A Multi-Protocol Software-Defined Networking Solution for the Internet of Things

Tryfon Theodorou, George Violettas, Polychronis Valsamas, Sophia Petridou, and Lefteris Mamatas

The authors propose MINOS, a multi-protocol SDN platform for IoT that implements service awareness utilizing appropriate SDN abstractions and interfaces for logically centralized network control of diverse and resource-constrained IoT environments, network protocols that are deployable and configurable on demand, and a GUI that provides a bespoke dashboard and a real-time visualization tool.

Digital Object Identifier:
10.1109/MCOM.001.1900056

Abstract

IoT has evolved from an experimental to a backbone technology able to connect myriads of people, things, and services for a large range of businesses. At the same time, the emergence of SDN can ideally handle IoT challenges for elasticity, heterogeneity, and mobility, offering an architecture that abstracts decision making away from the data plane and providing a programmable network facility. Along these lines, we propose MINOS, a multi-protocol SDN platform for IoT that implements service awareness utilizing appropriate SDN abstractions and interfaces for logically centralized network control of diverse and resource-constrained IoT environments, two network protocols that are deployable and configurable on demand, and a 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 dynamic network conditions, support of individual protocol configurations per node, and flexibility to accommodate new protocols and control algorithms. Our results demonstrate MINOS as an enabling platform for two protocols, CORAL-SDN and Adaptable-RPL, which, in comparison with the state-of-the-art IoT routing protocol RPL, improve the packet delivery ratio with relatively small control overhead.

Introduction

Nowadays, Internet of Things (IoT) technology holds a cardinal role as an enabler for a highly diverse set of services in respect to their *requirements*, including extremely high data rates, ultra low latency, low power consumption, large number of connected devices, and high mobility. Paramedics, a typical e-health example, demand high data rates in real time to support live video streaming to hospitals. On the other hand, wide sensors' deployments gathering ground and atmospheric measurements in large areas prioritize scalability issues over data rates. Traffic prioritization is a crucial requirement in harsh working environments that use IoT devices' deployments for safety reasons (e.g., prevent or face accidents); in this case, connectivity is of paramount importance. Finally, applications with mobile IoT devices, such as drones or human wearables, strive for efficient solutions (e.g., neighbor discovery and routing) that handle mobility and take into account constraints such as the remaining battery power.

Relevant applications in the literature [1] 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 [2]. Apparently, there is no single protocol or communication mechanism that can address multi-application requirements hosted by IoT networks. In response to the need for agile and configurable solutions, software-defined networking (SDN) provides a new, elastic network paradigm that can transform the traditional network backbones into flexible service delivery platforms. We define *three IoT research challenges* that SDN can ideally handle.

Elasticity: This is needed to appropriately deploy and configure different network protocols toward satisfying applications' requirements; moreover, to adapt to the network context environment (i.e., responding to an IoT network's feedback) by enforcing strategies for flexible and individual IoT devices' configuration, which improves performance and resource allocation while reducing cost.

Heterogeneity: This is required 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.

Mobility: This is for handling issues raised by IoT devices' mobility and consequent connectivity handovers (e.g., additional *control overhead* to maintain the topology), which become "costly" without suitable dynamic routing adjustments. Furthermore, mobility-aware mechanisms should not overload possible coexisting static nodes.

Contribution

Addressing the aforementioned challenges, we move forward in implementing service awareness, a networking research key requirement. This article presents the MINOS, an SDN platform aimed at providing elasticity for heterogeneous and/or mobile IoT deployments, through the operation and dynamic configuration of different protocols. We experimented with CORAL-SDN, a pure SDN protocol, and Adaptable RPL, which is an SDN-

The authors are with the University of Macedonia.

like protocol. In detail, MINOS introduces the following unique features.

An *SDN-based architecture* decoupling the data from the control plane This is an important design step toward elasticity for the following reasons: It keeps a network's heterogeneity transparent to the control and application planes, employs programmable interfaces for getting cross-layer measurements and enforcing appropriate strategies for adaptable topology and flow control, and implements software controllers providing logically centralized control though reducing management cost and complexity.

Two network protocols that benefit from the SDN-based architecture (demo videos for both protocols are available online [3]):

- CORAL-SDN [4], a software-defined OpenFlow-like protocol introducing adaptive topology control and routing strategies for IoT. CORAL-SDN dynamically enforces adaptive combinations of topology discovery and control algorithms, leveraging a network's elasticity. The impact of on-the-fly decisions is reflected in the results discussion below, demonstrating achievements in terms of packet delivery ratio (PDR) and control overhead.
- Adaptable-RPL [5], an evolutionary extension of the IPv6 Routing Protocol (RPL) for Low-Power and Lossy Networks (LLNs) [6] with improvements for mobile and heterogeneous IoT networks. The MINOS platform handles core RPL parameters both dynamically (i.e., in real time) and individually (i.e., mobile vs. static nodes). The results below show that adaptations in the RPL configuration, which mitigate the mobility issues, can efficiently tune the protocol's performance trade-offs, such as PDR vs. control overhead.
- A graphical user interface (GUI) consisting of a bespoke dashboard and a real-time visualization tool. Such a facility maintains the global network view, providing on-the-fly configuration options, visualizing dynamic topologies, and illustrating real-time measurements. In addition, it can be extended in a straightforward manner through the user interface to support new algorithms, and network protocol parameters and measurements, allowing scalable evolution.

This work is partially inspired by related software-defined wireless sensor networking (SDWSN) approaches, including:

- SDN-WISE [7], a stateful SDWSN solution
- Soft-WSN [8], a platform implementing basic SDN features, that is, topology and device management over application, control, and infrastructure layers
- TinySDN [9], which implements a distributed control plane SDN architecture for a wireless sensor network (WSN)
- Whisper [10], which introduces an SDN controller transmitting routing and scheduling messages compatible to RPL to manipulate its operation

The challenges of high complexity and high overhead that SDN architecture brings to LLNs are the main drawbacks of those works, although a recent proposal, mSDN, [2] moderates this effect by introducing a lightweight SDN framework.

Our CORAL-SDN protocol further improves the network overhead through advanced topology control and routing algorithms, utilizing a separate control channel. Trading performance with cost, these solutions are suitable for network installations with similar requirements, such as the use case discussed in the following section. In contrast to the fully centralized control approaches, the distributed protocol solutions, like [11, 12], suggest extensions and mechanisms improving the operation of RPL under mobility and high data traffic, respectively. Our Adaptable-RPL protocol, harmonized to the latter approach, improves RPL through centralized SDN-like adjustments in the protocol configuration. These solutions retain low complexity, making them suitable for public heterogeneous networks while preserving full compatibility with the RPL protocol; however, they may achieve lower improvements compared to a pure SDN solution, indicated by the results below.

Since the related works are mainly adjusted to specific network scenarios or contexts, we introduce a facility that accommodates and adapts multiple IoT protocols because there in no "single protocol fitting all services" solution, and addresses different application and network requirements through dynamic protocol adaptations (e.g., expressing particular network conditions and device constraints). These two characteristics constitute the main novelties of the MINOS platform. To the best of our knowledge, MINOS is the first software-defined multi-protocol platform for IoT networks [3].

In the following, a motivating use case scenario highlighting the advantages of the proposed platform and protocols is presented in the next section. We then elaborate on the MINOS platform's architecture, while comparative results are presented. Concluding remarks and future work insights are discussed in the final section.

Motivating Use Case Scenario

To motivate our proposal, we discuss a smart city use case, representing an ecosystem with large numbers of interconnected IoT devices ("motes"), characterized by a wide range of communication challenges, such as device types heterogeneity (e.g., coexisting mobile, wireless, and sensor networks), mobility behavior (e.g., humans, drones, and vehicles with different mobility patterns), and application message-exchange patterns (e.g., many-to-one vs. one-to-one communication).

In such a scenario, motes equipped with sensors (i.e., pollution and weather related, along with traffic conditions) monitor the smart city's facilities and infrastructures. Cases include monitoring an urban area for air pollution or a (smart) traffic control system, where a number of wirelessly connected motes collect physical world data, delivering them to a central point for further processing. Each mote can operate as either an endpoint or a forwarding device. A centralized SDN platform can make decisions according to the "global view" and enforce the necessary communication strategies (e.g., to prioritize crucial information to reach its destination on time). Such an SDN-inspired model can increase the response time and network operation reliability. Simultaneously, it contributes to the up-scaling of a smart city infrastructure where new innovative ideas

The MINOS platform implements service awareness by bringing together SDN with IoT technologies. It adapts IoT networks to the application requirements by deploying the appropriate network protocol on demand, while considering the network environment constraints by dynamically configuring the protocol in use.

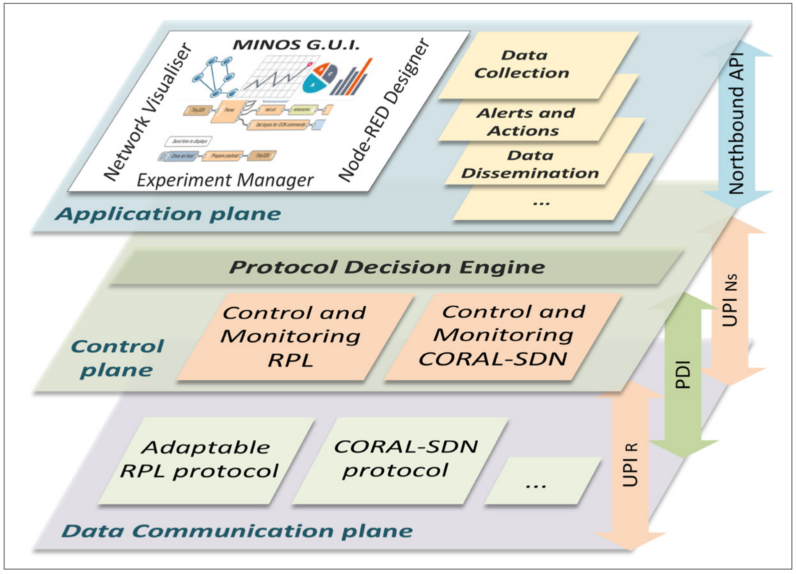


Figure 1. The MINOS architecture.

can be enabled from challenging communication approaches, including:

- A drone flying around to collect data from fixed or mobile IoT sensors about the city's conditions or incidents (e.g., weather, noise, pollution, traffic, car accidents) can safely transmit crucial information.
- Human wearables or sensors embedded to smartphones or vehicles with diverse mobility patterns have to remain seamlessly connected to multiple networks along their way.

Arguably, an SDN-inspired platform, through the necessary abstractions, can support individual protocol deployments and configurations per groups of IoT nodes and realize adaptive communication strategies, mitigating the above challenging communication issues. The smart city use case highlights that there is no single-protocol solution that can address the variety of requirements originated by dissimilar IoT applications. It also highlights the need for a service-aware platform accommodating multiple protocols and their on-demand deployment, along with network conditions' dynamic adaptations. The relevant MINOS solution is detailed below.

The MINOS Platform

The MINOS platform implements service awareness by bringing together SDN with IoT technologies. It adapts IoT networks to the application requirements by deploying the appropriate network protocol on demand, while considering the network environment constraints (e.g., limited node resource availability, mobility caused connectivity issues) by dynamically configuring the protocol in use. Figure 1 presents a high-level view of the MINOS architecture. Aligned to the typical three-tier SDN paradigm, it consists of the following planes, described bottom-up.

The Data Communication Plane: accommodates multiple dynamically configurable protocols supporting diverse IoT devices operating either in real testbeds or in the Cooja emulator. It also provides radio and network protocol measurements to the upper layer, as well as on-demand protocol deployment.

The Control Plane: controls the network protocols in 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 the *protocol decision engine*, which selects the protocol and its main configuration based on the application requirements, and protocol-specific control and monitoring components that are responsible for the runtime configuration adaptations and tracking the performance of the corresponding protocols, respectively.

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.

Below we elaborate on the MINOS planes, interfaces, and functionalities.

The MINOS Planes and Interfaces

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.

CORAL-SDN [4]: is implemented in the context of the MINOS platform to provide adaptability to a range of IoT applications and network constraints (e.g., mobility and signal issues) through four alternative protocol mechanisms: (i) two network topology discovery and control algorithms, the first based on node advertisements, and the second on neighbor requests; and (ii) two types of flow establishment rules, configuring the full path or the next hop only, respectively. For example, in a fixed topology with one mobile node, it is resource-expensive to configure the whole path each time a single node changes its attachment point; instead, the next hop flow establishment option is more suitable. CORAL-SDN also supports other configuration options; for example, the link quality estimation method, the use of acknowledgments, and the control messages' interval time for topology control.

Adaptable-RPL [5]: augments RPL, considered as the de facto routing standard for IoT, 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 RPL's performance issues in mobile environments [5], 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 the *control plane* with real-time measurements on the protocols' performance, or the configuration values of important parameters from both the radio (e.g., received signal strength indicator — RSSI or link quality indicator — LQI) and network viewpoints (e.g., number of packet drops). The MINOS southbound interfaces handle these interactions through utilizing novel application programming interfaces (APIs) provided by the WiSHFUL project [13]; 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 adjustment capabilities, providing full support to our protocols via the MINOS platform.

The *control plane* triggers the protocol deployment in an interchangeable manner through the protocol deployment interface (PDI). This plane currently supports two alternative ways of on-demand protocol deployment: *proactive*, for network 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 *reactive*, for rapid protocol deployments, where a double-protocol stack above the link layer dynamically selects one of the two alternative protocols (i.e., trading memory for quick protocol switching). Currently, we support proactive protocol deployment through device-specific Ansible scripts updating IoT devices' firmware. The reactive deployment is an ongoing work; hence, we released a first beta version of the double protocol stack [3]. Our plans also include experimentation with over-the-air protocol deployment approaches (e.g., utilizing elf contiki libraries).

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

The control plane performs runtime 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 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 construct and maintain an abstract representation of the infrastructure network (i.e., a network connectivity structure with runtime node or link information), such as the devices' battery level, or link quality measurements (e.g., RSSI or LQI); and 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 dynamically perform local topology 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 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 (e.g., through link coloring), but with minor impact on the other coexisting nodes transmitting measurements to the sink node. This is another interesting extension of the current work.

The 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, to specify the communication type required by the application, the number and main node capabilities, as well as the global performance goals. Currently, the PDE makes decisions based on hard-coded protocol strategies aligned with our experimentation analysis, but the results of an extensive experimentation analysis with more scenarios and a wide range of network conditions will furthermore enrich its decision making potential. Our short-term plans include a relevant machine learning algorithm (i.e., neural network) inspired by [14].

Lastly, the application plane specifies through the *northbound API* the requirements of the application to be realized by MINOS. This 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: many-to-one for typical WSN *data collection* applications, where a set of nodes gather periodic measurements destined to a single sink node; one-to-many for *data dissemination* applications, where the sink node spreads data to all nodes in the network; and 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 fixed or mobile, battery-pow-

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 select manually the protocol and its corresponding parameters; then the visualization outcome varies according to the protocol deployed.

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	Version 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 [15]	Canvas 750 × 750 m
Duration	1 h	

Table 1. Experimentation Setup.

ered or not, and 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.

The MINOS GUI

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. Representative screenshots are available online [3]. The dashboard performs the overall monitoring, while providing advanced functionality and configuration options through three sub-modules.

The Experiment Manager: providing configuration options related to available network protocols and experimentation setups.

The Network Visualizer: illustrating the network's topology, experiments' progress, and results.

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 via short client-side Node.js scripts.

Evaluation

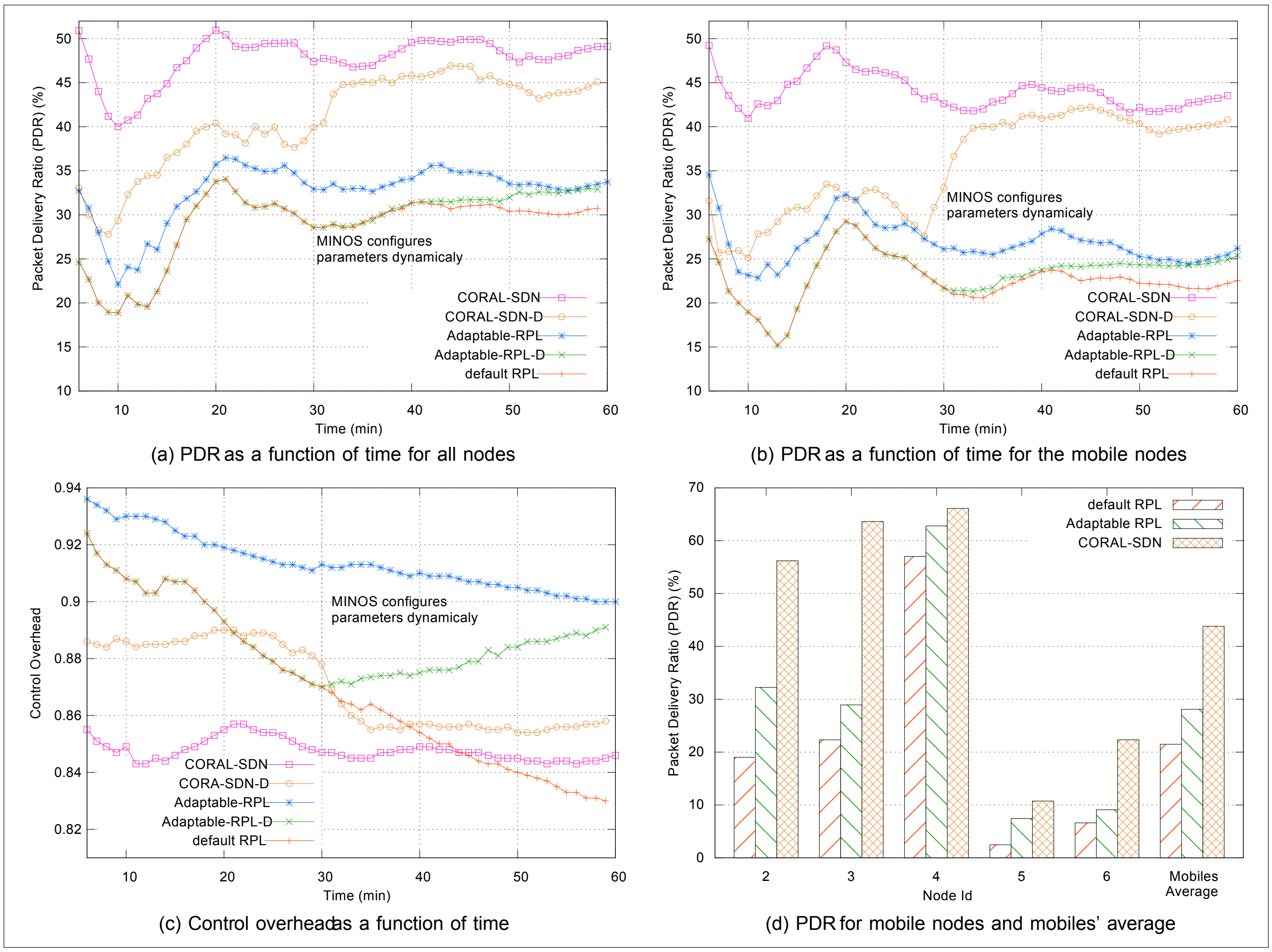
We evaluated MINOS for handling mobility and heterogeneity in an IoT experimentation setup; for example, we assume a smart city scenario similar to the one described earlier, where static nodes are located on buildings or stations, coexisting with mobile ones being placed in vehicles (all equipped with sensors offering city-monitoring facilities, e.g., temperature, humidity, noise, pollution). Through the MINOS GUI we can proactively deploy our protocols (i.e., either *CORAL-SDN* or *Adaptable-RPL*). Once one of them is deployed each time, we proceed with extracting results regarding two metrics: the PDR (i.e., the proportion of packets delivered against total packets sent), and 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 in contrast to the MINOS platform against traditional IoT deployments.

Experimentation Setup

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 bus routes, available via the MONROE project [15]. All of 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® Core™ i5-2410M, 2.30 GHz processor), along with real traces order of magnitude limitations, allow experimentation on this scale. Table 1 summarizes all experiments' settings.

Methodologically, we conducted the same scenario five times as follows:
- The first run employs the default RPL to provide a "ground truth" curve (red line in Figs. 2a–2c).
- The next two runs consider Adaptable-RPL, where the MINOS either dynamically configures its parameters on the fly (via the UPIs) at 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).
- The last two runs provide results for CORAL-SDN, where the MINOS again either dynamically adapts its parameters at 30 min (CORAL-SDN-D purple curve) or configures the protocol when the experiment starts (orange curve).

The MINOS interventions' impacts are clearly shown in Figs. 2a–2c, where after 30 min both the dynamically adapted scenarios (i.e., Adaptable-RPL-D and CORAL-SDN-D) are progressively converging to 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 on 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 [4], and configures the topology maintenance parameter rate at 4 s 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



(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 2. MINOS experimental results for PDR and control overhead: 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.

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.

Experimental Results

Figures 2a–2c 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 the x-axis to demonstrate the ability of 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, it achieves an improvement up to 7.74 percent for Adaptable-RPL and 19.4 percent for CORAL-SDN in the whole network (Fig. 2a), which rises up to 8.15 percent for Adaptable-RPL and 21.5 percent for CORAL-SDN for the mobile nodes (Fig. 2b). This outcome also highlights the benefits of offloading the *control overhead* to the static nodes. Since the mobility pattern for 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 percent in Fig. 2b. Another interesting outcome is that our platform can achieve better PDR with relatively small control overhead (Fig. 2c) when employing the CORAL-SDN protocol. Figure 2c presents the average PDR (over a period of 60 min) for each mobile node when 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 [4], 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 flooding the network with control messages. We indicatively found improvements in the PDR of mobile nodes 2, 3, and 6, up to 37.1, 41.3, and 15.7 percent, respectively. This happens because CORAL-SDN succeeds in routing messages through their neighboring mobile nodes. However, as previously indicated, this performance comes 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 criti-

We adopt the SDN approach to build an architecture that 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 to respond to dynamic network conditions. MINOS follows a modular architecture, and its planes serve as further experimentation place holders.

cal 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 Adaptable-RPL could be more suitable than CORAL-SDN by steadily offering a higher PDR compared to the default RPL protocol (Figs. 2a, 2b), since the mobile nodes remain connected to the topology for longer periods. The improved PDR is traded for the increased control overhead (Fig. 2c), occasionally tolerated if, for example, the mobile nodes are powered by the hosting vehicle.

Conclusions and Future Work Insights

This article presents the 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 that 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.

In the evolution of this work we consider:

- Using the IoT-LAB of the FIT experimental platform, and the CityLab Fed4FIRE+ testbed, both allowing multihop deployments
- Evolving sophisticated decision mechanisms for (reactive) protocol deployment and network control
- Accommodating more protocols and optimal switching among them
- Investigating performance trade-offs in respect to the network scale

Acknowledgment

This work is partially supported by the H2020 NECOS project (grant agreement number 777067) and the open call schemes of the WiSHFUL (grant agreement number 645274) and MONROE (grant agreement number 644399) projects.

References

[1] I. Yaqoob *et al.*, "Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges," *IEEE Wireless Commun.*, vol. 24, no. 3, June 2017, pp. 10–16.
[2] M. Baddeley *et al.*, "Evolving SDN for Low-Power IoT Networks," *Proc. 4th IEEE Conf. Network Softwarization and Workshops*, June 2018, pp. 71–79.
[3] "MINOS Open-Source Software and Demo Videos"; https://github.com/SWNRG/minos, accessed June 18, 2019.
[4] T. Theodorou and L. Mamatas, "Software Defined Topology Control Strategies for the Internet of Things," *Proc. IEEE Conf. Network Function Virtualization and Software Defined Networks*, Nov. 2017, pp. 236–41.
[5] G. Violettas, S. Petridou, and L. Mamatas, "Routing Under Heterogeneity and Mobility for the Internet of Things: A Centralized Control Approach," *Proc. IEEE GLOBECOM*, 2018, pp. 1–7.
[6] T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF RFC 6550, Mar. 2012.
[7] L. Galluccio *et al.*, "SDN-WISE: Design, Prototyping and Experimentation of a Stateful SDN Solution for WIreless Sensor Networks," *Proc. IEEE INFOCOM*, Apr. 2015, pp. 513–21.
[8] S. Bera *et al.*, "Soft-WSN: Software-Defined WSN Management System for IoT Appllication," *IEEE Sys. J.*, 2016.
[9] B. T. de Oliveira and C. B. Margi, "TinySDN: Enabling tinyOS to Software-Defined Wireless Sensor Networks," 2016, pp. 1229–37.
[10] E. Municio *et al.*, "Whisper: Programmable and Flexible Control on Industrial IoT Networks," *Sensors*, vol. 18, no. 11, 2018, p. 4048.
[11] D. Carels *et al.*, "RPL Mobility Support for Point-to-Point Traffic Flows Towards Mobile Nodes." June 2015.
[12] Y. Tahir, S. Yang, and J. McCann, "BRPL: Backpressure RPL for High-Throughput and Mobile IoTs," *IEEE Trans. Mobile Comp.*, vol. 17, no. 1, Jan. 2018, pp. 29–43.
[13] P. Ruckebusch *et al.*, "WiSHFUL: Enabling Coordination Solutions for Managing Heterogeneous Wireless Networks," *IEEE Commun. Mag.*, vol. 55, no. 9, Sept. 2017, pp. 118–25.
[14] S. A. Seidel *et al.*, "Analysis of Large-Scale Experimental Data from Wireless Networks," *Proc. IEEE Conf. Comp. Commun. Wksps.*, Apr. 2018, pp. 535–40.
[15] O. Alay *et al.*, "MONROE: Measuring Mobile Broadband Networks in Europe," *Proc. IRTF & ISOC Wksp. Research and Application of Internet Measurements*, 2015.

Biographies

Tryfon Theodorou is a Ph.D. candidate in the Department of Applied Informatics, University of Macedonia, Greece. His area of expertise includes wireless sensor networks, software-defined networks, information security, and IoT.

George Violettas is pursuing a Ph.D. in the area of software-defined mobile networks at the University of Macedonia. He holds an M.Sc. degree in applied informatics from the same university.

Polychronis Valsamas is pursuing a Ph.D. in the area of cloud computing and the network edge at the University of Macedonia. He holds an M.Sc. degree in system engineering and management, Democritus University of Thrace.

Sophia Petridou is an assistant professor in the Department of Applied Informatics, University of Macedonia. Her main research interests are in the areas of optical and wireless networks.

Lefteris Mamatas is an assistant professor in the Department of Applied Informatics, University of Macedonia. His research interests lie in the areas of software-defined networks, network functions virtualization, and mobile edge and fog computing.