UNIVERSITY OF MACEDONIA

SCHOOL OF INFORMATION SCIENCES

DEPARTMENT OF APPLIED INFORMATICS

Software-Defined Networks for Wireless Devices with Constrained Resources

Ph.D. Dissertation

of

Tryfon Theodorou

Supervisor: *Asst. Prof.* Lefteris Mamatas
Thessaloniki, September 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ

ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΑ ΔΙΚΤΥΑ ΓΙΑ ΑΣΥΡΜΑΤΕΣ ΣΥΣΚΕΥΕΣ ΜΕ ΠΕΡΙΟΡΙΣΜΕΝΟΥΣ ΠΟΡΟΥΣ

Διδακτορική Διατριβή

του

Θεοδώρου Τρύφωνα

Θεσσαλονίκη, Σεπτέμβριος 2020

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 multihop 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 thesis, we propose an approach that advances Low-power WSN to the era of IoT through centralized, programmable network control and operation, aligned to the SDN paradigm that brings the following novelties in contrast to the state-of-the-art 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 that employ heterogeneous mobile or static constraint devices, through supporting alternative topology discovery and control as well as flow establishment and routing mechanisms that dynamically adapt to different network deployments. We explore our proposed mechanisms and processes implementing a series of modular open-source frameworks (i.e., *CORAL-SDN*, *VERO-SDN*, *SD-MIoT* and *MINOS*). We evaluate the outcomes through a series of simulated experiments of alternative IoT use-case scenarios that include different topologies, network sizes, mobility characteristics, and high-volume transmissions with alternative communication patterns. The results prove our initial hypothesis as they verify that the SDN based programmable routing and control helps to overcome the challenges that IoT applications bring to WSN, providing flexible, efficient and robust performance with reduced control overhead.

Keywords: Software-Defined Networks, Internet of Things, Mobility, Wireless Sensor Networks, Software-Defined Wireless Sensor Networks, Out-of-band Control

Περίληψη

Το πεδίο της διδακτορικής διατριβής βασίζεται στη μελέτη των Ασύρματων Δικτύων Περιορισμένων Πόρων όπως τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks, WSN). Τα δίκτυα αυτά παρουσιάζουν ιδιαίτερο ενδιαφέρον, καθώς βρίσκουν πληθώρα εφαρμογών σε πολλούς τομείς με κυριότερο το Διαδίκτυο των Πραγμάτων (Internet of Things, IoT). Το Διαδίκτυο των Πραγμάτων, αποτελεί κεντρική τεχνολογία της λεγόμενης 4ης βιομηχανικής επανάστασης και δημιουργεί πλήθος νέων και σύνθετων απαιτήσεων στις δικτυακές επικοινωνίες καθώς ενσωματώνει εφαρμογές που εκτός από την βιομηχανία εφαρμόζονται και στην καθημερινότητα των ανθρώπων. Τα κύρια πρωτόκολλα δρομολόγησης που χρησιμοποιούνται σήμερα στο Διαδίκτυο των Πραγμάτων, όπως το πρωτόκολλο δρομολόγησης RPL για Ασύρματα Δίκτυα χαμηλής ισχύος και περιορισμένων πόρων, λόγω της κατανεμημένης αρχιτεκτονικής τους, παρουσιάζουν σοβαρές αδυναμίες στην ευέλικτη διαχείρισή τους και κατ' επέκταση στην εφαρμογή τους στις σύγχρονες απαιτήσεις (δηλ. διαλειτουργικότητα, κινητικότητα, ετερογένεια και ποιότητα υπηρεσίας - QoS). Σήμερα στην αιχμή της έρευνας και της εξέλιξης των πρωτοκόλλων δρομολόγησης καθώς και της βελτίωσης των μεθόδων διαχείρισης δικτύων βρίσκονται τα Προγραμματιζόμενα Δίκτυα (Software-Defined Networks, SDN). Τα δίκτυα αυτά βρίσκουν πρακτική εφαρμογή σε δίκτυα υποδομής και υποστηρίζονται από μεγάλες εταιρείες του χώρου και αποτελούν ένα νέο ερευνητικό πεδίο, το οποίο ενδεχομένως μπορεί να εφαρμοστεί και σε άλλους τύπους δικτύων. Ωστόσο, τα Προγραμματιζόμενα Δίκτυα παρουσιάζουν σημαντική επιβάρυνση στην κίνηση του δικτύου λόγω της πληθώρας των μηνυμάτων ελέγχου που χρειάζονται για να λειτουργήσουν. Για το λόγο αυτό δεν ενδείκνυται η απευθείας εφαρμογή τους σε τύπους δικτύων όπως τα Ασύρματα Δίκτυα Αισθητήρων καθώς επιβαρύνουν τους ήδη περιορισμένους πόρους τόσο των συσκευών όσο και του μέσου μετάδοσης του δικτύου. Η διατριβή επικεντρώνεται στην εφαρμογή τεχνικών Προγραμματιζόμενων Δικτύων μέσω του σχεδιασμού και της υλοποίησης αλγορίθμων και πρωτοκόλλων, τόσο στο επίπεδο του κεντρικού διαχειριστή (controller) του δικτύου, όσο και στο επίπεδο της επικοινωνίας δεδομένων του δικτύου (data-plane). Ο στόχος είναι η βελτίωση της διαχείρισης και λειτουργίας των Ασύρματων Δικτύων Αισθητήρων στο περιβάλλον του Διαδικτύου των Πραγμάτων λαμβάνοντας υπόψη χαρακτηριστικά όπως η κίνηση των κόμβων του δικτύου, η ταχύτητα και η ποιότητα επικοινωνίας μεταξύ τους. Οι ερευνητικοί στόχοι επικεντρώνονται στον σχεδιασμό, υλοποίηση και την αξιολόγηση προγραμματιζόμενων πρωτοκόλλων ελέγχου δρομολόγησης που λειτουργούν με χαμηλή επιβάρυνσή ελέγχου (δηλ. μειωμένο αριθμό μηνυμάτων ελέγχου) μέσω προτεινόμενης καινοτόμου λύσης που υποστηρίζει ξεχωριστό κανάλι ελέγχου μεγάλης εμβέλειας. Επιπλέον, τα πρωτόκολλα αυτά ακολουθώντας τις αρχές των Προγραμματιζόμενων Δικτύων υποστηρίζονται από λογισμικό ελεγκτή δρομολόγησης και διαχείρισης του δικτύου που βελτιώνει την ποιότητα λειτουργίας του δικτύου, σε ένα ευρύ φάσμα σεναρίων που χρησιμοποιούν κινητές ή στατικές συσκευές με περιορισμένους πόρους. Η χρήση κεντρικού διαχειριστεί για τον έλεγχο του δικτύου επιτρέπει τη δημιουργία και υποστήριξη καινοτόμων αλγορίθμων για την εύρεση της τοπολογίας του δικτύου και τον έλεγχο της δρομολόγησης των δεδομένων, που μπορούν να λειτουργούν εναλλακτικά και να προσαρμόζονται δυναμικά στις ιδιαιτερότητες και τα χαρακτηριστικά του δικτύου. Οι προτεινόμενοι μηχανισμοί και διαδικασίες υλοποιούνται σε μια σειρά ολοκληρωμένων λύσεων που υλοποιήθηκαν στα πλαίσια της διατριβής και διατίθενται με την άδεια ανοιχτού κώδικα (δηλ., CORAL-SDN, VERO-SDN, SD-MIoT και MINOS).

Η αξιολόγηση των προτεινόμενων λύσεων γίνεται μέσω μιας σειράς πειραμάτων που προσομοιώνουν εναλλακτικά σενάρια για το IoT χρησιμοποιώντας Προγραμματιζόμενα Δίκτυα Αισθητήρων και περιλαμβάνουν διαφορετικές τοπολογίες, μεγέθη δικτύου, χαρακτηριστικά κινητικότητας και μεταδόσεις μεγάλου όγκου δεδομένων, με εναλλακτικά μοτίβα επικοινωνίας. Τα αποτελέσματα επαληθεύουν την αρχική υπόθεση, καθώς επιβεβαιώνουν ότι η εφαρμογή των Προγραμματιζόμενων Δικτύων για τον έλεγχο δρομολόγησης και διαχείρισης του δικτύου βοηθά να ξεπεραστούν οι προκλήσεις που εισάγουν οι σημερινές εφαρμογές στα Ασύρματα Δίκτυα, παρέχοντας ευέλικτη, αποτελεσματική και στιβαρή απόδοση, με μειωμένη επιβάρυνση ελέγχου.

Λέξεις Κλειδιά: Ασύρματα Δίκτυα, Διαδίκτυο των Πράγματων, Ευφυή Προγραμματιζόμενα Δίκτυα

Acknowledgments

The doctorate dissertation is a road of loneliness that you cannot cross alone.

I had the honor of having Prof. Lefteris Mamatas as my academic supervisor. I would like to express my gratitude for his guidance and support, as, without it, I would not have been able to complete this endeavor. His endless energy and commitment to hard work motivated me the most.

I would also like to thank my supervisors Prof. Alexander Chatzigeorgiou for his support and advice when I most needed them during the challenging turns of this journey as well as Prof. Sofia Petridou for helping me especially when I was writing scientific papers.

I was fortunate to have George Violettas as a fellow traveler, friend, and colleague in this long, steep uphill road. In many cases, it was not easy to tell if he was pushing me or I was pulling him and vice versa until the end — George, "we did it."

Most importantly, I would like to thank my wife, Maria, as, without her constant support and patience, this work would not have been possible.

However, I would like to dedicate this work to my two boys Lazaros and Ioannis. I hope that the effort and commitment they witnessed from my side on this work during these years will be a raw model for their future endeavors.

Contents

List of Figures

List of Tables

List of Algorithms

1 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, 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 industrial monitoring, environmental and/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, mobility, 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.

An approach that targets the above challenges, exploits new flexible network architectures, such as the *Software-Defined Networking* (SDN) [5]. SDN uses logically centralized software, hosted in network nodes called SDN controllers, to control the behavior of a network by reducing the network configuration and management complexity. SDN was originally implemented for wired networks operating in cloud data centers. Recent research endeavors [6] are blending SDN and SDN-like architectures with Low-Power WSN technologies forming a new approach for SDN called *Software-Defined Wireless Sensor Networks* (SDWSN). The SDWSN paradigm brings new ways in the WSN control, management and operation. Although such approaches inherit the advantages of the traditional OpenFlow-based solutions over fixed networks, they are not yet fully aligned with the unique requirements of the IoT networks, e.g., the resource-constraint devices and the lossy nature of the wireless medium [7].

In this chapter, we highlight the importance of WSN towards the IoT era and the challenges posed by this imperative transition. Moreover, we feature the SDN paradigm and underline the aesthesia that can bring into the IoT. Further in, we list the aims and objectives of this dissertation along with its contributions, as well as the publications and achievements gained through its course. The chapter concludes with the overall dissertation structure.

1.1 The era of Internet of Things

Nowadays, the IoT technology holds a cardinal role as an enabler for a highly diverse set of services with respect to their requirements, including extremely high data-rates, ultra-low latency, low power consumption, a large number of connected devices, and high mobility. It has evolved from an experimental to a backbone technology able to connect myriads of people, things, and services for a broad range of businesses. Furthermore, IoT systems and devices define a huge area of innovation, allowing people to develop and design products, even at home [3]. In fact, it is characterized by explosive growth that is rapidly changing our world.

Table 1 indicatively presents IoT use-cases characterized by different network limitations, that express particular network conditions and device constraints, and application requirements. As such, *paramedics*, a typical e-health example, crucially demand high data-rates in real-time to support live video streaming to hospitals. Moreover, a wide deployment of sensors measuring ground and atmospheric metrics in large areas, as part of an *agriculture* application, prioritizes scalability issues over data-rates. Traffic prioritization is a crucial requirement for a *harsh working environment* that uses IoT devices' deployments for safety reasons, i.e., to prevent or face an accident; in this case connectivity and QoS in traffic is of paramount importance. Finally, applications with mobile IoT devices, such as drones or humans with wearables, strive for efficient solutions, e.g., neighbor discovery and routing, that handles mobility and takes into account device constraints, such as the remaining battery power.

Table 1: IoT Use-Cases

IoT Use-Case	Communication Constraints	Application Requirements
E-Health	reliability and performance	high data-rates, low latency and high priority
Agriculture	quality of service, scalability	low data-rates, low power, low priority, large range
Harsh Workplace	signal issues, reliability	traffic prioritization
Mobile IoT	mobility handling, scalability, resource-constraints and location awareness	resource efficiency

Relevant scenarios can be also found in the literature [4], where traditionally these applications are categorized in line with the type of communication as follows: Data Collection for many-to-one, Alerts and Actions for point-to-point and Data Dissemination for one-to-many communication.

A key enabling technology for the IoT suitable to implement a range of networking applications integrated to the traditional Internet infrastructure is the WSN [1]. In the next section, we present the main features of WSN and the challenges of employing them in IoT applications.

1.2 Low-Power Wireless Sensor Networks

WSNs consisting of wireless sensor nodes, mainly used to measure and monitor real-world physical conditions of interest, in high precision and large scale. Among the variety of WSN types, the *low-power wireless sensor networks with constraint devices* are prominent because they can be used in a variety of IoT applications utilizing characteristics like simplicity, low-cost communication, and multi-hop wireless connectivity with limited power and throughput requirements [8]. We explain that in this dissertation with the term WSN we refer to the low-power wireless multi-hop sensor networks with constraint devices.

A typical WSN, as in Fig. 1 consists of multiple sensor nodes, called motes, scattered across a field that measure physical phenomena like temperature, humidity, atmospheric pressure, and deliver the measured values to a central node (aka the Sink node) through multi-hop wireless transmissions.

For the communication among the resource-constrained nodes using IPv6, Internet Engineering Task Force (IETF) that leads standardization of communication

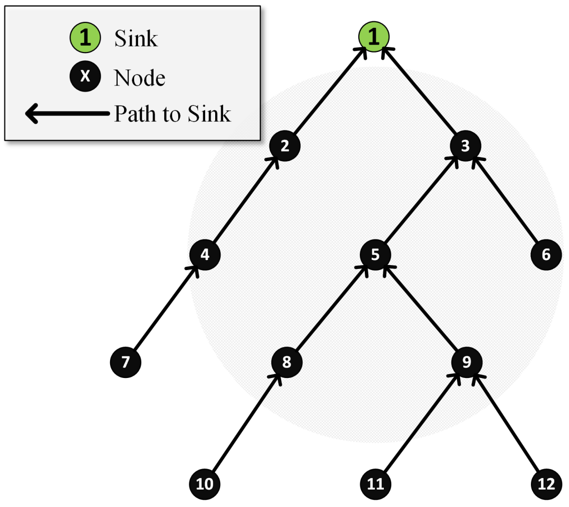


Figure 1: Wireless Sensor Network

protocols for Low Power and Lossy Networks (LLN) [9], [10] propose the IETF-IoT protocol stack. In Table 2, we depict the IETF-IoT layers in contrast to the TCP/IP protocol stack.

Table 2: WSN-IoT vs TCP/IP protocol stack

	IETF IoT Protocol Stack	**TCP/IP Protocol Stack**
Application Layer	IETF COAP, MQTT	HTTP, FTP, SMTP...
Transport Layer	UDP	TCP, UDP
Network Layer	RPL, IPv6	IPv4, IPv6
Adaptation Layer	IETF 6LoWPAN	-
Data-link Layer	IEEE 802.15.4 MAC	Network Interface &
Physical Layer	IEEE 802.15.4 PHY	Network Access

The majority of protocol stacks implemented for Low-Power WSN, use the IEEE 802.15.4 standard in their lower layers [11]. IEEE 802.15.4 was initially introduced in 2003, aiming to provide ultra-low complexity, low-cost, and extremely low power wireless connectivity characteristics among inexpensive, fixed, portable and moving devices [12]. It mainly describes the operation of the Physical Layer and the Data Link Layer. In the Physical Layer, IEEE 802.15.4 operates in two radio bands:

- the Low-band, with two Sub-GHz frequency options, the 868 MHz frequency, with 1 channel and 20 kb/s bandwidth, and the 915 MHz frequency with 10 channels and 40 kb/s bandwidth, and
- the High-band, in the free 2.4 GHz frequency, offering 16 channels and 250 kb/s bandwidth.

Depending on the frequency, the radio interface can achieve transmissions in a range from few tenths of meters to a few kilometers. Both versions of the Physical Layer use a common packet structure, enabling the definition of a common Data Link Layer interface. The latter operates a Medium Access Control (MAC) sub-layer that controls the access to the radio channel using the CSMA-CA mechanism. The IEEE 802.15.4 MAC is responsible for flow control, frame validations, and network synchronization, as well as implementing acknowledged frame delivery.

To enable IP connectivity in resource-constrained sensor networks, IETF IoT stack utilizes the Adaptation Layer, an intermediate layer between the Network and Data-link layers. The IPv6 over Low-Power WPAN (6LowPAN) Working Group [13] established for the Adaptation Layer the *6LoWPAN* protocol that applies IPv6 optimization over the MAC and Physical Layers of IEEE 802.15.4. 6LowPAN manages to maintain compatibility with IPv6 and mitigates the IPv6 large-packet-size problem by using header-compression and packet-fragmentation.

Routing Over Low power and Lossy networks (ROLL) Working Group [14] develops the distance vector protocol *Routing over Low Power and Lossy Networks* (RPL) protocol [15]. RPL is the de facto standard routing protocol for Low-Power WSN that, along with the IPv6 protocol, constitutes the IETF Network Layer. In particular, 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 [16]. 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). Furthermore, 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. In this thesis, we focus mainly on processes and mechanisms related to the Network Layer and particularly on routing and data forwarding processes.

The IETF Transport Layer favors UDP over TCP protocol because it is energy efficient, and aligns with the majority of IoT application requirements (i.e., monitoring and data collection). IETF IoT stack also supports the TCP protocol and, although not included in Table 2, there are protocols, such as the Application Layer protocol MQTT that relies on it.

Lastly, on the Application Layer, IETF proposes various protocols such as CoAP or MQTT. Each of these protocols, as discussed in [17], has different advantages and disadvantages and make their use better or worse, depending on the IoT application scenario.

Although IETF protocol stack contributes to the network's operation and performance we argue that the imperative transition of WSN to the era of the IoT and the increasing complexity of the IoT applications has laid bare the shortcomings of the existing WSN protocols revealing a number of exacerbating challenges related to network control and operation. For example, back in 2009, when IETF specified RPL's architectural characteristics, the communication patterns were mainly motivated by the need to support monitoring applications and their routing requirements. However, although monitoring is still a key IoT application, the plethora of emerging IoT applications requires an enhanced network protocol operation that deals successfully with all communication patterns (i.e., one-to-one, one-to-many, and many-to-one). Right after, we list the major challenges WSN facing today in the context of the IoT.

1.2.1 The Challenges of Low-Power WSN in the IoT era

IoT brings new potentials in humans' everyday life activities through a variate of applications with challenging communication requirements for the underlying WSN framework, such as QoS, mobility, elasticity, heterogeneity, interoperability, adaptability, security, and energy efficiency [18], [19]. In detail, modern IoT applications require from the Low-Power WSNs:

1. *QoS*: to implement efficiently multiple communication patterns with reduced end-to-end delays, maintaining robust packet delivery and reduced control overhead, utilizing advanced network topology control mechanisms, flexible flow control rules, and intelligent routing-forwarding decisions that support alternate paths enforcing the effectiveness and stability of network's data communication.
2. *mobility*: to handle issues raised from IoT devices' mobility and consequent connectivity hand-overs (e.g., additional *control overhead* to maintain the topology), which become “costly” without suitable dynamic routing adjust-

ments. Furthermore, mobility-aware mechanisms should not overload possible co-existing static nodes.

3. *elasticity*: to appropriately deploy and configure the protocol dynamically toward satisfying the applications' requirements; moreover, to adapt to the network context environment (i.e., responding to the IoT's network feedback) by enforcing strategies for flexible and individual IoT devices' configuration, that improve performance and resource-allocation whilst reducing cost.
4. *heterogeneity*: to integrate hardware (e.g., communication interfaces) and software (e.g., messaging protocols like CoAP) particularities, as well as nodes' characteristics (e.g., battery-powered or not). Carefully designed abstractions are needed to hide heterogeneity and allow devices to export common features to the higher control and application planes.
5. *interoperability*: to dispose of appropriate application interfaces and abstraction layers using compatible communication formats to work with other products or systems, at present or in the future.
6. *adaptability*: to provide access to networking open-source software resources and tools that maintain modular architectures allowing easy incremental updates without restrictions.
7. *security*: to protect the network operation and data from several types of attacks against different layers of the network stack [20] and in diverse application domains, including critical infrastructure systems. Security issues like authentication, authorization, integrity, and confidentiality are prominent in the IoT.
8. *energy efficiency*: to introduce exceptional requirements in routing protocols that should maintain low energy consumption through efficient protocol algorithms and mechanisms in terms of processing power and memory capacity [21].

We clarify that in this dissertation, we propose solutions and mechanisms that address the first six of the above challenges, while we leave the issues related to security and energy efficiency as future work.

The need for introducing applications with diverse, potentially stringent performance requirements calls for flexible and performance-efficient network protocols driven from sophisticated, low-overhead network control features that overcome the above challenges. In this context, SDNs [5] are employing new flexible, logically-centralized network architectures, as discussed in the next section.

1.3 Software-Defined Networks

SDNs provide a new, elastic network paradigm that transforms traditional network backbones into flexible service-delivery platforms and improves the network's utilization [5]. They exploit a logically centralized network architecture that decouples network control from the data plane using the OpenFlow protocol [22].

The primary entity of SDN is the controller, a network control software that has the complete picture and control of the entire network. The data plane is responsible for the network's data forwarding process employing the rules in the routing table. The data forwarding rules in the routing table are updated by the controller utilizing the OpenFlow protocol messages.

The controller provides the means to optimize the network operation in real-time and respond quickly to network usage changes without the need for manual configurations to existing infrastructure or purchase of new hardware. Additionally, this programmable approach allows the network to interact with applications and effectively reshape the network based on their needs.

Although SDNs and the prominent OpenFlow protocol were introduced for infrastructure networks, recent research endeavors [23], [24] are blending SDN architectures with WSN technologies, forming a new approach called *S*oftware-*D*efined *W*ireless *S*ensor *N*etworks (*SDWSN*), considered in the next section.

1.3.1 Software-Defined Wireless Sensor Networks

The SDWSN paradigm brings new prospects to WSN architecture, control, management, and operation [25]. It allows new, improved WSN routing and topology control protocols, utilizing the global centralized network view. The offloading of network control intelligence from motes to a central controller reduces the computational process and memory footprint for the low memory and processing power devices. Their centralized network control and programmability features allow for efficient, bespoke network protocol operation to the particular requirements of IoT applications.

To this end, efforts presented in [26] and [27], 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 [23], [24] 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 [28]. 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.

However, SDN is not fully aligned with the unique requirements of the IoT networks and poses additional challenges not present in the conventional WSN networks. In particular, SDN control communication messages with the controller increases the number of control packets in the network drastically. This further impairs the resource-constraint nodes as well as the low quality and lossy nature of the wireless communication medium [7]. To this end, in the next section, we present the goals and contributions of this thesis.

1.4 Evolving SDWSN in IoT

Inspired from the above, in this dissertation, we investigate the expansion of the SDN concept to IoT networking facilities, like the Low-Power WSNs. To mitigate the challenges discussed earlier, we propose innovative protocol architectures and mechanisms for WSNs that fully align with the SDN paradigm. We also design and implement novel SDWSN frameworks, and through extensive evaluation, we verify our initial objectives.

In this section, we: (i) specify the objectives of this thesis; (ii) summarise its contribution; (iii) record our scientific publications that support the results and conclusions; and (iv) report awards we achieved in the context of this research work.

1.4.1 Objectives

In this dissertation, we aim to achieve the following objectives:

- To identify the main challenges that IoT brings to the already existing Low-power WSN solutions, particularly for the state-of-the-art WSN routing protocols like RPL.
- To address those challenges, as described in section 1.2.1, utilizing the SDN paradigm and avoiding the negative consequences, like the control overhead, that SDN brings into the IoT constrained-devices.
- To design novel SD-based architectures and mechanisms that successfully adapt and fuse with the Low-power WSN environment restrictions and peculiarities.
- To develop open-source IoT frameworks and SDWSN protocols that provide improved IoT network operation as well as new perspectives and grounds to the modern IoT applications.
- To demonstrate the efficiency of the proposed solutions by measuring and evaluating the results of various network communication metrics over realistic and versatile use-case scenarios.
- To suggest further improvements and key research areas that now and in the future will exploit the benefits that the centralized control brings to WSN and IoT applications.

1.4.2 Contribution

Here we enlist a summary of the contribution of this work.

- The topology discovery process and the representation graph of the network's connectivity are key features in WSN operation as well as one of the essential inputs for an SDN controller's decision-making mechanisms. As such, we implemented two novel topology discovery algorithms [29]: (i) the Node's Advertisement Flooding (TC-NA) algorithm; and (ii) the Node's Neighbors Requests from the Controller (TC-NR) algorithm. These algorithms realize topology discovery and maintenance processes specifically designed to operate in the SDN for WSN context, taking into account the advantages and flexibility of the centralized SDN approach and the unique characteristics of the multi-hop wireless communication medium. We evaluated such topology control strategies using our novel SDN experimentation facility for IoT [30]. The results demonstrate new significant improvements in WSNs management, control features and performance in terms of topology construction time and less topology maintenance overhead.

- To amalgamate SDN in the context of WSN we designed and developed *CORAL-SDN* [31], an SDN solution for WSNs which: (i) uses modular intelligent centralized control mechanisms to adjust the protocol's functionalities dynamically; (ii) supports elasticity to the challenging requirements of the WSNs; (iii) maintains a scalable architecture; and (iv) exhibits improved network management and operation in terms of performance and resource utilization. *CORAL-SDN* provides a suitable environment for hands-on experimentation, featuring the *CORAL-SDN* protocol operation in real test-beds and highlighting the improvements that SDN brings to IoT.
- Towards reduced control communication overhead with improved robust packet delivery and reduced End-to-end data transmission delay applying multiple communication patterns, we evolved *CORAL-SDN* to *VERO-SDN* [32]. *VERO-SDN* is an 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 wireless control channel that introduces one-hop communication with the SDN controller, through adopting a double protocol stack and appropriate routing control; (ii) an SDN protocol and controller dynamically maintaining the IoT topology based on the two aforementioned topology discovery algorithms; and (iii) two types of flow-rule establishment processes able to either configure only the *Next Hop (FE-NH)* of the node's forwarding table or the *Complete end-to-end Path (FE-CP)*, configuring the forwarding tables of all nodes participating in the path. Moreover, *VERO-SDN* evolves towards: (i) supporting alternative application communication patterns (i.e., many-to-one, one-to-many, one-to-one [33]); (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. The impact of *VERO-SDN* operation is reflected in our extended evaluation (i.e., discussed in Chapter 5), highlighting *VERO-SDN's* achievements in terms of packet delivery ratio, network overhead, and end-to-end delivery time.

- En route to the need of *Mobile IoT* environments in this dissertation, we propose *SD-MIoT* [34], an open-source SDN solution that suggests novel SD-based mechanisms for *Mobile IoT* environments. In detail, it consists of a modular SDN controller and an OpenFlow-like protocol that builds up on top of *VERO-SDN* protocol, supporting: (i) *MOB-TD*, a mobility-aware topology discovery mechanism utilizing a hybrid of globally- and locally-executed topology discovery processes; (ii) routing policies adapted to mobility, employing data forwarding prioritization based on the nodes' mobility status; (iii) a hybrid combination of reactive and proactive flow-rule establishment methods; and (iv) *MODE*, a novel intelligent algorithm that detects passively in real-time the network's mobile nodes, utilizing the SDN controller's connectivity graph. We provide extensive evaluation results over realistic scenarios, further confirming the suitability of *SD-MIoT* for mobile IoT environments and demonstrating reliable operation in terms of successful data packet delivery with low control overhead in Chapter 5.
- In the direction of IoT challenges like heterogeneity, elasticity, scalability and as well as mobility, we designed and developed MINOS [35], a multi-protocol SDN platform for IoT that implements service-awareness utilizing: (i) appropriate SDN abstractions and interfaces for logically-centralized network control of diverse and resource-constraint IoT environments; (ii) two network protocols that are deployable and configurable on-demand; and (iii) a Graphical User Interface (GUI) that provides a bespoke dashboard and a real-time visualization tool. Due to its components, MINOS enables experimentation with novel network control features and protocols that realize optimized routing over heterogeneous IoT nodes; application of real-time strategies as a response to the dynamic network conditions; support of individual protocol

configurations per node; and flexibility to accommodate new protocols and control algorithms.

1.4.3 Published Work

The outcomes of this dissertation have been documented in a number of papers that have been published in the following scientific journals and international conferences.

1.4.3.1 Scientific Journals

J.1 T. Theodorou, L. Mamatas, "SD-MIoT: A Software-Defined Networking Solution for Mobile Internet of Things," *IEEE Internet of Things Journal*, 2020, doi:10.1109/JIOT.2020.3027427.

J.2 T. Theodorou, L. Mamatas, "A Versatile Out-of-Band Software-Defined networking solution for the Internet of Things," *IEEE Access*, vol. 8, pp. 103710-103733, Jun. 2020.

J.3 T. Theodorou, G. Violettas, P. Valsamas, S. Petridou, Lefteris Mamatas, "A Multi-Protocol Software-Defined Networking Solution for the Internet of Things," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 42-48, Oct. 2019.

1.4.3.2 International Scientific Conferences

C.1 T. Theodorou, L. Mamatas, "Software-Defined Topology Control Strategies for the Internet of Things," *Conference on Network Function Virtualization and Software Defined Networks - (NFVSDN)*, IEEE, Berlin Germany, Nov. 2017, pp. 236–241.

C.2 T. Theodorou, L. Mamatas, "CORAL-SDN: A Software-Defined Networking Solution for the Internet of Things," *Conference on Network Function Virtualization and Software Defined Networks - (NFVSDN) 2017*, IEEE, Berlin Germany, Nov. 2017.

C.3 G. Violettas, T. Theodorou, S. Petridou, A. Tsioukas, and L. Mamatas, "An Experimentation Facility Enabling Flexible Network Control for the Internet of Things," *International Conference on Computer Communications - (INFO-*

COM), IEEE, Atlanta USA, May. 2017.

1.4.4 Awards

We participated in the *Elastic Wireless Networking Experimentation (eWINE) Grand Challenge* [36], in Oulu, Finland June 2018. The *eWINE* Grand Challenge, was addressed to the research community and highly trained industry professionals, providing the grounds for competition among solutions that deal with elastic connectivity and routing that can scale to a high number of users in a short time-span through the use of an agile infrastructure utilizing intelligent software mechanisms and flexible hardware.

In this competition we received the 1^{st} runner-up award presenting the *Intelligent Network Control for the Internet of Things (INTER-IOT)* project. INTER-IOT demonstrates our research outcomes related to SDN-based efficient end-to-end wireless communication with dynamically optimized routing for heterogeneous mobility-aware networks. In detail, INTER-IOT is based on *CORAL-SDN* [31], our protocol described in following sections, which implements elastic network adaptations, such as SDN-based network discovery, topology maintenance and routing, over IoT devices to improve performance, reduce cost and resource utilization. Moreover, *INTER-IOT* is exploiting the *WiSHFUL* [37] infrastructure and the *eWINE* innovative intelligent facilities [38]. The project was presented and tested on the IMEC w-iLab.2 test-bed [39], using our CORAL experimentation facility [30].

1.4.5 Conference Tutorials

In order to disseminate and demonstrate examples of the architectures and mechanisms developed in this dissertation we successfully conducted a half-day Conference Tutorial with the title "Softwarized Internet of Things with Lightweight Clouds and Practices," at the 3rd IEEE Conference on Network Function Virtualization and Software Defined Networks in Berlin, Germany on Monday $6th$ of November 2017 (Tutorial-#3 [40]). The tutorial provided knowledge and hands-on experience to participants particularly in developing IoT solutions in an SDN controlled environment deployed in the edge of the infrastructure network using a lightweight clouding paradigm with single-purpose network functions.

1.5 Dissertation Outline

The remainder of this dissertation is organized as follows.

Chapter 2 presents the state-of-the-art solutions addressing the SDWSN and relevant WSN protocols as well as background information to the reader.

Chapter 3 elaborates on the proposed mechanisms and techniques we designed and implemented in this dissertation towards the integration of SDN with WSN. We present in detail the architectural characteristics of *CORAL-SDN* and *VERO-SDN*, including their main components, functionalities and protocol mechanisms.

Chapter 4 provides the design details of *SD-MIoT's* components, algorithms, and its essential operation in Mobile IoT environments. Additionally, we describe the architecture of the mutli-protocol approach that *MINOS* brings in scaled heterogeneous IoT environments.

Furthermore, an extensive evaluation is covered from Chapter 5, illustrating the performance and reliability advantages of our solutions. For each evaluation scenario, we provide motivating use-case scenarios highlighting the advantages of the proposed platforms and protocols.

Finally, Chapter 6 concludes the dissertation while also discussing further improvements and research challenges identified through the experience gained from the implementation of this research endeavor.

2 Literature Review

In this chapter, we present a comprehensive review of the existing related to this dissertation research works. We mainly aim at SDWSN protocols and communication platforms, supporting our descriptions with necessary background information and highlighting a range of prominent issues related to the contributions of this work like network mobility and out-of-band control.

2.1 SDN inspired WSN Protocols

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 Section 1.3.1. The first one proposes SDN-inspired network control facilities that operate on top of existing WSN protocols, such as RPL [15], 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.

2.1.1 Evolutionary approaches to softwarized IoTs

The first category of proposals is *evolving the WSN towards the SDN world.* Oliveira et al. [41], discuss the synergy between the SDWSN protocol TinySDN [42], [43] with RPL and how they can benefit from each other to improve performance in IoT deployments. The authors identify the benefits that a centralized approach provides to WSN towards improved resource management. In particular, they suggest a centralized route definition mechanism that coexists with the *Collection Tree Protocol* (CTP) [44], the routing protocol of TinyOS [45] operating system, that improves network lifetime.

μSDN [26] presents a lightweight SDN framework for low-power wireless IPv6 IoT networks, supporting interoperability with existing distributed routing protocols such as RPL. The control overhead challenge that SDN introduces is mitigated through an architecture that uses a number of overhead reduction functions and optimization techniques. μSDN, while maintaining comparable performance and scalability with RPL-based IEEE 802.15.4-2012 networks, provides the network with the SDN architecture opportunities like Global Knowledge, Network (Re)Configurability and Virtualization. Through a scenario within a simple network under intermittent interference, the authors show how μSDN provides redundancy to priority flows, achieving QoS for critical network flows with a reduction in latency and jitter compared to RPL.

A relevant to the above solution is *Whisper* [46], which manipulates RPL and 6TiSCH operation using a controller that remotely controls nodes' forwarding and cell allocation, sending carefully computed routing messages that are fully compatible with the protocols run in the network, i.e., RPL. *Whisper* achieves rerouting around low-battery devices and provides run-time-defense to jamming attacks, operating internal algorithms that aim for efficient network operation, reducing the number of messages sent by the controller, to make the exerted control as lightweight as possible.

Recent work from Violettas et al. [47], investigate two SDN-like routing control strategies utilizing the controller's global network view. More precisely, they investigate the *Moderate RPL control* that enables dynamic configurations from the controller of RPL's parameters to improve its operation in mobile environments and the *Deep RPL control* that utilizes a new parent selection Objective Function that enforces direct point-to-point paths through prioritization based on link-coloring.

Summarizing the above evolutionary softwarized IoT approaches, we clarify that the prime advantage of such solutions is the backward compatibility with the traditional protocols. However, they fail to significantly improve the protocol's performance because they cannot fully exploit the SDN features due to interoperability and backward compatibility limitations.

2.1.2 Revolutionary approaches to softwarized IoTs

The second category of solutions follows the reverse path through *bringing the SDN paradigm to the WSN environments*, i.e., radically changing the network environment [48].

An early work by Luo et al. [49] is proposing the *Sensor OpenFlow*, a conceptual OpenFlow-based protocol [22] as a solution to WSN-inherent problems, i.e., application-specific deployment, underutilization of resources, rigidity in policy changes, and difficult network management. This research endeavor is the first that suggests the application of the OpenFlow protocol in WSN. The same paper is identifying key technical challenges of *Sensor OpenFlow*, e.g., its increased control overhead. On the same track, Gante et al. [50] elaborate on benefits that SDN brings to WSN, e.g., enhanced network management, advanced topology discovery, and energy-saving. The authors propose a theoretical architecture for an SDWSN, where the controller is integrated at the base station.

Early work from Costanzo et al. [51] proposes *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. With SDN-WISE [52], Galluccio et al. extend 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. SDN-WISE aims to simplify the management of the network towards the development of novel applications, and the experimentation of new networking solutions in WSNs. In [53], the authors present a demo of SDN-WISE and provide access to the protocol's open-source and examples in (https://sdnwiselab.github.io/).

Towards a better integration of the SDN-enabled IoT with the fixed SDN environments, Anadiotis et al. [54] 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 [55] 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 [42], [43] implements a distributed TinyOS-based control plane architecture based on multiple controllers. The authors propose an SDN architecture with hierarchically organized multiple controllers aiming to manipulate the increased control traffic between network nodes and the controller. The protocol operates under the TinyOS [45] operating system for low-power devices. In [56], *TinySDN* enabled nodes are enhanced with the *Spotled*, a 2-level hierarchy of physically distributed global and local SDN controllers. The local *Spotled* controllers use local information to reply to nodes in their area while a global controller oversees the whole network. This architecture strives to reduce control traffic by reducing the distance that control messages travel in the WSN multi-hop environment.

Soft-WSN [57] 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. Additional to SDN centralized control, the authors propose device control mechanisms that schedule the sensing task and delay duration, as well as the active sleep period of the IoT devices.

Alves et al. [58] propose *IT-SDN*, a SDWSN protocol inspired from *TinySDN*. *IT-SDN* suggests an architecture that separates the southbound communication protocol in two parts, i.e., neighbor discovery, and controller discovery. It is implemented for Contiki OS [59], and the source code is provided to the research community as open-source for experimentation. The paper [60] provides a comparison between IT-SDN and RPL routing protocol evaluating metrics related to data delivery, data delay, control overhead, and energy consumption. The results show that IT-SDN operation is sufficient only in small network setups, i.e., less than 15 nodes. The authors suggest future work solutions that will improve its performance, such as proactive flow setup or multiple node configuration through a single control packet.

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 control messages over the LLN multi-hop medium.

Most recent work from Baddeley et al. [61], [62] towards the alienation of control from data messages, suggests *AtomicSDN*. This SDWSN solution proposes a time-sliced mechanism that separates the SDN control communication from the WSN data plane layer messages using designated flooding periods for the control messages. The evaluation results demonstrate improved network latency, reliability, and low energy consumption.

SDSense [63] implements an SDWSN architecture that separates the network management in static and dynamic network events. The slow-changing static phenomena (i.e., topology control) managed from a logically centralized controller using the SDN approach. Whereas the rapidly changing dynamic phenomena (i.e., congestion control) are handled using an agile approach of local controllers responsible for neighboring nodes. To reduce collisions and provide reliable performance, *SDSense* is using Time Division Multiple Access (TDMA) transmission that schedules the transmissions in times of globally synchronized slots. To optimize the bandwidth allocation, it uses a centralized Network Utility Maximization algorithm balancing between centralized solutions and the need for rapid reaction to dynamic changes. The authors evaluate *SDSense* through simulations demonstrating improvements in network performance over other solutions. However, their implementation is not available for comparison against other SDN solutions. Besides, the assumption that topology control is a static network process does not apply in all network environments, e.g., mobile.

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 unreliable control plane operation and a data communication plane encumbered with control messages. Moreover, none of them operates on mobile environments or tackles network mobility or heterogeneity issues.

Considering the advantages that the SDN paradigm brings to WSN, in this dissertation, we propose an SDN-based solution that employs a separate out-of-band control channel to overcome the above issues. As such, in the next section, we present research works that adapt the approach of utilizing a second radio communication channel to control network functions.

2.2 Out-of-Band Control

Out-of-Band control separates network control communication from user data and passes all control data through a separate communication line. It requires dual-network interfaces and communication medium. Its application in wireless networks is rather easy as the communication medium can accommodate multiple channels easily. 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 [64] and [65], present *WaCo*, a wake-up radio COOJA extension tool that allows the exploration of a second radio channel. The separate 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. The results confirm that wake-up technology has great potential to offer significant energy savings without compromising on reliability and latency. However, the solution implements only a binary on-off control message that cannot be used in more composite applications.

WASP [66] is an SDN inspired framework that implements a data plane for mobile ad-hoc networks using the Wi-Fi Direct and manipulates its operation through an LTE-based control plane. The controller, communicating through the wide-area radio, controls mobile nodes' traffic using neighbor information provided by the mobile devices. WASP architecture can be used in different networks and applications; however, it currently considers only smartphone devices.

Gu et al. [67] 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 communication between the network nodes and a base station using LoRaWAN. The central base station improves the packet delivery ratio by ad hoc interventions to forwarding decisions of the node's routing Collection Tree Protocol (CTP). The authors plan to systematically study network performance and manageability issues as well as the impact on node energy consumption.

Inspired by the above research projects, we came up with the idea of utilizing two separate radio channels, one for control and one for data communication. Hence, as described in Chapter 3, we materialize an SDWSN framework where the SDN controller communicates with the network nodes through a separate one-hop out-of-band communication channel.

2.3 Mobility in WSNs

Network mobility is a key feature that characterizes today's IoT applications. However, WSNs were predominately used only for data collection applications with static network nodes. As Such, RPL [15] the de-facto routing solution for constrained devices and WSNs was not originally designed for mobile environments [68], [69]. Several studies surveyed in [70], [33], aiming to improve RPL performance in mobile topologies. The majority suggests adaptations, including explicit identification of mobile nodes, improvements of preferred node parent selection for its topology graph formation, and adaptation of solicitation messages' interval for neighbor discovery.

Bouaziz et al. [19] identify, among others, three challenges that WSN protocols must transcend for a sufficient operation in a mobility context: (i) robust *topology control*, with reference to network connectivity and coverage area; (ii) elastic *routing* and *forwarding*, regarding alternate paths and stability; and (iii) efficient *QoS*, in terms of data loss rate and end-to-end delay.

Related research works that aim to address those challenges propose the adjustment of the network's protocol configuration parameters to the mobility environment characteristics [71]. For example, they increase the frequency of the node solicitation control messages, or avoid routing paths through mobile nodes, or intensify the forwarding rules' updates. Such solutions, although they improve the routing efficiency, flood the network with control messages and increase the protocol's footprint, eventually reducing the network's performance, especially in LLN environments like Low-Power WSN.

In particular, Fotouhi et al. with *mRPL* [72] and its improved version *mRPL+* [73] implements a hand-off handling topology control mechanism for RPL, where the mobile nodes can proactively disconnect from existing attachment points and connect to more suitable ones, based on RSSI measurements. The solution maintains multi-hop efficiency using traffic awareness filters as well as add-ons for best route flow establishment towards the sink and routing loops avoidance. The mechanisms reduce significantly packet loss and hand-off delays while maintaining backward compatibility with the RPL protocol.

Another study [74] inspired by the [75] suggests *BPRL*, a dual routing protocol, which can adaptively switch between RPL and *Backpressure* routing protocol [76], depending on network conditions. Two adaptive algorithms *QuickBeta* and *QuickTheta* are used to improve node mobility and balance throughput, respectively. Although it is not always the case in WSN applications, the solution requires a constant flow of data in order to maintain the functionality of the Backpressure routing protocol.

Recent work *EMA-RPL* [77] introduces a proactive process that can anticipate and predict the movement of mobile nodes by comparing the radio signal strength to its point of attachment and applying an energy-efficient parent replacement strategy. To maintain efficient downwards-routing uses two types of DAO messages to inform previous and new parent node for its actions. To avoid routing interruptions, EMA-RPL configures all mobile nodes as leaf-nodes. However, this configuration option requires knowledge of mobile and fixed nodes before starting up the network operation.

Similar to the above, in [78], the authors propose a predefined medical application context where the mobile nodes are configured as leaf nodes to exclude message-forwarding through the latter. At the same time, considering more general cases, they enhance the trickle timer algorithm [79] (i.e., the topology refresh mechanism of RPL) and adopt the preferred parent election metric to improve the mobile node disconnection periods.

MARPL [80] enhances the mobility management of the RPL protocol based on neighbor variability metrics. The design of MARPL encompasses mobility detection and preferred parent unavailability detection mechanisms. The protocol adjusts the DIO and DIS messages as well as the trickle algorithm to detect disconnections and changes caused by the mobile nodes. The authors report results that indicate improved performance in terms of overhead, residual energy, and packet delivery rate in comparison with RPL and protocols mentioned above, like mRPL.

An SDN-inspired solution by Violettas et al. [27], configures the RPL protocol parameters based on the particular behavior of the nodes (e.g., whether they are mobile or not) from a centralized controller that forms closed *monitor-decide-configure* control loops. In practice, the controller dynamically changes the trickle algorithm configuration parameters like I_{mins}, and boost-up network solicitation and discovery messages to timely detect network connectivity changes. The solution provides elasticity in RPL's functionality by tackling mobility issues on-the-fly through efficient decisions of protocol's performance trade-offs, e.g., between successful packet delivery and routing overhead.

We argue that, although the above suggestions improve RPL's performance to some extent, further improvements are hindered because it preserves architectural features that do not fit well to mobile network conditions, like the topology control trickle algorithm. Moreover, tweaking RPL parameters towards mobile topology requirements inflates the frequency of node solicitation control messages and intensifies the forwarding rules' updates. These practices, although they improve the routing efficiency, they flood the network with additional control messages that impact on the network performance, especially for LLN network environments. Moreover, we point out that: (i) none of the above mobility aware solutions integrate the SDN paradigm in their architectures; and (ii) none of the SDWSN frameworks discussed in Section 2.1.2 operates on mobile environments or tackles network mobility issues.

Motivated from the above open research area and considering the advantages that the SDN paradigm can bring to WSN, in this dissertation, we suggest mobility dedicated mechanisms that handle WSN mobility scenarios using the SDN paradigm in the prospect of the IoT era, discussed in Chapter 4.

2.4 Mobility Detection

The majority of the research solutions discussed in the previous section to mitigate challenges like the excess of control messages, apply separate policies only to mobile nodes. Thus, the majority of them operate with the prerequisite knowledge of the mobile nodes in a network. To this end, we identify four different strategies that address this aspect, briefly described below:

- *Human-in-the-loop* proposals are based on an external observer (e.g., network administrator) that acquires each node type (i.e., mobile or fixed) and either pre-configures the protocol operation [70] or utilize additional network control messages, in real-time, to adjust each node's protocol operation according to its type [27]. The main disadvantages are the external intervention to protocol operation and the lack of flexibility.

- *Hardware-based* solutions are utilizing information from specialized hardware (e.g., GPS antenna or accelerometer sensors) to realize the nodes' mobility behavior and adapt protocol operation accordingly [81]. Although this solution can successfully recognize the mobile nodes, the specialized hardware increases the implementation and deployment complexity.
- *Protocol-based* strategies are using mobility metrics (e.g., radio signal strength indicators, or neighbor nodes connectivity changes). When the measurements exceed pre-configured thresholds, such proposals adjust the protocol topology discovery periods [82], e.g., the trickle or reverse trickle timer [69]. Although this method excels in flexibility compared to the previous approaches, decision-making on a node level can lead to false positives, because of the mobility metrics affinity with the behavior of neighbor nodes.
- *Intelligent approaches* exploiting Artificial Intelligence / Machine-Learning techniques. For example, *Dribble* [83] suggests a learn-based timer scheme selector for mobility management in IoT, utilizing an intelligent Multi-Layer Perceptron classifier, on a node level, to detect mobility patterns. We argue that memory and processing-intensive algorithms, such as intelligent classifiers, result in excessive consumption of energy and hardware resources for the IoT devices. Moreover, the limited node-level visibility constrains the algorithm's input to local information only.

Inspired from the aforementioned intelligent approach and exploiting the SDN paradigm features, we propose in Chapter 4 our intelligent solution that suggests a new passive mobility detection algorithm. The algorithm compared to the above solution successfully operates without any human or hardware intervention or any additional overhead to the network or the low power devices, assisted by the SDN global network view.

Concluding this chapter, we highlight the main limitations that the current state-of-the-art solutions face towards the adoption of the SDN paradigm to the WSN environment:

- SDN-inspired solutions on top of existing WSN protocols like RPL do not fully exploit SDN potentials because of underlying backward compatibility restrictions and inherited limitations.

- SDN-like solutions struggle because of the WSN lossy communication medium impediment that undermines critical SDN operation features, i.e., the communication with the SDN controller. Considering the experience gained from the studies described in this chapter, we present in the next Chapter 3 our proposed solutions.
- To our knowledge, there is no SDN solution for WSN, proficient in handling network mobility issues. In Chapter 4, we demonstrate our SDN mechanisms that address the challenges of mobile network environments.

3 Evolution of Software-Defined Wireless Sensor Networks

In the previous chapter, we described the state of the art solutions as well as trends and challenges of WSN's evolution towards the new era of IoT. In this chapter, we elaborate on our proposal that suggests the evolution of WSN through the adoption of the SDN paradigm, applying network programmability with centralized control, aiming to improve the network's QoS. In detail, we propose an operational framework that in contrast to the related works brings the following novelties: (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 (i.e., one-to-many, many-to-one and one-to-one) through supporting two alternative topology discovery and two flow establishment mechanisms.

The network *topology* and *routing control* are both 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. Moreover, we elaborate on the main network control features and their associated protocol aspects, i.e., residing at the controller and the data plane, respectively.

The proposed mechanisms are implemented in our OpenFlow-like SDWSN solution *VERO-SDN* and evaluated through a series of simulations presented in Chapter 5. In the subsections below, we present a high-level overview of those mechanisms, their architecture, as well as the protocol operation and interfaces, including a detailed description of the associated software and hardware components.

Moreover, we describe the main network control features of *VERO-SDN*[1] platform and their associated protocol aspects, i.e., residing at the controller and the data plane, respectively.

3.1 Proposing Out-of-Band Control in SDWSN

In contrast to the other approaches implementing the SDN paradigm over the IoT as discussed in Chapter 2, we suggest rather than using the same medium to communicate data and control messages (i.e., in-band control), to split out the network communication control to a separate dedicated radio channel (i.e., out-of-band control). We argue that the separation of the control channel is an important feature of an SDN solution for IoT, for the following reasons: (i) the control channel associates with different communication requirements compared to the data channel (e.g., in the 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.

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 [84] supports two radio interfaces [85]: (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. Although a number of IoT motes already support double radio interfaces like the aforementioned Zolertia RE-Mote devices, the utilization of the additional radio interface is not straightforward. Such exercise requires systemic adaptations, spanning from the protocol level to the involved SDN controller and its mechanisms.

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

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 the 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 [7]; 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 [64] and [65]. However, as we mention in Chapter 1, this aspect is not part of this dissertation context and deserves an independent study.

Although adopting a separate control channel is not a breakthrough, we are the first to suggest and develop a solution that utilizes it in the context of SDN in WSN. As a bottom line, we consider our secondary-channel approach as a viable option to overcome the existing shortcomings of overhead and reliability issues in the control channel of SDWSN solutions, which is currently an open problem. In Chapter 5, through a series of simulations, we realize the significant benefits that this design choice brings in terms of efficient resource allocation, communication reliability, and IoT application performance.

In the next section, we describe the design and implementation of *VERO-SDN* our SDN solution for WSN that utilize a separate out-of-band control channel to communicate the SDN OpenFlow-like control messages.

3.2 VERO-SDN an IoT solution that applies out-of-band control

Along these lines, we describe the operational components and features of *VERO-SDN* framework. The solution aims to improve the communication performance of a wide range of IoT applications while reducing the SDN control overhead utilizing an approach of a separate control channel.

The *VERO-SDN* framework is an evolution and advancement of the *CORAL-SDN* framework [31]. Both frameworks were developed as part of this research endeavor; nevertheless, *CORAL-SDN* was the first to set the foundations for the development of our SDN controller and the grounds to evaluate the early implementations of the OpenFlow-like network control mechanisms, i.e., topology discovery and routing control algorithms. Since all of *CORAL-SDN* mechanisms and functionalities are inherited, improved, and exhaustively evaluated from *VERO-SDN*, we will concentrate our description on the latter. For clarity, detailed descriptions and information about *CORAL-SDN* can be found in published papers [31] and [29] as well as in Github repository [86] along with deployment documentation and related videos.

In detail, *VERO-SDN* natively supports: (i) a separate 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 Chapter 5), in terms of packet delivery ratio, network overhead, and end-to-end delivery time.

Fig. 2 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 $2.4GHz$

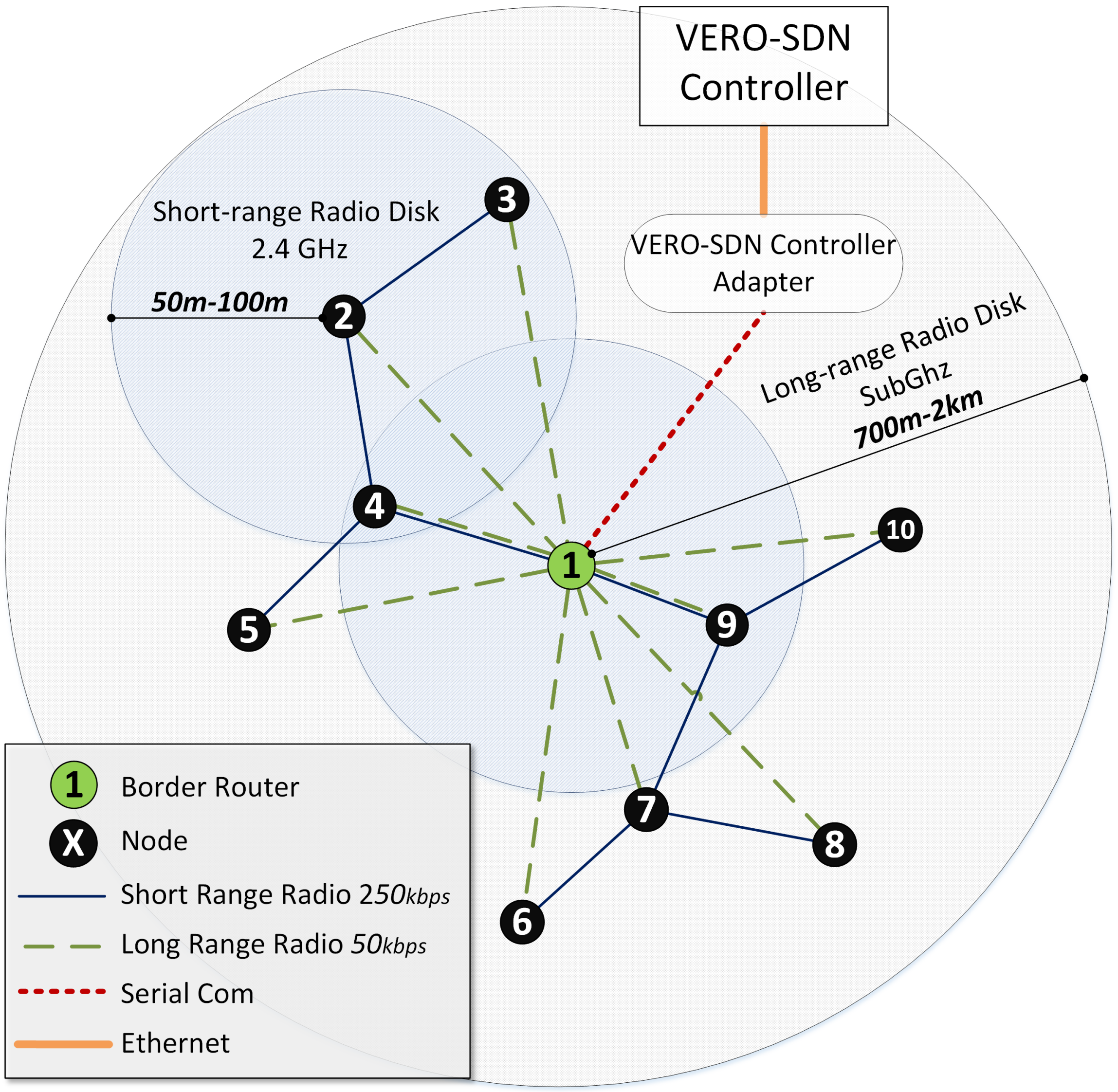

VERO-SDN Controller
VERO-SDN Controller Adapter
Short-range Radio Disk 2.4 GHz
50m-100m
Long-range Radio Disk SubGhz
700m-2km
1
2
3
4
5
6
7
8
9
10
Border Router
Node
Short Range Radio 250kbps
Long Range Radio 50kbps
Serial Com
Ethernet

Figure 2: *VERO-SDN* operational framework schema where each node espouses a dual-radio network interface, for long- and short-range communication.

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 as part of the protocol's scalability features in Section 3.7 at the end of this chapter.

- 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 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.

3.3 VERO-SDN Architecture

In this section, as in Fig. 3, we elaborate on a high-level representation of the *VERO-SDN* solution that consists of three planes, aligned to the typical SDN architecture [87]: (i) the *Application plane* providing high-level network management, monitoring, and the IoT applications; (ii) the *Control plane* manipulating 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*.

In the following subsections, we elaborate on the three *VERO-SDN* planes and their relevant functionalities.

3.3.1 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 [88], 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 operation, we developed our dashboard and visualization facility, the *VERO-SDN Dashboard*.

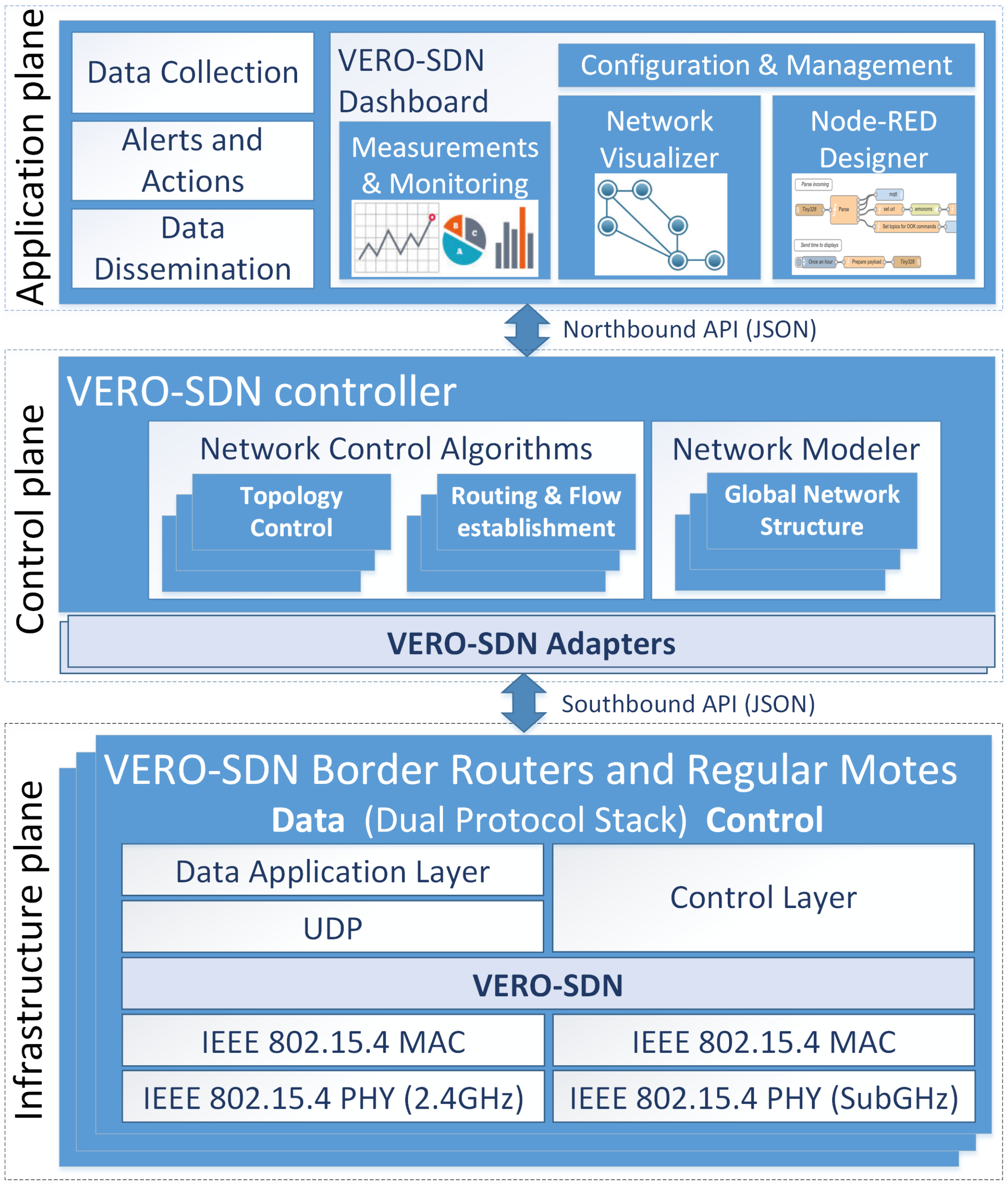
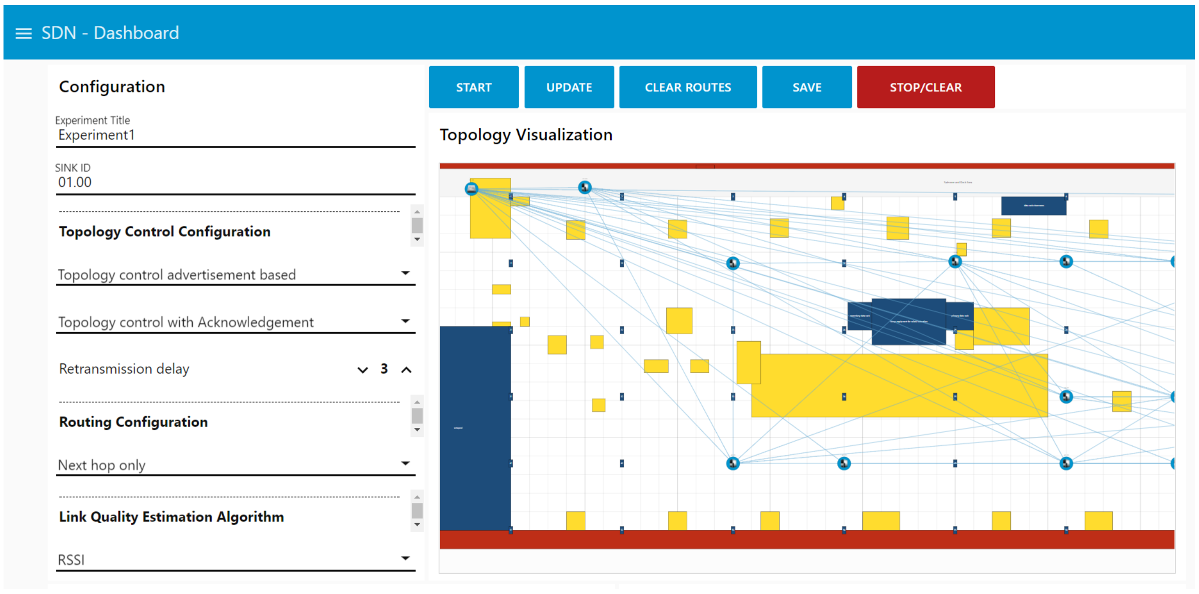


Figure 3: *VERO-SDN* architecture

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. 4. The *Dashboard* is implemented with the *Node-RED* framework [89], 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.



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 4: *VERO-SDN* dashboard GUI; left-hand-side, configuration parameters; right-hand-side, network topology graphical representation

3.3.2 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., [90]) based on the Java programming language. Its modular design easily accommodates new modules or algorithms, facilitating new functionalities.

3.3.3 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 [59], 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 [91]. 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 [92], 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 $2.4GHz$; 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 $868MHz$; and (ii) the *VERO-SDN* control layer that manipulates the control messages and operates the control processes.

In the next sections, we give a detailed description of the *VERO-SDN* protocol mechanisms and communication methods.

3.4 Protocol 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.

3.4.1 Northbound API

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

- The *Configuration Messages* that are outbound 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 outbound 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 [93] with the *Feature Extractor* approach for the link quality estimation and prediction proposed in [38]. 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 [36]. A further discussion on this matter is out of this dissertation's scope.

3.4.2 Southbound API

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 3, 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.

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 3, the maximum payload size in the protocol is in the *Neighbor Response* message with 27 bytes. Further details on the *southbound API* messages can be found in Appendix B.2.

Table 3: *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

3.5 Network Topology Control Mechanisms

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 [29] 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.

3.5.1 Topology Discovery using Node's Advertisement Flooding (TC-NA)

In Fig. 5, 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

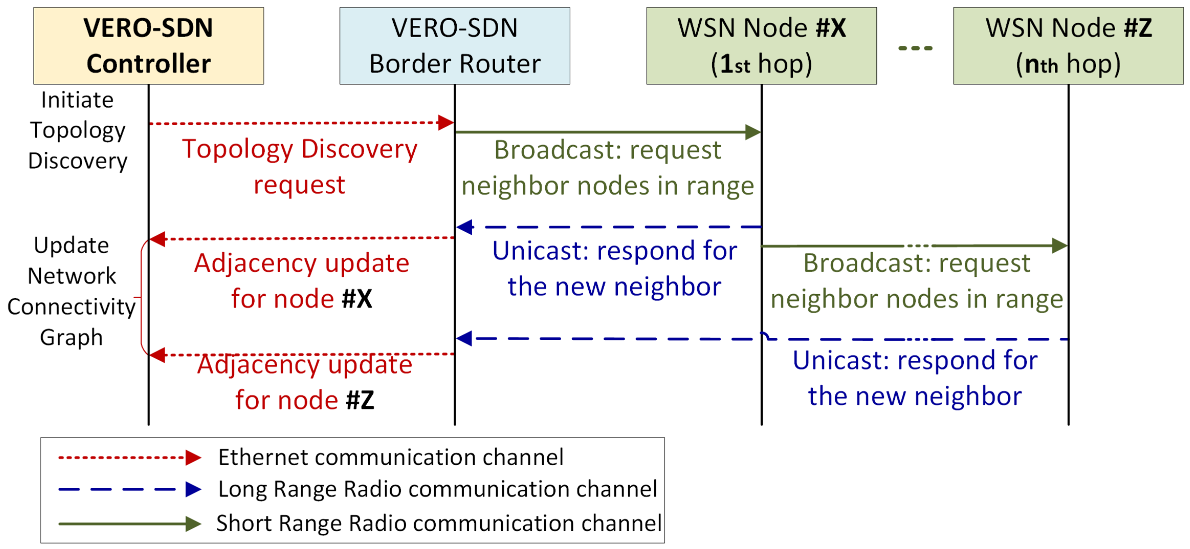

VERO-SDN Controller
VERO-SDN Border Router
WSN Node #X (1st hop)
WSN Node #Z (nth hop)
Initiate Topology Discovery
Topology Discovery request
Broadcast: request neighbor nodes in range
Update Network Connectivity Graph
Adjacency update for node #X
Unicast: respond for the new neighbor
Broadcast: request 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 5: Network topology discovery based on the Node's Advertisement Flooding algorithm (TC-NA)

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 *Con-*

troller'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.

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

```
1  tdID ← 0; // Initialize topology-discovery-run ID
2  ndi ← 0; // Initialize traffic message counter (global)
3  while true do
4      pktR ← receive();
5      if pktR.comm is Broadcast and pktR.radio is ShortR then
6          if pktR.type is "ND" then // ND=Neighbors' Discovery
7              ndi ← ndi + 1; // Increase traffic message counter
8              if tdID not equal pktR.tdID then // New topol. discovery
9                  tdID ← pktR.tdID; // Update topology-discovery-run ID
10                 ndi ← 1; // Restart traffic message counter
                   // Retransmit a Neighbors' Discovery beacon message
11                 pktB.type ← "ND";
12                 pktB.sender ← this.nodeAddress;
13                 pktB.BRaddress ← pktR.BRaddress;
14                 pktB.tdID ← pktR.tdID;
15                 pktB.retDelay ← pktR.retDelay;
16                 broadcast(pktB, ShortR, pktR.maxD);
17             end
               // Respond to BR New Neighbor message
18             pktS.type ← "NB"; // NB=New Neighbor
19             pktS.sender ← this.nodeAddress;
20             pktS.receiver ← pktR.BRaddress;
21             pktS.data ← pktR.rssi&pktR.LQI&this.nodeEnergy;
22             unicast(pktS, LongR, pktR.maxD);
23         end
24     end
25 end
26 Thread unicast(pktS, radio, maxD)
27     sleep(rand(maxD));
28     $send_{unicast}(pktS, radio)$;
29 Thread broadcast(pktB, radio, maxD)
30     sleep(rand(maxD));
31     if ndi <= maxT then // maxT=Maximum Traffic Constant
32         $send_{broadcast}(pktB, radio)$;
33     end
```

The *TC-NA* operation has similarities to the RPL's topology discovery algorithm [15]. 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 [94]. 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 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 backward 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 Chapter 5, we provide simulations highlighting the improved performance of *TC-NA* compared to RPL, with different network topology scenarios.

3.5.2 Topology Discovery with Node's Neighbors Requests solicited from the Controller (TC-NR)

In Fig. 6, 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

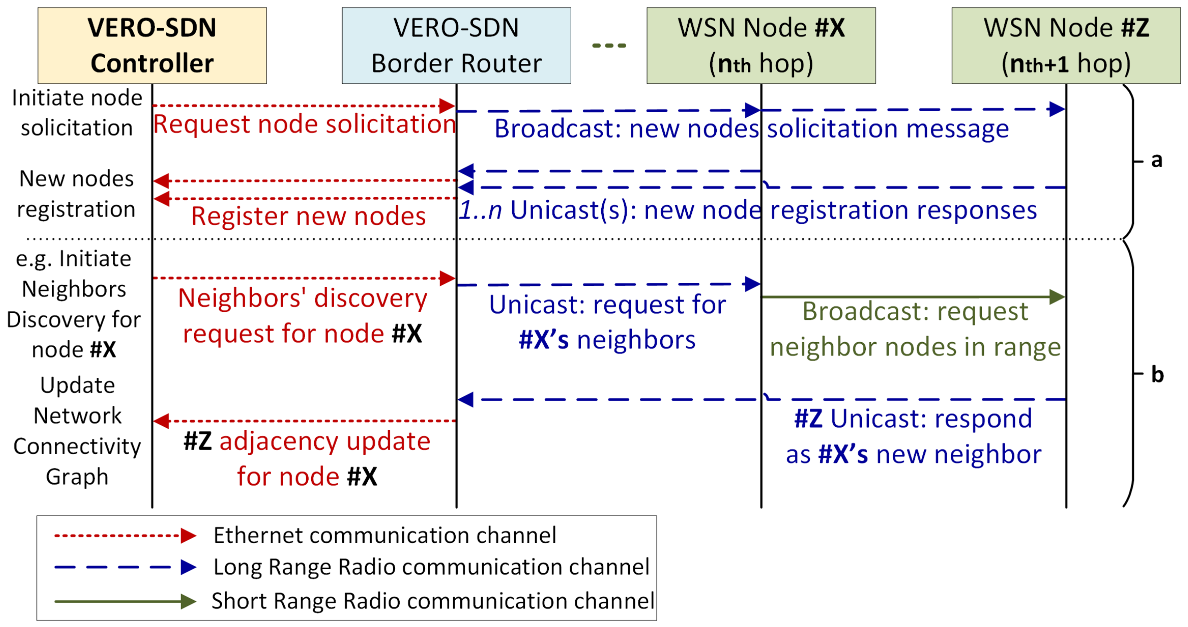

VERO-SDN Controller
VERO-SDN Border Router
WSN Node #X (nth hop)
WSN Node #Z (nth+1 hop)
Initiate node solicitation
Request node solicitation
Broadcast: new nodes solicitation message
New nodes registration
Register new nodes
1..n Unicast(s): new node registration responses
a
e.g. Initiate Neighbors 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 6: Network topology discovery with Node's Neighbors Requests solicited from the Controller (TC-NR)

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 *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.

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

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 Chapter 5, 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 described in Chapter 4.

3.5.3 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

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));
    $send_{broadcast}(pktB, radio)$;
```

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* [15]. The trickle timer, in order to reduce the control 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 [47].

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 5.1, 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* can utilize 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 can regulate the number of control messages needed to achieve a vivid picture of the network's topology, especially in networks where topology changes dynamically. More details of this hybrid strategy that avoid overloading the network with control data will be discussed in the next chapter in Section 4.1.2 as part of the polices used to mitigate topology changes of mobile networks.

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 Chapter 5, 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.

We point out that *VERO-SDN* topology discovery and maintenance mechanisms provide 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. More details on that matter we describe in Section 4.1.3.

In the next section, we elaborate on *VERO-SDN* network control mechanisms related to routing and flow establishment.

3.6 Network Routing Mechanisms

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 3. 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 [95]). 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 offers 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 [96]) 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 3. The *Controller* selects the best path by making use of the Dijkstra algorithm [97] 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, i.e.,*next-hop only* (FE-NH) and *complete path* (FE-CP) flow rule establishment described in the next subsections.

3.6.1 Next-hop only flow rule establishment

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.

For example, in Fig. 7, we depict a simple network scenario with four nodes and a *BR* in three consecutive transmission cases, with the assumption that initially, all forwarding tables are empty:

1. Data message ($3 \rightarrow 5$): For a data packet transmission request from node 3 to 5, node 3 reports to the *Controller* with a *missing-route* request, and receives back an *add-route* response. The response is recorded into the forwarding table and the data packet is forwarded to the next node 4. Since node 4 has no routing information, the previous process is repeated and the data packet is transmitted successfully to node 5.
2. Data message ($2 \rightarrow 5$): In this case, the forwarding tables of nodes 3 and 4 already know how to handle a message to node 5. As a result, the *missing-route* request is issued only from node 2.

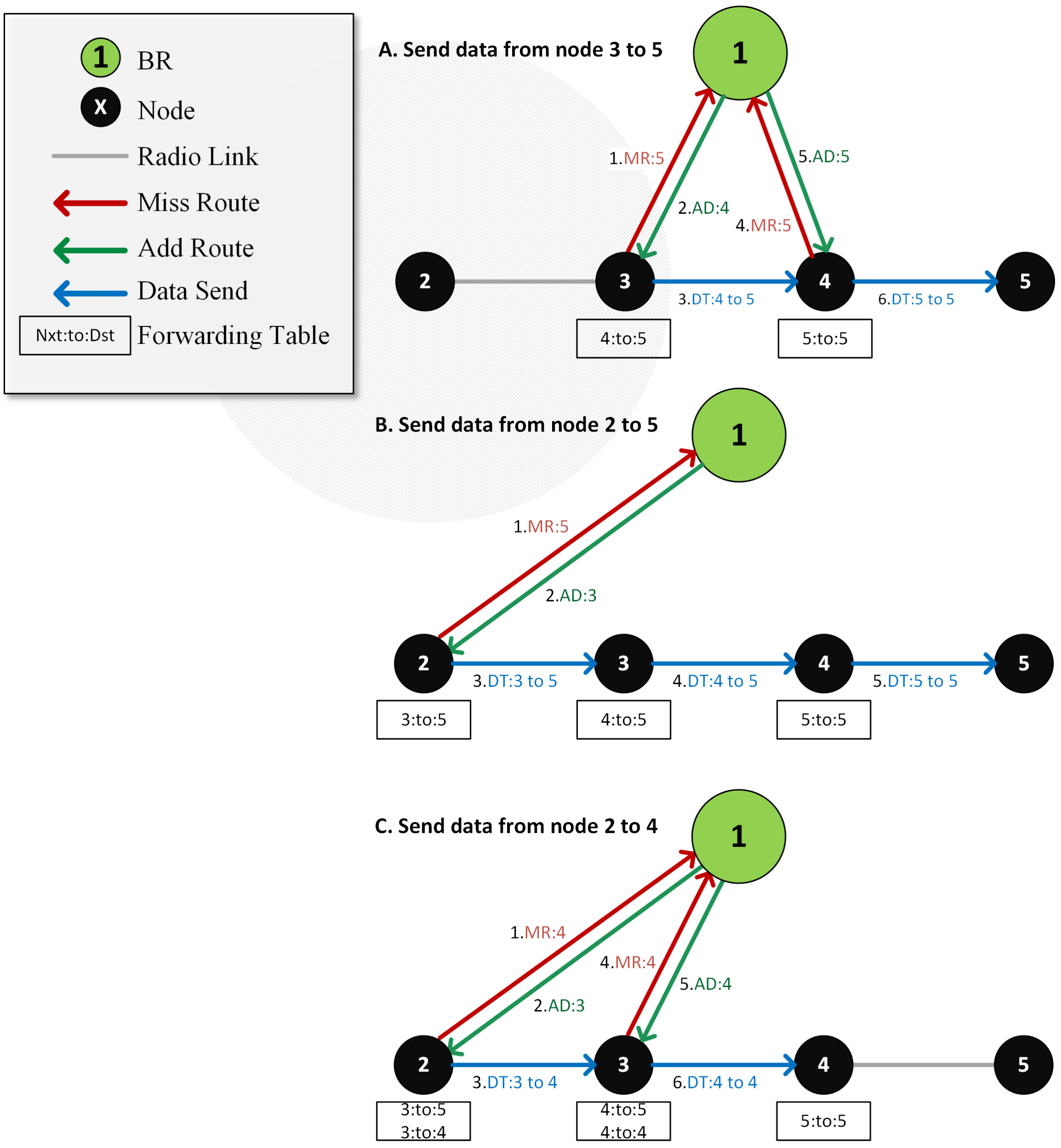


Figure 7: Three examples of *VERO-SDN* Next-hop only flow rule establishment (FE-NH) in steps

3. Data message (2 → 4): In the third case, we observe a request to a new destination, node 4. As in the first case, the nodes involved one by one issue *missing-route* requests and their forwarding tables are updated accordingly with new records.

The *FE-NH* method resembles the OpenFlow flow establishment process. Its main drawback is the high end-to-end delay, especially when the network is in the initial phase and all nodes issue missing route requests. However, as we see in the second example above, the problem is alleviated when the network has previously established forwarding routes.

3.6.2 Complete Path flow rule establishment method

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.

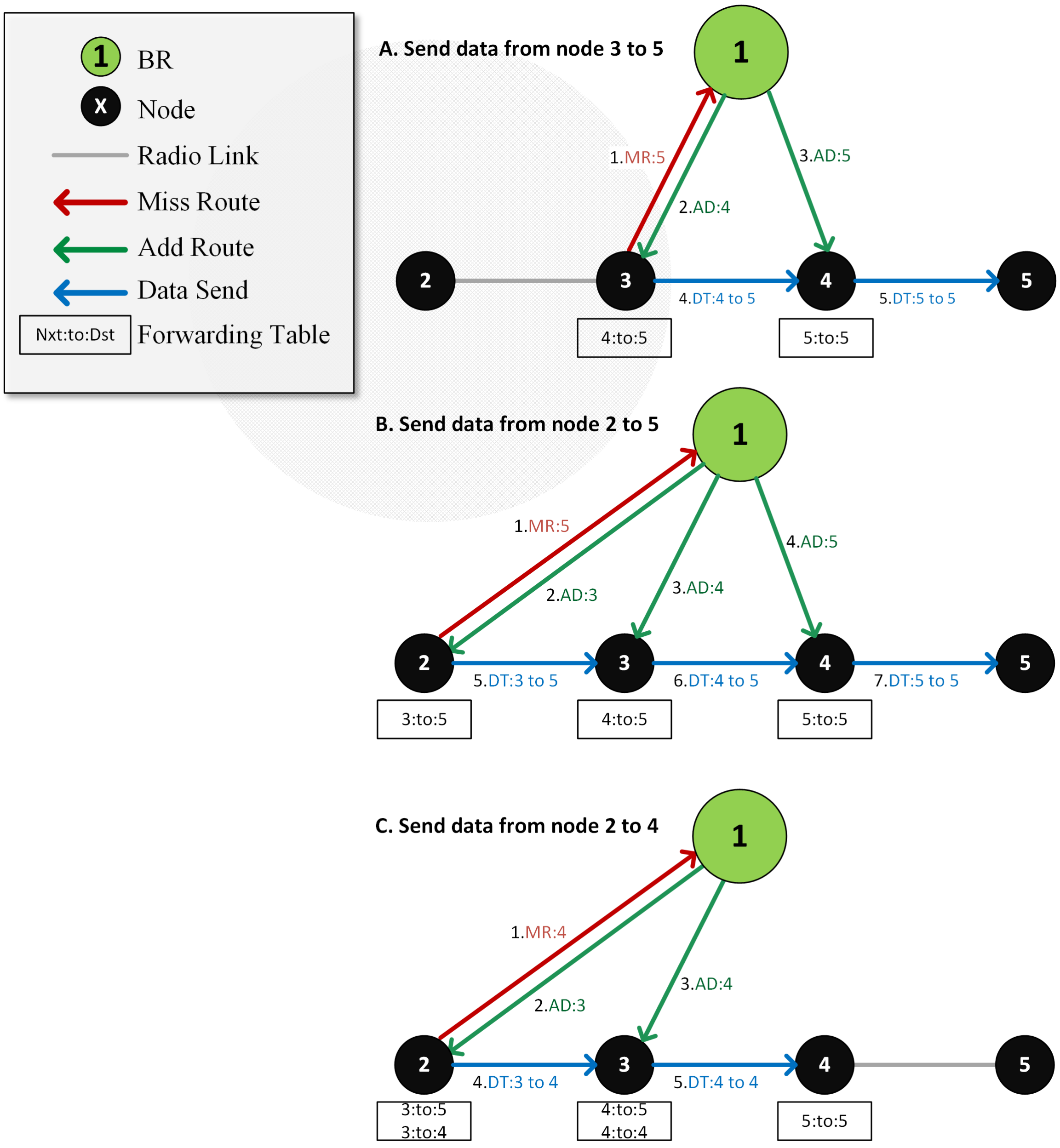


Figure 8: Three examples of *VERO-SDN* Next-hop only flow rule establishment (FE-CP) in steps

For example, for the same network with the previous subsection in Fig. 8, we observe that in the first transmission $3 \rightarrow 5$, the first *missing-route* request from node 3 results *add-route* responses to all of the nodes that participate in the path, i.e., nodes 3 and 4. In this case, the network transmits one less *missing route* control message than the *FE-NH* method. However, in the second case $2 \rightarrow 5$, we observe that *FE-CP* results two additional *add-route* responses to nodes 3 and 4 that is not required as both nodes' forwarding tables are already updated.

With *FE-CP*, the *Controller* maintains absolute control over the entire route, with the drawback of sending more control packets. This method is crucial for network installations that require traffic prioritization communication paths. 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.

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* ability to establish efficient routing paths in every direction, considering the WSN specificities, confirms that *VERO-SDN* is capable of operating in various applications utilizing any communication model (i.e., many-to-one, one-to-one, or one-to-many).

3.7 VERO-SDN Scalability extensions

IoT applications typically require large-scale deployments in terms of device numbers and geolocation. However, this is a major challenge for centralized solutions like SDWSNs. On the one hand, the distance of the *Controller* from the nodes affects the response time of control messages, and on the other hand, the number of devices that the *Controller* can concurrently support sets a ceiling to the network size.

To overcome these challenges, we propose the deployment of multiple *BRs*, which are acting as a neighborhood provosts and balance the control messages load by splitting the network into smaller groups of nodes. This architectural choice naturally fits *VERO-SDN* framework, due to the fact that this solution also extends the network deployments beyond the maximum communication range of the long-radio channel.

Fig. 9 depicts the extended multi-BR *VERO-SDN* framework that illustrates two *BRs*, the green node 1 and the red node 2 serving the network nodes in each

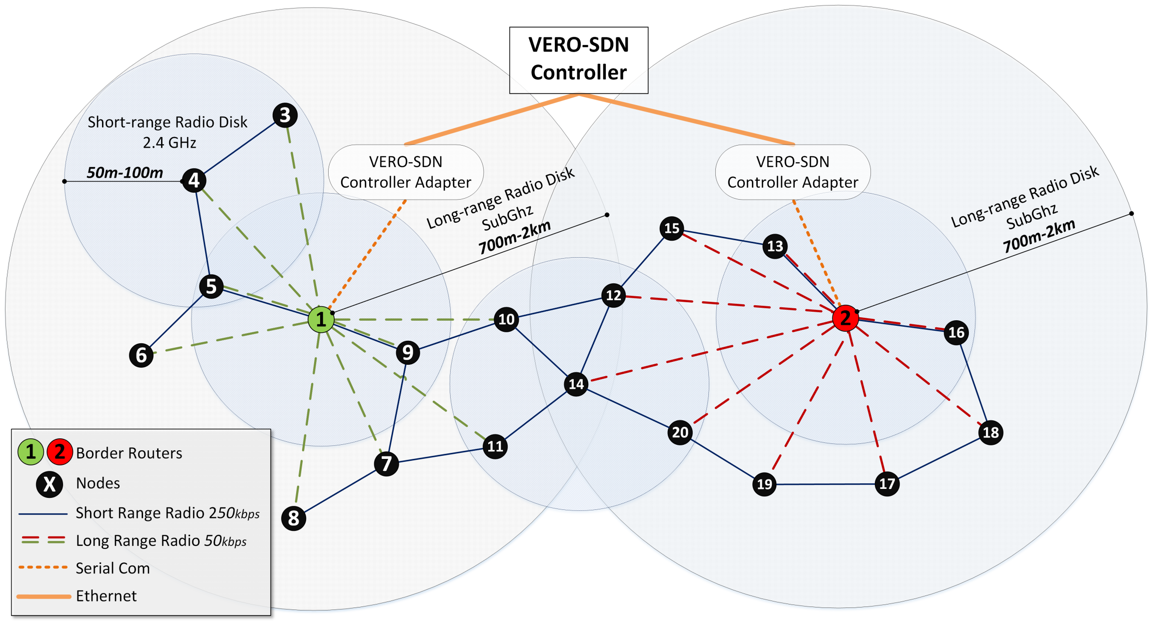

VERO-SDN Controller
VERO-SDN Controller Adapter
VERO-SDN Controller Adapter
Short-range Radio Disk 2.4 GHz
50m-100m
Long-range Radio Disk SubGhz 700m-2km
Long-range Radio Disk SubGhz 700m-2km
Border Routers
Nodes
Short Range Radio 250kbps
Long Range Radio 50kbps
Serial Com
Ethernet

Figure 9: Multiple Border Routers Framework

control area. In detail, we enhance the *VERO-SDN Controller* to support many *BR* nodes connected through low latency infrastructure lines, i.e., fiber-optics or fast Ethernet connections, to avoid delays in the propagation time of control messages. For sites that require wireless communication due to landscape or infrastructure reasons, wireless connections such as *5G* can be used instead. The positions of the *BRs* require planning from the administrator to achieve maximum coverage with a minimum number of *BRs*. Moreover, we enhance each *southbound API* control message by including a *BR* identification number. This way, each node registers to one *BR*, and each *BR* is responsible for the nodes in its long-range radio coverage area. In areas where more than one long-range radio signal of *BRs* overlaps, the nodes register to the *BR* node that reaches them first during the topology discovery process.

The above enhancements allow *VERO-SDN* to be deployed in large-scale geographical areas maintaining robust communication for the control messages. For example, considering that each *BR* supports 100 nodes (e.g., in Chapter 5 we conduct successfully experiments with 90 nodes on one *BR*) with 10 BR we can support networks with 1000 nodes. Theoretically, the protocol is designed to support 100 *BRs*, but such large-scale deployments are considered outside of the dissertation goals. Concluding, very-large-scale deployments of SDWSNs is an open research area, and we suggest as future *VERO-SDN* extensions, the formation of hierarchical

BR structures, or the operation of distributed *Controllers*.

This chapter has presented our proposed mechanisms and methods that evolve WSNs towards the IoT environments. We elaborated on the architectural decisions that amalgamate the SDN paradigm with Low power and Lossy WSNs using a separate long-range wireless channel enabling one-hop communication with the *SDN Controller*. The discussed algorithms and methods are implemented under an SDWSN framework *VERO-SDN*, while the resulted improvements en route to the initial challenges defined in Chapter 1 will be discussed in Chapter 5. In the next chapter, we demonstrate architectural enhancements and mechanisms that advance the *VERO-SDN* solution towards mobile and heterogeneous network environments.

4 Evolving SDWSNs in Mobile and Heterogeneous environments

Among the challenges we described in Chapter 1, *mobility* is prominent as a *sine qua non* for the emerging *Mobile Internet of Things* (MIoT) applications, including monitoring and tracking systems for a plethora of everyday human activities, including sports, entertainment, and healthcare [98], [99]. Such scenarios are characterized by constant changes in their physical topology, constituting challenging environments for the network routing protocols.

Moreover, *heterogeneity*, a common characteristic among IoT applications, undermine the deployment of WSNs, especially on a large scale, such as smart-city applications [100]. It is mainly caused by the diversity of hardware devices, the lack of turnkey solutions and the absence of standardization. To handle heterogeneity in an SDN context requires carefully designed abstractions to hide heterogeneity and allow devices to export common features to the higher control and application planes.

In this chapter, we propose two functional frameworks and their architectural planes, followed by a detailed description of the proposed mechanisms that realize the SDWSN platforms that aim at providing elasticity for heterogeneous and/or mobile IoT deployments. More precisely, we propose:

- *SD-MIoT* [34] an SDWSN solution that improves network operation and performance under mobility, described in Section 4.1, and
- *MINOS* [35] a Multi-protocol SDN platform for IoT that adapts the IoT network to the application requirements by deploying on-demand the appropriate network protocol while considering the constraints and required abstractions for elastic network management and optimized routing over heterogeneous IoT nodes, presented in Section 4.2.

4.1 SD-MIoT: a Mobility aware SDWSN for IoT

WSN protocols, such as RPL [15], are mainly focusing on data collection applications with static network nodes. Consequently, they appear inadequate for IoT and their emerging applications, characterized by a diverse range of communication patterns and mobility. As we commented in Chapter 2, solutions that attempt to improve RPL's efficiency in mobile environments are impaired by its original architecture. Additionally, in our knowledge, none of the available SDWSN solutions consider WSN scenarios with mobility characteristics.

Along these lines, we propose the *SD-MIoT* framework (Fig. 10) aiming to address the challenges that mobility brings to WSNs, including instability, control overhead and performance issues. Aligned to the SDN paradigm, *SD-MIoT* decouples control complexity from the network protocol and offloads it to a modular *SDN Controller* deployed at the surrounding fixed infrastructure. The *SD-MIoT Controller* implements programmable topology and routing control striving for improved QoS through robust packet delivery as well as reduced control overhead for WSNs with mobile nodes. At the infrastructure layer, it operates an OpenFlow-like protocol improving the quality of communication through supporting alternative topology discovery and flow establishment mechanisms adapted to the needs of a wide range of mobile WSN scenarios.

SD-MIoT is an open-source software [101] designed and implemented utilizing our earlier experience in SDWSN protocol design and development, gained from *VERO-SDN* framework described in the previous chapter. In particular, *SD-MIoT* inherits from *VERO-SDN* the novel SDN centralized architectural characteristics, including out-of-band control communication and its infrastructure plane mechanisms. Consequently, *SD-MIoT* builds up on the significant advantages of VERO-SDN in terms of reduced control overhead, robust operation, and support of alternative IoT application requirements. It suggests new innovative ideas and solutions that contribute to the transformation of traditional WSNs to solid infrastructure mediums for MIoT applications, including the *Mobility Control Component (MCC)* and the *MObility DEtection (MODE)* algorithm, briefly described below.

MCC is an *SD-MIoT Controller* module aiming for better support of network scenarios with mobile nodes. It adopts hybrid strategies for the fixed and mobile nodes, including: (i) diverse topology discovery algorithms to the mobile and static parts of the network, based on an up-to-date representation of the network, achieving reduced control overhead; (ii) routing prioritization policies considering the nodes' mobility status for the establishment of robust data packet forwarding

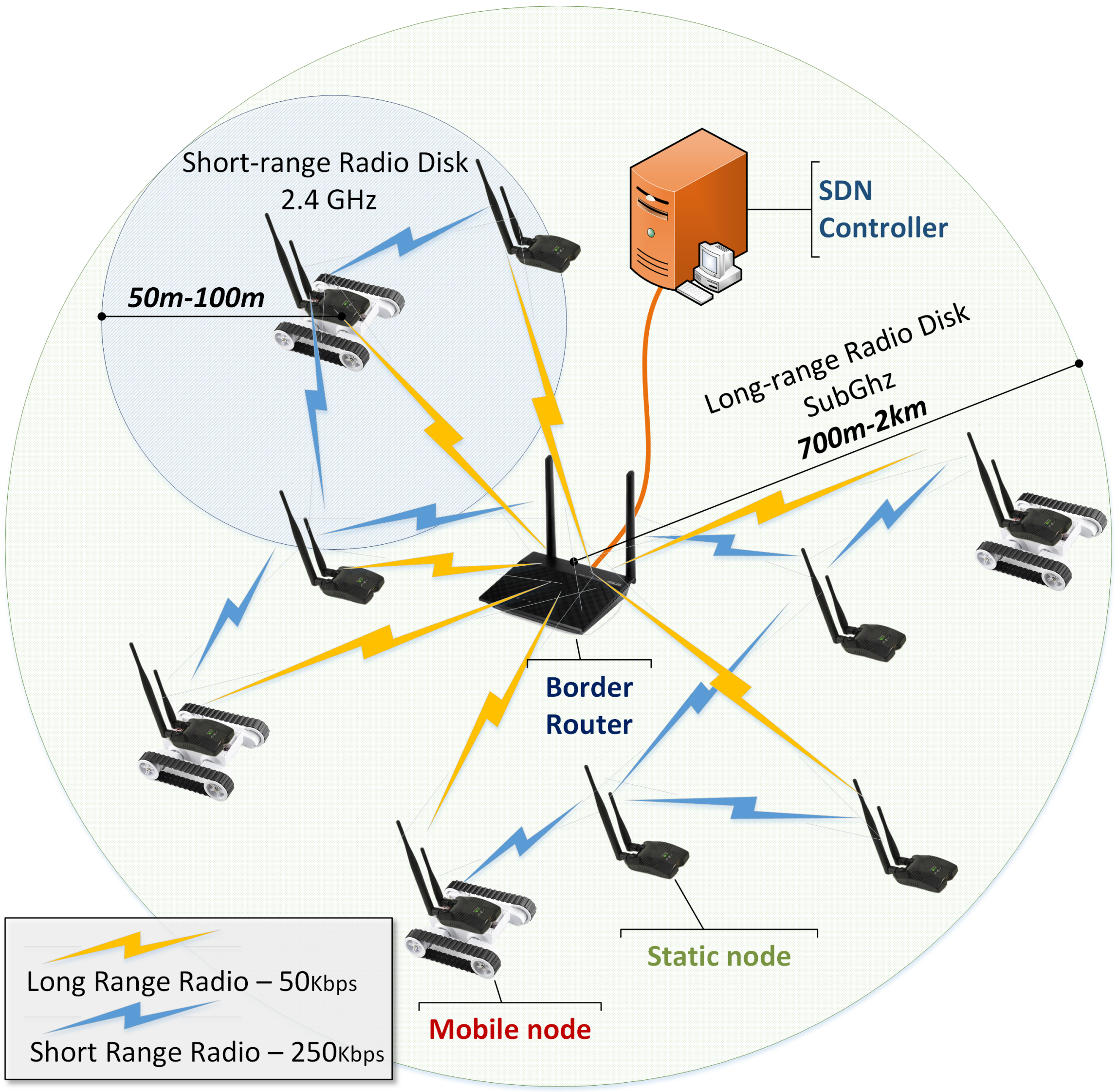


Figure 10: SD-MIoT operational framework schema

paths; and (iii) dynamically blending of reactive and proactive routing flow establishment techniques to maintain flow-rule establishment timeliness, especially for the mobile motes that avoid packet-loss and routing loops, elevating Packet Delivery Ratio (PDR) and QoS.

The MODE algorithm applies data analysis and classification methods (i.e., the K-means algorithm [102]) on adjacent matrices generated from the network's connectivity graph to passively detect which nodes are moving and which are static, at any given time. It operates at the application plane; as such, its outcome, apart from an essential input for our *MCC* mechanisms, can be further used from third-party IoT applications or other SDN controllers. To our knowledge, *MODE* is the first attempt towards a smart decision-making algorithm that passively detects the network's mobility behavior, based on an SDN framework.

In the following subsections, we describe in detail the *SD-MIoT* architecture, mechanisms, and operation.

4.1.1 SD-MIoT Architecture

Fig. 11 depicts a high-level view of *SD-MIoT* architectural structure with its basic components and interfaces, marking as yellow its novel mobility handling functionalities. Aligned to the typical three-tier SDN paradigm [87], it consists of the following planes, described bottom-up as follows:

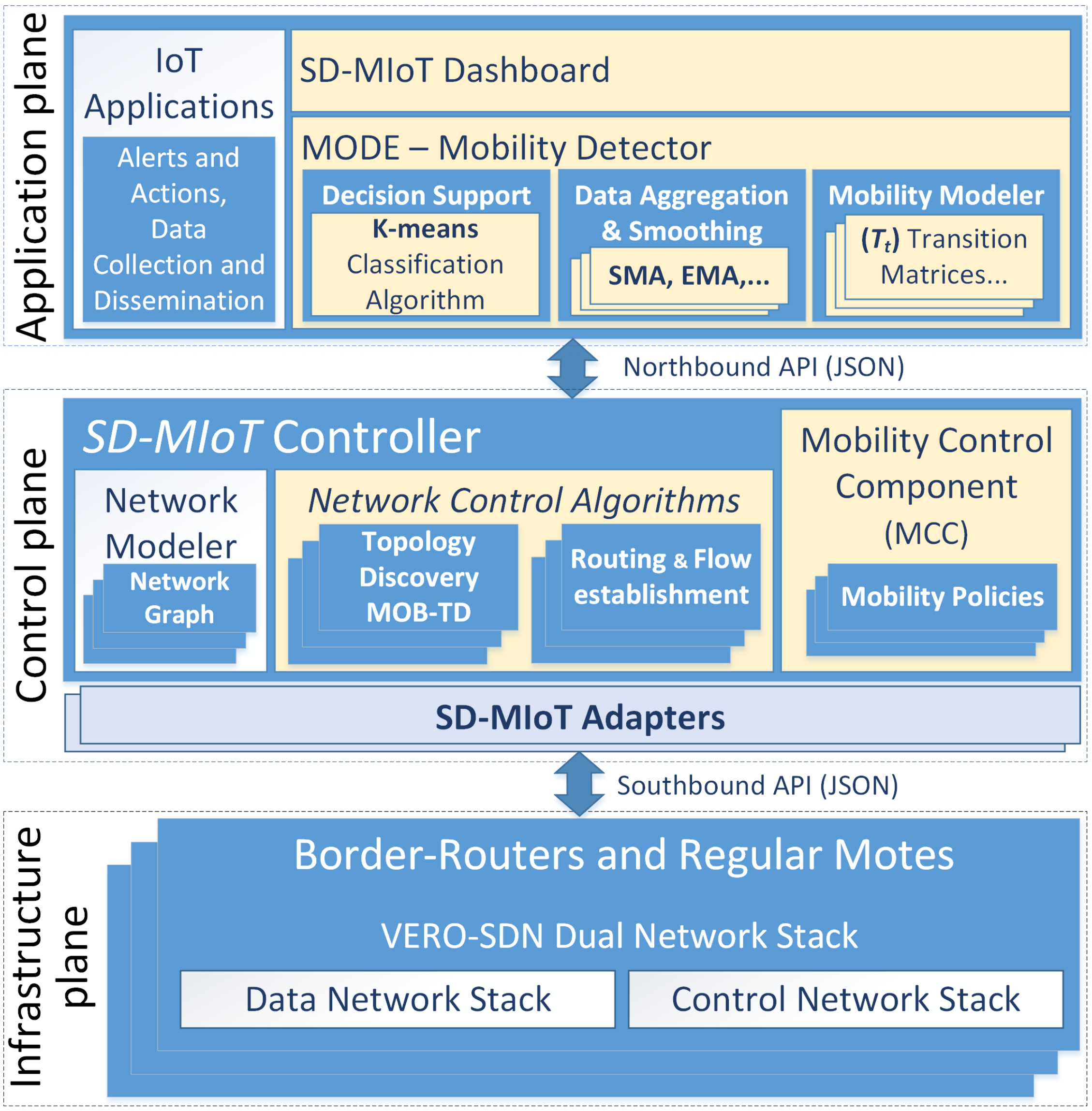

Application plane
IoT Applications
Alerts and Actions, Data Collection and Dissemination
SD-MIoT Dashboard
MODE – Mobility Detector
Decision Support
K-means Classification Algorithm
Data Aggregation & Smoothing
SMA, EMA,...
Mobility Modeler
(T_t) Transition Matrices...
Northbound API (JSON)
Control plane
SD-MIoT Controller
Network Modeler
Network Graph
Network Control Algorithms
Topology Discovery MOB-TD
Routing & Flow establishment
Mobility Control Component (MCC)
Mobility Policies
SD-MIoT Adapters
Southbound API (JSON)
Infrastructure plane
Border-Routers and Regular Motes
VERO-SDN Dual Network Stack
Data Network Stack
Control Network Stack

Figure 11: *SD-MIoT* architecture

1. *Infrastructure plane* is comprised of regular and border-router IoT motes. *SD-MIoT* utilizes out-of-band radio network control through the *VERO-SDN's*

[32] dual network stacks for *control* and *data communication* channels, offering standardized low-power wireless communication and media access control through the IEEE 802.15.4 Physical and Media Access Layers. The network layer operates four alternative protocol mechanisms: (i) two network topology discovery algorithms, the *Node's Advertisement Flooding (TC-NA)* algorithm, that implements a global network discovery based on a node-to-node broadcast advertisement process, and the *Node's Neighbors Requests solicited from the Controller (TC-NR)* algorithm, that acquires connectivity information through targeted requests from the SDN controller to specific nodes or network areas; and (ii) two types of flow-rule establishment processes able to either configure only the *Next Hop (FE-NH)* of the node's forwarding table or the *Complete end-to-end Path (FE-CP)*, configuring the forwarding tables of all nodes participating in the path. We note that *SD-MIoT* infrastructure plane is inherited from the *VERO-SDN* protocol, described in detail in Chapter 3, i.e., *SD-MIoT* builds up on *VERO-SDN's* robust and with reduced overhead network control capabilities. The latter is utilized from novel mobility handling mechanisms. The firmware of IoT devices is implemented in C programming language for Contiki-OS 3.0 [59], either operating in real test-beds or the Cooja emulator.

2. *Control plane* maintains the *SD-MIoT Controller* that manipulates the abstracted anatomy of the infrastructure network and utilizes sophisticated *Network Control Algorithms*, acclimatized to mobile WSN environments. In particular, it handles the following tasks: (i) maintains an abstract view of the network connectivity, the *Network Graph*, through the *Network Modeler* module using a hybrid topology discovery process adapted to mobile topologies, using two kinds of topology discovery algorithms (i.e., local and global); (ii) performs network routing and flow control decisions utilizing proactive and reactive flow establishment methods; (iii) adjusts the data-plane protocol parameters dynamically; and (iv) incorporates a variety of cross-layer connectivity information, like RSSI and LQI. The Controller's adaptation within the mobility context is orchestrated from the *MCC* module, described in detail in the next subsection. The *SD-MIoT Controller* is implemented in

Java, and it is designed in a modular, scalable approach, i.e., also supporting multiple *BR* motes that accommodate new algorithms and intelligent functionalities in a straightforward manner.

3. *Application plane* provides IoT applications (i.e., Data Collection, Alerts and Actions, or Data Dissemination [88]), high-level network management and monitoring (i.e., the *SD-MIoT Dashboard*, a highly flexible GUI), and the intelligent *MODE* module passively detecting, in real-time, the moving network nodes, discussed in Section 4.1.3.

The *SD-MIoT planes* communicate through the northbound and southbound API using JSON messages, which detailed technical specifications are provided in [101].

In the next two subsections, we describe each of the *MCC* and *MODE* mobility handling mechanisms separately, due to their significance in the operation of *SD-MIoT*.

4.1.2 MCC: Mobility Control Component

Here, we introduce the *Mobility Control Component (MCC)* that augments SDN Controller with mobility-aware features, i.e., to address the stability and performance issues due to physical topology changes. *MCC* module applies protocol control policies that differentiate between fixed and mobile nodes, dynamically adapting *SD-MIoT* topology discovery and flow establishment processes to particular network conditions. In detail, *MCC* applies three policies, i.e., *Mobility Topology Discovery*, *Mobility Routing Prioritization*, and *Mobility Flow-rules Establishment*, detailed below.

4.1.2.1 MOB-TC: Hybrid Dynamic Topology Discovery in Mobile Environments

This policy aims at a mobility-friendly topology monitoring process that maintains a vivid and up-to-date representation of the network's connectivity, notwithstanding the dynamic changes imposed by the moving nodes. The *Mobility Topology Discovery* policy is implemented from the *Mobility Topology Discovery (MOB-TD)* algorithm. The main concept of this algorithm is the early topology change detection using frequent topology maintenance requests to mobile nodes only, avoiding overloading the rest of the network with control messages.

In Algorithm 3, we depict the main steps of *MOB-TD* operation. *MOB-TD* utilizes interchangeably the topology discovery algorithms mentioned in the previous chapter, i.e., *TC-NA* (Algorithm 1) and *TC-NR* (Algorithm 2). In detail, is carrying out infrequent global topology discoveries using *TC-NA*, i.e., for the static part of the network, and frequent local topology discoveries using *TC-NR*, i.e., for the mobile part only.

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 Dashboard GUI. The *TRt* parameter of *TC-NA* is configured considering the requirements of a fixed network.

In the interval between the global topology detection processes, *MOB-TD* executes *local* network topology-discovery-runs to realize the dynamic changes caused by mobility, using the *TC-NR* algorithm. In practical terms, the latter requests the adjacent nodes for the moving nodes only. This targeted approach 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. The interval of these 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. This hybrid strategy avoids the overloading of the network with unnecessary control data for the fixed areas of the network.

The prerequisite for the operation of *MOB-TD* is the awareness of mobile nodes. We comply with this need by introducing the *MODE* algorithm, i.e., detailed in Section 4.1.3.

Algorithm 3: *MOB-TD* – Mobility Topology Discovery

```
Input: TRt - global topology refresh time
Input: TTRt - local (mobile) topology refresh time
Output: G - Network Connectivity graph structure
while true do
    sleep(TTRt); // wait for TTRt seconds
    // retrieve list of mobile nodes from MODE (Algorithm:4)
    mobN ←MODE(G,5);
    if isExpired(TRt) then // global topology discovery
        G ←TC_NA(); // call TC-NA [29]
    else // local topology discovery for mobile nodes
        foreach element m in mobN do
            G ←TC_NR(m); // call TC-NR [29]
        end
    end
end
```

4.1.2.2 Routing Prioritization for Mobile Environments

SD-MIoT takes advantage of the SDN paradigm and its unique routing decision-making capabilities. Utilizing the *Controller's* computational resources, it is capable of analyzing the network connectivity graph with exhaustive search algorithms, e.g., Dijkstra, and a multitude of weighted-parameters (i.e., RSSI, LQI, or node's energy). As such, we enhance *SD-MIoT* with a routing prioritization policy that considers mobility as an important routing criterion, i.e., to make efficient routing decisions in mobile IoT contexts. In detail, it gives routing priority to the fixed nodes over mobile nodes, taking into account the node type (i.e., fixed or mobile). As a result, it gives a low priority to data-message forwarding through mobile nodes, mitigating the adverse effects that mobile nodes can cause to the data packet delivery, as shown in the next chapter. This policy favors network installations where the majority of nodes are fixed, like the two use-case scenarios described in Section 5.3.1. In case there are no fixed points available in the node's range, the *Controller* decides based on the next available rule (e.g., RSSI). Since this aspect deserves further analysis, it is part of our future plans described in Chapter 6.

4.1.2.3 Reactive and Proactive Flow Establishment

To further align *SD-MIoT* operation to mobile network environments, *MCC* proposes enhanced *flow-rule establishment processes*, manipulating the *forwarding table* of the nodes. In a nutshell, it implements a routing policy that employs a combination of *reactive* and *proactive* flow establishment mechanisms.

In *Reactive* routing path establishment protocols, e.g., OpenFlow [22] or Lightweight On-demand Ad-hoc Distance-vector routing protocol LOADng [95], a next-hop decision process is activated (e.g., a miss-route message to an SDN controller and a relevant response) each time a node initiates a data packet transmission to an unknown destination. These protocols achieve point-to-point communication between all nodes, with the main drawback being the delay in establishing a rule for the first time. In static networks, this is not a significant disadvantage, since such rules do not frequently change. Nonetheless, they are not suitable for dynamic network environments, because outdated flow-rules in forwarding tables reduce the network PDR, while a frequent forwarding rule establishment increases end-to-end delay time and control messages overhead.

Proactive routing protocols, including RPL [15], build a tree of connected nodes using distance vectors (i.e., Destination Oriented Directed Acyclic Graphs – DODAGs), where nodes maintain paths through their parent node towards a root (or sink) node. Such protocols operate efficiently and achieve low end-to-end delays, when the data are always routed to known and predetermined destinations, while they fail to create optimal point-to-point paths between nodes, because of complexity issues. Moreover, in dynamic environments, although they can actively refresh their routing tables by adjusting the protocol's timers (e.g., the reverse trickle timer [69]), the protocol overhead is substantially increased.

The *Mobility Flow-rules Establishment* policy exploits the advantages of both methods, utilizing them interchangeably based on the nodes' mobility behavior. As such, it applies a *reactive* flow-rule establishment for fixed nodes, while mobile nodes update their forwarding table *proactively*, whenever there is a topology change. The result is a rigorous routing mechanism that avoids data packet loss and routing loops due to obsolete flow-rules, while maintaining a high PDR. We argue that such versatile mechanisms are enabled by the flexibility and centralized view that *SD-MIoT* brings in Mobile IoT environments.

In Chapter 5, we demonstrate through simulations the significant improvements of *MCC* policies, in terms of successful packet delivery and reduced amount of control messages. However, *MCC* operation heavily depends on the accurate identification of mobile nodes. In this respect, we present in the next section, a novel mechanism capable of identifying the moving nodes in a network.

4.1.3 MODE: Mobility Detection for Mobile Internet of Things

To support the mechanisms of *MCC* module, we design, implement, and evaluate a novel application plane component, the *MObility DEtector (MODE)*. *MODE* separates in real-time fixed from mobile nodes using cluster-analysis. Particularly, it applies the K-means data-mining vector-quantization method on a time-series collection of network adjacency matrices provided by the Controller's *Network Modeler* module through the northbound API (Fig. 11), i.e., as snapshots of the network's connectivity graph data-structure.

4.1.3.1 The Concept of Transition Matrices

To further elaborate on *MODE*, we consider the following simple network of four nodes (i.e., three fixed and one mobile), as shown in Fig. 12. A snapshot of the network's connectivity map at a given time t is represented by a graph $G_t(V,E)$, with V vertices representing the network nodes and E edges representing the radio links. The *Mobility Modeler* module of the *Controller* (Fig. 11) converts each $G_t(V,E)$ into a two dimensional symmetric $V \times V$ matrix, defined from the graph theory as the *Adjacent matrix* A_t of G. Each $A_{t(i,j)}$ element contains values of zero and one as in (4).

$$A_{t(i,j)} = \begin{cases} 1 \; \forall i,j \in V \;\; \textit{if } i,j \textit{ are connected (edge)} \\ 0 \; \textit{otherwise} \end{cases} \tag{4}$$

To designate the network connectivity changes, we monitor a series of the n most recent *Adjacent matrices* $\{A_{t-(n-1)},\ldots,A_{t-1},A_t\}$. To realize the amount of connectivity changes for each node, at a given time t, we calculate a set of two-dimensional $V \times V$ square matrices, the *Transition matrices* $\{T_{t-(n-2)},\ldots,T_{t-1},T_t\}$, as the subsequent subtractions in absolute values of the n *Adjacent matrices*, in pairs of consecutive times, as in Equation (5).

$$\begin{aligned} T_t &= \|A_t - A_{t-1}\| \\ T_{t-1} &= \|A_{t-1} - A_{t-2}\| \\ &\vdots \\ T_{t-(n-2)} &= \left\|A_{t-(n-2)} - A_{t-(n-1)}\right\| \end{aligned} \tag{5}$$

The *Transition matrices*, over time, serve as memory snapshots that depict all network topology changes and constitute the primary data input for the *MODE* decision-making process.

For example, as exemplified in Fig. 12, node 4 at time $t = 1$ is connected to node 2 and then moving from left to right at time $t = 2$, disconnecting from node 2 and connecting to node 3. The *Transition matrix* of this example, at time $t = 2$, is marked as T_2 and calculated in (6).

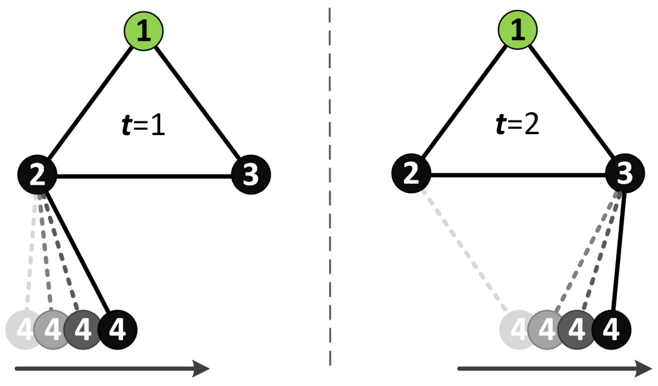


Figure 12: Simple mobility detection scenario example

$$T_2 = \left\| \overset{A_{t=2}}{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & \mathbf{0} \\ 1 & 1 & 0 & \mathbf{1} \\ 0 & \mathbf{0} & \mathbf{1} & 0 \end{bmatrix}} - \overset{A_{t=1}}{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & \mathbf{1} \\ 1 & 1 & 0 & \mathbf{0} \\ 0 & \mathbf{1} & \mathbf{0} & 0 \end{bmatrix}} \right\| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & 0 \end{bmatrix} \tag{6}$$

We are now analyzing matrix T_2 per row. Each row represents the connectivity changes of one node (i.e., first row for node 1, second row for node 2, etc.). In the first row, all elements have a zero value, indicating no connectivity changes for node 1, so we assume it is a fixed node. In the fourth row, two elements have the value one, indicating that node 4 changed its connectivity status twice. As a result, we hypothesize that node 4 is a mobile node. However, both nodes 2 and 3, i.e., in the second and the third row, have one connectivity change due to the mobility of node 4 and any attempt to decide for their mobility status is at stake.

To overcome the above false-positive identification challenge and enable accurate decisions about nodes' mobility, we enhance *MODE* with advanced processes implemented by *Data Aggregation-and-Smoothing* and *Decision Support* modules (Fig. 11), i.e., applying intelligent data analysis and data-mining classification methods in *Transition matrices'* data, respectively. We further describe these modules in detail in the next subsection.

4.1.3.2 Mobility Detection Algorithm

In Algorithm 4, we detail in a unified pseudocode the three modules (i.e., *Mobility Modeler*, *Data Aggregation-and-Smoothing*, and *Decision Support*) that *MODE* employs to detect the mobile nodes.

Initially, the *Mobility Modeler* (i.e., lines 1 to 7) converts the input connectivity graph G to its $N \times N$ Adjacency matrix J, i.e., N is the number of network nodes, and adds it into list A. Subsequently, it calculates the *Transition matrices* through subtracting the matrices in A, as in Equation (5), and inserts them into queue T.

The second part of algorithm (i.e., lines 8 to 12) implements the *Data Aggregation-and-Smoothing* process, which further refines our data using the *moving average* method, which apart from aggregation it is also used for normalizing data anomalies. From the variety of moving average methods, we select the most common, the *Simple Moving Average (SMA)*. *SMA* is implemented by adding a set of data and then dividing them by the number of observations n (aka *SMA* window), considering the same weight to each observation [103]. In our case, we apply *SMA* on the latest n observed *Transition matrices* in queue T, as in Equation (7).

$$T_{SMA(t)} = \frac{1}{n}\sum_{i=0}^{n-1} T_{(t-i)} \tag{7}$$

The $T_{SMA(i,j)}$ $\{i, j \in N\}$ elements of the $N \times N$ *Transition Matrix* T_{SMA} at a given time t, contain the values representing the number of connectivity changes between each pair of (i, j) nodes. The value of *SMA* window n is a prominent figure since it affects the sensitivity of our algorithm in terms of early or late mobility detection, as well as the number of false-positive decisions. For example, small n values increase the sensitivity in detecting topology changes earlier, as well as the chances of false positives, whereas large n values result in late detection of changes. For this reason, in our experimentation analysis in Section 5.4, we dedicate a further analysis that fine-tunes the operation of our algorithm.

To calculate the total mobility behaviour for each node (i.e., total connectivity changes between node x to all other nodes), we sum up the values of each row x in $T_{SMA(N \times N)}$ and insert the results in vector $sumT_{SMA(N)}$, as in Equation (8).

$$sumT_{SMA(x)} = \sum_{i=0}^{N-1} T_{SMA(x,i)} \tag{8}$$

The main *Decision Support* module algorithm, described in lines 13 to 20 of Algorithm 4, detects the mobile nodes using the K-means clustering algorithm [104]. Although K-means is one of the first unsupervised clustering algorithms, it is still widely used in data-mining applications, because it is relatively simple and can easily adapt to new application areas, while it is characterized by a good intuition about the data structure.

We use the K-means algorithm to separate the elements of $sumT_{SMA} = \{x_1, x_2, ..., x_N\}$ data-set in K number of clusters. We point out that the prior knowledge of K value is very important for the efficiency of K-means algorithm. Since we expect to classify the network nodes into two groups (i.e., mobile and fixed), the K parameter equals to 2. K-means randomly selects two centroid points $\{c_1, c_2\}$ from the data-set and calculates the squared distance between each element of $sumT_{SMA}$ and the centroid points. Then, it classifies each element in one of the two clusters $\{C_1, C_2\}$ based on the shortest squared distance between the element and the cluster's centroid point, as in Equation (9).

$$\begin{aligned} C_1 &= \{\{x\} \mid \forall x \in sumT_{SMA}, (x - c_1)^2 \leqslant (x - c_2)^2\} \\ C_2 &= \{\{x\} \mid \forall x \in sumT_{SMA}, (x - c_2)^2 < (x - c_1)^2\} \end{aligned} \tag{9}$$

The algorithm refines the values of its centroid points with the mean values of each cluster and repeats the classification based on the new centroid values. This process continues until the elements of both $\{C_1, C_2\}$ remain the same, between

Algorithm 4: *MODE* – Mobility Detector

```
   Input: $G$ – network-connectivity graph
   Input: $n$ – moving average window size
   Output: $mD$ – list of mobile nodes
   Static Parameter: $A$ – list of Adjacency matrices
   // A. Mobility Modeler module
1  $J \leftarrow G.getAdjacencyMatrix()$;                    // $J$ is 2D Adj. Matrix
2  $A.addNewItem(J)$;             // add $a$ in queue $A$ of Adj. Matrices
   // To maintain $n$ matrices in queue $A$, remove older items
3  if $A.count() > n$ then $A.removeOldestItem()$;
4  if $A.count() \geq 2$ then
5      for $t \leftarrow 2$ to $A.count()$ do
           // calculate Transition matrix and add it to queue $T$
6          $T.addNewItem(\left|A[t] - A[t-1]\right|)$;
7      end
       // B. Data Aggregation-and-Smoothing module
       // Average Transition matrices in window $n$ using SMA
8      foreach element $e$ of queue $T$ do
9          $T_{sum} \leftarrow T_{sum} + e$;          // sum up all Transition matrices
10     end
11     $T_{sma} \leftarrow T_{sum}/T.count()$;
       // Calculate vector $sumT$ by adding each column of $T_{sma}$
12     $sumT_{sma} \leftarrow sumPerColumn(T_{sma})$;
       // C. Decision Support module
       // Split in two clusters using K-means data-mining
          algorithm
13     $C \leftarrow wekaKmeans(sumT_{sma}, 2)$;
       // Check for the case of one cluster, where all nodes are
          fixed
14     if $\left|avg(C[1]) - avg(C[2])\right| < 1$ then
15         $mD \leftarrow null$;                                  // all fixed
       // The cluster with the highest average contains the mobile
          nodes
16     else if $avg(C[1]) > avg(C[2])$ then
17         $mD \leftarrow C[1]$;                               // mobile nodes
18     else
19         $mD \leftarrow C[2]$;                               // mobile nodes
20     end
21     $return(mD)$;                    // return the list of mobile nodes
22 end
```

two consecutive runs.

Setting up apriori $K = 2$ creates an issue in the special case where all nodes remain static. In this situation, although there is one cluster, the algorithm still tries to divide the nodes into two based on minor differences. For this reason, we enhance *MODE* algorithm with an additional condition exploiting a particular feature derived from the application's scope: the *Transition matrices* for static networks contain values very close to zero since there are no recorded changes in the network topology. Therefore, both clusters contain similar values, which are also close to zero. As in Equation (10), the *MODE* algorithm subtracts the average values of the two clusters C_1 and C_2 with number of elements n_1 and n_2, respectively.

$$\left\| \frac{1}{n_1} \sum_{i=1}^{n_1} C_{1(i)} - \frac{1}{n_2} \sum_{k_2=1}^{j} C_{2(j)} \right\| < 1 \tag{10}$$

When the result is less than the value of one, we conclude that there is no mobility action in the network. Otherwise, the algorithm returns the set of mobile nodes to the *Controller*. Moreover, the one-cluster case with all nodes being mobile is not possible in our implementation, since the *SD-MIoT* border router node is always a fixed node. Finally, although Algorithm 4 employs two-dimensional array calculations, we omit the second dimension from the pseudocode, for simplicity.

4.1.3.3 MODE: Implementation Insights - Discussion

To our knowledge, we are the first to propose an unsupervised algorithm utilizing the network connectivity behavior to detect in real-time its mobile nodes, as an important SDWSN feature for Mobile IoT. *MODE* is implemented as an open-source software [101] using the Java programming language. For the K-means clustering, we integrated the WEKA machine-learning library [105] in our solution. The modular design of *MODE* component accommodates improvements easily; therefore, we suggest the study of alternative intelligent mechanisms, including neural networks, to either improve the current solution or to enhance it towards new applications (i.e., detection of mobility patterns). Finally, we annotate that a similar methodology based on the network's *Transition matrices*, as proposed in this dissertation, can be used in the detection of other network conditions and abnormalities (i.e., radio interference, physical obstacles, or security threats).

4.2 MINOS: a Multi-protocol Framework for IoT

Addressing today's multi-application requirements hosted by IoT networks by a single protocol or communication mechanism is not feasible. In response to the need for agile and configurable solutions, SDN provides an elastic network paradigm that can transform the traditional network backbones into flexible service-delivery platforms.

Understanding the challenges resulting from the plethora of IoT hardware solutions and application requirements, we propose *MINOS* an SDN platform aiming at providing elasticity for heterogeneous IoT deployments, through the operation and dynamic configuration of different protocols. We design and implement *MINOS* using an SDN-based architecture that decouples the data from the control plane. Briefly, we explain that this architectural decision keeps the network's heterogeneity transparent to the control and application planes and employs programmable interfaces for getting cross-layer measurements. Moreover, enforcing appropriate strategies for adaptable topology and flow-control utilizing a software controller providing logically-centralized control with reduced management cost and complexity. The platform operates two network protocols that are benefited by the SDN-based architecture:

1. the *CORAL-SDN* [31], a software-defined OpenFlow-like protocol, introducing adaptive topology control and routing strategies for IoT. The CORAL-SDN dynamically enforces adaptive combinations of topology discovery and control algorithms, leveraging network's elasticity.
2. the *Adaptable-RPL* [27] developed by G. Violettas as part of the CORAL research project [106], an evolutionary extension of RPL with improvements for mobile and heterogeneous IoT networks.

The main novelties of the *MINOS* platform are: i) accommodates and adapts multiple IoT protocols because there is no "single protocol fitting all services" solution; and ii) addresses different *application and network requirements* through dynamic protocol adaptations (e.g., expressing particular network conditions and device constraints). To the best of our knowledge, MINOS is the first Software-Defined Multi-protocol platform for IoTs [107].

In the following subsections, we elaborate on the *MINOS* platform's architecture.

4.2.1 MINOS Architecture

The MINOS platform implements service-awareness by bringing together SDN with IoT technologies. It adapts IoT networks to the application requirements by on-demand deploying the appropriate network protocol, while considering the network environment constraints (e.g., limited node resource availability, mobility caused connectivity issues) by dynamically configuring the protocol in use. Fig. 13 presents a high-level view of the MINOS architecture.

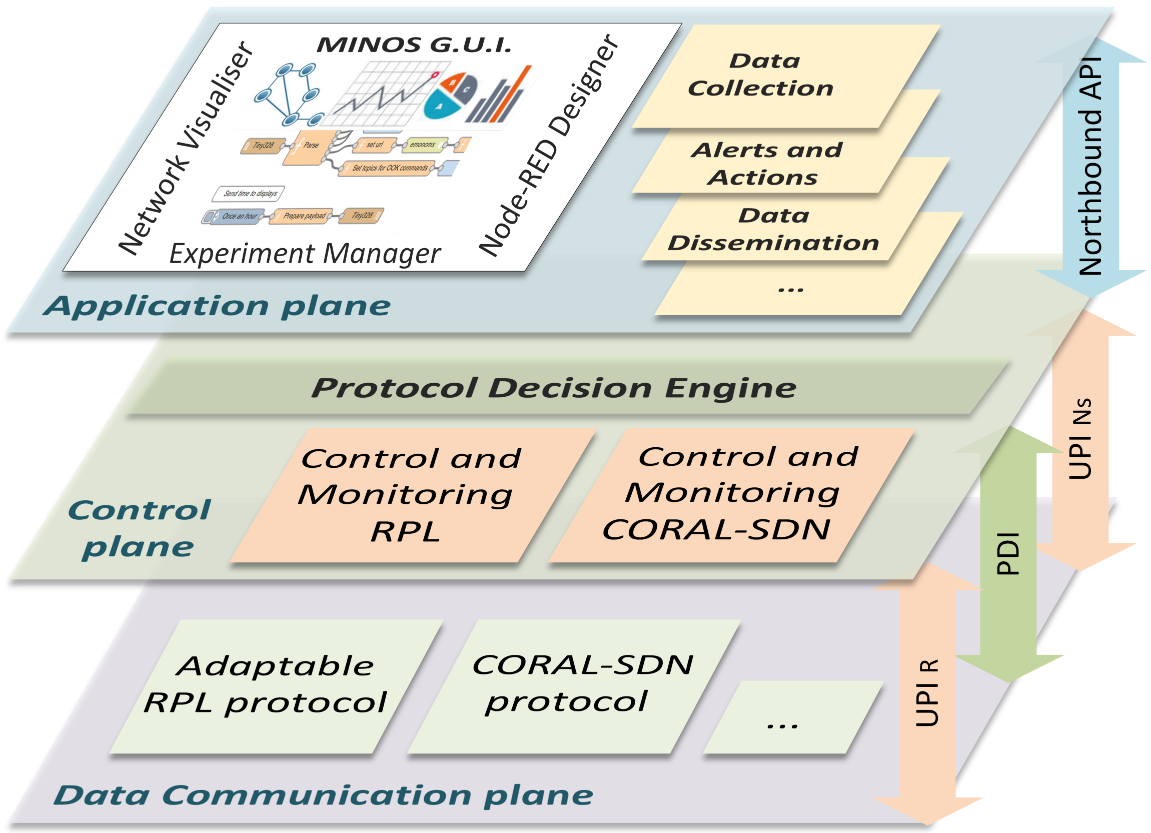


Figure 13: The MINOS architecture

Aligned to the typical three-tier SDN paradigm, it consists of the following planes, described bottom-up as follows:

1. The *Data Communication plane* accommodates multiple dynamically-configurable protocols supporting diverse IoT devices operating either in real test-beds or in the Cooja emulator. It also provides radio and network protocol measurements to the upper layer, as well as on-demand protocol deployment.
2. The *Control plane* controls the network protocols at real-time based on information coming from both the *Application plane* (i.e., application requirements) and the *Data Communication plane* (i.e., network constraints). It

consists of: (i) the *Protocol Decision Engine* that selects the protocol and its main configuration based on the application requirements; and (ii) protocol-specific control and monitoring components that are responsible for the run-time configuration adaptations and tracking the performance of the corresponding protocols, respectively.

3. The *Application plane* specifies the category of IoT application used (i.e., Data Collection, Alerts and Actions, or Data Dissemination) and the particular IoT node requirements, for example regarding their energy constraints and mobility support.

In the next subsections, we present each of the MINOS planes, interfaces, and functionalities individually.

4.2.1.1 Data Communication plane

The bottom layer of the MINOS architecture (i.e., the *Data Communication plane*) supports multiple IoT protocols, real-time configuration and measuring of the protocols, as well as their deployment on-demand. MINOS currently supports two protocols:

1. the *CORAL-SDN* is implemented in the context of the MINOS platform to provide adaptability to a range of IoT applications and network constraints (e.g., signal issues) through WSN protocol mechanisms adapted to the operation of the SDN paradigm. Mechanisms like network topology discovery and maintenance, as well as flow rule establishment methods, control the routing and forwarding processes of the network. The protocol's operation is fine-tuned from MINOS to achieve alternative types of behavior that adapt to the needs and characteristics of the application. *CORAL-SDN* also supports other configuration options; for example, the link quality estimation method, the usage of acknowledgments, and the control messages' interval time for the topology control.
2. the *Adaptable-RPL* augments RPL, with dynamic reconfigurability to extend its applicability to alternative use-cases and dynamic network environments (e.g., it improves its responsiveness to sudden changes in the network conditions). At this point, MINOS adjusts a number of important RPL parameters

(e.g., I_{min}, $I_{doubling}$) reflecting the responsiveness but also the communication overhead of the protocol, and the choice of the Objective Function to use for the distance-vector functionality of RPL. For example, in order to tackle the RPL's performance issues in mobile environments [27], MINOS adjusts the I_{min} parameter differently for nodes with particular characteristics, so communication overhead is offloaded from the mobile to the fixed nodes.

Furthermore, the *Data Communication plane* provides to the *Control plane* real-time measurements on the protocols' performance, or the configuration values of important parameters from both the radio (e.g., RSSI or LQI) and network view-points (e.g., packet drops' number). The MINOS southbound interfaces handle these interactions by utilizing novel APIs provided by the WiSHFUL project [108]. In particular, the *Universal Network and Radio Control Interfaces* (UPIs), that is one UPI_N per protocol, for network layer variables, and UPI_R for the radio channel. The WiSHFUL facilities provide hooks for dynamic adaptations in IoT communication protocols (e.g., RPL), and abstractions tackling the network or device heterogeneity. Following the WiSHFUL platform evolution, we further enriched its protocol adjust-ment capabilities, providing full support to our protocols via the MINOS platform.

4.2.1.2 Control plane

The *Control plane* triggers the protocol deployment in an interchangeable manner through the *Protocol Deployment Interface (PDI)*. This plane currently supports two alternative ways for the on-demand protocol deployment: (i) *proactively*, for net-work operation scenarios with relaxed time constraints, where protocol changes are applied through updates in IoT devices' firmware (i.e., a low memory-footprint approach with moderate deployment time); and (ii) *reactively*, for rapid protocol de-ployments, where a double-protocol stack above the link-layer dynamically selects one of the two alternative protocols (i.e., trading memory for quick protocol switch-ing). Currently, we support proactive protocol deployment through device-specific Ansible [109] scripts updating IoT devices' firmware. The reactive deployment is an on-going work, as our plans include experimentation with over-the-air protocol deployment approaches (e.g., utilizing elf Contiki-OS libraries).

The modular MINOS architecture allows easy existing protocols modifications (i.e., support of extra mechanisms or parameters) or further additions of new ones. As such, we are at the early stages of experimenting with an adaptable version of the Back-Pressure Routing (BPR) protocol. Our protocol implementations support diverse IoT hardware (e.g., RM090, and Zolertia Z1 devices).

The *Control plane* performs run-time network control and monitoring of the network environment through the *Protocol-specific Control and Monitoring* components, while it implements the service-awareness of MINOS, based on the application requirements originating from the *Application plane*, and being handled by the *Protocol Decision Engine (PDE)*.

The *Protocol-specific Control and Monitoring* components implement technology-specific local control loops for a subset, or all nodes, monitoring the behavior of the network while adjusting a rich set of network protocols' parameters to achieve the performance goals set by a particular application. At this point, MINOS supports two relevant components, reflecting the centralized network control features of the *CORAL-SDN* and *Adaptable-RPL* protocols:

- The *CORAL-SDN Control and Monitoring* component implements SDN controller functionalities that: (i) construct and maintain an abstract representation of the infrastructure network (i.e., a network connectivity structure with run-time node or link information), as for example the devices' battery level, or link quality measurements (e.g., RSSI or LQI); and (ii) perform centralized control of the data flows and define dynamic forwarding rules, responding to changes in the aforementioned network's abstract view, while matching the application requirements. For example, such control features perform dynamically topology local adjustments in the case of mobile nodes, to reduce the corresponding communication overhead.
- The *Adaptable-RPL Control and Monitoring* component collects measurements from the RPL protocol and triggers dynamic adaptations in the protocol parameters after changes in the network behavior, or to match application performance requirements. For example, an identification of mobile nodes coming from the *Application plane* results in different I_{min} parameter values for these particular nodes; that is to offload communication overhead to the fixed nodes with a power source. Furthermore, if there is a need to prioritize a

particular node-to-node communication (e.g., collecting data from all nodes to the sink may trigger an alert between two nodes), the MINOS platform may initiate minor topology changes through enforcing a new RPL Objective Function that prioritizes such communication.

The *Protocol Decision Engine (PDE)* selects the protocol to deploy, its enabled mechanisms (i.e., supported by the particular protocol) and initial configuration parameters, based on an *Application plane* request, that is to specify the communication type required by the application, the number and main node capabilities, as well as the global performance goals. Currently, *PDE* takes decisions based on hard-coded protocol strategies aligned with our experimentation analysis. However, in the future, this task can be carried out from a relevant machine-learning algorithm (i.e., neural network) inspired by [110].

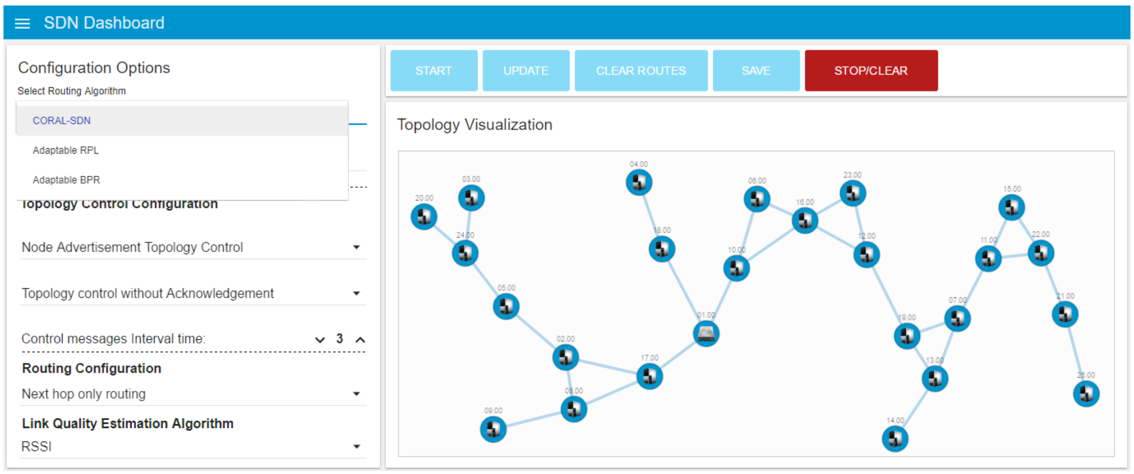
4.2.1.3 Application plane

Lastly, the *Application plane* specifies through the *Northbound API* the requirements of the application to be realized by MINOS. Such process is handled from the method: *configure_network(communication_type, nodes_configuration, prioritized_KPIs)*, where:

- The *communication_type* parameter actually defines the communication type, which can be: (i) many-to-one for the typical WSN *Data Collection* applications, where a set of nodes gather periodic measurements destined to a single sink node; (ii) one-to-many for *Data Dissemination* applications, where the sink node spreads data to all nodes in the network; and (iii) point-to-point for *Alerts and Actions* scenarios, where a node has to urgently interact with only one node. There is also the case of hybrid applications that support diverse communication methods for different parts of the network.
- The *nodes_configuration* parameter defines the number and characteristics of each IoT node, such as: (i) fixed or mobile; (ii) battery-powered or not; (iii) information on the particular resource constraints (e.g., maximum firmware size).
- The *prioritized_KPIs* parameter specifies the global performance goals for the application to be supported by the IoT network. For example, an e-health

application may request high bandwidth and low latency.

4.2.2 MINOS GUI


SDN Dashboard

Configuration Options

Select Routing Algorithm

- CORAL-SDN
- Adaptable RPL
- Adaptable BPR

Topology Control Configuration

Node Advertisement Topology Control

Topology control without Acknowledgement

Control messages Interval time: 3

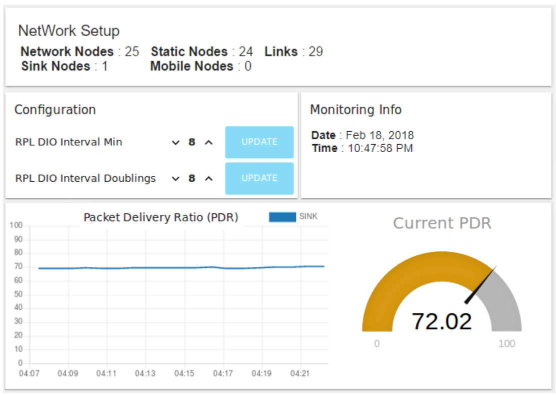
Routing Configuration

Next hop only routing

Link Quality Estimation Algorithm

RSSI

(a) Network's topology graph produced by the MINOS


NetWork Setup

Network Nodes : 25 Static Nodes : 24 Links : 29
Sink Nodes : 1 Mobile Nodes : 0

Configuration

RPL DIO Interval Min 8

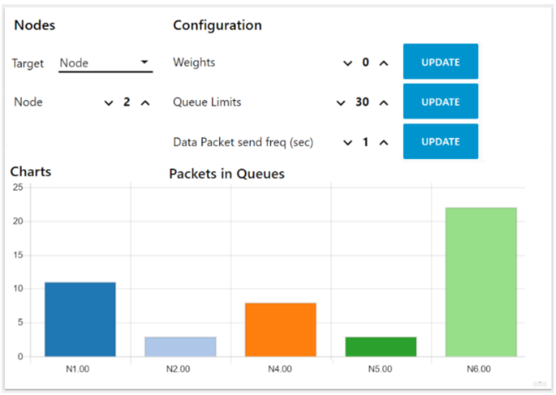
RPL DIO Interval Doublings 8

Monitoring Info

Date : Feb 18, 2018
Time : 10:47:58 PM

Current PDR 72.02

(b) PDR measurement given by the adaptable RPL protocol


Nodes

Target Node

Node 2

Configuration

Weights 0

Queue Limits 30

Data Packet send freq (sec) 1

(c) Queued packets' statistics provided by the adaptable BPR protocol

Figure 14: The MINOS Dashboard options

The MINOS platform interacts with the user through a highly flexible GUI implemented in the Node-RED platform. The GUI allows the operator to override the *PDE* and manually select the protocol and its corresponding parameters; then, the visualization outcome varies according to the protocol deployed, as in Fig. 14. The dashboard performs the overall monitoring while providing advanced functionality and configuration options through three sub-modules:

1. the *Experiment Manager*, providing configuration options related to available network protocols and experimentation set-ups;
2. the *Network Visualizer*, illustrating the network's topology, experiments' progress and results, and;

3. *the Node-RED Designer*, offering a library of the basic MINOS features implemented as Node-RED nodes and workflows. Hence, such features can be easily configured through the user interface or parameterized short client-side Node.js scripts.

In this chapter, we presented the design, implementation, and operational features of two novel SDWSN frameworks that advance SDWSN protocols in IoT environments with prominent challenges like mobility and heterogeneity: i) *SD-MIoT* that propose innovative centralized mechanisms which extend our *VERO-SDN* SDWSN solution towards mobile environments with dynamic topologies; and ii) *MINOS* a multi-protocol platform that integrates under a three-tier SDN-like architecture an elastic network management system that aims to optimize routing over heterogeneous IoT nodes.

In the following chapter, we present our experimental results and validate the impact of the proposed solutions.

5 Evaluation and Outcomes

In this chapter, we provide our extensive evaluation and analysis of results that highlight the performance advantages of the proposed solutions. The chapter is organized into five sections that contain independent experimental scenarios and evaluation results. Each section includes the corresponding evaluation methodology, simulation setup, performance metrics, and evaluation results and discussion. To further motivate our choices of experimental evaluations, we propose realistic use case scenarios discussed at the beginning of each section. Briefly, in this chapter, we elaborate on the evaluation of:

- *VERO-SDN* platform and its corresponding network control mechanisms. Our main goal is to show that we 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. 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, discussed in sections 5.1 and 5.2.
- *SD-MIoT* platform and its corresponding network mobility control mechanisms. We elaborate on our evaluation analysis in two sections. In section 5.3, we investigate the overall network operation in terms of packet delivery and control overhead, aiming to verify the robust routing performance and reduced control overhead of *SD-MIoT*. Whereas, in subsection 5.4, we devote an independent analysis of the effectiveness of *MODE* algorithm with regards to the success ratio of discovering the mobility conditions of the nodes.
- *MINOS* platform against handling mobility and heterogeneity in an IoT experimentation setup of a smart city use-case scenario with static or mobile nodes equipped with sensors (i.e., temperature, humidity, light noise, pollution to offer city monitoring facilities). In Section 5.5, we employ the evaluation of *MINOS* using two network protocols, i.e., the CORAL-SDN and the Adaptable RPL, and we extract results regarding two metrics: the PDR and the control

Overhead.

As a point of reference to contrast our solutions with the traditional IoT deployments, 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, e.g., [27], [33], [73].

5.1 Evaluating Topology Discovery in Networks with Fixed nodes

The accurate and timely representation of the network's topology is a critical protocol mechanism. In the following subsections, we evaluate the topology discovery performance of *VERO-SDN*.

5.1.1 Use-case scenario: Smart Traffic Lights

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. When it comes to applications that require all types of communication patterns (i.e., one-to-one, one-to-many, and many-to-one), RPL faces performance and reliability issues, which raises the question on whether RPL can support today's IoT requirements [111].

VERO-SDN supports all communication patters just as right, taking advantage of SDN's elasticity. To motivate our network evaluation scenarios, we consider a use-case that can accommodate the features and functionalities offered by *VERO-SDN*.

For example, in a smart traffic light system as in Fig. 15 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. As in this example, the variability in communication patterns can further improve the network's

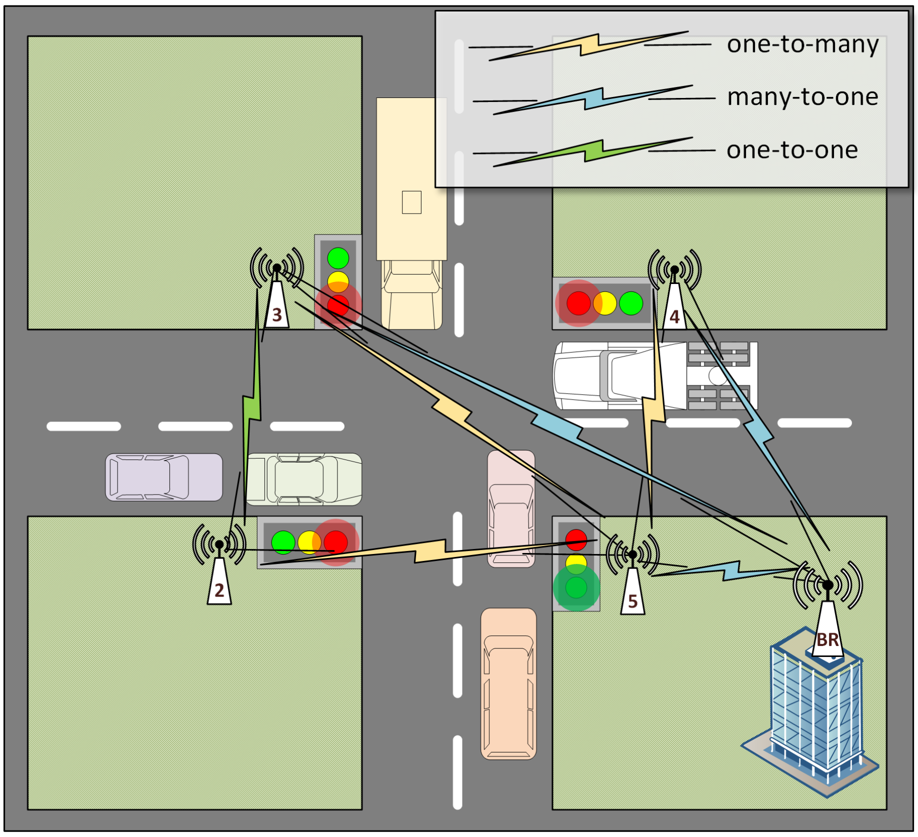


Figure 15: Smart Traffic Lights Use-case scenario

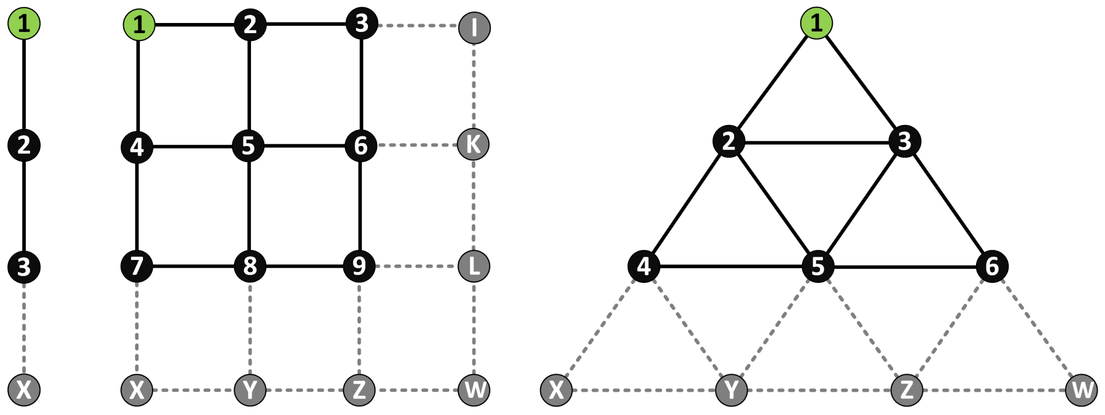
performance due to the direct communication paths among nodes that avoid the meddling of a sink node.

However, the possibility of alternative routes requires a very good knowledge of the network's connectivity. As such, the duration time for the discovery of the entire network topology, as the main aspect of topology discovery, 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. In the following subsections, we aim to gather results through simulations for each one of them within different network terrains.

5.1.2 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. 16a), a *Rectangular Grid* (Fig. 16b), and a *Triangular Grid* (Fig. 16c).

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 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.



(a) Linear G_1 **(b)** Rectangular Grid G_2 **(c)** Triangular Grid G_3

Figure 16: 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

In Table 4, 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:

Table 4: 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

- 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.

5.1.3 Simulation Setup

In our simulations, we use the *Cooja* [112] simulator with emulated Zolertia Z1 IoT devices. Cooja is a Linux based cross-layer WSN simulator for Contiki OS, which enables the creation of virtual WSN scenarios. In Tables 5 and 6, 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 section 3.5.

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.

Table 5: 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 [59]	ver 3.0

5.1.4 Performance Metrics

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

Table 6: 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 [59]	ver 3.0

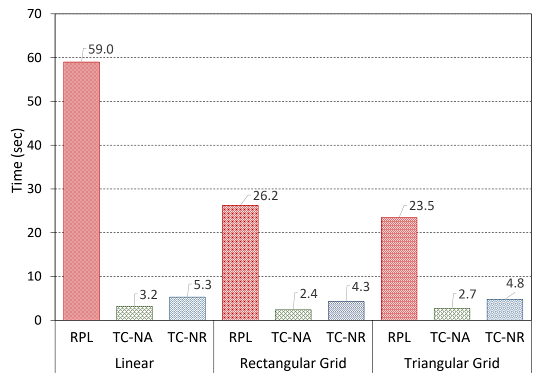
- *Topology Discovery Duration (TDD) Time*: The *TDD* time represents the total duration of time in seconds required to collect and construct the connectivity graph for the entire network.
- *Topology Discovery Control (TDC) Overhead*: The *TDC* overhead represents the total number of control messages 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* [113], 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.

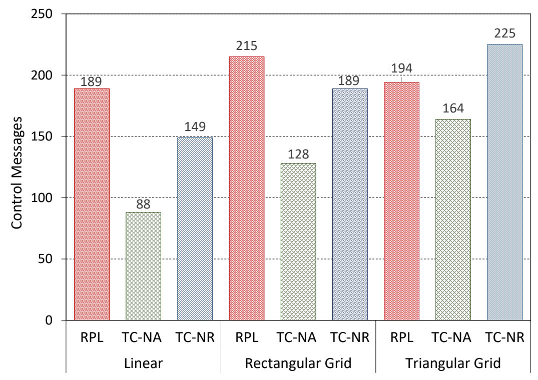
We carry out 15 simulation runs for each scenario and calculate the average values for both metrics. For clarity purposes, we omit the standard deviation values in the figures, since they are insignificant in all scenarios, in contrast to the performance differences we observed.

5.1.5 Evaluation Results and Discussion

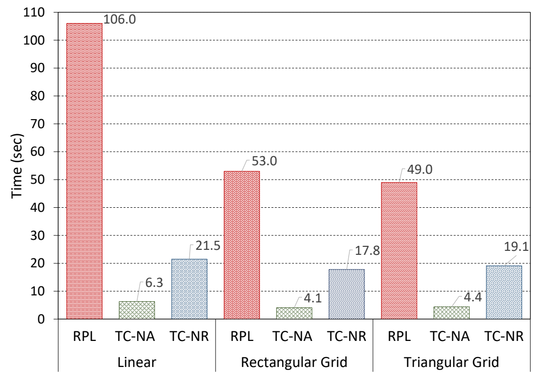
In Fig. 17, we illustrate the network topology discovery evaluation results for all the three topology scenarios. We depict our results using bar charts that exhibit each algorithm's performance based on the aforementioned metrics, i.e., *TDD* time and *TDC* overhead, classified per topology scenario for both networks of 30 (i.e., Fig. 17a and Fig. 17b) and 90 nodes (i.e., Fig. 17c and Fig. 17d).



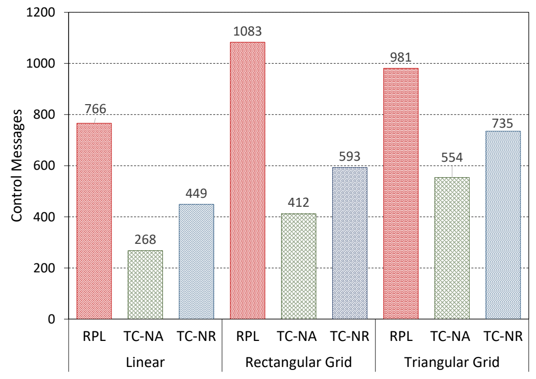
(a) Total discovery duration time for networks of 30 nodes



(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 17: Network discovery evaluation results for RPL, *VERO-SDN TC-NA* and *TC-NR* with linear, rectangular and triangular grid network topologies

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 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. 17b). 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. 17d) 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. 17a 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. 17c, 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. 17a 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. 17c), 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 4. 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.

5.2 Evaluating Routing Control in Networks with Fixed Nodes

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 the following subsections, 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.

5.2.1 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. 18. 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.

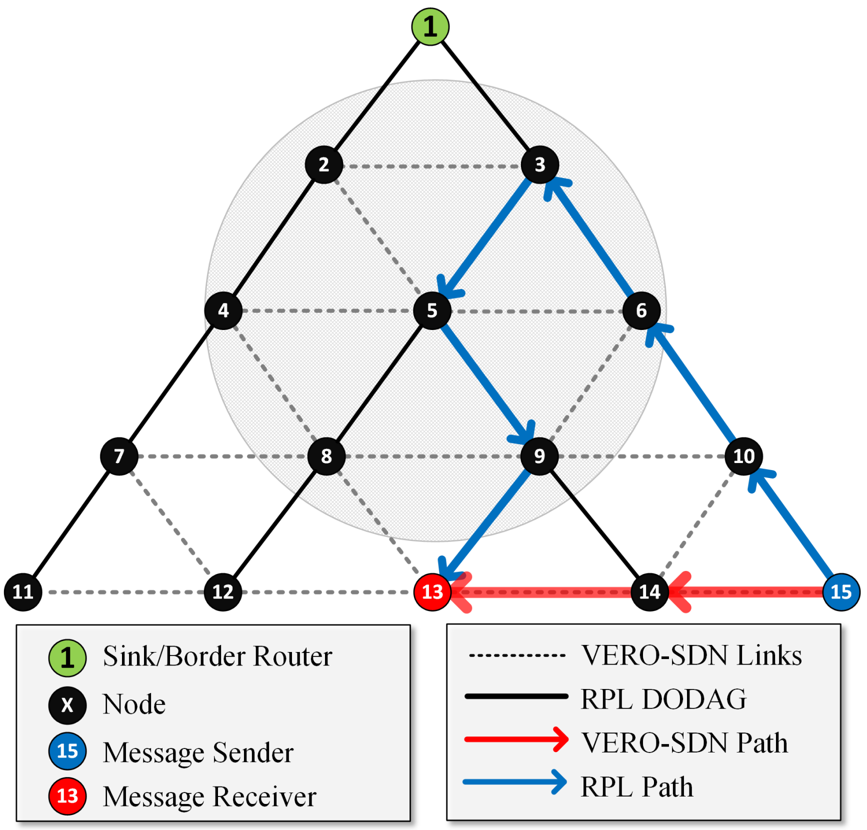


Figure 18: Routing evaluation scenario with a triangular grid network topology

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.

Table 7 enlists the values of the above three metrics obtained from the Distance Matrices of three triangular graphs. One of 15 nodes, like in Fig. 18 and two subgraphs 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.

Table 7: 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	

5.2.2 Simulation Setup

To justify the above theoretical insights, we create the network shown in Fig. 18 and conduct simulations based on the same simulation setup environment described in subsection 5.1.3. The protocol parameters for RPL and *VERO-SDN* protocols are recorded in Tables 5 and 6, 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.

5.2.3 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. It is calculated as the ratio of *received* data messages R_x over *sent* data messages S_x transferred among the motes, as in (11).
$$PDR = \frac{\sum R_x}{\sum S_x} \tag{11}$$
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 [114] 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 com-

munication 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 (12).

$$EED_{(N,N)} = \begin{bmatrix} 0 & d_{1,2} & d_{1,3} & \cdots \\ d_{2,1} & 0 & d_{2,3} & \cdots \\ d_{3,1} & d_{3,2} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{12}$$

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 (13).

$$TEED_{(N,N)} = \sum_{m=1}^{M} EED_m \tag{13}$$

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 (14), we depict the *NOD* metric formula for a network of *N* nodes. Undelivered packets are excluded from the calculations.

$$NOD = \sum_{i=1}^{N} \sum_{j=1}^{N} TEED_{(i,j)} \tag{14}$$

5.2.4 Evaluation Results of *all-to-all* scenario

In Fig. 19, 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.

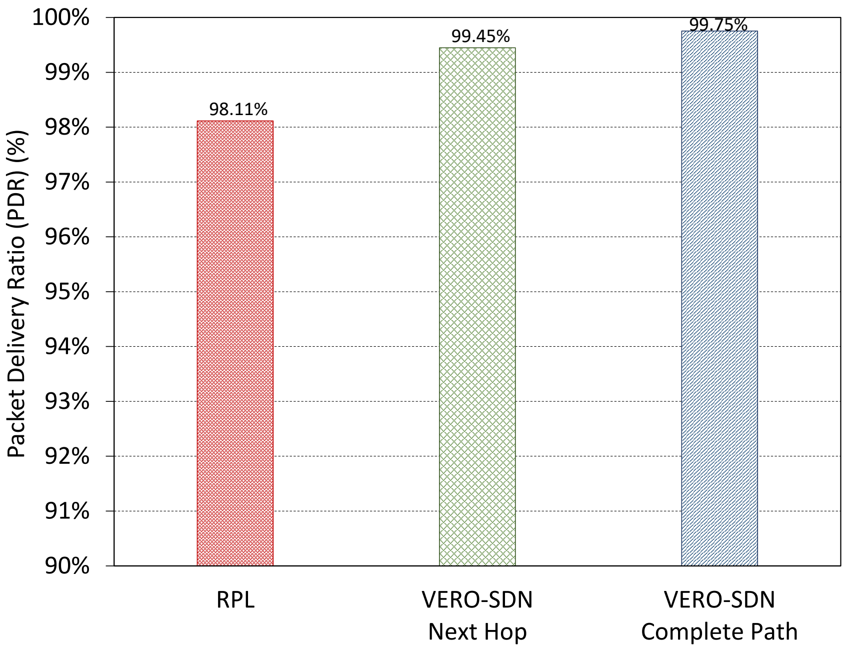


Figure 19: The average *PDR* per protocol

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. 20):

- $TEED_{(15,13)}$ (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. 18 the flows that each protocol selects. The dark blue arrows illustrate RPL's flow through the established DODAG, while the

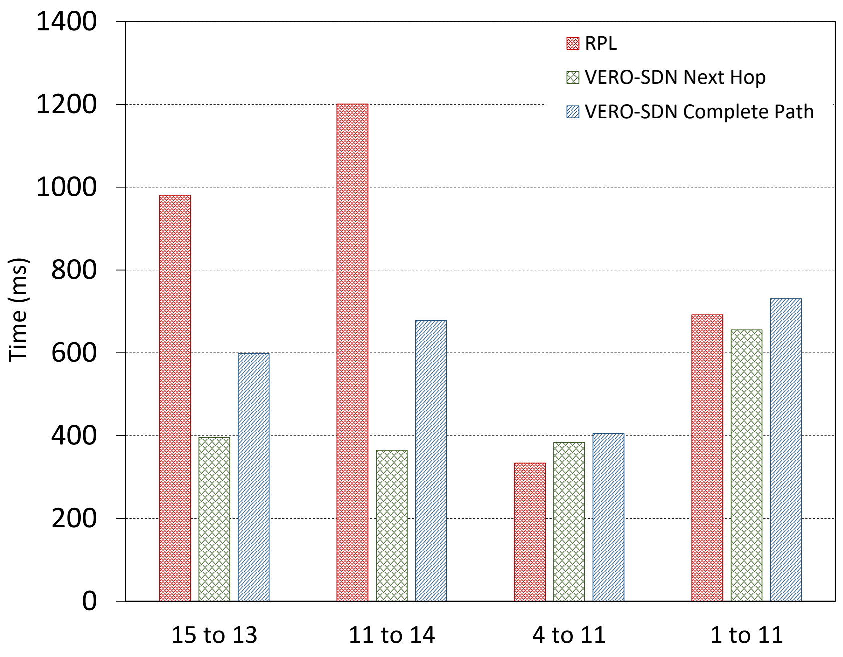


Figure 20: Total End-to-End Delay (TEED) time for selected paths from node X to node Y

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 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. 21 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.

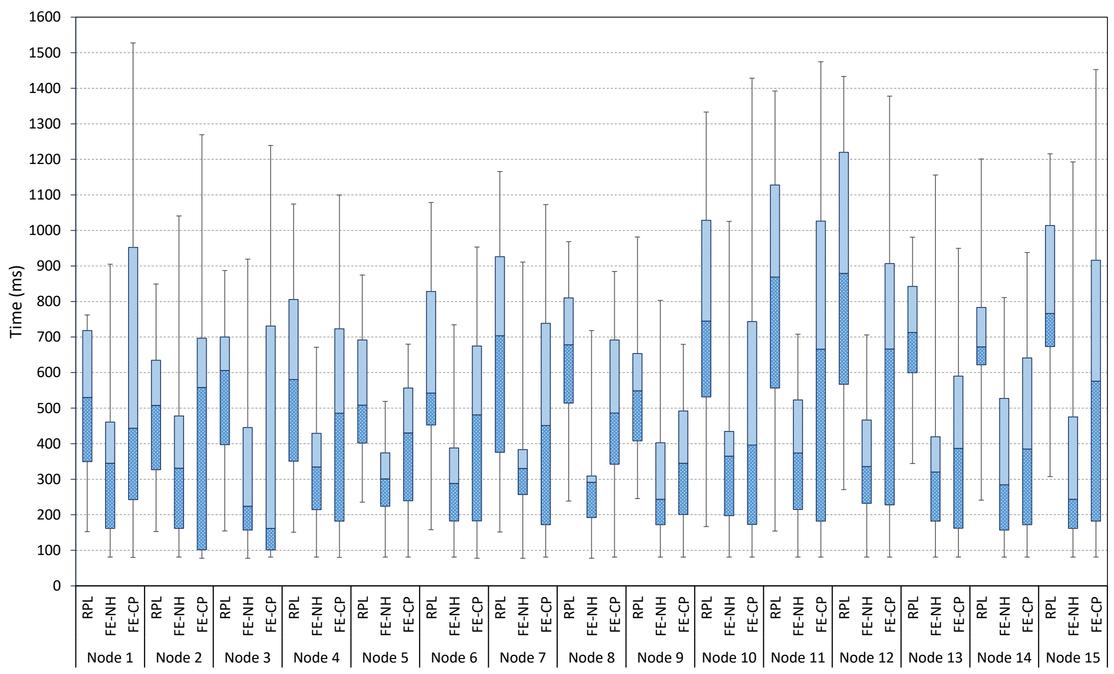


Figure 21: The range of *TEED* values from each node to all other network nodes

The network overall end-to-end delay NOD time can be seen in Fig. 22. 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. 23, we observe a summary per-protocol of the *TEED* values in quartiles through the box diagram. From this chart, we observe that 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,

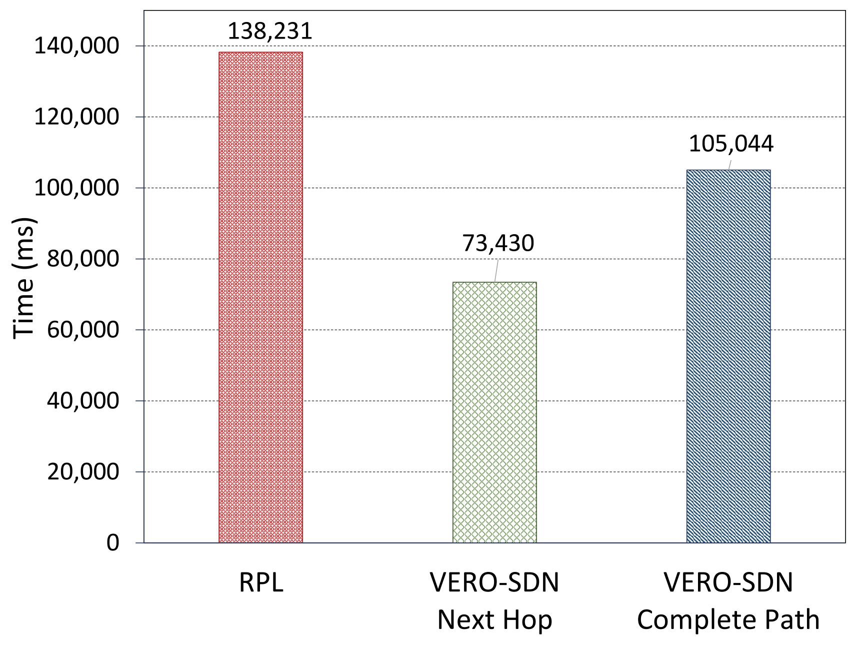


Figure 22: The Network Overall End-to-end Delay (NOD), *all-to-all* scenario

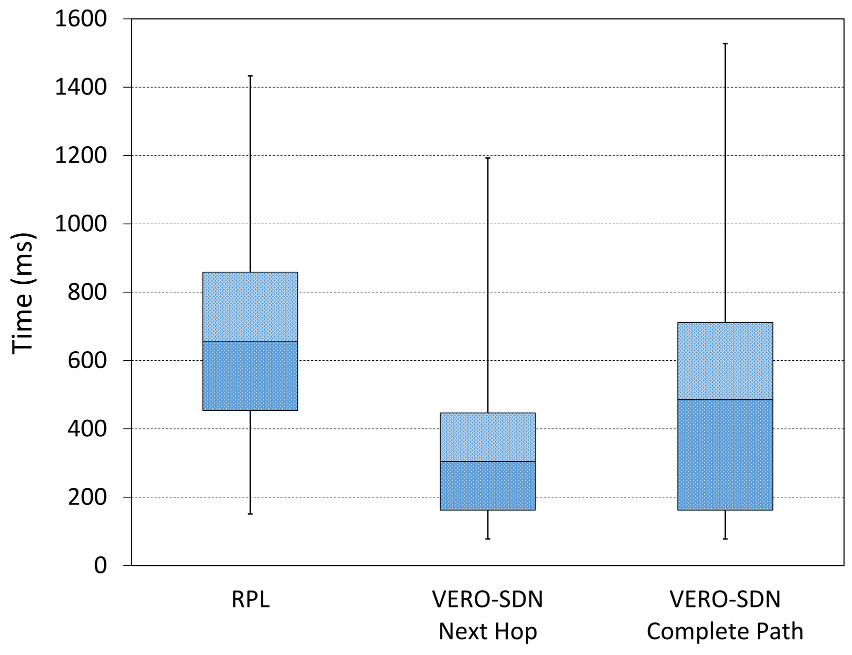


Figure 23: The range of *TEED* values per protocol, *all-to-all* scenario

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. 19, 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.

5.2.5 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. 24 and 25). 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. 24, 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.

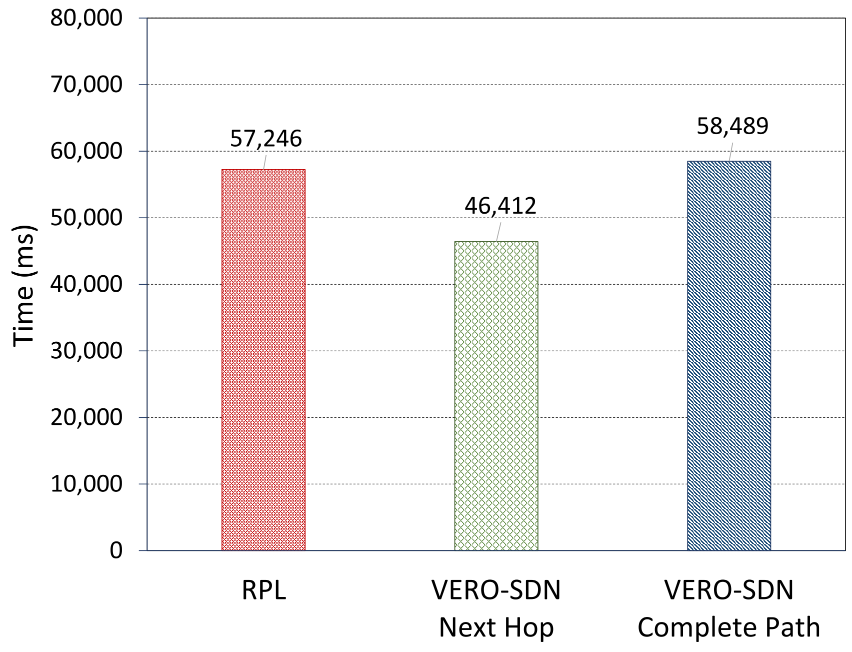


Figure 24: The Network Overall End-to-end Delay (NOD), *many-to-one* scenario

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 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. 25) 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

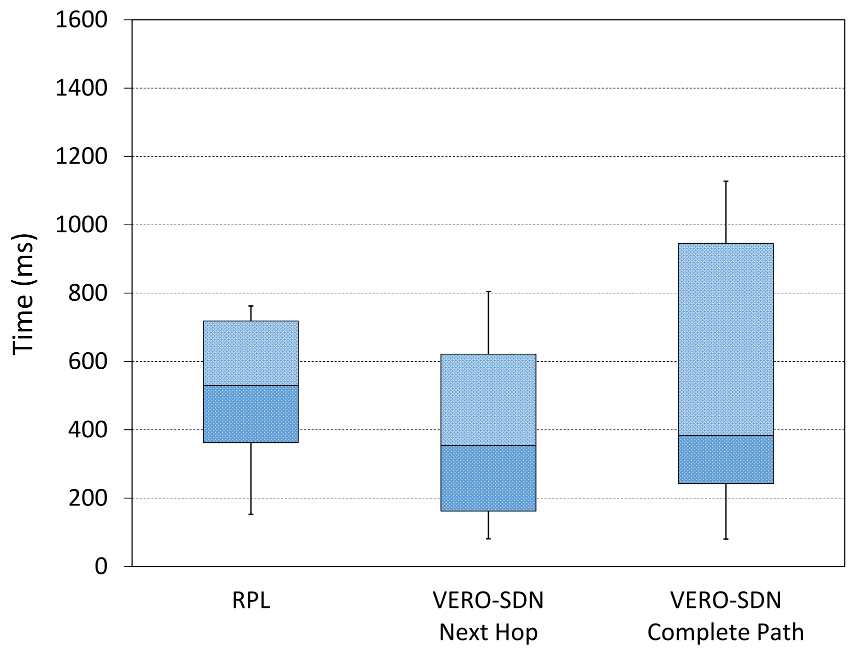


Figure 25: The range of *TEED* values per protocol, *many-to-one* scenario

performance.

Right after, we carry on with the performance evaluation of the *SD-MIoT* proposed mechanisms and processes in network scenarios that include mobile nodes.

5.3 Evaluating Routing Control in Mobile Network environments

In this section, we provide our extensive evaluation analysis highlighting the performance advantages of *SD-MIoT* and its corresponding network mobility control mechanisms. Our main goal is to achieve robust packet delivery performance and reduced network control overhead for mobile IoT communication environments. Therefore, we carried out a variety of realistic simulations utilizing real human mobility traces as well as synthetic based on the *random waypoint* (RWP) mobility model [115]. Our proposal is compared against RPL, and the results highlight the improvements of *SD-MIoT* in terms of successful data delivery with efficient network control overhead.

5.3.1 Use-case scenarios: Extreme-Sports & Amusement park

To further motivate our experimentation analysis, we discuss two representative use-cases, i.e., the *extreme-sports activity* and the *smart amusement park* scenarios. We emphasize the research challenges that emerge in each case and how to tackle them by the appropriate WSN deployment. Briefly, both use-cases contain a border router node and a mixture of mobile or fixed regular nodes in linear and grid topologies. The network's operation is characterized by devices' mobility issues and the requirement for robust network operation. Specifically, the extreme sports activity emerges the need for network operation with routing and forwarding stability, reliable data delivery, and enhanced QoS due to the critical environment for human life. In contrast the amusement park requires traffic prioritization and reliable network connectivity due to intensive data transmission challenges during peak times and dynamic topology changes. We further elaborate on the two use-cases below, and we evaluate the performance of *SD-MIoT* in terms of packet delivery ratio and control overhead for both scenarios through simulations in the same section.

5.3.1.1 Extreme sports use-case

Action or Extreme sports are activities that often involve speed, height, and a high level of physical exertion combined with uncontrollable factors like the weather and terrain, including mountains, snow, water, and wind. This combination is also well known for involving a high degree of risk for the participants. Mountain sports, e.g., skiing, mountain running, and mountaineering, are among the most famous extreme sports that attract people of all ages, providing satisfaction and unique experiences. Mount Olympus, the Greek mythical "Mountain of the gods," attracts approximately $10,000$ climbers per year. Moreover, several running racing events are held in mountains annually, i.e., the *Olympus Marathon* [116] and the *Olympus Mythical Trail* [117], a unique trail running event of about 100 *km* length and 6700 *m* altitude to overcome. A high-risk sport requires excellent preparation and use of safety measures and precautions to reduce the probability of accidents, which unfortunately may occur in physically isolated areas with low proximity to medical facilities. In such cases, exchanging timely information between the injured party and the rescuers is of paramount importance. Inspired by this, we investigate and evaluate the effectiveness of *SD-MIoT*, in such a harsh and dangerous terrain.

In this context, we suggest the deployment of a WSN using a linear topology of wireless fixed-position environmental sensor nodes placed in a "W" shape, to achieve maximum radio coverage in width, along both sides of the trail. The runners use wearable mobile sensor nodes capable of measuring vital signs and delivering alerts. The nodes can either operate as forwarding devices or information sinks. Fig. 26 depicts on a Google map [118] a 40 *min* duration and 1800 *m* distance of real mobility trace data [119], recorded at the highest altitude and most dangerous part of the Olympus Mythical Trail race in 2016, located on the Olympus mountain peak named “Mytikas". At the start of the running trail, we assume a racing campsite, where we locate the network control facilities. A race monitoring software application utilizing the many-to-one traffic pattern of *SD-MIoT*, continuously collects environmental conditions and runners' vital signs. Respectively, when necessary, the application reacts actuating alerts using one-to-one or one-to-many traffic patterns.

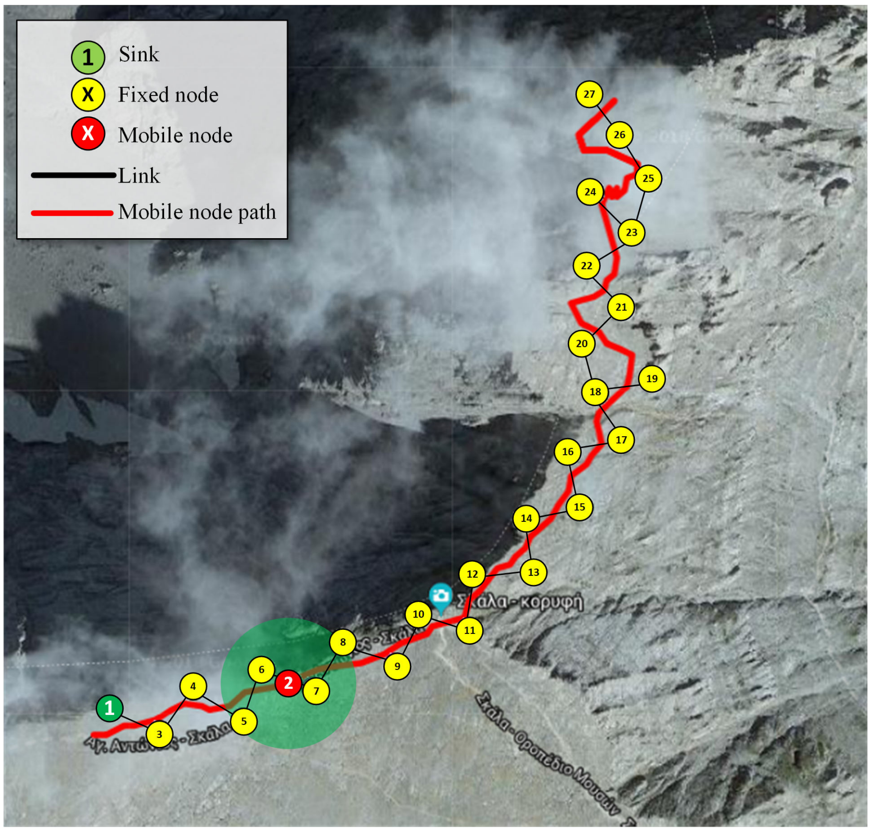


Figure 26: Olympus Marathon Mythical Trail, extreme sports activity scenario, Linear topology with 1 border router, 1 mobile, and 25 fixed nodes

The key challenge in this scenario is the network reliability and QoS in terms of timely and without interruption delivery of data for all nodes, i.e., fixed or mobile. Since we focus on mobility issues, we consider outside of this research context the physical threats or other challenges related to the harsh mountain environment, which may also affect the reliability of the network. *SD-MIoT* addresses the above problem using two main features: (i) adaptive topology discovery mechanisms utilizing timely and reliable knowledge of the network and targeting specific parts of the network, i.e., the mobile nodes, to achieve robust communication with reduced control overhead; and (ii) proactive, dynamic deployment of forwarding rules and configurations to mobile nodes, based on centralized intelligent decision-making, utilizing an abstract network representation graph of multivariate weights.

5.3.1.2 Amusement park use-case

Amusement or thematic parks are typical enterprises in which innovative IoT applications can leverage their business operations and enhance the customer experience to a level that was not possible in the past, using the traditional management and operation methods. In Fig. 27 [120], we consider a *water-park* being monitored from a central control room using two types of monitoring nodes, i.e., wireless nodes placed at fixed positions in a grid topology and wearable devices used for monitoring visitors (e.g., their location in the park), delivering useful notifications and alerts, as well as exchanging messages among users.

There is a wide variety of applications that can be potentially applied in this scenario; however, introducing several challenges to the network infrastructure, for example: (i) safety and surveillance applications (i.e., helping parents to find lost children), demand network reliability and robustness; (ii) customer service improvement through the big-data collection and predictive analytics applications (i.e., park activities, traffic congestion analysis, and queue waiting time reduction), require centralized control and processing power; and (iii) proximity marketing applications using real-time information and alerts (i.e., providing information tips and promotion notifications based on the time of day or the location of the visitors), involve low latency end-to-end communication.

To confront these challenges, especially in dynamic network environments with mobile nodes, the *SD-MIoT* framework enables an OpenFlow-like centralized network protocol management. By offloading the control operations from the resource-constrained devices to the fixed infrastructure, *SD-MIoT* takes accurate mobility-

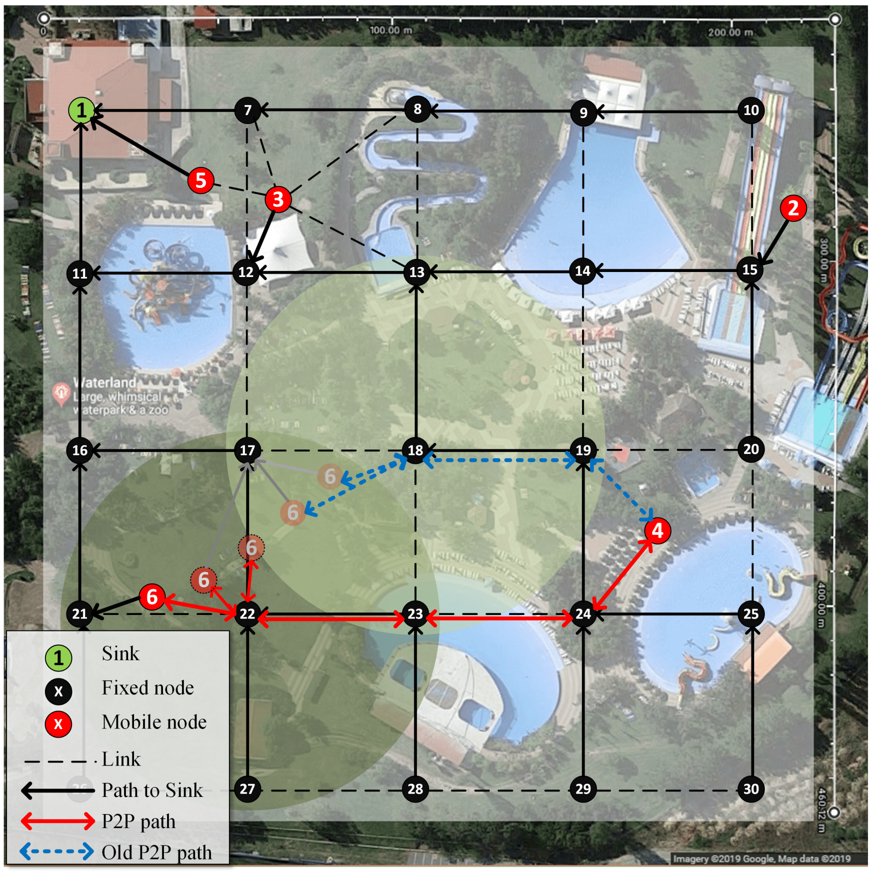


Figure 27: Water Park smart amusement park scenario, Grid topology with 1 border router, 5 mobile, and 24 fixed nodes

aware network control decisions, utilizing the global network view. In this context, routing prioritization policies adaptable to the type of device in terms of mobility behavior can establish efficient paths, e.g., forwarding the information from wearable devices, preferably through fixed nodes to avoid communication disruptions.

Moreover, in Fig. 27, we demonstrate cases of routing decisions that *SD-MIoT* can tackle, utilizing its novel mechanisms and a timely, robust view of the network topology. Mobile node 2 employs the closest fixed node 15 to deliver data to the sink. Although mobile node 3 can opportunistically forward packets through another mobile (i.e., node 5), it selects a path through nodes 12 and 11 to guarantee a robust delivery. Furthermore, node 4 exchanges information with node 6 and the network Controller realizes that node 6 moves away from node 18, initially serving their communication. The Controller is proactively establishing a new forwarding path to handle this situation, i.e., through nodes 24, 23 and 22. To sum up, such facilities can benefit from *SD-MIoT* centralized management framework and its intelligent network control and routing decisions based on link quality estimations, along with its adaptive topology discovery to mobile environments.

5.3.2 Evaluation Methodology & Simulation Setup

Here, we investigate the capability of *SD-MIoT* to address the main research challenges defined in Chapter 1, like mobility and network's QoS. We consider two distinct scenarios corresponding to the use-cases defined above. Our two sets of simulations probe into two main network operation processes, i.e., topology discovery and flow management, unveiling the improvements that *MCC* brings in mobile network environments. These essential routing protocol factors are associated with successful packet delivery and are tightly bonded with the QoS of the network.

In this context, we compare *SD-MIoT* against RPL. Although RPL primarily targets fixed environments [19], as we explain at the beginning of this chapter, we use it as a point of reference for our results and as a common denominator for comparisons with other works in the future.

Table 8: Simulation parameters and configuration setups

Parameters		**Configurations**
Extreme-sports	Mobility model	Real Traces [119]
	Linear topology	$500 \times 500\ m$
	Border routers	1 node
	Fixed	25 nodes
	Mobile	1 node
Water-park	Mobility model	RWP max.speed 5 km/h
	Linear topology	$300 \times 300\ m$
	Border routers	1 node
	Fixed	24 nodes
	Mobile	5 nodes
Data	Packet size	128 $Bytes$
	Period of fixed nodes	6 data-packets/h
	Period of mobile nodes	120 data-packets/h
Network	Transport	UDP
	Network	SD-MIoT / RPL
	Physical/MAC	IEEE 802.15.4
Hardware	Radio Interface	2.4 GHz
	Radio Range	50 m
	Mote	Zolertia $Z1$
Simulation	Simulator	COOJA [112]
	Duration	40 min
	OS	Contiki ver 3.0

In Table 8, we show the experimental setups of both simulation scenarios. We highlight that the extreme-sports scenario, as described earlier, considers a linear topology with one mobile node utilizing real traces, and through its simplicity, aims to provide proof-of-concept outcomes. While the water-park scenario, probes into a real-life grid topology scenario using synthetic mobility traces generated from the RWP model, reproducing human mobility characteristics (i.e., speed less or equal to 5 km/h), for five nodes. Moreover, for RPL we configure the trickle timer to its default values as implemented in Contiki OS v3.0 (i.e, $I_{min} = 12$, $I_{doublings} = 8$ and the redundancy constant $k = 10$). Whereas for *SD-MIoT*, we set up the *MOB-TD* parameters $TRt = 10\ min$ and $TTRr = 10$, which result in a $TTRt = 60\ sec$ for the mobile nodes. Since we focus our study on routing processes and algorithms under mobility, we assume a radio environment with data loss only related to distance and no other signal issues. This configuration provides explicit conclusions related to the network layer mechanisms only and creates a deterministic environment, so there is no need to validate the statistical accuracy of our results. Finally, we select higher data transmission rates for mobile nodes (i.e., 120 $data\ packets/h$) compared to the fixed ones (i.e., 6 $data\ packets/h$), because we concentrate our analysis towards the mobility aspects, due to the fact that it suits the requirements of our use-cases.

5.3.3 Performance Metrics

In our experiments, we use two metrics:

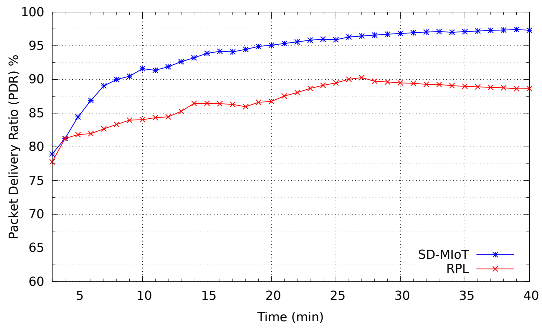
1. *Packet Delivery Ratio (PDR)*, to measure *SD-MIoT's* reliability in terms of message delivery success ratio as in Equation 11. Higher *PDR* values declare reliable transmission, whereas lower values reveal deficiencies in routing processes since, in our simulations, we assume no radio signal issues.
2. *Control Messages Overhead (CMO)*, to show the efficiency of our solution. This metric is a ratio calculated as the number of control messages C_x (i.e., routing packets) propagated by each mote throughout the network and the total number of packets received by the destinations $T_x = R_x + C_x$, as in (15).

$$CMO = \frac{\sum C_x}{\sum T_x} \tag{15}$$

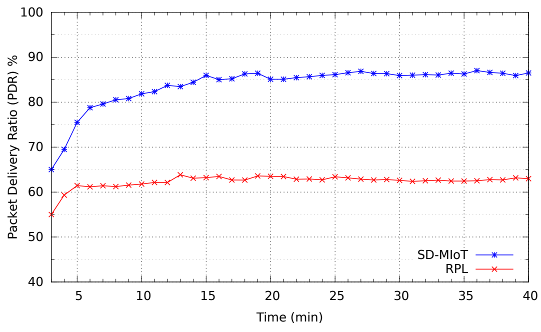
Since the value of this metric is scenario-specific, we use it only for comparisons between simulation results within a scenario.

5.3.4 Evaluation Results & Discussion

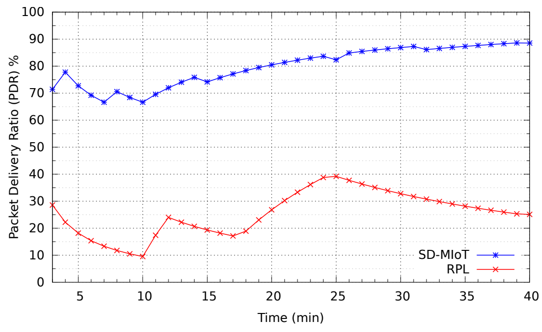
In Fig. 28, we illustrate the simulation results of both use-case scenarios. In detail, in Fig. 28a, for the *extreme-sports* scenario with one mobile node, we observe that at the end of simulation (i.e., 40 *min*), in terms of *PDR* performance, *SD-MIoT* achieves 97.5% in reliable data transmission.



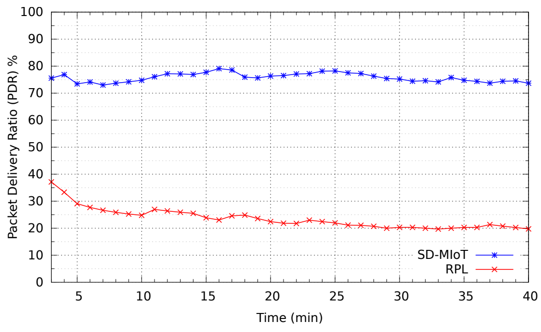
(a) Extreme-sports – PDR as a function of time for all nodes



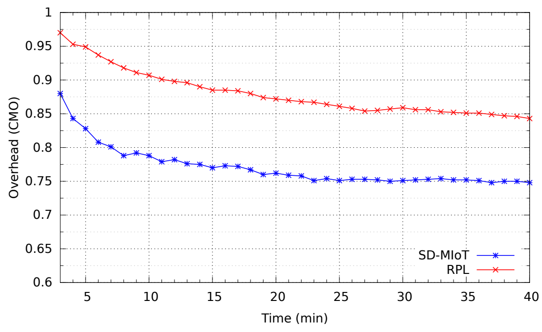
(b) Water-park – PDR as a function of time for all nodes



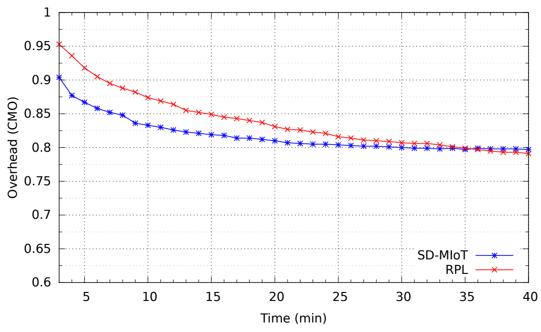
(c) Extreme-sports – PDR as a function of time for the mobile node



(d) Water-park – PDR as a function of time for the mobile node



(e) Extreme-sports – Control Overhead as a function of time



(f) Water-park – Control Overhead as a function of time

Figure 28: Extreme-sports & Water-park use-case results

This result is an early justification of our expectations for reliable performance of our solution, while in the same chart, we remark that RPL produces worse performance by 9%. The severity of the effect that mobility has in the network reliability is predominately evident for RPL in Fig. 28c, where we observe that the mobile node succeeds in delivering only one-fourth of the transmitted data (i.e., $PDR = 25\%$). On the contrary, the mobile node with *SD-MIoT* fails to deliver only 10% of its data messages. Moreover, the control overhead produced from both protocols, as shown in Fig. 28e, demonstrates similar characteristics to the *PDR* results. In particular, *SD-MIoT* shows efficient operation with reduced *CMO* by 10% compared to RPL overhead. This outcome is important because it comes from RPL having no specific control strategies to reinforce mobility. If we assume that we apply any of the solutions focusing on RPL mobility behavior, the *CMO* metric will only get worse due to the fact that the majority of them increase the frequency of DIO control messages [47]. A relevant comparison between an early version of *SD-MIoT's* and *Adaptable RPL* [35], supports this claim. As a bottom line, the discussed results of *extreme-sports* simulation justify our initial hypothesis to suggest *SD-MIoT* in a critical for the human-life use-case scenario that requires reliability and QoS.

The *water-park* simulation scenario with five mobile nodes, exhibits outcomes in analogy to the *extreme-sports* scenario results detailed above, verifying the consistency of *SD-MIoT* operation. In detail, in Fig. 28b, although *SD-MIoT's PDR* achieves lower values than in the *extreme-sports* scenario, it reaches 86% of successful packet delivery, with an evident improved performance above 20%, compared to RPL. Considering the mobile nodes separately from the rest of network nodes, in Fig. 28d, we uncover why RPL performance is drastically reduced, since the mobile nodes manage to deliver only 20% of the transmitted data, with *SD-MIoT* resulting almost a 70% PDR. Since the number of topology discovery control messages of *SD-MIoT* depends on the number of mobile nodes in the network, as we explain in Section 4.1.2, we observe that the produced overhead at the end of the simulation is slightly worse than RPL's (Fig. 28f). Nevertheless, this is expected since the higher number of mobile nodes requires more control messages to detect the dynamic changes in the network topology.

In conclusion, our results confirm the suitability of *SD-MIoT* in mobile IoT environments, improving the routing efficiency and reducing the control overhead. Proceeding with the evaluation of SD-MIoT in the next section, we extend upon the presentation of MODE, the Mobility Detector of the SD-MIoT platform.

5.4 Evaluating Mobility Detector MODE

To investigate the effectiveness of *MODE* algorithm, we conduct a proof-of-concept experimentation analysis confirming the algorithm's accuracy in detecting node mobility. Particularly we focus on its operational characteristics isolated from factors that affect the network connectivity, e.g., external radio interference. Thereafter, we elaborate on the methodology and experimentation arrangements we follow in our evaluation.

5.4.1 Evaluation Methodology & Simulation Setup

We conduct simulations aligned to the *water-park* use-case scenario (i.e, Section 5.3.1.2) and considering a network of 30 nodes with one border router, 24 fixed nodes, 5 mobile nodes, and the parameters defined in Table 9. The mobile nodes are moving according to the RWP model and follow a predefined mobility pattern with starts and stops, as shown in Fig. 29a. The main goal of our evaluation is to assess the ability of *MODE* to successfully detect, at any given time, the mobility-condition of each node in the network (i.e., whether they move or stay motionless).

Table 9: Simulation setup of MODE evaluation

Settings	**Description**	**Configuration**
Clustering method	K-means	2 clusters
Data-Smoothing method	SMA	window size 5
Recurrence	Interval time (*TTRt*)	60 *sec*
Network nodes	Zolertia Z1	30 motes
Network deployment	Grid topology	300×300 *m*
Network protocol	SD-MIoT	[101]
Mobility model	Random Way Point	max speed 5 km/h
Mobility pattern	5 mobile, 25 fixed	Figure 29a
	Duration	80 *min*
Simulation	Simulator	COOJA [112]
	OS	Contiki-OS 3.0 [59]

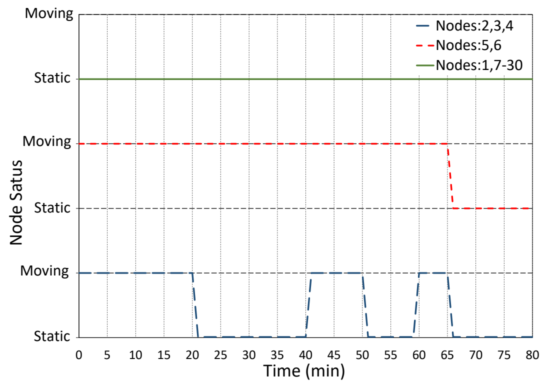
5.4.2 Evaluation Metrics, Results & Discussion

We initially justify the validity of *Transition Matrices*, as the primary criterion for identifying the moving nodes. In Fig. 29b, we depict a chart with the aggregated values of $sumT_{SMA}$ per node over time. As described in the previous chapter in subsection 4.1.3, these values represent the total connectivity changes for each node, emerging from Equation (8), and they constitute the basic input for K-means clustering process. From the chart, we observe significantly higher values for the moving nodes, compared to the fixed ones. For example, nodes 5 and 6 consistently maintain values above 4 until the 65 *min*, when their values start dropping as the nodes stop moving. We observe similar behavior for nodes $2,3$ and 4, since their values drop when they stop at $20,50$ and 65 *min*. Correspondingly, they increase again when they start moving at 40 and 60 *min*. The fixed nodes maintain values varying from zero to two, mainly due to surrounding moving nodes that affect their adjacency matrices. The combined result of Fig. 29b and 29a is a visual justification of how *MODE* utilizes the *Transition Matrices* as a decision-making criterion.

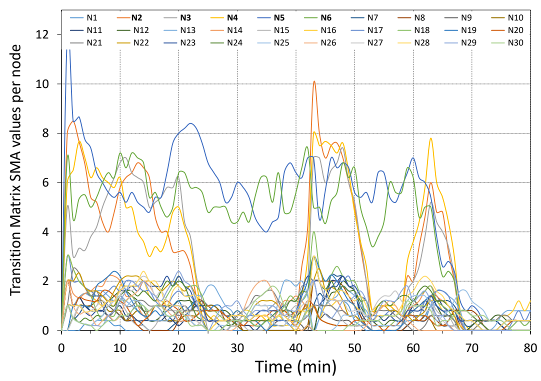
To evaluate the efficiency of *MODE* algorithm, we use the *Mobility Condition Discovery Success Ratio (mSR)* metric, expressing the ratio of nodes with a successful identification on their mobility condition at a specific point in time. In particular, at time t in a network of N nodes, we calculate the *mSR* value based on Equation (16). The mobility pattern is reflected on vertex $P_t = \{n_1, n_2, ..., n_N\}$, where $n \in N$ describes the *actual* mobility condition for each node, specifically $n = 0$ for static and $n = 1$ for moving nodes. The result of *MODE* algorithm is represented by vertex $D_t = \{d_1, d_2, ..., d_N\}$, where $d \in N$ expresses the *detected* mobility condition for each node, with $d = 0$ for static and $d = 1$ for moving nodes.

$$mSR_t = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 1 \; if \; P_t(i) = D_t(i) \\ 0 \; otherwise \end{cases} \tag{16}$$

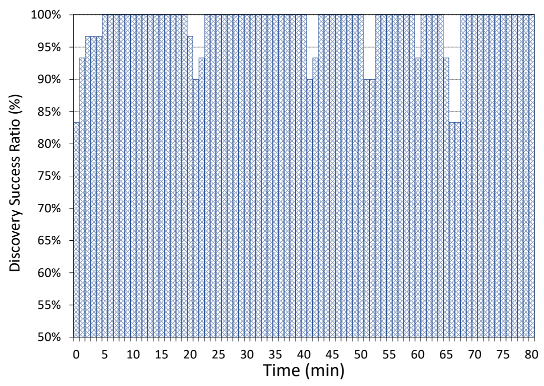
The *mSR* results are presented as percentages in Fig. 29c for all network nodes and in Fig. 29d for the five mobile ones. We clarify that *MODE* starts with the assumption that all nodes are static and its first execution run is placed at time



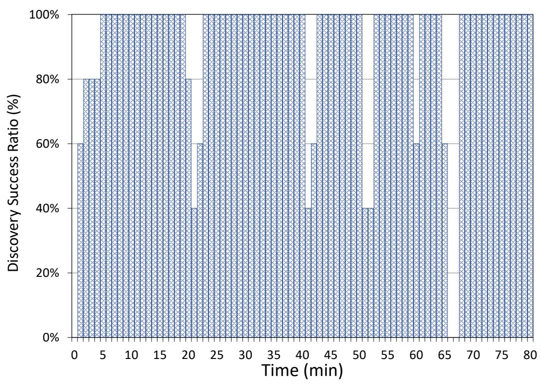
(a) Mobility scenario timeline for 30 nodes



(b) Transition Matrix average values using Simple Moving Average



(c) Discovery Success Ratio for All nodes



(d) Discovery Success Ratio for Mobile nodes

Figure 29: Simulation results of *MODE* algorithm in the *water-park* use-case scenario, using *SMA* on Transition Matrices series with window size 5

0. From the first two samples, *MODE* starts detecting mobility in the network and at the 6^{th} run (i.e., at 5 *min*, as we sample every minute), the algorithm detects the mobility condition of all nodes successfully. During the first stop of nodes 2, 3 and 4, *MODE* needs 3 *min* to realize the change, while it takes 2 *min* to detect that they start moving again, i.e., at 40 *min*. We observe a similar behavior during the second stop and start of the same nodes, as well as at 65 *min*, i.e., when all nodes stop moving. Furthermore, we note the absence of false-positives for any of the fixed nodes.

We now investigate the effect of *SMA* window parameter on the algorithm's efficiency. We are aiming to experimentally optimize this parameter, based on the *Overall Mobility Condition Discovery Success Ratio (SmSR)* metric. In detail, the latter represents the overall success ratio for all nodes, considering the total duration of the simulation. We calculate *SmSR* as the average of all mSR_t values, as shown in Equation (17). We execute *MODE* in *TTRt* time intervals and $S = \{t_0, t_1, ..., t_r\}$ contains all execution times t (i.e., $t_1 = t_0 + TTRt$), with r its total number.

$$SmSR = \frac{1}{r}\sum_{i=0}^{r} mSR_{S_i} \tag{17}$$

In Fig. 30, we illustrate the results of five simulations with different *SMA* window sizes $n = 1, 3, 5, 10$, and 15. The $n = 5$ window size produces the best result, with $SmSR = 98.3\%$. Whereas window values lower than 5 make *MODE* very sensitive to minor changes, i.e., producing false positives, while values above 5 jeopardize its capability to rapidly detect changes in a node's mobility condition.

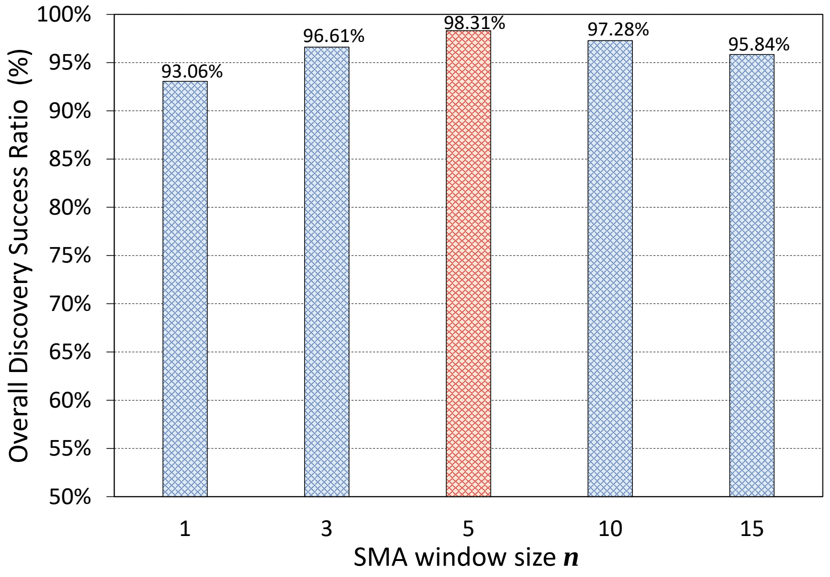


Figure 30: Overall Mobility Condition Discovery Success Ratio (SmSR) for different window values n in SMA Transition Matrices

Finally, we clarify that *MODE* success ratio is highly related to its alignment to the topology refresh rate of the protocol. As in Algorithm 3, we suggest that *MODE* execution interval should follow the protocol's topology refresh rate, which in this evaluation is 1 *min*. Although this topology discovery rate is challenging for the majority of IoT protocols, we argue that it is feasible for *SD-MIoT*, because of the hybrid operation of *MOB-TD*.

Immediately after, broadening the study on the effectiveness of the solutions proposed in this dissertation, we proceed with the evaluation of the MINOS multi-protocol platform.

5.5 Evaluating Routing Control in Heterogeneous Network environments

We evaluated MINOS for handling mobility and heterogeneity in an IoT experimentation setup; for example, we assume a smart city scenario similar to the one described in the next subsection, where static nodes are located on buildings or stations, co-existing with mobile ones being placed in vehicles (all equipped with sensors offering city-monitoring facilities, e.g. temperature, humidity, noise, pollution).

Through MINOS GUI we can proactively deploy our protocols (i.e. either the *CORAL-SDN*, or the *Adaptable-RPL*). Once one of them is deployed each time, we progress with extracting results regarding two metrics: (i) the *PDR* (i.e. the proportion of packets delivered against total packets sent), and (ii) the *control overhead* (i.e. ratio of control packets to the total packets).

In our comparative analysis, we use the *default RPL* as a baseline protocol, that is to contrast the MINOS platform against traditional IoT deployments.

5.5.1 Use-case scenario: Smart-Cities

This section discusses a representative use-case, i.e., the the smart city scenario. Briefly, the smart city scenario is characterized by devices' heterogeneity and mobility issues, as well as a requirement for elastic network operation.

A smart city is a large ecosystem with a high number of connected subsystems and components. It is characterized by a wide range of communication challenges, such as *heterogeneity* in terms of device type (e.g., mobile, wireless and sensor networks coexist), *mobility* behavior (e.g., humans, drones or vehicles are characterized by diverse mobility patterns) or application requirements (e.g., delay-sensitive or real-time); and *elasticity* requirements (e.g., dynamic establishment of routing paths in response to topology changes due to mobility). We argue that an SDN-like platform can provide the necessary abstractions to support individual protocol configurations per node and realize adaptive communication strategies mitigating the above issues.

In the scenario illustrated in Fig. 31, a variety of heterogeneous wireless IoT-devices (i.e., temperature, water and light sensors or flame detectors) are moni-

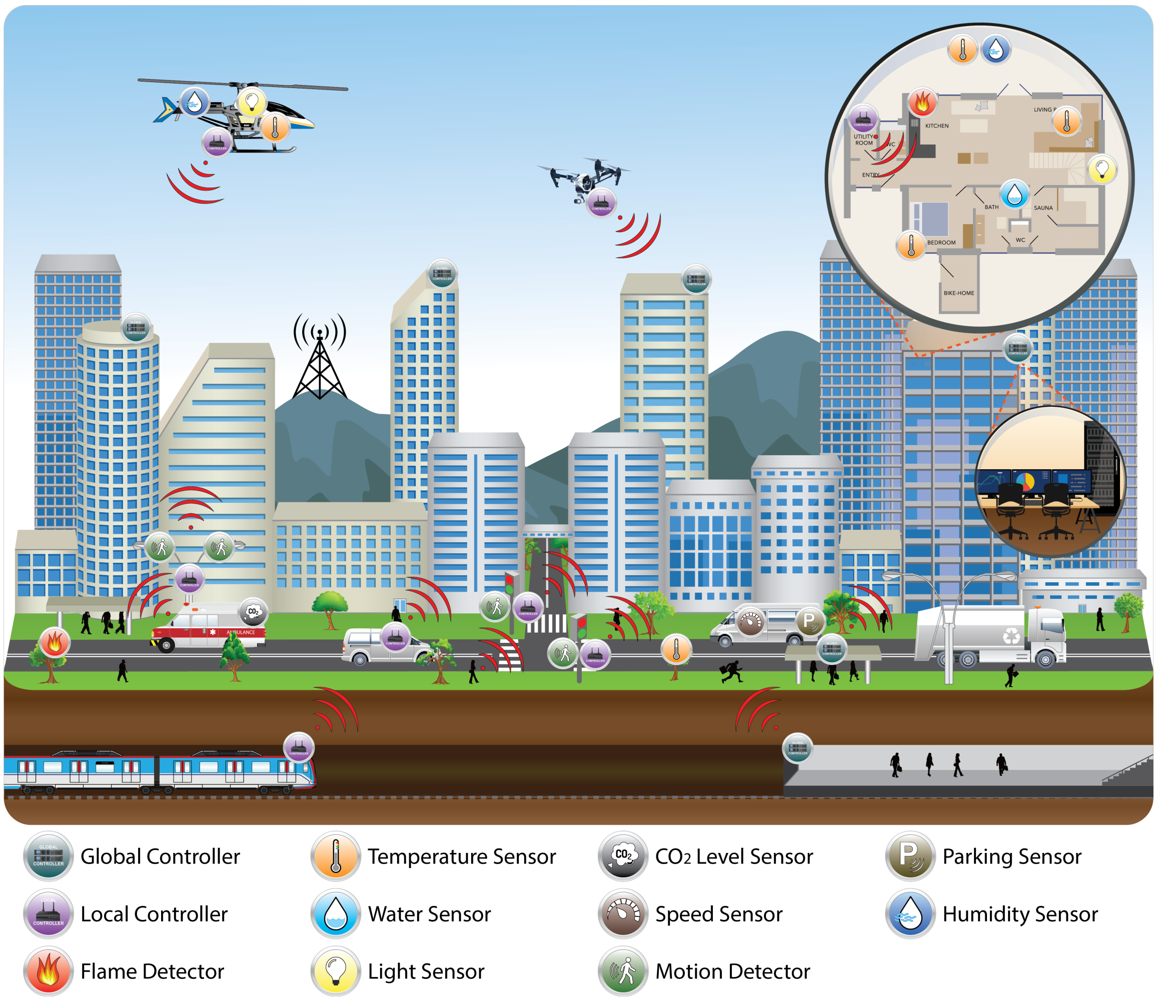


Figure 31: The smart-city use-case

toring a (smart) apartment. Each wireless node can operate as an end-point or a forwarding device. Once a local controller in each apartment gathers and processes the data, it sends a message about the apartment's conditions to the global controller. The latter resides at a central control room and is responsible for the entire building. In case, for example, the global controller receives a crucial message about a flame or a smoke detection, indicating the possibility of blaze, it transmits a prioritized message triggering the necessary actions (e.g., activating an alarm for the building evacuation).

Other examples can be: a) a drone hosting a local controller is flying around to collect data from fixed or mobile IoT sensors about the city's conditions or incidents (e.g., weather, traffic, noise, pollution, car accidents); and b) human wearables, sensors embedded to smartphones or vehicles moving with diverse mobility patterns and being connected to multiple networks along their way. The local con-

Table 10: Experimentation Setup

Layer Setting	**Description**	**Notes**
Transport	UDP	Packet size 60 B
Network	IPv6/Rime	RPL/CORAL-SDN
Adaptation	6LoWPAN	
MAC	CSMA	
Physical	IEEE 802.15.4	Channel 26
	Radio Duty Cycle	128 Hz
IoT Motes	Zolertia Z1	Transceiver 2.4 GHz
OS	Contiki-OS	ver 3.0
Simulation	Cooja	
TX/RX	100%	Reliable Radio
Traffic load	Mobile: 120 data pckt/h	
	Fixed: 6 data pckt/h	
Mobility Model	real traces [121]	Canvas 750×750 m
Duration	1 h	

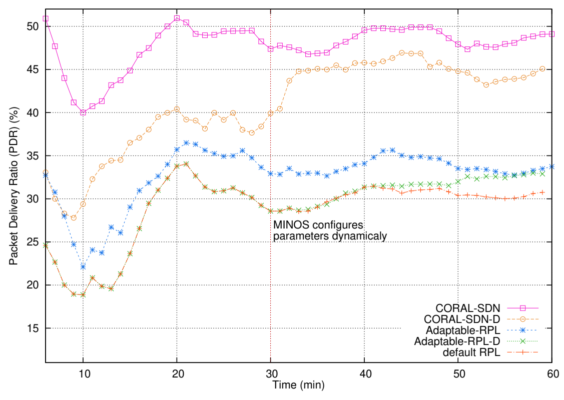
trollers communicate with a global controller which in turn considers the detected network conditions, according to the "big picture", and enforces the necessary communication strategy, e.g., prioritizes crucial information to reach to a central station.

We highlight in the next subsections that MINOS protocol can accommodate a number of relevant IoT network configurations through dynamically adjusting its parameters either per network or device. Moreover, we emphasize on the research challenges that emerge in each case and should be tackled by a relevant management platform.

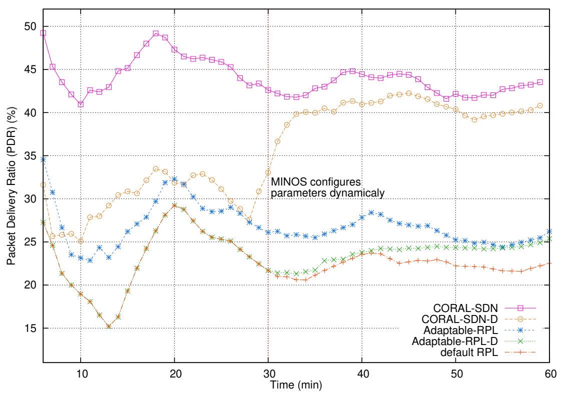
5.5.2 Methodology & Metrics

The protocols, accommodated in the *Data Communication* plane of the MINOS platform, are evaluated in a network with 21 IoT nodes (1 sink, 5 mobile, 15 static nodes), exploiting real-traces extracted from Stockholm city buses' routes, available via the MONROE project [121]. All our experiments are deterministic, hence we do not need to evaluate the results' statistical accuracy with multiple experiment runs. The combination of Cooja emulator scalability issues and the available processing power (we used an Intel(R) Core(TM) i5-2410M, 2.30 GHz processor), along with real-traces order of magnitude limitations, allows experimentation of this scale. Table 10 summarizes all experiments' settings.

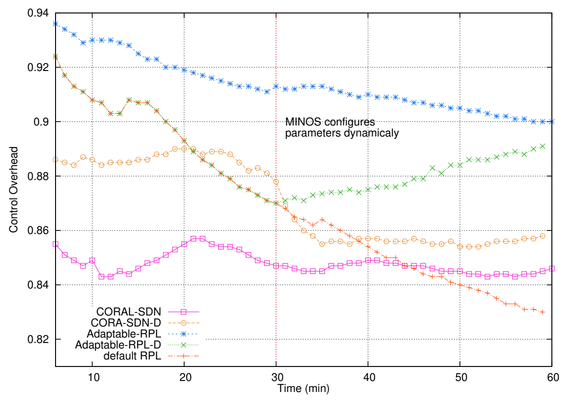
Methodologically, we conducted the same scenario five times as follows: (i) the



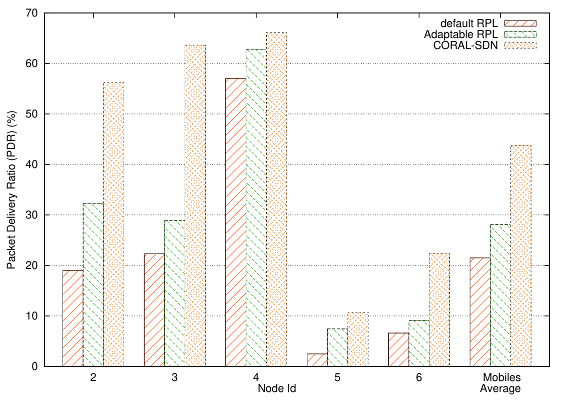
(a) *PDR* as a function of time for all nodes



(b) *PDR* as a function of time for the mobile nodes



(c) *Control overhead* as a function of time



(d) *PDR* for mobile nodes and mobiles' average

Figure 32: MINOS experimental results for *PDR* and *control overhead*

first run employs the *default RPL* to provide a "ground truth" curve (red line in Figs. 32a-32c graphs); (ii) the next two runs consider the *Adaptable-RPL*, where the MINOS either Dynamically configures its parameters on-the-fly (via the UPIs) at the 30 *min* (results are indicated by the *Adaptable-RPL-D* green curve), or it adapts its configuration parameters from the beginning of the experiment (blue curve), and; (iii) the last two runs provide results for the *CORAL-SDN*, where the MINOS again either Dynamically adapts its parameters at the 30 *min* (*CORAL-SDN-D* purple curve), or it configures the protocol when the experiment starts (orange curve). The MINOS interventions' impacts are clearly shown in Figs. 32a-32c, where after 30 *min* both the dynamically adapted scenarios (i.e. the *Adaptable-RPL-D* and *CORAL-SDN-D*), are progressively converging to the *Adaptable-RPL* and *CORAL-SDN* respectively, which in turn, are constantly superior to the *default RPL* which keeps the default parameters (i.e. $I_{min} = 12$ and $I_{doubling} = 8$). For the *Adaptable-RPL-D* scenario, after the 30 *min*, the *Control and Monitoring* RPL component of MINOS triggers only the sink and fixed nodes to look in a more frequent basis for disconnected mobile nodes, that is tuning $I_{min} = 8$ and $I_{doubling} = 0$. Similarly, for the *CORAL-SDN-D* scenario, also after 30 *min*, the corresponding component employs the *neighbor request* topology control algorithm that considers the neighbor requests [31], and configures topology maintenance parameter's rate at 4 *sec* for the mobile nodes and 10 *min* for the static ones. These last values could be adjusted by an intelligent algorithm or configured by the administrator to match the typical mobility patterns of buses. The results demonstrate the positive impact of dynamically tuning the aforementioned parameters through the MINOS platform on the *PDR*. Even in low density networks, there is a trade-off between *PDR* and *control overhead*, reflected differently in the two protocols.

5.5.3 Evaluation Results & Discussion

Figures 32a-32c show the performance of *default RPL*, *Adaptable-RPL*, and *CORAL-SDN* protocols, in terms of *PDR* and *control overhead*. We use these graphs with the time parameter on x-axis to demonstrate the ability of the MINOS to dynamically configure its protocols. An important observation derived by all sub-figures is that our platform and protocols outperform the default routing protocol regarding the *PDR*; especially in the case of the mobile nodes, achieves an improvement up to 7.74% for the *Adaptable-RPL* and 19.4% for the *CORAL-SDN* in the whole network (Fig. 32a), which rises up to 8.15% for the *Adaptable-RPL* and 21.5% for the *CORAL-SDN* for the mobile nodes (Fig. 32b). This outcome also highlights the benefits of offloading the *control overhead* to the static nodes. Since the mobility pattern for the nodes $2-6$ is a moving buses' emulation, there are long time periods of no connectivity for the mobile nodes because of radio limitations. This explains the fact that *PDR* does not exceed 50% in Fig. 32b. Another interesting outcome is that our platform can achieve better *PDR* with relatively small *control overhead* (Fig. 32c), in case of employing the *CORAL-SDN* protocol. Fig. 32d presents the average PDR (over a period of 60 *min*) for each mobile node when the *Adaptable-RPL* and *CORAL-SDN* are used to tune their mobility-aware parameters from the beginning of the experiments. The bars clearly show that the mobile nodes can be potentially double advanced by the MINOS treatment compared to the standard RPL handling (e.g. nodes $2, 3$).

In general, the MINOS platform by selecting the *CORAL-SDN* protocol [31] achieves a high *PDR* with a marginally worse *control overhead*. In practice, the advanced topology construction algorithms of the protocol reduce the discovery time, while avoiding to flood the network with control messages. We indicatively found improvements in the *PDR* of the mobile nodes 2, 3 and 6, up-to 37.1%, 41.3%, and 15.7% respectively. This happens because the *CORAL-SDN* succeeds to route messages through their neighboring mobile nodes. However, as previously indicated, this performance is brought with the additional cost of equipping the devices with a separate control channel. This investment is sensible in use-case scenarios requiring reliability in routing crucial messages (e.g. a smoke detection alarm, or critical infrastructure sectors).

Then again, in public heterogeneous networks such as the smart city environment where the extra hardware cost may not be reasonable, low complexity solutions like the *Adaptable-RPL* could be more suitable than the *CORAL-SDN* by steadily offering a higher *PDR* compared to the default RPL protocol (Figs. 32a, 32b), since the mobile nodes remain connected to the topology for longer periods. The improved *PDR* is traded for the increased *control overhead* (Fig. 32c), occasionally tolerated if, for example, the mobile nodes are powered by the hosting vehicle.

In this chapter, we presented an extensive evaluation analysis of the proposed solutions that contributes to confirming the initial goals of this dissertation. In a nutshell, we simulated several realistic scenarios using a plethora of different use-cases and metrics carefully selected to draw objective and accurate conclusions in our analysis. The results show significant improvements in a multitude of network performance metrics that ameliorate the overall network QoS.

In the next chapter, we conclude this dissertation by summarizing its outcomes and contributions. Withal, we enlist ideas and research areas for potential future extensions of our work that arose from the experience and knowledge gained through this endeavor.

6 Conclusions and Further Work Discussion

6.1 Summary and Conclusions

We are at the dawn of a new era of advanced network management integrating IoT with traditional technologies like WSN and emerging technologies like 5G networks. Notwithstanding, today's IoT applications impel requirements like the catholic demand for ultra-low latency, high bandwidth, and energy efficiency over heterogeneous networks and devices.

In this dissertation, we advocate the idea that the future belongs to platforms that bringing elasticity, intelligence, and central management into mobile and heterogeneous IoT networks, utilizing the SDN paradigm for network control and management. Fusing those technologies is not a straightforward task as new challenges emerge like the excessive use of control messages transmitted over the low power and lossy communication environment.

In Chapter 2, we have described in detail the progress of the research community to date on the above issues, highlighting their strengths and weaknesses and citing the most important work, many of which have been a source of inspiration for us. In Chapters 3, 4, 5, we delve into these problems and propose, design, develop, and evaluate three novel SDWSN solutions that incorporate architectures and mechanisms for IoT networks, which are fully aligned with the SDN model. In detail we have presented respectively:

- *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*.

- *SD-MIoT* solution and its algorithms, mitigating the research issues arisen in the rapidly growing research area of mobile IoT, including the unreliable and high-overhead communication due to the dynamicity in the topology introduced by mobile nodes. *SD-MIoT* addresses these issues through: (i) a timely picture of the network topology in dynamic environments; (ii) reliable forwarding decisions based on mobility-aware routing prioritization; and (iii) blended reactive and proactive flow rule establishment processes. Such features are backed from a novel intelligent *MODE* algorithm, designed to accurately and timely detect nodes under mobility. We validated the robust with low control overhead performance of our solution in challenging realistic Mobile IoT use-cases, as well as the *MODE* detection accuracy. In our understanding, *SD-MIoT* is the first *SDWSN* solution design for mobile IoT environments and since we have released *SD-MIoT* as an open-source, we expect further improvements and experiments from the scientific research community.
- *MINOS* platform, a multi-protocol SDN facility, implementing service-awareness as a feature that amplifies the cardinal role of IoT technology. It stands between revolutionary approaches fully exploiting the SDN paradigm to provide centralized control, and evolutionary ones, which enhance IoT-oriented mechanisms with SDN-inspired functionalities to keep their pros, while moderating their inabilities in terms of elasticity, heterogeneity, and mobility. We adopt the SDN approach to build an architecture which can host multiple IoT protocols while having the functional components to deploy them on-demand according to the application requirements, and configuring them in real-time responding to dynamic network conditions. *MINOS* follows a modular architecture and its planes serve as further experimentation place-holders. Furthermore, our platform can be easily extended in a straightforward manner

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 control decisions based on data inputs from the protocols.

Open and flexible architectures like those discussed in the dissertation 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 our work and results 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 our implementation as an open-source in the Github platform [90], [101], [107], and [86], providing additional technical information and deployment instructions omitted for reasons of clarity form this text.

6.2 Future Work Discussion

Emerging technologies for 5G networks such as Mobile Cloud, Edge and Fog Computing, together with lightweight clouds and network slicing, have a significant impact on future IoT evolution. 5G networks integrate networking, computing and storage resources into one unified programmable infrastructure. This unification allows for optimized and more dynamic usage of server and network resources and the convergence of fixed, mobile, and broadcast services.

Here, we discuss a number of topics that we suggest as future work that can further improve the impact and applicability of the proposed SDWSN frameworks, including energy efficiency, security 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 define a non-exhaustive list of topics that our proposed platforms can accommodate to enable future research:

- *Energy Efficiency:* Energy conservation is an essential feature in the operation of IoT networks. Although at this stage one could argue that our solutions may introduce additional energy consumption because of the usage of two radio channels, research works like [64] and [65] showed that a centralized system with a separate control channel, like those described in this dissertation, 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 SDWSN platforms in large scale network scenarios. Although we have tested our facility with networks of 90 nodes in the context of this dissertation, as we discussed already in Section 3.7 we anticipate the operation of our solutions like *VERO-SDN* engaging in experiments with 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 evident reliable operation of our solutions motivates us to further study its robustness in scenarios with unreliable and challenging communication environments (e.g., IoT for critical infrastructure), by conducting experiments aligned to relevant industrial use-cases.
- *Predictive routing and flow establishment*: This emerging flow establishment technique attempts to proactively set up routing paths and flow rules based on intelligent decision-making modules that utilize the SDN global network view and the *Controller's* excessive computational power. For example, such research endeavors can be a natural evolution of the *SD-MIoT* framework.
- *Mobility trajectory prediction*: We envisage to augment *MODE* functionality with the addition of mobility behavior modeling and relevant pattern extraction, aiming to provide advanced mobility trajectory prediction, as an additional *Controller's* feature.
- *Extensive experimentation*: In the evolution of this work, we plan to perform large scale experiments in realistic test-beds with mobile nodes, including w-iLab.2 [39] and IoT-LAB [122] that both allowing multi-hop deployments.
- *Security*: Last but not least, the general network view of SDWSN can enable the detection of intruders or malicious nodes (e.g., spot a node advertising

signal levels that are significantly different from its neighbors). Furthermore, researchers could implement machine-learning solutions to tackle security issues.

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 supporting a dual-radio channel communication will use this infrastructure terrain as the catalyst for expanding further the integration between 5G and IoT networks. Additionally, the modular architecture of our solutions and the well-designed API interfaces can also enable the latest developments in networks, such as the network data plane slicing.

As a bottom line, we consider the solutions proposed in this dissertation as enabling platforms for research on SDN-like capabilities for IoT devices. Our goal is to keep building on top of them, keep integrating new intelligence modules, and exploit them to enable new unique protocol features.

References

[1] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless Sensor Networks: A survey on recent developments and potential synergies," *The J. of Supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[2] S. Li, L. Da 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, 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, 2017.

[5] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

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

[7] 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. and Elect. Eng.*, vol. 66, pp. 274–287, 2018.

[8] J. Zheng and M. J. Lee, "A comprehensive performance study of ieee 802.15. 4," *Sensor network operations*, vol. 4, pp. 218–237, 2006.

[9] Z. Sheng, S. Yang, Y. Yu, A. V. Vasilakos, J. A. McCann, and K. K. Leung, "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 91–98, 2013.

[10] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comp. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.

[11] IEEE, "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, Apr. 2016.

[12] J. A. Gutiérrez, “On the use of ieee std. 802.15.4 to enable wireless sensor networks in building automation,” *International Journal of Wireless Information Networks*, vol. 14, no. 4, pp. 295–301, 2007.

[13] “IPv6 over Low power WPAN (6lowpan),” Accessed: Jul. 10, 2020. [Online]. Available: https://datatracker.ietf.org/wg/6lowpan/about/

[14] “Routing Over Low power and Lossy networks (roll),” Accessed: Jul. 10, 2020. [Online]. Available: https://datatracker.ietf.org/wg/roll/about/

[15] 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,” RFC 6550, IETF, Tech. Rep., Mar. 2012.

[16] T. Tsvetkov and A. Klein, “Rpl: Ipv6 routing protocol for low power and lossy networks,” *Network*, vol. 59, pp. 59–66, 2011.

[17] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, “A survey on application layer protocols for the internet of things,” *Trans. on IoT and Cloud Comput.*, vol. 3, no. 1, pp. 11–17, 2015.

[18] D. Singh, G. Tripathi, and A. J. Jara, “A survey of internet-of-things: Future vision, architecture, challenges and services,” in *IEEE world forum on Internet of Things (WF-IoT)*. IEEE, 2014, pp. 287–292.

[19] M. Bouaziz and A. Rachedi, “A survey on mobility management protocols in Wireless Sensor Networks based on 6LoWPAN technology,” *Comput. Commun.*, vol. 74, pp. 3–15, 2016.

[20] I. Tomić and J. A. McCann, “A survey of potential security issues in existing wireless sensor network protocols,” *IEEE Internet of Things J.*, vol. 4, no. 6, pp. 1910–1923, 2017.

[21] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, “Energy-efficient routing protocols in wireless sensor networks: A survey,” *IEEE Commun. surveys & tutorials*, vol. 15, no. 2, pp. 551–591, 2012.

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

[23] K. Sood, S. Yu, and Y. Xiang, “Software-Defined Wireless Networks opportunities and challenges for Internet-of-Things: A review,” *IEEE Internet of Things J.*, vol. 3, no. 4, pp. 453–463, Aug 2016.

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

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

[26] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for Low-Power IoT networks," in *4th IEEE Conf. on Netw. Softwarization and Workshops*, Jun. 2018, pp. 71–79.

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

[28] R. C. Alves, D. A. Oliveira, G. A. N. Segura, and C. B. Margi, "The Cost of Software-Defining Things: A Scalability Study of Software-Defined Sensor Networks," *IEEE Access*, vol. 7, pp. 115 093–115 108, 2019.

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

[30] G. Violettas, T. Theodorou, S. Petridou, A. Tsioukas, and L. Mamatas, "An experimentation facility enabling flexible network control for the Internet of Things," in *IEEE Conf. on Comput. Commun. (INFOCOM).* IEEE, 2017, pp. 992–993.

[31] T. Theodorou and L. Mamatas, "CORAL-SDN: A Software-Defined Networking solution for the Internet of Things," in *IEEE Conf. on Netw. Function Virtualization and Softw. Defined Netw. (NFVSDN)*, Nov 2017, pp. 1–2.

[32] T. Theodorou and L. Mamatas, "A Versatile Out-of-Band Software-Defined networking solution for the Internet of Things," *IEEE Access*, vol. 8, pp. 103 710–103 733, Jun 2020.

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

[34] T. Theodorou and L. Mamatas, "SD-MIoT: A Software-Defined Networking Solution for Mobile Internet of Things," *IEEE Internet of Things J.*, 2020, doi: 10.1109/JIOT.2020.3027427.

[35] 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.

[36] "eWINE Elastic Wireless Networking Experimentation grand challenge awards," Accessed: Jun. 10, 2020. [Online]. Available: https://ewine-project.eu/grand-challenge/

[37] C. Fortuna, P. Ruckebusch, C. Van Praet, I. Moerman, N. Kaminski, L. DaSilva, I. Tinirello, G. Bianchi, F. Gringoli, A. Zubow *et al.*, "Wireless software and hardware platforms for flexible and unified radio and network control," in *European Conf. on Netw. and Commun. (EUCNC 2015)*, 2015, pp. 712–717.

[38] 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, 2016.

[39] "IMEC w-iLab.2 testbed," Accessed: Jun. 10, 2020. [Online]. Available: http://wilab2.ilabt.iminds.be

[40] "Tutorial: Softwarized Internet of Things with Lightweight Clouds in Practice," Accessed: Jun. 10, 2020. [Online]. Available: https://nfvsdn2017.ieee-nfvsdn.org/program/tutorials/#tutorial3

[41] B. T. de Oliveira, R. C. A. Alves, and C. B. Margi, "Software-defined wireless sensor networks and internet of things standardization synergism," in *2015 IEEE Conf. on Standards for Commun. and Netw. (CSCN)*. IEEE, 2015, pp. 60–65.

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

[43] B. T. de Oliveira and C. B. Margi, "TinySDN: Enabling TinyOS to Software-Defined Wireless Sensor Networks," *XXXIV Simpósio Brasileiro de Redes de Computadores*, pp. 1229–1237, 2016.

[44] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 1, pp. 1–49, 2013.

[45] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "TinyOS: An operating system for sensor networks," in *Ambient intelligence*. Springer, 2005, pp. 115–148.

[46] 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.

[47] 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. 132 173–132 192, 2019.

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

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

[50] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on Software-Defined Networking," in *27th Biennial Symp. on Commun. (QBSC)*. IEEE, 2014, pp. 71–75.

[51] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks (SDWN): Unbridling SDNs," in *European Workshop on Softw. Defined Netw.*, 2012, pp. 1–6.

[52] 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 *IEEE Conf. on Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 513–521.

[53] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Reprogramming Wireless Sensor Networks by using SDN-WISE: A hands-on demo," in *2015 IEEE Conf. on Comput. Commun. Workshops (INFOCOM WKSHPS)*. IEEE, 2015, pp. 19–20.

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

[55] 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, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618312192

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

[57] 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, 2016.

[58] R. Alves, D. Oliveira, G. Nez, and C. B. Margi, “It-sdn: Improved architecture for sdwsn,” in *XXXV Brazilian Symposium on Computer Networks and Distributed Systems*, 2017.

[59] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki-a lightweight and flexible operating system for tiny networked sensors,” in *29th Ann. IEEE Int. Conf. on Local Comput. Netw.* IEEE, 2004, pp. 455–462.

[60] C. B. Margi, R. C. Alves, G. A. N. Segura, and D. A. Oliveira, “Software-defined wireless sensor networks approach: Southbound protocol and its performance evaluation,” *Open J. of Internet Of Things (OJIOT)*, vol. 4, no. 1, pp. 99–108, 2018.

[61] 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. 96 019–96 034, 2019.

[62] 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. of the 2019 Int. Conf. on Embedded Wireless Syst. and Netw.* Junction Publishing, 2019, pp. 206–207.

[63] I. Haque, M. Nurujjaman, J. Harms, and N. Abu-Ghazaleh, “SDSense: An agile and flexible SDN-based framework for wireless sensor networks,” *IEEE Trans. on Vehicular Technol.*, vol. 68, no. 2, pp. 1866–1876, 2018.

[64] 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 *2015 IEEE 11th Int. Conf. on Wireless and Mobile Comput., Netw. and Commun. (WiMob)*, Oct 2015, pp. 322–328.

[65] R. Piyare, T. Istomin, and A. L. Murphy, “WaCo: A Wake-Up Radio COOJA Extension for Simulating Ultra Low Power Radios,” in *Proc. of the 2017 Int. Conf. on Embedded Wireless Syst. and Netw.* USA: Junction Publishing, 2017, pp. 48–53.

[66] M. Kaplan, C. Zheng, M. Monaco, E. Keller, and D. Sicker, “WASP: A software-defined communication layer for hybrid wireless networks,” in *2014 ACM/IEEE Symposium on Architectures for Netw. and Commun. Syst. (ANCS)*, Oct 2014, pp. 5–15.

[67] C. Gu, R. Tan, X. Lou, and D. Niyato, "One-hop out-of-band control planes for low-power multi-hop wireless networks," in *2018 IEEE Conf. on Comput. Commun. (INFOCOM)*, April 2018, pp. 1187–1195.

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

[69] C. Cobarzan, J. Montavont, and T. Noel, "Analysis and performance evaluation of RPL under mobility," in *IEEE Symp. on Comput. and Commun. (ISCC)*. IEEE, 2014, pp. 1–6.

[70] I. El Korbi, M. B. Brahim, C. Adjih, and L. A. Saidane, "Mobility enhanced RPL for wireless sensor networks," in *3rd Int. Conf. on the Netw. of the Future (NOF)*. IEEE, 2012, pp. 1–8.

[71] A. Oliveira and T. Vazão, "Low-power and lossy networks under mobility: A survey," *Comput. Netw.*, vol. 107, pp. 339–352, 2016.

[72] H. Fotouhi, D. Moreira, and M. Alves, "mRPL: Boosting mobility in the Internet of Things," *Ad Hoc Networks*, vol. 26, pp. 17–35, 2015.

[73] H. Fotouhi, D. Moreira, M. Alves, and P. M. Yomsi, "mRPL+: A mobility management framework in RPL/6LoWPAN," *Comput. Commun.*, vol. 104, pp. 34–54, 2017.

[74] Y. Tahir, S. Yang, and J. McCann, "BRPL: Backpressure RPL for high-throughput and mobile IoTs," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 29–43, 2017.

[75] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[76] M. J. Neely and R. Urgaonkar, "Optimal backpressure routing for wireless networks with multi-receiver diversity," *Ad Hoc Networks*, vol. 7, no. 5, pp. 862–881, 2009.

[77] M. Bouaziz, A. Rachedi, A. Belghith, M. Berbineau, and S. Al-Ahmadi, "EMA-RPL: Energy and mobility aware routing for the Internet of Mobile Things," *Future Generation Comput. Syst.*, vol. 97, pp. 247–258, 2019.

[78] F. Gara, L. B. Saad, R. B. Ayed, and B. Tourancheau, "A new scheme for RPL to handle mobility in wireless sensor networks," *Int. J. of Ad Hoc and Ubiquitous Comput.*, vol. 30, no. 3, pp. 173–186, 2019.

[79] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, “The Trickle Algorithm,” IETF, Tech. Rep. RFC 6206, Mar. 2011. [Online]. Available: https://tools.ietf.org/html/rfc6206

[80] V. de Figueiredo Marques and J. Kniess, “Mobility Aware RPL (MARPL): Providing Mobility Support for RPL Protocol,” in *Anais Principais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, 2019, pp. 211–223.

[81] G. Han, J. Jiang, C. Zhang, T. Q. Duong, M. Guizani, and G. K. Karagiannidis, “A survey on mobile anchor node assisted localization in wireless sensor networks,” *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2220–2243, 2016.

[82] B. P. Santos, O. Goussevskaia, L. F. Vieira, M. A. Vieira, and A. A. Loureiro, “Mobile matrix: Routing under mobility in iot, iomt, and social iot,” *Ad Hoc Netw.*, vol. 78, pp. 84–98, 2018.

[83] B. P. Santos, P. H. Rettore, L. F. Vieira, and A. A. Loureiro, “Dribble: A learn-based timer scheme selector for mobility management in IoT,” in *IEEE Wireless Commun. and Netw. Conf. (WCNC)*. IEEE, 2019, pp. 1–6.

[84] “Zolertia RE-Mote platform,” Accessed: Dec. 10, 2019. [Online]. Available: https://github.com/Zolertia/Resources/wiki/RE-Mote

[85] R. Wallace, “Achieving optimum radio range,” Texas Instruments Incorporated, Dallas, Texas 75265, Tech. Rep. SWRA479A, 9 2017.

[86] “CORAL-SDN open-source software and demo videos,” Accessed: May. 1, 2020. [Online]. Available: https://github.com/SWNRG/coral-sdn

[87] Open Networking Foundation, “SDN architecture 1.1,” 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_Architecture_issue_1.1.pdf

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

[89] “Node-RED,” Accessed: Dec. 10, 2019. [Online]. Available: https://nodered.org

[90] “VERO-SDN open-source software and demo videos,” Accessed: May. 1, 2020. [Online]. Available: https://github.com/SWNRG/vero-sdn

[91] “Contiki OS enhanced with dual-radio features open-source software,” Accessed: May. 08, 2019. [Online]. Available: https://github.com/clovervnd/Dual-radio-simulation

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

[93] "Weka 3: Machine Learning Software in Java," Accessed: Jun. 10, 2020. [Online]. Available: https://www.cs.waikato.ac.nz/~ml/weka

[94] 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, no. 2-3, pp. 153–167, 2002.

[95] J. V. Sobral, J. J. Rodrigues, R. A. 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, 2019.

[96] 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 *2018 24th European Wireless Conf.*, May 2018, pp. 1–6.

[97] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[98] E. Borgia, "The internet of things vision: Key features, applications and open issues," *Comput. Commun.*, vol. 54, pp. 1–31, 2014.

[99] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. Kwak, "The Internet of Things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[100] L. Lan, R. Shi, B. Wang, and L. Zhang, "An iot unified access platform for heterogeneity sensing devices based on edge computing," *IEEE Access*, vol. 7, pp. 44 199–44 211, 2019.

[101] "SD-MIoT open-source software and demo videos," [Accessed: May. 10, 2020]. [Online]. Available: https://github.com/SWNRG/SD-MIoT

[102] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[103] C. D. Kirkpatrick II and R. Julie, "Moving averages," in *CMT Level II The Theory and Analysis of Technical Analysis*. John Wiley & Sons, 2017, ch. 2.

[104] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[105] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*, 4th ed. Morgan Kaufmann, 2016.

[106] "2nd Open Call Scheme of the Wireless Software and Hardware platforms for Flexible and Unified radio and network controL project (WiSHFUL)," Accessed: Jul. 20, 2020. [Online]. Available: http://www.wishful-project.eu/OpenCall2.html

[107] "MINOS open-source software and demo videos," [Accessed: Jun. 18, 2020]. [Online]. Available: https://github.com/SWNRG/minos

[108] P. Ruckebusch, S. Giannoulis, D. Garlisi, P. Gallo, P. Gawlowicz, A. Zubow, M. Chwalisz, E. De Poorter, I. Moerman, I. Tinnirello *et al.*, "Wishful: Enabling coordination solutions for managing heterogeneous wireless networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 118–125, 2017.

[109] "Ansible is Simple IT Automation Platform," Accessed: Jun. 10, 2020. [Online]. Available: https://www.ansible.com/products/automation-platform

[110] S. A. Seidel *et al.*, "Analysis of large-scale experimental data from wireless networks," in *IEEE Conf. on Comput. Commun. Workshops*, Apr. 2018, pp. 535–540.

[111] 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, 2016.

[112] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *1st IEEE Int. Workshop on Practical Issues in Building Sensor Netw. Appl. (SenseApp)*, 2006.

[113] "6LoWPAN Troubleshooting with Foren6," Accessed: Apr. 10, 2020. [Online]. Available: https://cetic.github.io/foren6

[114] G. Almes, S. Kalidindi, and M. Zekauskas, "A One-Way Delay Metric for IP Performance Metrics (IPPM)," IETF, Tech. Rep. RFC 7679, Jan. 2016. [Online]. Available: https://tools.ietf.org/html/rfc7679

[115] P. Nain, D. Towsley, B. Liu, and Z. Liu, "Properties of random direction models," in *Proc. IEEE 24th Ann. Joint Conf. of the IEEE Comput. and Commun. Societies*, vol. 3. IEEE, 2005, pp. 1897–1907.

[116] "Olympus Marathon," [Accessed: Feb.15, 2020]. [Online]. Available: https://www.olympus-marathon.com

[117] "Olympus Mythical Trail," [Accessed: Dec.10, 2019]. [Online]. Available: https://www.omt100.com

[118] Google-maps, “Olympus mountain, Mytikas pick $2,919m$, Greece,” [Accessed: Jun. 10, 2020]. [Online]. Available: https://goo.gl/maps/Udbq2fmX6HE1Td9P9

[119] “Olympus Trails on GPSies.com,” [Accessed: Jul. 24, 2019]. [Online]. Available: https://www.gpsies.com/

[120] Google-maps, “Waterland water park, Thessaloniki, Greece,” [Accessed: Jun. 10, 2020]. [Online]. Available: https://goo.gl/maps/J3orcXAEw3wScCubA

[121] O. Alay *et al.*, “MONROE: Measuring mobile broadband networks in Europe,” in *IRTF & ISOC Workshop on Res. and Appl. of Internet Measurements*, 2015.

[122] “IoT-LAB,” Accessed: Jun. 10, 2020. [Online]. Available: https://www.iot-lab.info

Appendices

A Funding

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.

B Implementation Insights

B.1 Control Plane Class Diagrams

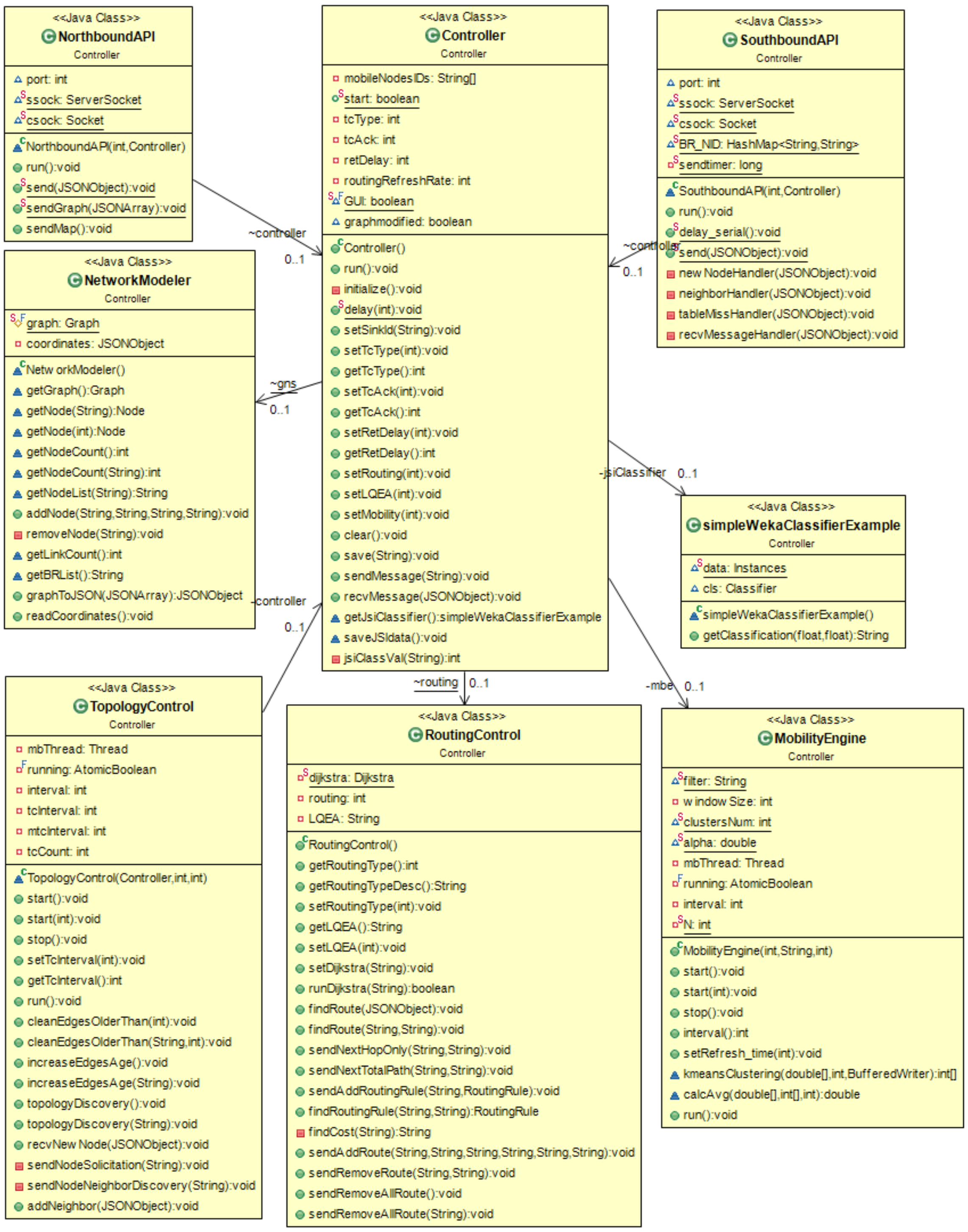


Figure B.1: VERO-SDN Controller Class Diagram

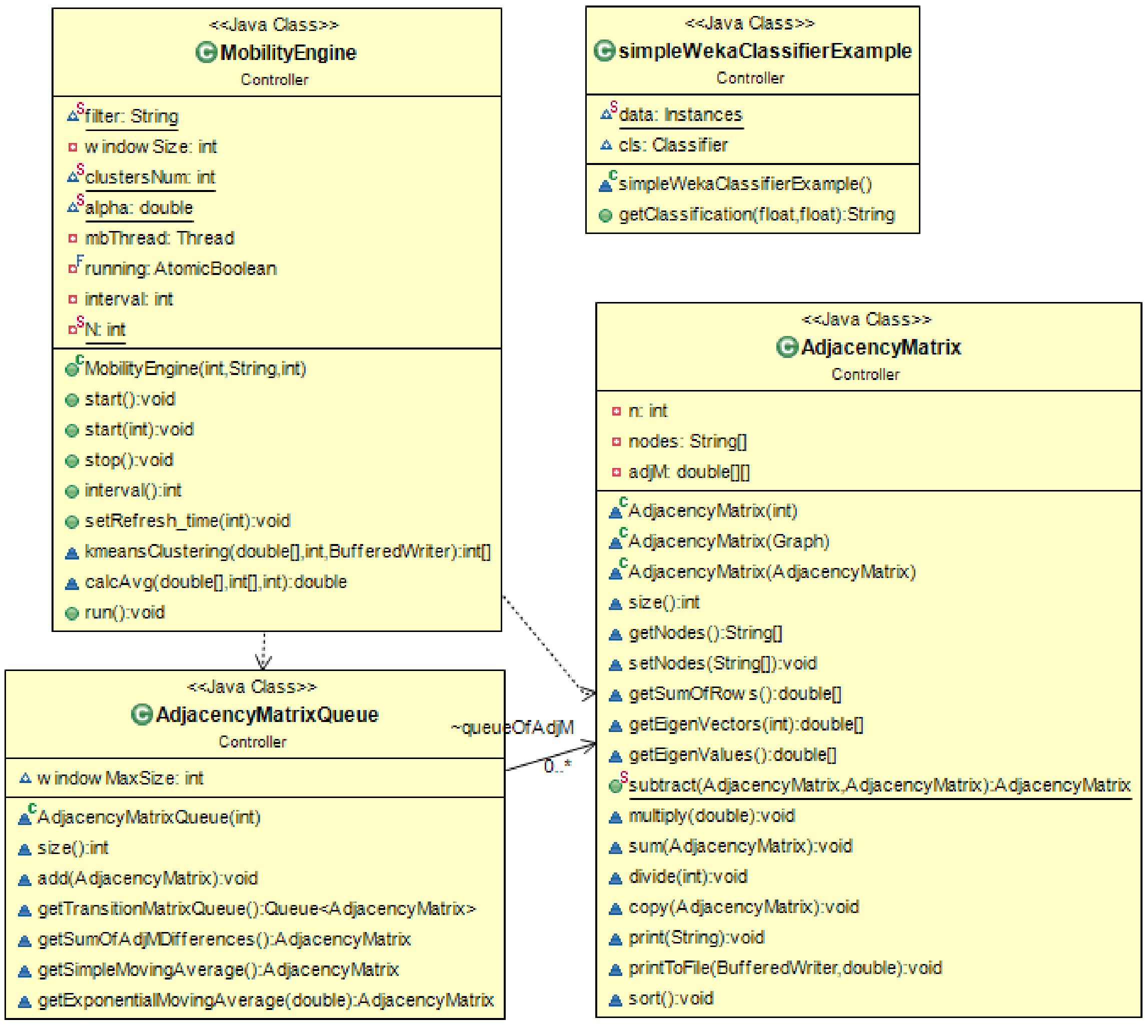

<<Java Class>>
MobilityEngine
Controller
filter: String
windowSize: int
clustersNum: int
alpha: double
mbThread: Thread
running: AtomicBoolean
interval: int
N: int
MobilityEngine(int,String,int)
start():void
start(int):void
stop():void
interval():int
setRefresh_time(int):void
kmeansClustering(double[],int,BufferedWriter):int[]
calcAvg(double[],int[],int):double
run():void
<<Java Class>>
simpleWekaClassifierExample
Controller
data: Instances
cls: Classifier
simpleWekaClassifierExample()
getClassification(float,float):String
<<Java Class>>
AdjacencyMatrix
Controller
n: int
nodes: String[]
adjM: double[][]
AdjacencyMatrix(int)
AdjacencyMatrix(Graph)
AdjacencyMatrix(AdjacencyMatrix)
size():int
getNodes():String[]
setNodes(String[]):void
getSumOfRows():double[]
getEigenVectors(int):double[]
getEigenValues():double[]
subtract(AdjacencyMatrix,AdjacencyMatrix):AdjacencyMatrix
multiply(double):void
sum(AdjacencyMatrix):void
divide(int):void
copy(AdjacencyMatrix):void
print(String):void
printToFile(BufferedWriter,double):void
sort():void
<<Java Class>>
AdjacencyMatrixQueue
Controller
windowMaxSize: int
AdjacencyMatrixQueue(int)
size():int
add(AdjacencyMatrix):void
getTransitionMatrixQueue():Queue<AdjacencyMatrix>
getSumOfAdjMDifferences():AdjacencyMatrix
getSimpleMovingAverage():AdjacencyMatrix
getExponentialMovingAverage(double):AdjacencyMatrix
~queueOfAdjM
0..*

Figure B.2: SD-MIoT Mobility Detector Class Diagram

B.2 Southbound API Data-Plane Packet Descriptions

Table 11: Protocol Packet Fields - Description

Field Code	Description	Data Type	Size (bytes)
PTY	Packet Type	String	2
BID	Border Router Address	Address	4
NID	Node Address	Address	4
NBR	Neighbor Address	Address	4
TCT	Topology Control Algorithm Type	Unsigned Short Integer	1
ACK	Acknowledgement request	Unsigned Short Integer	1
RET	Control Message Retransmission Delay	Unsigned Short Integer	1
RSS	Radio Signal Strength Indicator	Integer	4
LQI	Link Quality Indictor	Integer	4
ENG	Node Battery Energy	Integer	5

Table 12: Southbound API Messages with Examples

Message Type	**Code**	**Size (bytes)**	**Payload Fields (as in Table 11)**	**Packet Example**
New Border Router Solicitation Request	BR	2	PTY(2)	{PTY:BR}
New Border router registration	BR	6	PTY(2), BID(4)	{PTY:BR, BID:01.00}
New Node Solicitation Request	NN	6	PTY(2), BID(4)	{PTY:NN, BID:01.00}
New Node response	NN	15	PTY(2), BID(4), NID(4), ENG(5)	{PTY:NN, BID:01.00, NID:0X.00 ,ENG:12345}
Neighbor Request TC-NA	ND	9	PTY(2), BID(4), TCT(1), ACK(1), RET(1)	{PTY:ND, BID:01.00, TCT:1, ACK:1, RET:2}
Neighbor Request TC-NR	ND	13	PTY(2), BID(4), NID(4), TCT(1), ACK(1), RET(1)	{PTY:ND, BID:01.00, NID:0X.00, TCT:1, ACK:1, RET:2}
Neighbor Response	NB	27	PTY(2), BID(4), NID(4), NBR(4), RSS(4), LQI(4), ENG(5)	{PTY:NB, BID:01.00, NID:0X.00, NBR:0Z.00, RSS:94, LQI:105, ENG:13002}
Missing Forwarding Rule	MR	14	PTY(2), BID(4), NID(4),DID(4)	{PTY:MR,BID:01.00,NID:03.00,DID:01.00}
Add Forwarding Rule	AD	23	PTY(2), BID(4), NID(4), DID(4), NXH(4), CST(3), SEQ(2)	{PTY:AD,BID:01.00,NID:02.00,DID:01.00, NXH:01.00,CST:107,SEQ:00}
Replace Forwarding Rule	AR	23	PTY(2), BID(4), NID(4), DID(4), NXH(4), CST(3), SEQ(2)	{PTY:AR,BID:01.00,NID:02.00,DID:01.00, NXH:01.00,CST:107,SEQ:00}
Remove Specific Forwarding Rule	RM	14	PTY(2), BID(4), NID(4),DID(4)	{PTY:RM,BID:01.00,NID:09.00,DID:03.00}
Remove All Forwarding Rule Records Unicast	RM	14	PTY(2), BID(4), NID(4),DID(4)	{PTY:RM,BID:01.00,NID:09.00,DID:00.00}
Remove All Forwarding Rule Records Broadcast	RM	14	PTY(2), BID(4), NID(4),DID(4)	{PTY:RM,BID:01.00,NID:00.00,DID:00.00}