SD-MIoT: A Software-Defined Networking Solution for Mobile Internet of Things

Article *in* IEEE Internet of Things Journal · September 2020

DOI: 10.1109/JIOT.2020.3027427

2 authors:

Tryfon Theodorou
University of Macedonia
20 PUBLICATIONS **122** CITATIONS
SEE PROFILE

Lefteris Mamatas
University of Macedonia
100 PUBLICATIONS **882** CITATIONS
SEE PROFILE

SD-MIoT: A Software-Defined Networking Solution for Mobile Internet of Things

Tryfon Theodorou, *Member, IEEE,* and Lefteris Mamatas, *Member, IEEE*

***Abstract*—The Internet of Things (IoT) brings new potentials in humans' everyday life activities through enabling applications with stringent communication demands, including mobile IoT applications. State of the art routing protocols for the IoT, such as IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), have not been originally designed for such challenging applications. Recent proposals blend the Software-Defined Networking (SDN) paradigm with IoT, enabling better-informed and bespoke routing control, matching the application requirements.**

In this paper, we propose *SD-MIoT*, an open-source SDN solution for mobile IoT environments that consists of a modular SDN controller and an OpenFlow-like protocol. *SD-MIoT* detects passively in real-time the network's mobile nodes through *MODE*, an intelligent algorithm that utilizes the connectivity graph of the SDN controller. It incorporates novel mobility-aware topology discovery mechanisms, routing policies and flow-rule establishment methods, all of them balancing control overhead with routing robustness, according to nodes' mobility behavior detected by *MODE*. We provide extensive evaluation results over two 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.

***Index Terms*—Mobile Internet of Things, Software-Defined Wireless Sensor Networks, Mobility, Mobility Detection**

I. Introduction

INTERNET of Things (IoT) is characterized by explosive growth, changing our world rapidly. IoT systems and devices define a huge area of innovation, allowing people to develop and design products, even at home [1]. A special type of Low-Power and Lossy Networks (LLN), the *Wireless Sensor Networks* (WSNs), are becoming nowadays a key enabling technology for IoT, introducing a new range of networking applications integrated to the traditional Internet infrastructure [2]. WSNs connect tens or hundreds of tiny wireless network devices, called motes, at the edge of conventional network infrastructures, capable of measuring and interacting with the physical world through sensors and actuators.

However, WSN protocols, such as Routing over Low Power and Lossy Networks (RPL) [3], are mainly focusing on applications with a many-to-one communication between sensor devices and measurement sinks. Consequently, they appear inadequate for IoT and their emerging applications, characterized by a diverse range of communication patterns and performance requirements. The transition of WSN to the IoT brings a number of challenges, e.g., interoperability, mobility, heterogeneity, and Quality of Service (QoS) [4], [5]. *Mobility* is prominent as a *conditio 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 [6], [7]. Such scenarios are characterized by constant changes in their physical topology, constituting challenging environments for network routing protocols. Bouaziz et al. [8] identify, among others, three challenges that mobile WSN protocols must transcend for a sufficient operation: (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.

The need for applications with diverse, potentially stringent performance requirements calls for flexible and performance-efficient network protocols driven from sophisticated, low-overhead network control features. In this context, *Software-Defined Networks* (SDNs) [9] are employing flexible, logically-centralized network architectures that decouple network control from the data plane. Although SDNs and the prominent OpenFlow protocol [10] were introduced for infrastructure networks, they also bring significant advantages to mobile and wireless networks [11], including resource optimization for dynamic conditions, automation in its operations, granular control and policy management, innovation and openness. Furthermore, SDN is considered as a typical enabler for mobility handling in different contexts, including Mobile [11], Access [12], [13], and Vehicular networks [14].

Recent research endeavors [15], [16] are blending SDN architectures with WSN technologies, forming a new approach called *S*oftware-*D*efined *W*ireless *S*ensor *N*etworks (*SDWSN*). The SDWSN paradigm brings new prospects to WSN architecture, control, management, and operation [17], enabling: (i) improved WSN routing and topology control protocols, utilizing a global network view; (ii) offloading network control intelligence from motes to a central controller, i.e., reducing the resource-demands of constrained IoT devices; and (iii) programmability features allowing for efficient, adaptable network protocol operation to the particular requirements of IoT applications. Here, we argue and demonstrate that SDWSN provides the ground for novel mobility handling mechanisms that exploit abthove capabilities. For example, an SDN controller may detect the mobility behavior of nodes and adapt its routing strategies, accordingly.

However, SDN poses additional challenges not present in the conventional WSN networks, including the increased amount of control packets that impairs the low quality of radio communication. To this end, our SDWSN framework *CORAL-*

The authors are with the Department of Applied Informatics, University of Macedonia, 156 Egnatia Street, GR-546 36 Thessaloniki, Greece (e-mails: {theodorou, emamatas}@uom.edu.gr).

SDN, initially presented in the demo paper [18], suggests novel centralized topology discovery mechanisms [19]. Our recent work, *VERO-SDN* [20], demonstrates with adequate analysis and results an SDWSN framework that achieves reduced control overhead, maintaining efficient routing and forwarding decisions with low end-to-end communication delays, utilizing an out-of-band control communication channel. Nevertheless, none of the available SDWSN solutions in our knowledge, considers WSN scenarios with mobility characteristics.

Along these lines, we propose the *SD-MIoT* framework 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 [21] designed and implemented utilizing our earlier experience in SDWSN protocol design and development, gained from *VERO-SDN* framework [20]. 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 DEtector (MODE)* algorithm, introduced 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 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 [22]) 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.

We demonstrate the entire *SD-MIoT* solution and validate the impact of its novel mechanisms through an extensive evaluation based on scenarios inspired by realistic use-cases. Briefly, our proposal is compared against RPL, the state of the art WSN routing protocol, in a variety of realistic simulation scenarios, utilizing both real and synthetic human mobility traces. Our results highlight the improvements of *SD-MIoT* in terms of robust packet delivery and reduced network control overhead. An initial investigation of *SD-MIoT* features is presented as part of the Multi-Protocol Software-Defined Networking Solution for the Internet of Things (MINOS), which is a multi-protocol enabling platform for the IoT [23], illustrating limited early experimental results, contrasting such features with RPL and its mobility-friendly extension *Adaptable RPL* [24], [25]. Moreover, we conduct a proof-of-concept experimentation analysis confirming the accuracy of *MODE* algorithm in detecting node mobility.

The remainder of this paper is organized as follows: in Section II we propose two realistic MIoT use-case scenarios, emphasizing on the impact of our work; Section III gives an overview of related WSN protocols addressing mobility issues and SDWSN solutions; Section IV details *SD-MIoT's* components, algorithms, and its basic operation; Section V provides an extensive evaluation through a series of simulated scenarios, illustrating the results and achievements of our research proposal; and, finally, Section VI concludes the paper, while also discussing our next steps.

II. Use-case Scenarios

To further motivate our proposal, we discuss two representative use-cases, i.e., the *extreme sports* and the *smart amusement park*, while emphasize on the research challenges that emerge in each case. In Section V, we evaluate through simulations the efficiency of *SD-MIoT* and its mechanisms, in terms of handling most of these challenges.

A. Extreme sports use-case

Extreme sports are activities that involve a high level of physical exertion combined with uncontrollable factors like the weather and terrain, including mountains, snow, water, and wind. Such high-risk sports require safety measures and precautions to reduce the probability of accidents, which unfortunately may occur in physically isolated areas with low proximity to medical facilities. Consequently, exchanging timely information between the injured party and the rescuers is of paramount importance, e.g., geo-location information. Here, we assume the *Olympus Mythical Trail* [26], a unique trail running event of about $100km$ length and $6700m$ altitude to overcome. 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. 1 depicts on a Google map a 40 min duration and 1800 m distance of real mobility trace data [26], recorded at the highest altitude and most dangerous part of the Olympus Mythical Trail race in 2016. 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.

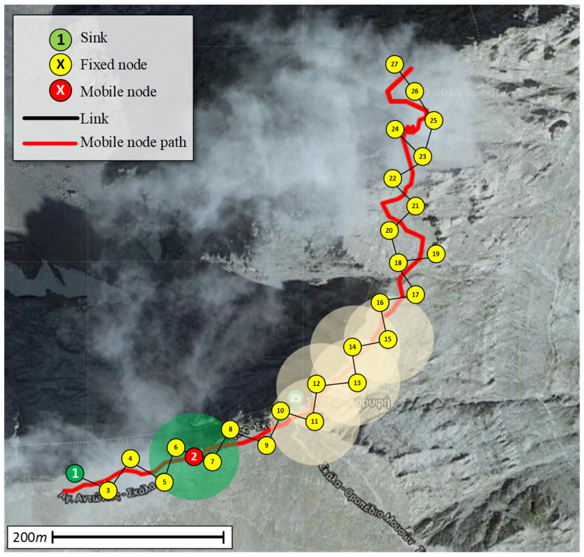


Fig. 1. Olympus Marathon Mythical Trail [26] *extreme sports* scenario – Linear topology with 1 border router, 1 mobile, and 25 fixed nodes.

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. Although our mobility handling mechanisms could also be supported by geo-location capabilities, we consider such features as means to produce application-layer data only. *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.

B. Smart 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. 2, we consider a *smart amusement 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.

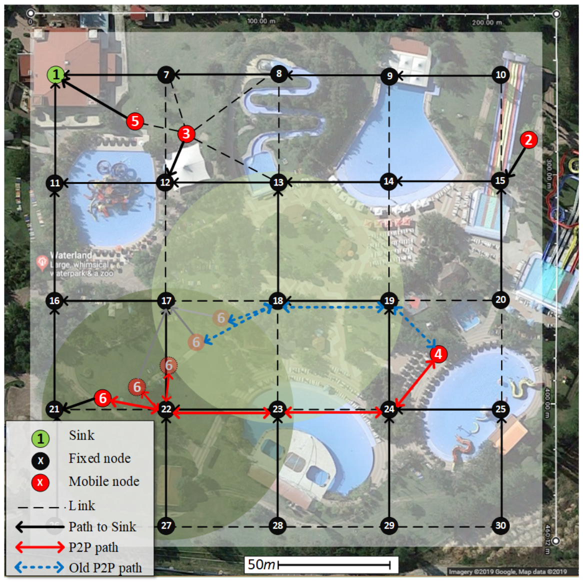


Fig. 2. *Smart amusement park* scenario – Grid topology with 1 border router, 5 mobile, and 24 fixed nodes.

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, demanding network reliability and robustness; (ii) customer service improvement through predictive analytics applications, require centralized control and processing power; and (iii) proximity marketing applications, involving 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-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. 2, 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. Our flow establishment capabilities consider the quality of communication among potential nodes rather than physical distance, which could sometimes be misleading (e.g., choosing close nodes with an obstacle between each other rather than more distant nodes with better radio conditions).

III. Related Works

RPL [3] is a de-facto routing solution for constrained devices and WSNs. However, it was not originally designed for mobile environments [27], [28], such as those discussed in the previous section. The majority of studies aiming to improve RPL performance in mobile topologies suggest 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 [29], [27].

In Table I, we present five important RPL-based solutions that are relevant to our proposal, including the core contributions of each one of them. Such proposals can be interoperable with RPL. However, although they improve performance to some extent, its further improvements are hindered from the orientation of RPL to fixed networks. As such, they preserve architectural features that do not fit well to mobile network conditions, like the topology control trickle algorithm [30]. Moreover, tweaking RPL parameters towards mobile topology requirements inflates the frequency of node solicitation control messages and intensifies the updates of the forwarding rules, i.e., increasing control overhead for mobile environments.

There is also a recent trend towards the adoption of the SDN paradigm in IoT. In Table I, we give the main contributions of eleven important SDN-based frameworks, including two conceptual proposals [36], [35] highlighting requirements for SDN-based IoTs. The main issues of such works are: (i) the challenges of high-complexity and high-overhead that SDN architecture brings to WSNs; and (ii) the reduced reliability because of multi-hop control-message transmissions over resource-constrained devices.

All prior frameworks advance the idea of integrating the SDN paradigm to WSNs in terms of providing new relevant architectures, protocols and control mechanisms. Nevertheless, none of them is evaluated in mobile environments or attempts to tackle network mobility issues (i.e., besides the conceptual platform [36]), although many attempt to tackle control overhead issues, including [20], [42], [38]. Considering the advantages that SDN paradigm brings to WSNs, we propose a mobility-aware platform and relevant mechanisms that target WSN mobility scenarios in the prospect of the IoT era. To our knowledge, *SD-MIoT* is the only SDN solution for WSNs, that proficiently handles network mobility issues. In the next section, we elaborate on the relevant system and its mobility-handling mechanisms.

An additional issue is the prerequisite knowledge of the mobile nodes in a network, assumed from many of the above solutions. To this end, we identify four different strategies to 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 [29] or utilizes additional network control messages, in real-time, to adjust each node's protocol operation according to its type [25]. 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 [44]. 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 [45], e.g., the trickle or reverse trickle timer [28]. 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* [46] 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.

IV. The SD-MIoT Framework

In this section, we begin with an overview of *SD-MIoT* functional framework and architectural planes, followed by a detailed description of the proposed mobility management mechanisms that improve network operation and performance.

A. Architecture Overview

Fig. 3 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 [47], it consists of the following planes, described bottom-up as follows:

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* [20] dual network stacks for *control* and *data communication* channels, offering standardized low-power wireless communication and media access control through the IEEE

TABLE I
RELATED WORKS AND SOLUTIONS

Solution	**Main contribution**	**RPL based**	**SDN based**	**Address mobility**
mRPL+ [31]	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.	✓		✓
[32]	Operates a dual routing protocol, which can adaptively switch between RPL and *Backpressure* routing protocol, depending on network conditions, to improve throughput and node mobility support.	✓		✓
EMA-RPL [33]	Introduces a proactive process able to anticipate and predict the movement of mobile nodes by comparing the radio signal strength to its point of attachment and apply an energy-efficient replacement strategy.	✓		✓
[34]	Excludes message-forwarding though mobile nodes by configuring them as leaf nodes, enhances trickle timer and adopts the preferred parent election metric to improve the mobile node disconnection periods.	✓		✓
Adaptable-RPL [25], [24]	Configures RPL protocol parameters according to node mobility behavior by an SDN-inspired centralized Controller based on *monitor-decide-configure* closed-loops.	✓		✓
Sensor OpenFlow [35]	Presents a conceptual OpenFlow-based protocol, targeting key technical challenges for its core components, including a mitigation of control messages overhead.		✓	
[36]	Recapitulates on the benefits that SDN brings to WSNs, including enhanced network management, advanced topology discovery, and improved sensor node mobility handling.		✓	✓
SDWN [37]	Identifies requirements that SDN solutions need to contemplate in a WSN context, e.g., Radio Duty-Cycling (RDC), in-network data aggregation, and flexible definition of rules, to overcome challenges like control messages overhead, and energy efficiency.		✓	
SDN-WISE [38]	Employs stateful routing tables and proactive routing decisions to reduce interactions with the controller and improve the flow-control decisions, attempting to overcome the identified challenges mentioned above.		✓	
TinySDN [39]	Implements a TinyOS-based SDN architecture with a distributed control plane of multiple controllers for WSN.		✓	
Spotled [40]	Enhances *TinySDN* with a hierarchy of physically distributed global and local SDN controllers, aiming the reduction of control messages.		✓	
Soft-WSN [41]	Improves topology control of WSNs through SDN management policies that can be modified in run-time, i.e., to deal with dynamic requirements.		✓	
AtomicSDN [42]	Utilizes a Synchronous Flooding time-sliced mechanism, i.e., allocating designated flooding periods for the control messages that separates SDN control from data plane messages, improving network latency, reliability, and energy efficiency.		✓	
SDWSN6Lo [43]	Operates a centralized routing protocol based on the shortest path algorithm, utilizing an altered IPv6 addressing scheme and packet format to avoid fragmentation overheads.		✓	
VERO-SDN [20]	OpenFlow-like protocol improving the quality of communication supporting alternative topology discovery and flow establishment mechanisms and reduced control overhead through inherent protocol support of a long-range control channel.		✓	

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 our recent research work [20], i.e., *SD-MIoT* builds up on *VERO-SDN's* robust and with reduced overhead network control capabilities. The latter are utilized from novel mobility handling mechanisms. The firmware of IoT devices is implemented in C programming language for Contiki OS v3.0 [48]. We selected to use Contiki because: (i) it is open-source, well documented and provides an RPL implementation; (ii) it is frequently used in relevant research initiatives; and (iii) it supports both real experimentation and simulations.

2) *Control plane* maintains the *SD-MIoT Controller* that manipulates an 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 the Received Signal Strength Indicator (RSSI) and the Link Quality Indicator (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 a hierarchy of border-router 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 [49]), 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 IV-C.

The *SD-MIoT planes* communicate through the northbound and southbound API using JSON messages, which detailed technical specifications are provided in [21].

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*.

B. *SD-MIoT 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

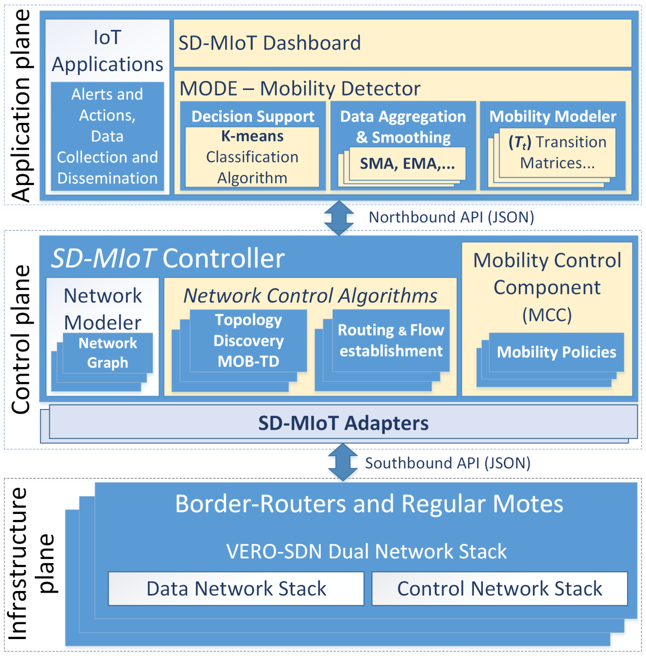

Application plane
IoT Applications
Alerts and Actions, Data Collection and Dissemination
SD-MIoT Dashboard
MODE – Mobility Detector
Decision Support
K-means Classification Algorithm
Data Aggregation & Smoothing
SMA, EMA,...
Mobility Modeler
(T_t) Transition Matrices...
Northbound API (JSON)
Control plane
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

Fig. 3. *SD-MIoT* architecture

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.

1) Mobility Topology Discovery policy: This policy aims at a mobility-friendly topology monitoring maintaining a vivid and up-to-date representation of the network topology, notwithstanding the dynamic changes imposed by the moving nodes. It is implemented from the *Mobility Topology Discovery (MOB-TD)* algorithm, which realizes 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 1, we depict the main steps of *MOB-TD* operation. *MOB-TD* utilizes interchangeably the topology discovery algorithms mentioned in the previous subsection (i.e., *TC-NA* and *TC-NR*). 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 (1).

$$TTRt = \frac{60 \cdot TRt}{TTRr} \tag{1}$$

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 subsection IV-C.

Algorithm 1: *MOB-TD* – Mobility Topology Discovery

```
Input: $TRt$ - global topology refresh time
Input: $TTRt$ - local (mobile) topology refresh time
Output: $G$ - Network Connectivity graph structure
1  while true do
2      $sleep(TTRt)$; // wait for $TTRt$ seconds
       // retrieve list of mobile nodes from MODE (Algorithm:2)
3      $mobN \leftarrow$ MODE$(G, 5)$;
4      if $isExpired(TRt)$ then // global topology discovery
5          $G \leftarrow$ TC_NA(); // call TC-NA [19]
6      else // local topology discovery for mobile nodes
7          foreach element $m$ in $mobN$ do
8              $G \leftarrow$ TC_NR$(m)$; // call TC-NR [19]
9          end
10     end
11 end
```

2) Mobility Routing Prioritization policy: 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 Section V-B. This policy favors network installations where the majority of nodes are fixed, like the two use-case scenarios described in Section II. 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 Section VI.

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

Proactive routing protocols, including RPL [3], prepare the full topology graph in advance. They operate efficiently and achieve low end-to-end delays, when the data are routed to known destinations, making them suitable for static networks. However, in dynamic environments, although they can actively refresh their routing tables by adjusting protocol timers, the protocol overhead is substantially increased. On the other hand, *Reactive* routing control (e.g., [10] or [50]) activates a path or next-hop decision process each time an unknown destination is detected in a transmission, matching the characteristics of dynamic networks better. However, there is a need for a balance between outdated forwarding or routing rules reducing network PDR and the frequent rule changes increasing control overhead.

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 Section V, 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 subsection, a novel mechanism capable of identifying the moving nodes in a network.

C. SD-MIoT Mobility Detector

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. 3), i.e., as snapshots of the network's connectivity graph data-structure.

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. 4. 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. 3) 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 (2).

$$A_{t(i,j)} = \begin{cases} 1 \ \forall i, j \in V \ \textit{if } i, j \textit{ are connected (edge)} \\ 0 \ \textit{otherwise} \end{cases} \tag{2}$$

To designate the network connectivity changes, we monitor a series of the n most recent *Adjacent matrices* $\{A_{t-(n-1)}, ..., A_{t-1}, A_t\}$. 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* $\{T_{t-(n-2)}, ..., T_{t-1}, T_t\}$, as the subsequent subtractions in absolute values of the n *Adjacent matrices*, in pairs of consecutive times, as in Equation (3).

$$\begin{aligned} T_t &= \|A_t - A_{t-1}\| \\ T_{t-1} &= \|A_{t-1} - A_{t-2}\| \\ &\vdots \\ T_{t-(n-2)} &= \left\|A_{t-(n-2)} - A_{t-(n-1)}\right\| \end{aligned} \tag{3}$$

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. 4, 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 (4).

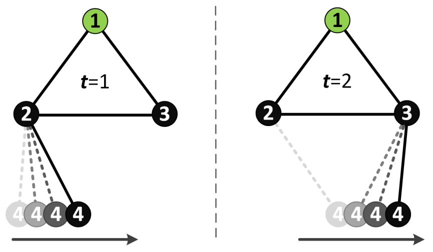


Fig. 4. Simple mobility detection scenario example.

$$T_2 = \left\| \overset{A_{t=2}}{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & \mathbf{0} \\ 1 & 1 & 0 & \mathbf{1} \\ 0 & \mathbf{0} & \mathbf{1} & 0 \end{bmatrix}} - \overset{A_{t=1}}{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & \mathbf{1} \\ 1 & 1 & 0 & \mathbf{0} \\ 0 & \mathbf{1} & \mathbf{0} & 0 \end{bmatrix}} \right\| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & 0 \end{bmatrix} \tag{4}$$

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. 3), i.e., applying intelligent data analysis and data-mining classification methods in *Transition matrices'* data, respectively.

In Algorithm 2, we detail in a unified pseudocode the main algorithms of 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 (3), 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 [51]. In our case, we apply *SMA* on the latest n observed *Transition matrices* in queue T, as in Equation (5).

$$T_{SMA(t)} = \frac{1}{n} \sum_{i=0}^{n-1} T_{(t-i)} \tag{5}$$

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 V-A, 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 (6).

$$sumT_{SMA(x)} = \sum_{i=0}^{N-1} T_{SMA(x,i)} \tag{6}$$

The main *Decision Support* module algorithm, described in lines 13 to 20 of Algorithm 2, detects the mobile nodes using the K-means clustering algorithm [52]. 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.

Algorithm 2: *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          $T.addNewItem(\lvert A[t] - A[t-1] \rvert)$;
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 $\lvert avg(C[1]) - avg(C[2]) \rvert < 1$ 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
```

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 (7).

$$\begin{aligned} C_1 &= \{\{x\} \mid \forall x \in sumT_{SMA}, (x-c_1)^2 \leqslant (x-c_2)^2\} \\ C_2 &= \{\{x\} \mid \forall x \in sumT_{SMA}, (x-c_2)^2 < (x-c_1)^2\} \end{aligned} \quad (7)$$

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 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 (8), 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.

$$\left\| \frac{1}{n_1}\sum_{i=1}^{n_1} C_{1(i)} - \frac{1}{n_2}\sum_{k_2=1}^{j} C_{2(j)} \right\| < 1 \quad (8)$$

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 2 employs two-dimensional array calculations, we omit the second dimension from the pseudocode, for simplicity.

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 [21] using the Java programming language. For the K-means clustering, we integrated the WEKA machine-learning library [53] in our solution.

V. Evaluation

In this section, we provide our extensive evaluation analysis that highlights 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 a real human mobility trace as well as synthetic based on the *random waypoint* (RWP) mobility model [54], all considering the discussed use-cases. In both of them, we consider multiple speeds which are typical for a human, i.e., the real speeds incorporated in the mobility trace and randomly picked out of a Uniform distribution at the range of $[0, 5]$ km/h, respectively.

We elaborate on our evaluation analysis in two subsections. In subsection V-A, 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. Whereas in subsection V-B, 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*.

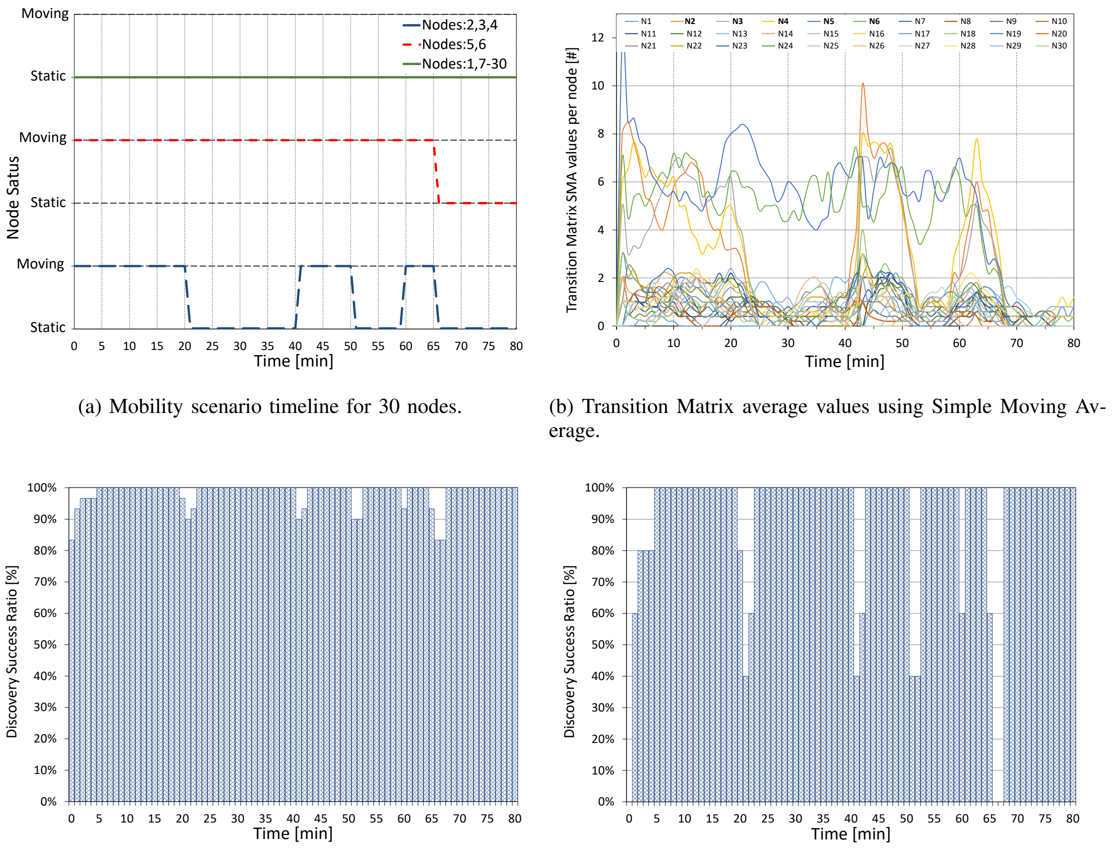
A. Mobility Detector Evaluation

We investigate the effectiveness of *MODE* algorithm through evaluating its feasibility in terms of mobility detection accuracy, particularly focusing on its operational characteristics isolated from factors that affect the network connectivity, e.g., external radio interference. We conduct simulations aligned to the *smart amusement park* use-case scenario (i.e., Section II-B) and considering a network of 30 nodes with one border router, 24 fixed nodes, 5 mobile nodes, and the parameters defined in Table II. The 80 min duration of our experiment deemed appropriate to evaluate *MODE's* detection accuracy with the following mobility behavior: the mobile nodes are moving according to the RWP model and follow a predefined mobility pattern with starts and stops, as shown in Fig. 5a. The nodes move at multiple speeds, i.e., $[0, 5]$ km/h. 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). Our experiments utilize the Zolertia Z1 mote, because it is resource-constraint (i.e., matching the requirements of LLN), well-supported by Contiki OS and frequently used in relevant research works.

TABLE II
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	[21]
Mobility model	Random Way Point	speed $[0, 5]$ km/h
Mobility pattern	5 mobile, 25 fixed	Figure 5a
	Duration	80 min
Simulation	OS / Simulator	Contiki Cooja [48]

We initially justify the validity of *Transition Matrices*, as the primary criterion for identifying the moving nodes. In Fig. 5b, we depict a chart with the aggregated values of $sumT_{SMA}$ per node over time. As described in Section IV-C, these values represent the total connectivity changes for each node, emerging from Equation (6), 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.



(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.

Fig. 5. Simulation results of *MODE* algorithm in the *smart amusement park* use-case scenario, using *SMA* on Transition Matrices series with window size 5.

The combined result of Fig. 5b and 5a 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 (9). 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.

$$mSR_t = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 \; if \; P_t(i) = D_t(i)\\ 0 \; otherwise\end{cases} \tag{9}$$

The mSR results are presented as percentages in Fig. 5c for all network nodes and in Fig. 5d 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 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 (10). 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.

$$SmSR = \frac{1}{r}\sum_{i=0}^{r} mSR_{S_i} \tag{10}$$

In Fig. 6, 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. Since data aggregation and smoothing mechanisms may affect the algorithm's efficiency, in the near future we plan to investigate the impact of alternative moving average techniques and their configurations on its operation, including the *Exponential Moving Average*.

Finally, we clarify that *MODE* success ratio is highly related to its alignment to the topology refresh rate of the protocol. As in Algorithm 1, 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*. Also, we can further fine-tune *MODE* efficiency by testing different recurrence intervals; nevertheless, this is not part of this paper's context.

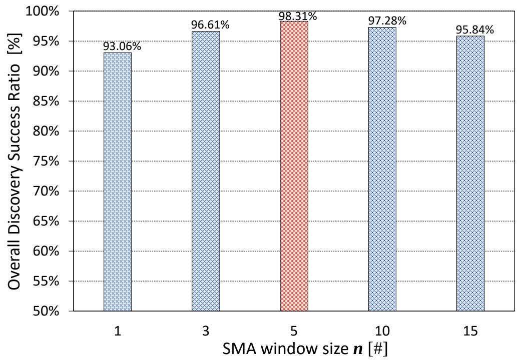


Fig. 6. Overall Mobility Condition Discovery Success Ratio (SmSR) for different window values n in SMA Transition Matrices.

B. Routing and Control Performance Evaluation

Here, we investigate the capability of *SD-MIoT* to address the main research challenges defined in Section I. We consider two distinct scenarios corresponding to the use-cases defined in Section II. Our two sets of simulations probe into two main network operation processes, i.e., topology discovery and flow management, unveiling the improvements *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, the state of the art WSN routing protocol. Although RPL primarily targets fixed environments [8], we use it as a point of reference for our results and as a common denominator for comparisons with other works in the future, since the majority of relevant research papers are compared with RPL, e.g., [27], [31], [25].

In Table III, we show the experimental setups of both simulation scenarios. We highlight that the *extreme sports* scenario, as described in Section II-A, considers a linear topology with one mobile node utilizing real traces, and through its simplicity, aims to provide proof-of-concept outcomes. While the *smart amusement park* scenario, as described in Section II-B, 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. Furthermore, we selected these data transmission rates for the following reasons: (i) matching the requirements of both use-cases, so their results can be comparable; (ii) assume a frequent measurement of humans vital signs (e.g., every $30\ sec$) for the extreme sports use-case and infrequent reporting from fixed nodes, e.g., on temperature; and (iii) consider heavy usage of park applications from mobile visitors and less demanding communication of static nodes (e.g., calculating average queue sizes in ticket counters). The duration of our experiments is $40\ min$, matching the size of the mobility trace data we used [26].

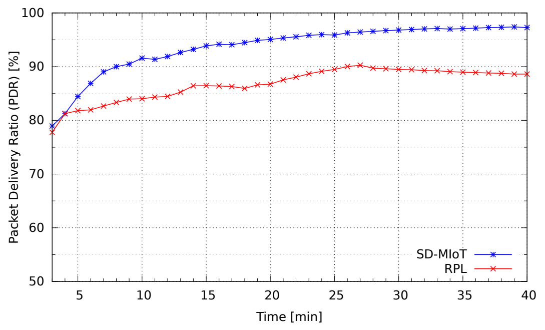
TABLE III
SIMULATION PARAMETERS AND CONFIGURATION SETUPS

Parameters		**Configurations**
Extreme sports	Mobility model	Real Traces [26]
	Linear topology	$500 \times 500\ m$
	Border routers	1 node
	Fixed	25 nodes
	Mobile	1 node
Amusement park	Mobility model	RWP / speed $[0, 5]\ km/h$
	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	OS / Simulator	Contiki Cooja [48]
	Duration	$40\ min$

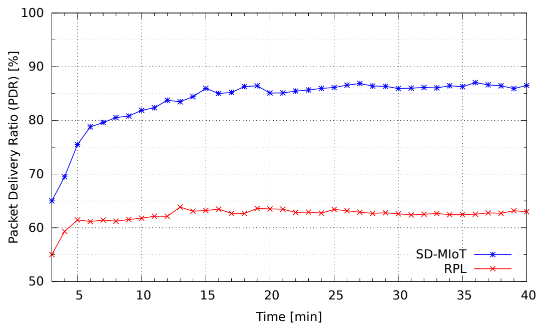
In our experiments, we use two metrics:

1) *Packet Delivery Ratio (PDR)*, to measure *SD-MIoT's* reliability in terms of message delivery success ratio. This metric is calculated as the ratio of *received* data messages R_x over *sent* data messages S_x, as in (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.

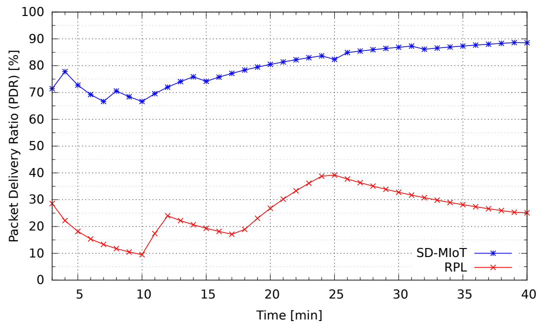
$$PDR = \frac{\sum R_x}{\sum S_x} \tag{11}$$



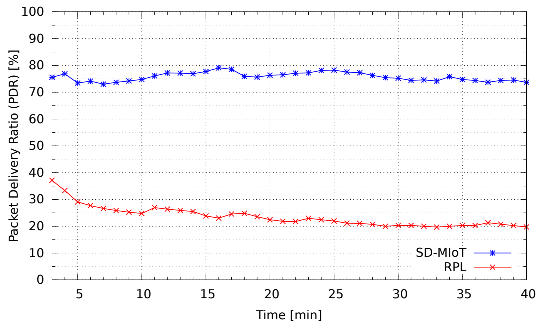
(a) Extreme sports – PDR as a function of time for all nodes.



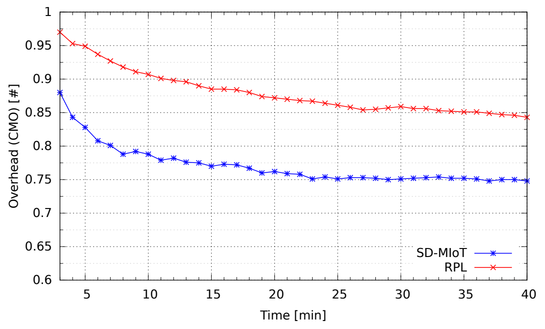
(b) Amusement park – PDR as a function of time for all nodes.



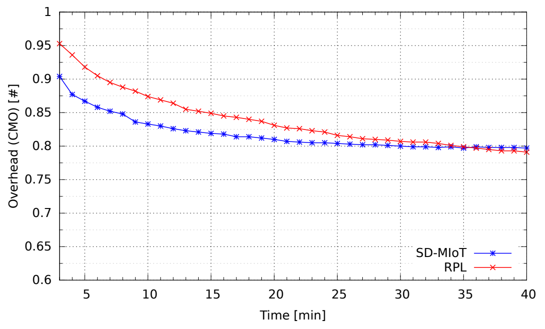
(c) Extreme sports – PDR as a function of time for the mobile node.



(d) Amusement park – PDR as a function of time for the mobile node.



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



(f) Amusement park – Control Overhead as a function of time.

Fig. 7. *Extreme sports* & *Amusement park* use-case results.

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 (12).

$$CMO = \frac{\sum C_x}{\sum T_x} \tag{12}$$

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

In Fig. 7, we illustrate the simulation results of both use-case scenarios. In detail, in Fig. 7a, 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. 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. 7c, 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. 7e, 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 [24]. A relevant comparison between an early version of *SD-MIoT's* and *Adaptable RPL* [23], 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 *smart amusement 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. 7b, 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. 7d, 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 IV-B, we observe that the produced overhead at the end of the simulation is slightly worse than RPL's (Fig. 7f). 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. In our understanding, *SD-MIoT* is the first *SDWSN* solution design for mobile IoT environments. Since we have released *SD-MIoT* as an open-source, we expect further improvements and experiments from the scientific research community.

VI. Conclusions and Further Work

We proposed and evaluated the *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 *MODE* detection accuracy.

We plan to carry out further investigations and advance the capabilities of *SD-MIoT*, including on:

- *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.
- *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, also potentially supported by geo-localization capabilities in the devices.
- *Industrial use*. The evident reliable operation of *SD-MIoT* motivates us to further study its robustness in industrial scenarios with unreliable and challenging communication environments, as well as vehicular or drone scenarios.
- *Extensive experimentation*. We plan to perform large scale experiments in realistic test-beds with mobile nodes, while also contrasting our solution with RPL proposals enhanced with mobility capabilities. Since *SD-MIoT* already required a significant implementation effort, we left this aspect out for future work.

Acknowledgment

This work is partially supported by the H2020 NECOS (grant agreement number 777067) and the open call scheme of the WiSHFUL (grant agreement number 645274) projects.

References

[1] 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.
[2] 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.
[3] T. Winter *et al.*, “RPL: IPv6 routing protocol for low-power and lossy networks,” RFC 6550, IETF, Tech. Rep., Mar. 2012.
[4] I. Yaqoob *et al.*, “Internet of Things architecture: Recent advances, taxonomy, requirements, and open challenges,” *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 10–16, 2017.
[5] 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.
[6] E. Borgia, “The internet of things vision: Key features, applications and open issues,” *Comput. Commun.*, vol. 54, pp. 1–31, 2014.
[7] 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.
[8] 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.
[9] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proc. of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.
[10] 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.
[11] Open Networking Foundation, “Openflow-enabled mobile and wireless networks,” Open Netw. Found., Palo Alto, CA, USA, Tech. Rep., Sep. 2013. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/03/sb-wireless-mobile.pdf
[12] S. Bera, S. Misra, and M. S. Obaidat, “Mobi-flow: Mobility-aware adaptive flow-rule placement in software-defined access network,” *IEEE Trans. on Mobile Comput.*, vol. 18, no. 8, pp. 1831–1842, 2018.
[13] H. Li, P. Li, and S. Guo, “Morule: Optimized rule placement for mobile users in sdn-enabled access networks,” in *2014 IEEE Global Commun. Conf.* IEEE, 2014, pp. 4953–4958.
[14] I. Yaqoob *et al.*, “Overcoming the key challenges to establishing vehicular communication: Is SDN the answer?” *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 128–134, 2017.
[15] 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.
[16] 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.

[17] 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.
[18] 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.*, Nov. 2017, pp. 1–2.
[19] 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.*, Nov. 2017, pp. 236–241.
[20] 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.
[21] "SD-MIoT open-source software and demo videos," [Accessed: May. 10, 2020]. [Online]. Available: https://github.com/SWNRG/SD-MIoT
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] "Olympus Trails on GPSies.com," [Accessed: Jul. 24, 2019]. [Online]. Available: https://www.gpsies.com/
[27] 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.
[28] 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.
[29] 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.
[30] 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
[31] 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.
[32] 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.
[33] 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.
[34] 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.
[35] 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.
[36] 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.
[37] 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.
[38] 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.
[39] 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.
[40] 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.
[41] 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.
[42] 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.
[43] F. F. Jurado-Lasso, K. Clarke, and A. Nirmalathas, "A Software-Defined Management System for IP-Enabled WSNs," *IEEE Syst. J.*, pp. 1–12, Oct. 2019.
[44] 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.
[45] 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.
[46] 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.* IEEE, 2019, pp. 1–6.
[47] 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
[48] 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.
[49] 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.
[50] 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.
[51] 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.
[52] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.
[53] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*, 4th ed. M. Kaufmann, 2016.
[54] 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.

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

Lefteris Mamatas (Member, IEEE) is currently an Assistant Professor with the Department of Applied Informatics, University of Macedonia, Greece. He leads the Softwarized and Wireless Networks Research Group (http://swn.uom.gr), University of Macedonia. He has worked as a Researcher with the University College London, U.K., the Democritus University of Thrace, Greece, and DoCoMo Eurolabs, Munich. He has participated in many international research projects, such as NECOS (H2020), FED4FIRE+ OC4 (H2020), WiSHFUL OC2 (H2020), MONROE OC2 (H2020), Dolfin (FP7), UniverSELF (FP7), and Extending Internet into Space (ESA). He has published more than 80 articles in international journals and conferences. His research interests include Software-Defined Networks, the Internet of Things and 5G Networks.