ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΚΤΥΑΚΟΣ ΕΛΕΓΧΟΣ ΚΑΙ ΑΣΦΑΛΕΙΑ ΝΕΑΣ ΓΕΝΙΑΣ ΓΙΑ ΤΟ ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

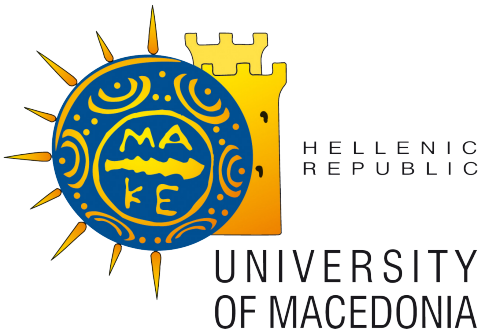
Διδακτορική Διατριβή

του

Βιολέττα Γεωργίου του Ευαγγέλου

Επιβλέπων: Ελευθέριος Μαμάτας, Επίκουρος Καθηγητής

Θεσσαλονίκη, Φεβρουάριος 2021



UNIVERSITY OF MACEDONIA

SCHOOL OF INFORMATION SCIENCES

DEPARTMENT OF APPLIED INFORMATICS

Next-generation Network Control and Security for the Internet of Things

Ph.D. Dissertation

of

George E. Violettas

Advisor: Assist. Prof. Lefteris Mamatas

Thessaloniki, February 2021

Ἀνερρίφθω κύβος…
Alea jacta est…

Στη μνήμη του Πατέρα μου, Βαγγέλη Βιολέττα

Στη ‘‘μου Σόφη’’ που με ανέχεται…

Στο Βαγγέλη και στη Δανάη για τις ώρες που τους στέρησα…

Με εκτίμηση, στο Λευτέρη Μαμάτα που μου στάθηκε στα δύσκολα. Δεν είναι πλέον ο επιβλέπων, είναι φίλος.

Στον Αλέξανδρο Χατζηγεωργίου. Η φιλία του με τιμά.

Στη Σοφία Πετρίδου. Χωρίς την επιμονή της, το αποτέλεσμα θα ήταν σαφώς κατώτερο.

Στην Ασημίνα Σουλτάνη. Αν δεν μου είχε πει μία μέρα να κάνω αίτηση στο ΕΑΠ, τιποτα από όλα αυτά δεν θα είχε συμβεί…

Σε όλους τους φίλους που έκανα από την πρώτη μέρα που πήγα στην Ο.Σ.Σ. του Ε.Α.Π. Στο Δημήτρη Θεοδωρίδη, και στο Βάιο Ζιώγα.

Και φυσικά στον Τρύφωνα Θεοδώρου με τον οποίο φάγαμε ‘‘ψωμί και αλάτι’’. Χωρίς τις ατελείωτες φορές που ο ένας πίεζε τον άλλου, χωρίς τις ατέρμονες συζητήσεις, τα ξενύχτια μέσω skype, αυτή η διατριβή δεν θα ήταν δύνατη. Φίλος στα δύσκολα…

Σε όλους αυτούς που με βοήθησαν.

Στους φίλους που γνώρισα στα ξένα.

Περίληψη

Αναμφίβολα, ένας από τους τομείς με τη μεγαλύτερη συμβολή στα δίκτυα 5G είναι το Διαδίκτυο των Πραγμάτων (ΔτΠ), δηλαδή ετερογενείς και χαμηλού κόστους αισθητήρες, ενεργοποιητές, αντικείμενα, καθώς και συσκευές περιορισμένων πόρων, διασκορπισμένες σε μεγάλες αστικές και αγροτικές εκτάσεις. Όλες αυτές οι συσκευές, ενώ αντιμετωπίζουν κάθε λογής προβλήματα όπως περιορισμένους πόρους, ασύρματη κάλυψη, ετερογένεια των λειτουργικών συστημάτων, προβλήματα λογισμικού και πρωτοκόλλων μεταξύ άλλων, προσπαθούν να επικοινωνήσουν ασφαλώς και αποτελεσματικά, με τελικό στόχο να πάρουν μετρήσεις και να αλληλοεπιδράσουν με το περιβάλλον, μετατρέποντας τις αναλογικές μετρήσεις σε ψηφιακή πληροφορία, για πρώτη φορά στην ανθρώπινη ιστορία.

Επιπλέον, τα ΔτΠ δεν επικοινωνούν απαραίτητα με τα "παραδοσιακά ασύρματα πρωτόκολλα" (π.χ. 802.11x) λόγω περιορισμών στη διαθέσιμη μνήμη, ενέργεια, επεξεργαστική ισχύ, κτλ. Επίσης αυτές οι συσκευές ευρισκόμενες εντός πεδίου, συνήθως αλλάζουν γρήγορα και δυναμικά την τοπολογία τους (π.χ. κινούμενοι κόμβοι. Τα παραπάνω αποκαλύπτουν την ανάγκη για νέα προσαρμοζόμενα πρωτόκολλα και τεχνικές επικοινωνίας που θα μπορούν να επιλύσουν τέτοια ενδιαφέροντα προβλήματα και να υπερβούν τους υφιστάμενους περιορισμούς και δυσκολίες. Αυτές οι λύσεις θα πρέπει αναγκαστικά να είναι συμβατές με το καθολικά χρησιμοποιούμενο πρωτόκολλο IPv6 όπως και υπάρχοντα πρωτόκολλα που ήδη κατέχουν δεσπόζουσα θέση (όπως π.χ., το IEEE 802.15.4). Αυτά τα πρωτόκολλα επιλύουν κάποια από αυτά τα προβλήματα, αλλά υπάρχει χώρος για πιο ακριβείς και καινοτόμες λύσεις. Για παράδειγμα, το αρκετά δημοφιλές πρωτόκολλο RPL αντιμετωπίζει περιορισμούς και προβλήματα, ειδικά επειδή βασίζεται στην κατανεμημένη εικόνα που απορρέει από την ίδια τη φύση των ΔτΠ δικτύων.

Η διατριβή αυτή εστιάζει σε προβλήματα που αναδύονται σε αυτά τα πρωτόκολλα όταν εμφανίζονται ακραίες ή ασυνήθεις συνθήκες. Υπάρχουν περιπτώσεις όπου οι προτεραιότητες του πρωτοκόλλου πρέπει να διαφοροποιηθούν προσωρινά, ώστε να εξυπηρετήσουν μία ανακύπτουσα ανάγκη για επικοινωνία σημείου-προς-σημείο μεταξύ των κόμβων, ή όταν κινούμενοι κόμβοι χρειάζεται να χρησιμοποιήσουν αποδοτικά το χαμηλό εύρος ζώνης που υπάρχει διαθέσιμο, και όλα αυτά κάτω από τη διαρκή ανάγκη για ασφάλεια.

Κάποια από αυτά τα θέματα μπορούν να αντιμετωπιστούν μέσω της εφαρμογής του παραδείγματος των Δικτύων Καθοριζόμενων απο το Λογισμικό (ΔΚΛ), (Software Defined Networks, SDN) στα ΔτΠ. Η εφαρμογή των ΔΚΛ επιφέρει κεντρική διαχείριση, καθολική κατόπτευση του δικτύου, κεντρική λήψη αποφάσεων, καλύτερη απόκριση σε αλλαγές, κινδύνους, κτλ., αλλά ακριβώς λόγω της καθολικής εικόνας επιφέρει και περισσότερη κίνηση στο δίκτυο λόγω της αποστολής της επιπλέον πληροφορίας. Κάτω από το πρίσμα του ΔτΠ, τα ΔΚΛ πρέπει να χρησιμοποιηθούν επιλεκτικά, μόνον εκεί όπου η ισορροπία μεταξύ του όποιου κέρδους και της πλεονάζουσας πληροφορίας, είναι σαφώς υπέρ του πρώτου.

Η διατριβή επίσης παρουσιάζει μία λεπτομερή καταγραφή των προβλημάτων ασφάλειας των δικτύων ΔτΠ. Ως αποτέλεσμα αυτής της βιβλιογραφικής μελέτης, κατασκευάστηκε ένα σύγχρονο λογισμικό Σύστημα Ανίχνευσης Εισβολών (Intrusion Detection System, IDS), ονόματι *ASSET*, εμπνευσμένο από το παράδειγμα των Δικτύων Καθοριζόμενων απο το Λογισμικό (ΔΚΛ), (Software Defined networks, SDN). Το σύστημα μπορεί να αντιμετωπίσει τις περισσότερες επιθέσεις ενάντια στο RPL πρω-

τόκολλο.

Πιο συγκεκριμένα, η διατριβή εξετάζει: (i), Τη δημιουργία και εφαρμογή λύσεων δρομολόγησης σημείου προς σημείου (point-to-point) μέσα στο πρωτόκολλο RPL, το οποίο είναι προσανατολισμένο σε δικτύωση ενός-προς-πολλούς (point-to-multipoint), με αφετηρία τους κόμβους του δικτύου, και κατάληξη στον κεντρικό κόμβο-δρομολογητή, (sink-node), (ii) Τη λειτουργία των ΔτΠ, αλλά και του RPL, κάτω από ιδιαίτερες και ακραίες συνθήκες, και πως αυτή η λειτουργία μπορεί να βελτιωθεί, τόσο από την παραμετροποίηση των κόμβων ανά περίπτωση (ad-hoc) αλλά και τα προβλήματα που ανακύπτουν από την ετερογένεια των κόμβων αυτών, σε επίπεδο κατασκευαστή, δυνατοτήτων επεξεργαστικής και αποθηκευτικής ισχύος, κτλ. Επίσης στο ίδιο πλαίσιο εξετάζονται οι δυνατότητες συνεργασίας μεταξύ διαφορετικών επιπέδων δικτύου και η μεταφορά δεδομένων κατά περίπτωση, και (iii), Την αποτελεσματικότητα που επιφέρει η κεντρική διαχείριση και η κατόπτευση του δικτύου στην αποτελεσματική αντιμετώπιση κόμβων-εισβολέων, και πολλών διαφορετικών επιθέσεων. Στο ίδιο πλαίσιο εξετάζεται αναλυτικά η αποτελεσματικότητα χρήσης εργαλείων από άλλους κλάδους στην υπόδειξη και αποκλεισμό των εισβολέων-κόμβων.

Τα πειραματικά αποτελέσματα της διατριβής δείχνουν τα εξής: (i) το πρωτόκολλο RPL, αν και αυτή την στιγμή έχει δεσπόζουσα θέση στο ΔτΠ, παρά ταύτα, έχει σοβαρές δυσλειτουργίες στη δρομολόγηση σημείου-προς-σημείο, και στη δρομολόγηση και διαχείριση κινούμενων κόμβων. Σε αυτόν τον τομέα, η βιβλιογραφία χρειάζεται εμπλουτισμό με καινούριες, καινοτόμες λύσεις και ιδέες, (ii), υπάρχουν πολλά ανοιχτά ζητήματα στο ΔτΠ όσον αφορά στις συσκευές, και πιο συγκεκριμένα στις δυνατότητες επεξεργασίας, αποθήκευσης και δικτύωσης, στη διάρκεια και στη διαχείριση της ενέργειας και της μπαταρίας, καθώς και στις διαλειτουργικές δυνατότητες και προβλήματα μεταξύ των διαφόρων επιπέδων δικτύου (network layers), καθώς και μεταξύ διαφόρων κατασκευαστών, δυνατοτήτων των συσκευών, και επιμέρους προσφερόμενων λύσεων,

(iii), τα ΔτΠ, έχουν πολλά ανοιχτά θέματα ασφαλείας, ειδικότερα όσον αφορά την παρακολούθηση και αναγνώριση επιθέσεων, ειδικά αυτών που εκμεταλλεύονται δομικές αδυναμίες σε δημοφιλή πρωτόκολλα όπως το ΡΠΛ, και κάποιες από αυτές δεν αντιμετωπίζονται ακόμη και με τη χρήση κρυπτογραφίας. Σε αυτό το πεδίο, τα Συστήματα Αναγνώρισης Εισβολής (ΣΑΕ,(Intrusion Detection System, IDS), παραμένουν μία αξιόπιστη λύση που χρειάζεται πολύ περισσότερη βιβλιογραφική και πειραματική έρευνα.

Λέξεις Κλειδιά: Ασύρματα Δίκτυα Αισθητήρων, Διαδίκτυο των Πράγματων, ΔτΠ, Δίκτυα Κινούμενων Κόμβων, Δίκτυα Καθοριζόμενα από το Λογισμικό, RPL, Ασφάλεια ΔτΠ

Abstract

Undoubtedly, one of the primary enablers of the 5G networks is the Internet of Things (IoT), i.e., low-cost and heterogeneous sensors, actuators, objects, and constrained devices, scattered across extensive urban or rural areas. All those devices, while facing all kind of problems from constrained resources, wireless coverage, heterogeneity of operating systems, applications, and protocols, among others, are struggling to efficiently and securely communicate, with the ultimate goal of measuring and interacting with the surrounding environment, transforming the analog readings to digital information, for the first time in human history.

Those IoT networks do not usually communicate with the "traditional" wireless protocols (e.g., IEEE 802.11x) because of restrictions such as memory, energy, complexity, etc. Also, those devices, when in the wild, could be rapidly and dynamically changing their topology (e.g., mobile sensors). The above reveals the need for new adapted protocols and communication techniques that will try to solve new exciting problems and overcome arising restrictions. All such proposals should also be compatible with the IPv6 and existing well-established already protocols (e.g., IEEE 802.15.4). Those existing protocols address some of the above issues, but yet, there is an open space for more accurate and innovative solutions. For example, the well-established RPL protocol faces limitations and problems, basically due to the distributed view inherited by the very nature of IoT networks.

The current dissertation focuses on problems arising within those protocols when an extreme or unusual condition or demand occurs. There are cases where the protocol's priorities need to be temporarily altered, to favor for example, an ad hoc emerged point-to-point communication need or when mobile nodes need to efficiently utilize the low bandwidth available, all under the everlasting need for security and safety.

Some of the above issues can be benefited from the application of the Software Defined Networks (SDNs) paradigm in the IoT. SDN brings central management, a catholic view of the network, centralized decision making, better response to changes, dangers, etc., but also entails increased network traffic because of this extra information. Under the IoT prism, SDN needs to be selectively used, only where the trade-off between possible gain and control overhead is clearly in favor of the first.

The dissertation also cites a thorough investigation of IoT security, and in particular, the security issues arising under the RPL protocol. As a result of this bibliographic overview, a state-of-the-art Intrusion Detection System named *ASSET* inspired by the network softwarization paradigm was constructed, able to confront most of the existing attacks against the RPL protocol.

More specifically, the dissertation examines: (i) creating and applying point-to-point routing algorithms and solutions for the RPL protocol, focusing on point-to-multipoint communications and routing from the network nodes to the sink node. (ii) the functionality of IoT and RPL protocol under harsh and difficult circumstances, and how to improve this functionality by ad hoc parameterization of the nodes and confront the problems arising from the heterogeneity of the nodes in manufacturer level, nodes functionalities, etc. Within the same context, possibilities of cooperation between different network layers and adaptable data transfer are examined. (iii) the effectiveness brought by the centralized man-

agement and real-time network monitoring to successfully confronting multiple intruders and many different attacks. Under the same scope, the usability and effectiveness of tools and techniques from other scientific areas to identify and exclude those intruders are thoroughly examined. Experimental results of this dissertation, among others, have shown the following: (i) Even though the RPL protocol holds a dominant position in IoT, it has severe issues and flows regarding point-to-point and mobile nodes routing. (ii) there are many open issues in IoT regarding the devices, especially towards processing, storing, and networking, along with the duration and optimization of batteries in the intra-functional capabilities and problems between different network layers, but also between manufacturers, device capabilities, and offered solutions. (iii) The IoT has many open security issues, especially regarding the monitoring and detection of attacks exploiting structural weaknesses of popular protocols such as RPL, since some of them cannot be confronted even with cryptography. In this area, IDSs remain a trustworthy solution needing more bibliographical and experimental research.

Keywords: Wireless Sensor Networks, Internet of Things, IoT, Mobile networks, Software-Defined Networks, RPL, IoT Security

Funding

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

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations & Symbols

Ελληνικά

ΔτΠ	Διαδίκτυο των Πραγμάτων
ΔΚΛ	Δίκτυα Καθοριζόμενα απο το Λογισμικό
ΣΑΕ	Σύστημα Αναγνώρισης Εισβολής

English

ASSET	A Softwarized intruSion dEtection SysTem for RPL
DAO	Destination Advertisement Object packet
DAO-ACK	DAO Acknowledgement packet
DIO	DODAG Information Object packet
DIS	DODAG Information Solicitation packet
DODAG	Destination-Oriented Directed Acyclic Graph
ETX	Expected Transmission Count
IDS	Intrusion Detection System
IoT	Internet of Things
LLNs	Low-power and Lossy Networks
MRHOF	Minimum Rank with Hysterisis Objective Function
OF	Objective Function
OF0	Objective Function Zero
P2P	peer-to-peer
PDR	Packet Delivery Ration
PDT	Packet Delivery Time
PLR	Packet Loss Ratio
RPL	Routing over Low Power and Lossy Networks
RTT	Round Trip Time
SDN	Software Defined Networks
TT	Trip Time
WSN	Wireless Sensor Networks

1 Introduction

Internet of Things (IoT) does rapidly develop. It is also the technological enabler for smart-x ecosystems and the next-generation advanced manufacturing, referred to as Industry 4.0 (I 4.0), including smart products, smart production, and smart services. For example, recent advances in communication technology, e.g., 5G Networks, along with the IoT in the era of the I 4.0, evolve the request for mass production and automation—firstly introduced during the previous waves of revolution—from the principle idea to connect everything in the production chain to the more sophisticated context of broader and more fine-grained interconnections [2]. Indeed, a network of geographically distributed factory branches requires flexible adaptation of production capabilities and sharing of resources and assets to improve orders' fulfillment. In such an automation ecosystem, secure and reliable data transfer among different entities is an essential but also critical issue. Such installations entail hundreds of scattered smart devices, sensors, and actuators communicating throughout IoT deployments.

In all the above cases, routing is an incredibly challenging network function, basically due to power, storage, memory, processing, and signal limitations of the connected IoT devices, and nowadays, because of characteristics such as large-scale deployment, dynamicity, heterogeneity and mobility, not to neglect security. Undeniably, today, more and more environmental, agriculture, or smart city applications require extended, reliable, and secure sensing coverage.

Examples and test cases include deployments where scattered sensors are gathering air and surroundings' critical measurements in large industrial facilities, need to communicate with other sensors for verification or to alert purposes (e.g., in case of a poisonous gas spike). Traffic prioritization is another 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. 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 consider constraints such as the remaining battery power. All those need to securely and accurately exchange data without the communication been interrupted or intercepted while confronting possible attacks and identifying and excluding potential intruders or alien nodes.

1.1 RPL Protocol

One of the protocols widely used to tackle such as the above issues, the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), has become the de facto standard for IoT routing today [3, 4]. RPL has been proven significantly mature to connect IPv6 devices, with reasonable control overhead and under challenging conditions, e.g., lossy links, heterogeneous and constraint devices, even under newfangled threats [5, 6]. RPL supports several configuration parameters and Objective Function (OF, i.e., how each node estimates and chooses the next hop) customization(s) that cover a wide range of alternative deployments [3, 7]. However, customizing such configurations is basically manual, generic, and often unpredictable in outcome terms.

Those examples highlight the need for routing control mechanisms that appropriately handle the requirements defined by the application, i.e., individual configuration of network nodes, or even a "surgical" change of the topology graph, driven by a more centralized view of the network. Although the main focus of RPL is to create routes from the nodes to the sink (the route node), it has a high customization capability, offering space for routing strategies that can successfully accommodate the end-to-end communication without violating the principles of the protocol.

From the above, it can be argued that if adapted accordingly, RPL can provide better support to scenarios involving mobility and efficiently address end-to-end communication, much like the scenarios exploited in this dissertation.

1.2 Security of IoT protocols

On the security front, RPL still has open issues, the most important of which are related to attacks that disrupt the IoT network's operation [8]. In fact, RPL is unavoidably exposed to a large number of attacks since it is based on the IPv6 open stack and uses mostly wireless media for the nodes' communication.

According to the literature [8], RPL-related attacks include malicious actions aiming at: (i) exhausting nodes' resources as a means of significantly reducing the network's lifespan and availability, (ii) disrupting the structure of the Destination-Oriented Directed Acyclic Graph (DODAG), upon which nodes' communication is based, affecting network's performance in respect to packet losses and end-to-end (E2E) delays.

In addition, by exploiting RPL's mechanisms, an intruder can gain access to the network and unleash attacks that originate from within the LLN. In such cases, encryption itself does not suffice to provide security [9]. On this front, the RPL standard specifies three modes of operation, i.e., unsecured mode, preinstalled mode, and authenticated mode [3], while it also defines mechanisms for data confidentially, data authenticity, and replay protection [10]. Although some recent research efforts focus on a partial implementation of RPL's security features [10, 11], up to this time, the majority of RPL implementations assume the unsecured mode of operation. Actually, the RPL security features are characterized as optional [3] and, according to [12, 13], future versions of RPL will address issues such as authenticated security.

1.2.1 Intrusion Detection Systems

Until today, one of the most realistic approaches to deal with attacks is the Intrusion Detection Systems (IDSs). IDSs refer to a set of methods designed toward: (i) *detecting an attack*, (ii) *identifying the attacker*, and (iii) *mitigating* the event. They aim to detect several attacks concurrently, and ideally, they can be extended to deal with attacks that are not originally included in their design goals. Compared to the standalone mechanisms, they require some degree of collaboration among the network's nodes [14].

Although the research field of IDSs in the IoT domain is generally vast, only a restricted subset of them is appropriate for Low-power and Lossy Networks (LLNs) [15, 16], i.e., they take into consideration limitations in respect to their lossy links, heterogeneous and resource-constraint devices. In fact, most of them have been proposed in the recent bibliography, i.e., from 2013 to 2020 [8, 14, 15].

An overview of these works makes clear that there is no one-for-all solution that succeeds in all three axes, i.e., to *detect* several attacks at once, to *identify* the intruder, and to *mitigate* the event, and at the same time, identify as many attacks as possible, be extendable while detecting the attacker accurately and rapidly.

Moreover, due to the distributed nature of the IoT and subsequently RPL's, such an IDS would be greatly benefited if provided with more centralized capabilities, given the catholic view of the network provided by such an installation.

1.3 Software Defined Networks Paradigm

A prominent solution that can contribute to improving the IDSs described above by providing a more centralized view of the network is Software Defined Networks (SDNs) and the prominent OpenFlow protocol, another newly emerged technology, that was initially described for large infrastructure networks, and thus, it is not fully aligned with the unique requirements of the IoT networks while posing additional challenges not present in the conventional networks.

In particular, SDN control communication messages with the controller increase the number of control packets in the network drastically. This further impairs the resource-constraint nodes as well as the low quality and lossy nature of the wireless communication medium [17]. Nevertheless, SDN-inspired paradigms provide a new, elastic network paradigm that can transform the traditional network backbones into flexible service delivery platforms. Four IoT research challenges that SDN can ideally handle are defined hereby:

- *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.
- *Security*: This is to handle issues arising from the distributed nature of IoT protocols. A centralized view of such a network offers a global picture, identifying discrepancies in specific network areas. Moreover, the centralized infrastructure can utilize computationally intense tools and applications for analysis and monitoring.

Arguably, an SDN-inspired IoT platform, through the necessary abstractions, can support individual protocol deployments and configurations per groups of IoT nodes and realize adaptive communication strategies, mitigating the previously mentioned challenging communication issues. In general, there is no single protocol solution that can address the variety of requirements originated by different IoT

applications. Hence, there is an open area for platforms accommodating multiple protocols, their on-demand deployment, along with network conditions' dynamic adaptations.

1.3.1 SDN-inspired Solutions for IoT

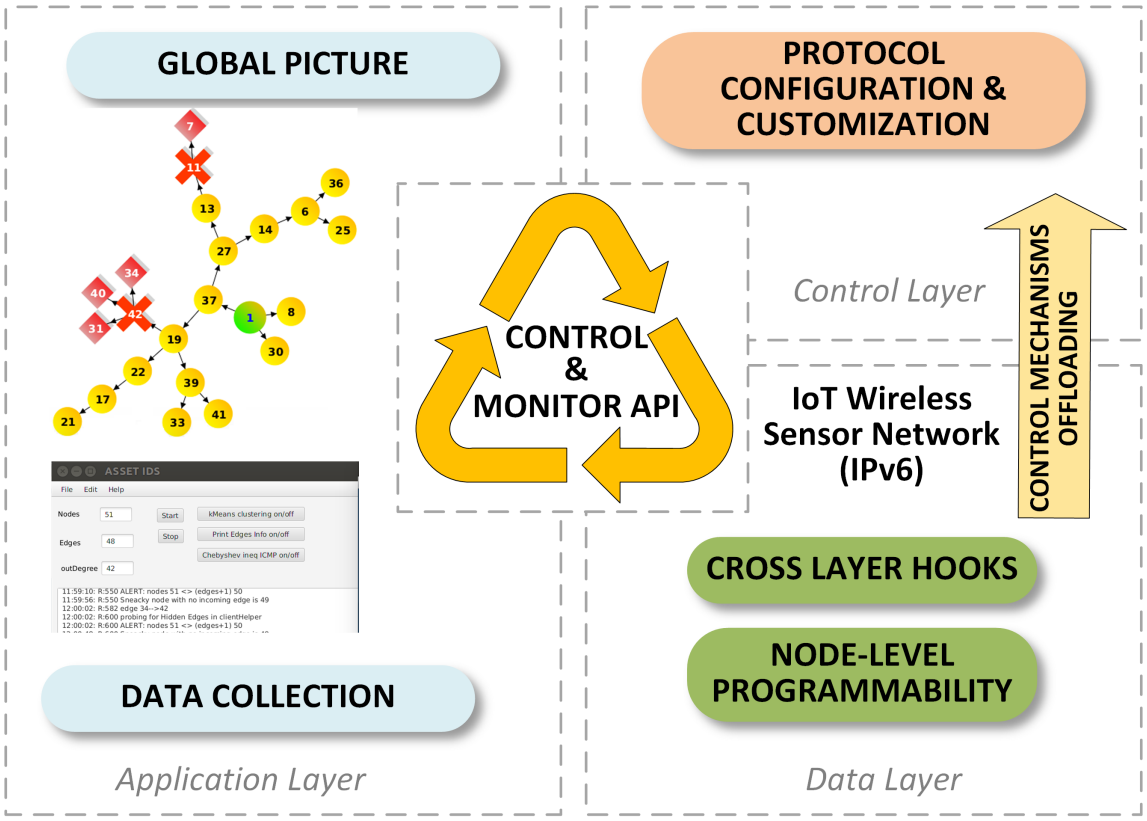


Figure 1: Architecture of SDN-like mechanisms implemented over IoT networks.

In several cases, IoT network nodes are constrained devices with limited wireless coverage. In such conditions, the separation of control and data channel is not possible since it would quickly exhaust such nodes' resources by the overload of control messages. Hence, the SDN paradigm is not entirely applicable. Nevertheless, suppose network-softwarization is applied where possible. In that case, it can lead to greater responsiveness, security, efficiency, and cost-effectiveness by making the network more flexible and simple by minimising dependence on hardware constraints while keeping control overhead at acceptable levels and improving security by providing centralized monitoring. Figure 1 provides an abstract view of the extra potentials offered by the softwarization of IoT networks. Such SDN-inspired solutions can provide network programmability and offloading control mechanisms from the nodes to the control layer of the centralized infrastructure. Along comes the ability to configure and dynamically altering the protocol parameters through cross-layer hooks installed in the data layer. Such functionalities allow the controller to read individual node data and consequently impose respective policies per node, while keeping the control overhead at acceptable levels since only certain functionalities were transferred centrally. A control & monitoring API provides the ability to the controller to maintain a vivid, live, and more secure view of the

network, offering centralized view, which leads to more accurate decisions, improved security, and the possibility of utilizing computationally demanding tools and mechanisms that cannot be used in the constrained nodes.

In a nutshell, such carefully abstracted solutions can benefit from the SDN-alignment, offering much more network management functionalities and dynamic parameterization of nodes via centralization.

1.4 Aims & Objectives

This dissertation aims to improve the end-to-end, mobility, and secure operation of the state-of-the-art RPL protocol with SDN-inspired architectures and mechanisms enabling a centralized elastic operation and intrusion detection, backed by node-level programmability and efficient control channel communication.

In detail, this dissertation, by enabling the SDN-paradigm and centralized monitoring of the network, aims to provide softwarized solutions for the IoT and especially for the RPL protocol, as the following:

- End-to-end communication within RPL, and the challenges such an endeavor faces since RPL was built upon the many-to-one principle. Under the same prism, mobility of the nodes is also challenging since RPL did not include such function in its initial specifications.
- Provide a systematic and up-to-date survey on RPL security challenges, specifically for Intrusion Detection Systems for RPL. Here, the bibliography was inadequate and somewhat outdated, so a fresh look under a new prism framework was essential.
- Based on the previous systematic survey, the need for a softwarized, central, state-of-the-art, SDN-like IDS was eminent. Such an IDS was indeed created by utilizing technologies and algorithms from other areas (machine learning, artificial intelligence, outlier detection).

Moreover, in this dissertation, the following objectives were set:

- To identify the main challenges that IoT brings to the Low-power WSN solutions, particularly with the state-of-the-art WSN routing protocols like RPL,
- To give the ability to RPL to dynamically alter crucial parameters to be more efficient under different circumstances,
- To identify the RPL's difficulties to deliver point-to-point information among nodes, especially under mobility successfully, and to present new, efficient, protocol-compatible ways to improve these issues,
- To describe the RPL's security issues and weaknesses and present ways to address those,
- To systematically review the current bibliography on Intrusion Detection Systems for the RPL protocol,
- To construct an innovative state-of-the-art SDN-inspired platform that can centrally represent RPL's logical network representation,
- To equip the above platform with intelligent, independent, new, and existing mechanisms, to tackle several RPL security challenges,
- To suggest further improvements and key research areas that now and in the future will exploit the benefits that the centralized control brings to IoT applications.

1.5 Published Work

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

1.5.1 Scientific Journals

J.1 **G. Violettas**, S. Petridou and L. Mamatas, "Evolutionary Software Defined Networking-Inspired Routing Control Strategies for the Internet of Things," in IEEE Access, vol. 7, pp. 132173-132192, 2019.

J.2 T. Theodorou, **G. Violettas**, P. Valsamas, S. Petridou, Lefteris Mamatas, "A Multi-Protocol Software-Defined Networking Solution for the Internet of Things," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 42-48, Oct. 2019.

J.3 G. Simoglou, **G. Violettas**, S. Petridou, L. Mamatas, "Intrusion Detection Systems for RPL Security: A Comparative Analysis", Computers & Security, vol 104, pp. 0167-4048, Elsevier, 2021.

J.4 **G. Violettas**, G. Simoglou, S. Petridou, L. Mamatas, "ASSET: A Softwarized intruSion dEtection SysTem for RPL" , Future Generation Computer Systems, Elsevier, 2021. (Under Revision).

1.5.2 International Scientific Conferences

C.1 Valsamas, P., Skaperas, S., **Violettas, G**., Theodorou, T., Petridou, S., Vardalis, D., & Mamatas, L. (2019). "Multi access edge computing for efficient content distribution and IoT services". In 2019 IEEE WCNC.

C.2 P. Valsamas, S. Skaperas, **G. Violettas**, T. Theodorou, S. Petridou, D. Vardalis, A. Tsioukas, and L. Mamatas, "Experimenting with Cloud and Network Orchestration for Multi-Access Edge Computing", IEEE Wireless Communications and Networking Conference, Marrakech, Morocco, Apr. 2019, demo paper.

C.3 **Violettas, G.**, Petridou, S., and Mamatas L., "Routing under heterogeneity and mobility for the Internet of Things: a centralized control approach", IEEE Global Communications Conference, IEEE Globecom, 09-12 December 2018, Abu Dhabi, UAE. Acc. Rate 38%.

C.4 **G. Violettas**, T. Theodorou, S. Petridou, A. Tsioukas, and L. Mamatas, "An Experimentation Facility Enabling Flexible Network Control for the Internet of Things," *International Conference on Computer Communications - (INFOCOM)*, IEEE, Atlanta USA, May. 2017, demo paper.

1.5.3 Conference Tutorials

T.1 **Violettas, G.**, Theodorou, T., and Mamatas, L. (2017b). Softwarized IoT with Lightweight Clouds in Practice (3hr). (Berlin, Germany: 3rd IEEE Conference on Network Functions Virtualization and Software Defined Networking (IEEE NFV-SDN 2017), 2017.

1.5.4 Awards

As a part of the SWN Research group (http://www.swn.uom.gr), we participated in the *Elastic Wireless Networking Experimentation (eWINE) Grand Challenge* [18], in Oulu, Finland June 2018, where we received the runner-up award.

Table 1: Github URLs for the code created under this dissertation.

#	Project	URL
1	CORAL	https://github.com/SWNRG/wishful-coral
2	Link Coloring	https://github.com/SWNRG/rpl_link_coloring
3	MINOS	https://github.com/SWNRG/minos
4	ASSET	https://github.com/SWNRG/ASSET
Adapted Contiki OS versions for ASSET		
4.1	https://github.com/SWNRG/contiki-standard-extra-functions	
4.2	https://github.com/SWNRG/contiki-malicious-controller-aware	
4.3	https://github.com/SWNRG/contiki-malicious-controller-aware-version-attack	
4.4	https://github.com/SWNRG/contiki-malicious	

1.5.5 Available Code

For the needs of this dissertation, several code and software were created, all freely available under GPLv3, enlisted in Table 1.

1.6 Dissertation Contribution & Outline

This dissertation's contribution and chapters are described here below:

- **Chapter 2** approaches RPL from the Software-Defined Networking (SDN) perspective, exploiting its high customization features to address the above-described inefficiencies. Building over the widely deployed RPL protocol, innovative solutions are presented while maintaining compliance with its standard.
  In detail, two routing control strategies are exploiting the global view of the network: (i) one that enables dynamic reconfigurability of crucial protocol parameters to improve its operation in mobile environments; and (ii) one utilizing a newly proposed RPL Objective Function (OF) that enforces direct *point-to-point* paths through link-coloring. Both solutions are implemented and evaluated on a novel centralized routing control facility.
  The subsequent experimental analysis considers hybrid scenarios with both fixed and mobile nodes, as well as many-to-one and *point-to-point communication*. The proposed mechanisms in which those experiments were tested do not require adaptations in the RPL standard and are being integrated into CORAL [19, 6], a novel centralized control facility supporting: (i) network control mechanism extensibility; (ii) abstracted protocol configuration APIs; and (iii) GUI interface and experimentation features for protocol configuration and measurement visualization. Moreover, CORAL, based on the WiSHFUL architecture [20], offers suitable abstractions and interfaces for dynamic protocol adaptations. An initial investigation of CORAL and the *Moderate RPL control* strategy can be found in [6], while a demonstration of this work is also described in [19]. Compared to the standard RPL in mobile topologies, the proposed solution achieves an improved packet delivery ratio of up to 33 percent and 21 percent for the mobile nodes and for the whole network, respectively, while maintaining RPL compliance. In the case of *point-to-point communication* in a random topology, the improvements rise to 32.7 percent for the trip-time and 42 percent for the round-trip time. In comparison, the

packet loss ratio for the same experiment is improved up to 75 percent in the non-storing mode.

- **Chapter 3** is an extensive literature review of RPL security challenges, focusing primarily on the existing solutions of Intrusion Detection Systems addressing those challenges.
  More precisely, a coherent investigation of RPL-related IDSs according to a novel conceptual framework that defines a three-step methodology was implemented.
  1. The first step concerns the *requirements' definition* that a successful IDS should address. The starting point is a better understanding of the problem IDSs tackle, i.e., the mitigation of attacks.
  2. The second step identifies the *best practices & gaps* out of an extensive literature review in respect to the defined design requirements. The goal is to realize the best approaches of existing works addressing the requirements, understand their evolution, and identify associated open issues. The 22 most recently introduced RPL-related IDSs in the literature ($2013-2020$) were thoroughly investigated. There is a discussion on their classification in respect to their detection method and their placement strategy. Then, a timeline of their evolution stages along with their principle qualitative (i.e., detection method, placement strategy) and quantitative features (i.e., number of attacks) was built, along with a discussion on requirements and classification criteria of each IDS.
  3. The last step involves a synthetic process producing the investigation's outcome, which is to *introduce design guidelines* for up-to-date IDSs. The outputs of the steps mentioned above are consolidated by first, including mapping the IDSs to the attacks they tackle. Secondly, there is a summarized comparison viewed under the design requirements introduced. For the attacks' mapping, both attack detection supported by simulations and those discussed conceptually only are considered. For compliance with the requirements, the respective authors' claims in the IDS' relevant articles were considered. There is also a comprehensive comparison that produces and elaborates on four crucial design guidelines for future up-to-date IDSs.

- **Chapter 4** deals with an innovative Intrusion Detection centralized controller, monitoring an IoT network. For the moment, the IDS is focusing on RPL protocol, but it can be easily expanded to cover existing or future such protocols. More specifically, *ASSET* (A Softwarized intruSion dEtection SysTem for RPL) is a novel hybrid anomaly and specification-based Intrusion Detection System (IDS) consisting of machine learning algorithms and well-known mechanisms and components, plenary operating together. *ASSET* is offering a holistic approach against many different types of RPL-related attacks, inspired by the centralized SDN paradigm, i.e., it offloads computational and communication overhead to the centralized infrastructure. At the same time, it follows a modular architecture, allowing easy future additions and adaptations. *ASSET* offers (i) a workflow hosting data-analysis system, (ii) a set of mechanisms for attacks' detection, attacker identification, and malicious activities mitigation, and (iii) an adaptable control & monitoring protocol al-

lowing centralized network supervision and bearable overhead. Moreover, the detailed experimental analysis conducted with *ASSET* offers (i) several proof-of-concept simulation results demonstrating the attacks' impact on the network control packets' overhead, along with *ASSET's* detection and mitigation responses, and (ii) a set of multiple experiments exhibiting how *ASSET* successfully detected as many as 13 different types of RPL-attacks with high accuracy and moderated cost, while remaining RPL-compatible.

- **Chapter 5** concludes the dissertation, also discussing further improvements and future challenges identified through the experience gained from the implementation of this research endeavor.

In summary, this dissertation contributes to the body of knowledge with:

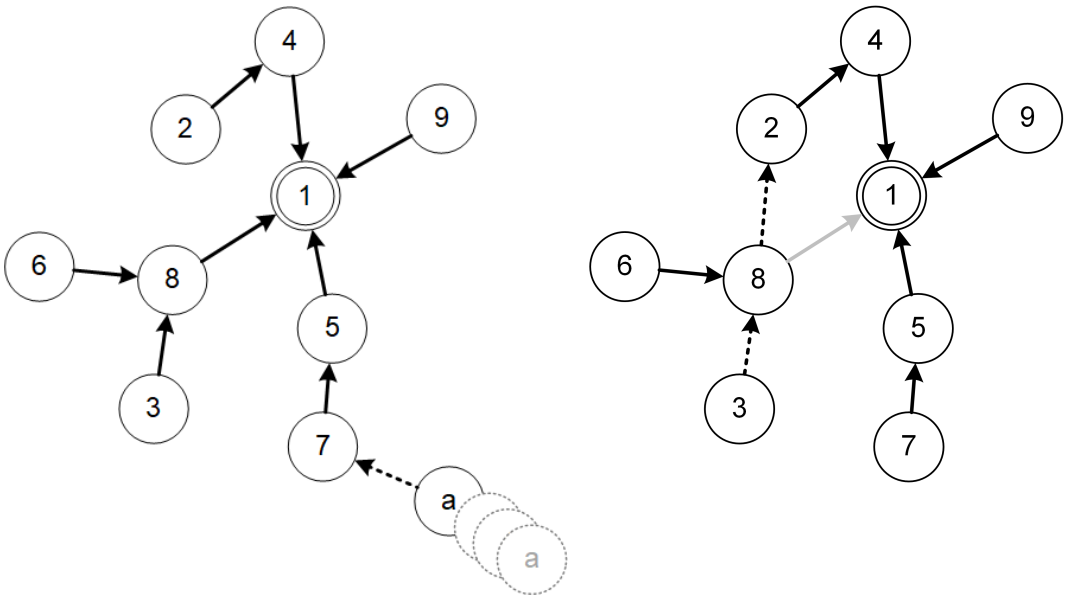
- New, innovative ways of tackling the point-to-point communication and mobility weaknesses of RPL by utilizing custom-made algorithms complying with RPL, under the prism of selective network-softwarization [5, 6],
- New, adaptable, dynamic parameterization of RPL, depending on the given network's characteristics, by considering the catholic network view provided again by utilizing the network-softwarization paradigm [21],
- A detailed categorization and deep analysis of RPL's security challenges, along with a detailed analysis of existing Intrusion Detection Systems for the same, and their characteristics [15],
- A state-of-the-art SDN-like, centralized IDS for IoT-RPL networks, mitigating more attacks than any other similar system described in the literature, also able to identify and ostracize the intruder in many cases [22].

2 End-to-End Communications & Mobility for RPL

This chapter presents the solutions designed for two fundamental open issues of IoT and RPL-protocol: (i) point-to-point communication in such networks, and (ii) how to handle more efficiently the mobility of nodes. More specifically, here are presented (i) a novel, custom-made, centralized infrastructure—CORAL—to monitor and adapt parameters of an IoT network in real-time, (ii) innovative solutions and algorithms regarding the problematic communication in RPL point-to-point, and (iii) solutions on adapting via CORAL the IoT network parameters to support better the nodes' mobility issues. Moreover, there is extensive testing of the proposed solutions via experiments conducted on the top of an IoT emulated network with various topologies, among them a random-placement one. Utilizing CORAL, the exact topology created by RPL with and without the offered solutions and algorithms is examined. The results, in all cases, show the superiority of those newly added solutions.

2.1 Introduction

Augmenting Wireless Sensor Networks (WSNs) with Internet connectivity enabled an essential class of IoT applications on area, machine, or people monitoring, such as for the protection of the environment, industrial processes control, and personalized healthcare systems [23, 24, 25]. In this context, *efficient routing* is a challenging issue, primarily due to the power, storage, memory, processing, and signal limitations of the connected devices [26].



(a) Mobility **(b)** *Point-to-point communication*

Figure 2: An abstract view of the RPL's performance issues

The most important relevant proposal is RPL, a distance-vector IPv6 routing protocol for Low-power and Lossy Networks (LLNs). In practice, RPL organizes network nodes, e.g., motes of a WSN, as a Destination-Oriented Directed Acyclic

Graph (DODAG) routed to a single destination called sink [3, 7]. The sink is the only node that can launch the DODAG's (re)construction based on a periodic exchange of routing control messages. The DODAG's maintenance is placed at the very core of the RPL's functionality. Hence, the following two essential aspects impact the network performance significantly: (i) efficient scheduling of the topology control messages from a dedicated algorithm called the trickle timer where I_{min} and $I_{doubling}$ are the particular RPL configuration parameters which tune the messages' time interval from I_{min} up to $I_{min} * 2^{I_{doubling}}$; and (ii) an appropriate node-parent selection and link quality estimation based on a relevant Objective Function (OF).

2.2 Motivation

The more complex IoT scenarios and applications become, e.g., involving *mobility* and communication beyond measurement collection from a single node, such as *point-to-point (P2P)*, the more they are questioning the applicability boundaries of the existing protocols' solutions, inherited by the WSNs. For example, the RPL topology maintenance features are particularly adjusted to energy efficiency and long-term periodic sensing over fixed motes' deployment [4].

In the case of the topology shown in Fig. 2a, the default trickle timer starts from roughly 4 *ms* (i.e., $I_{min} = 2^{12}$) and gradually increases (actually doubled), defining the interval time of the topology discovery control messages accordingly. This is a typical approach when a fixed topology of nodes (e.g., 2 to 9) gather and send measurements to the sink-node (e.g., node 1). Upon the appearance of a new mobile node (e.g., node *a*), the above "conservative" configuration of gradually increased interval time would lead to extensive delays in discovering the newcomer, since the trickle timer reaches relatively high values, e.g., up to 17.5 *min* (i.e., $2^{12} * 2^8$). On the other hand, flatly decreasing the nodes' trickle timer, e.g., independently of their mobility behavior, can lead to pointless energy consumption. We argue that the fixed nodes require different treatment regarding RPL configuration compared to the mobile ones. Hence, a centralized control approach defining RPL's parameters per node can be beneficial for the network.

Apart from data gathering, the same nodes' deployment could temporarily serve the need for direct communication between two nodes, as shown in Fig. 2b. For example, a node that monitors a threshold crossing value may urgently require to trigger an alarm to a node that is not the typical sink; for instance, we assume that the node 3 has to communicate with the node 2. RPL supports two approaches in implementing *P2P communication*: (i) the storing mode, where each node stores locally the DODAG; and (ii) the non-storing mode, where each node knows its parent and only the sink node maintains the DODAG. In both cases, the DODAG is constructed with the path towards the sink as a target, that is, $3 \rightarrow 8 \rightarrow 1$; paths between nodes are neither stored nor discovered in full. In such a case, forcing an emergency data flow from $3 \rightarrow 2$ simply requires that the node 8 should switch from the "parent" 1 to the new "parent" 2. This "surgical" switching, significantly improves the $3 \rightarrow 2$ communication. At the same time, it has a minor impact on the rest of nodes' communications, e.g., in Fig. 2b only the nodes 6 and 8 are affected by the change. We argue that many WSN deployments have temporal requirements for *P2P* apart from the typical many-to-one communication. Such requirements can be served by centralized routing control mechanisms which, exploiting the underlying RPL graph, could proceed with local amendments.

Figure 3: Urban environmental monitoring

RPL supports several configuration parameters and OF customization(s) that cover a wide range of alternative deployments [3, 7]. However, customizing such configurations is basically manual, global, and often unpredictable in terms of the outcome. The above two examples highlight the need for routing control mechanisms that appropriately handle the requirements defined by the application, i.e., configure network nodes individually, or "surgically" change the topology graph, an approach aligned to the Software-Defined Networking (SDN) paradigm. Such concepts are discussed below.

2.3 Contribution

Here, we suggest that the high customization capability of the RPL protocol can enable the appropriate SDN-inspired routing control strategies to extend the applicability of the protocol beyond the typical WSN scenarios. Hence, we propose two SDN-inspired network control mechanisms improving the RPL's behavior in mobile and *P2P communication* contexts, reflecting different depths in the protocol adaptation:

- The *Moderate RPL control* that enforces appropriate protocol configurations in an on-the-fly manner to improve mobile communication. For example, it allows mixing RPL configurations, i.e., treating differently mobile and fixed nodes. In a typical WSN data collection application, where battery-powered mobile nodes may be employed to widen the monitoring area, fixed nodes should broadcast topology discovery messages frequently to provide connectivity chances for the mobile ones, while the latter relaxes their discovery intervals to save energy.

Figure 4: The harsh workplace of a mine

- The *Deep RPL control* that establishes *P2P communication* in consistence to the RPL topology graphs, exploiting the feature of changing the OF. More specifically, it utilizes the idea of link coloring to appropriately color the network nodes and enforce direct communication paths based on those. When a measurement of a sensor-node crosses a threshold, it may trigger a rule in an alert and action application that defines an urgent message transmission to a particular recipient-node instead of the regular post to the sink-node. A centralized controller can specify the relevant direct path that a suitable OF can enforce.

Our proposed mechanisms do not require adaptations in the RPL standard and are being integrated into CORAL [19, 6], our novel centralized control facility supporting: (i) network control mechanism extensibility; (ii) abstracted protocol configuration APIs; and (iii) GUI interface and experimentation features for protocol configuration and measurement visualization. We implemented CORAL based on the WiSHFUL architecture [20], offering suitable abstractions and interfaces for dynamic protocol adaptations. An initial investigation of CORAL and the *Moderate RPL control* strategy can be found in [6], while a demonstration of this early work is described in [19]. Our experimental results in the current work confirm that extending the RPL with novel SDN-inspired routing control features can tackle its inefficient performance in the cases of heterogeneous topologies consisting of

both fixed and mobile nodes, as well as applications occasionally requiring *P2P communication*.

To highlight the impact of our proposal, we discuss two representative use-case scenarios in Section 2.4. The architecture of the centralized routing control facility, along with the two proposed mechanisms, are described in Section 2.5. Section 2.6 presents a detailed experimental analysis. The related works are discussed in Section 2.7. Finally, we provide concluding remarks and future work insights in Section 2.8.

2.4 Use-Case Scenarios

To further motivate our work, we elaborate on two use-cases with characteristics aligned to the capabilities of the proposed SDN-like routing control schemes: (i) an *urban environmental monitoring* deployment based on data collection; and (ii) a *harsh workplace* scenario with occasional alert and action communication requirements.

2.4.1 Data Collection in Urban Environments

In the first use-case, we consider an urban environment with several motes scattered downtown gathering measurements (e.g., pollution or weather-related). We assume that fixed nodes connected to a mains power supply coexist with battery-powered mobile ones for complete area coverage. Fig. 3 shows that the yellow-colored mobile nodes communicate through the red-colored fixed nodes, given that there is radio connectivity among them.

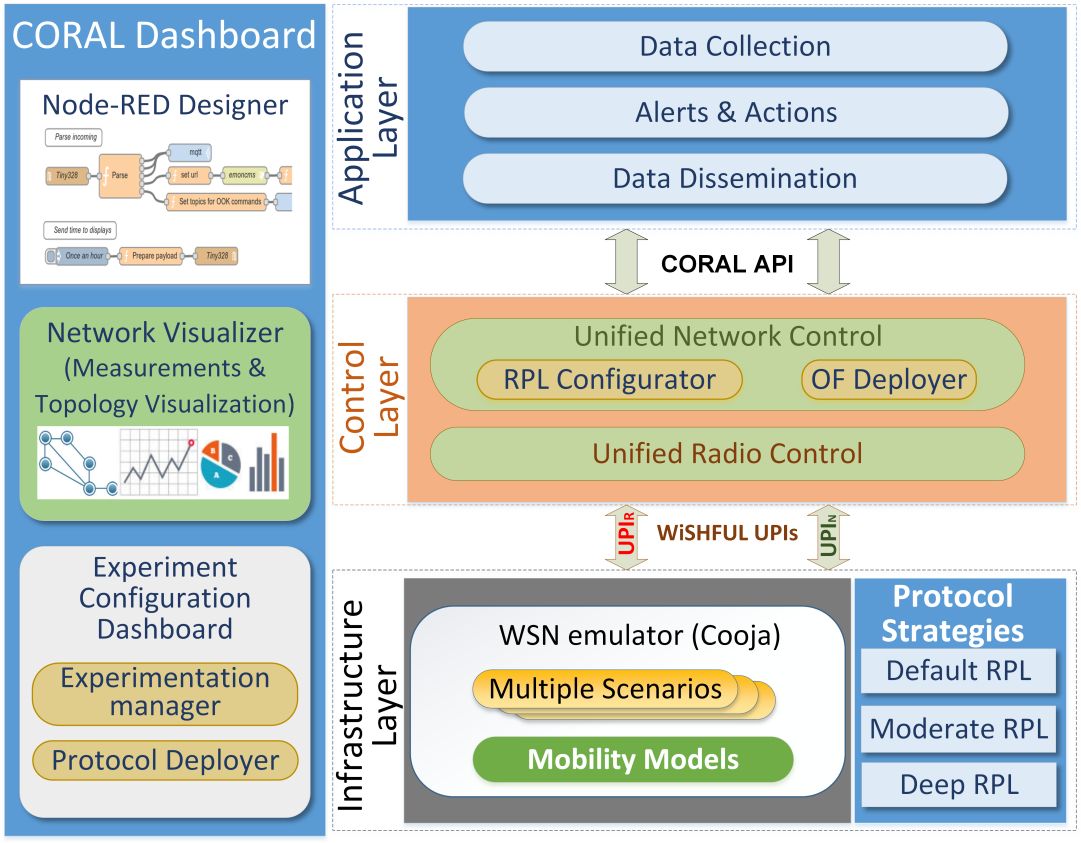


Figure 5: The CORAL architecture

According to our previous work [6], in cases like this, an optimal routing control mechanism should take into account the different resource-capabilities of the fixed and mobile nodes. For example, RPL should adapt the fixed nodes to probe more frequently for mobile ones into their vicinity, while the latter should be more conservative to preserve energy. Practically, a platform like CORAL should be able to tune, from a global viewpoint, the RPL routing configuration in respect to the nodes' mobility profile. This will allow an extended network coverage since the mobile nodes can be detected and incorporated into the monitoring network without delays.

2.4.2 Alerts and Actions in Harsh Workplaces

In the second use-case, we consider a harsh workplace environment like the mine shown in Fig. 4. A set of nodes are deployed in the area, such that the fixed ones monitor the environment (e.g., oxygen level) while the mobile wearables collect the miners' vital signs. Under typical operation, this deployment exhibits similar requirements with the first use-case.

However, suppose a miner suddenly faces an emergency, indicated by his vital signs monitoring device. In that case, there is a need for emergent communication between him and the operation center on the ground surface or to another worker with medical training. In such a case, high priority should be given to the messages originated from the miner in danger, even sacrificing the rest devices' communication. As identified in several works [12, 27] and shown in our experimental analysis below, RPL is inefficient for the direct communication between two end-points. Relevant scenarios have been described in [28, 29, 30], focusing on industrial cases of emergency, a large-scale industrial environment, and an oil refinery scenario with real-time constraints, respectively.

Here, we argue that a centralized routing control facility like CORAL can enforce direct communication between nodes while maintaining the typical RPL operation for the rest network. To tackle this issue, our approach employs a new OF that enables link coloring to enforce direct *P2P* paths. The centralized controller appropriately colors the nodes to change a single or more nodes' parents intentionally, that is, to enable the direct communication while being aligned to the RPL's RFC [3].

2.5 Centralized Routing Control

To improve the performance of RPL in mobile or *P2P communication* contexts, we propose CORAL, a novel routing control facility accommodating two alternative centralized control approaches, i.e., the *Moderate* and *Deep RPL control*. Here, we provide a high-level overview of the CORAL architecture and describe the novel routing control strategies we propose.

2.5.1 The CORAL Architecture

The CORAL facility implements SDN-inspired routing control to evolutionary enable the applicability of RPL to alternative IoT use-cases, such as those described in Section 2.4. Fig. 5 illustrates the CORAL architecture, which follows the typical three-tier SDN paradigm. We describe the three layers in a bottom-up fashion below:

- The *Infrastructure Layer* accommodates multiple scenarios with diverse topologies and network settings, including support of realistic mobility mod-

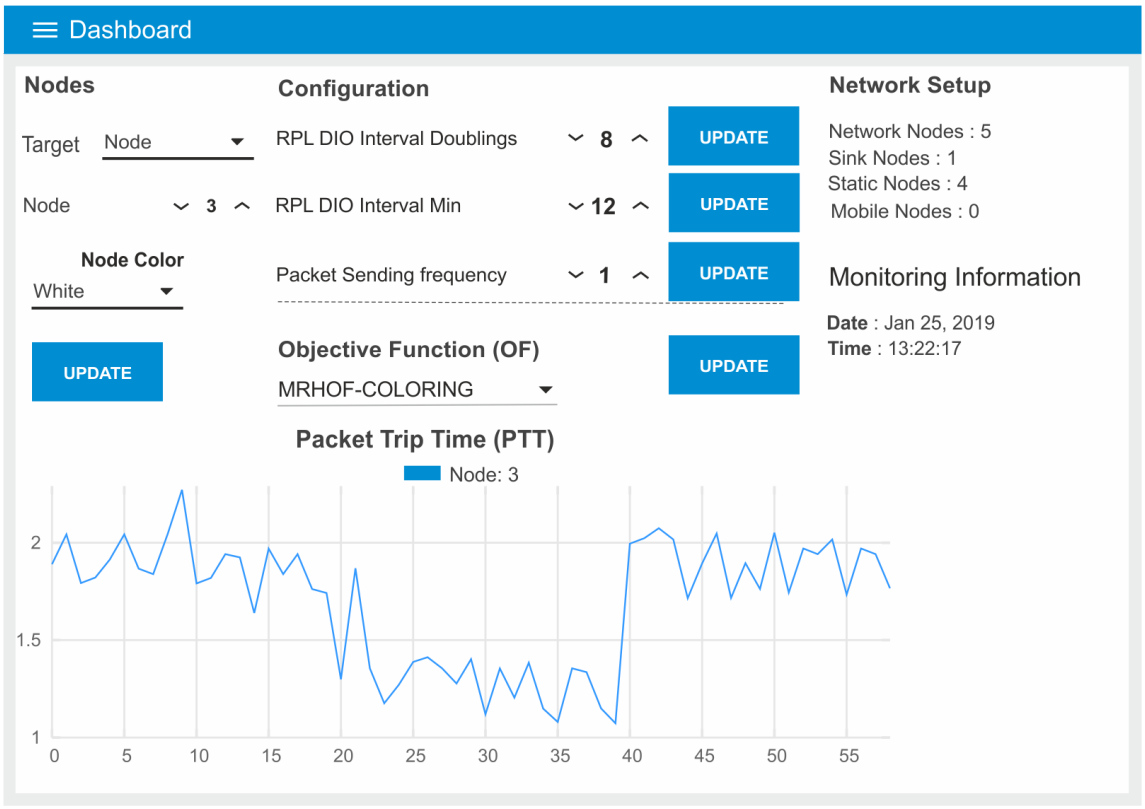


Figure 6: On-the-fly RPL configuration with the CORAL dashboard

els for the motes. Such scenario configuration can be both loaded at compile-time or dynamically updated in real-time, i.e., to implement dynamic IoT scenarios. In practice, we employ Ansible scripting for the offline Cooja emulator configurations and utilize the *Control Layer* for the online adaptations.

- The *Control Layer* offers abstracted and logically centralized control of the network. The network control abstraction accommodates two main components: (i) the *RPL Configurator* being responsible for configuring the RPL protocol (e.g., adapting parameters like the I_{min}) and running OF specific features (e.g., the link coloring in our case) to adhere with rapidly changing network demands; and (ii) the *OF Deployer* able to change the OF in real-time, when alternative communication requirements emerge. This layer currently supports protocol deployment via device-specific Ansible scripts updating the IoT devices' firmware (i.e., low memory-footprint approach with reasonable deployment time). Our plans also include experimentation with over-the-air protocol deployment approaches. The configuration/measurements of network protocol functionalities and radio are provided to the *Control Layer* through the *Universal Network and Radio Control Interfaces*, i.e., the UPI_N, for the RPL protocol, and UPI_R for the radio channel, developed by the WiSHFUL project [31]. UPIs are also utilized by the *Infrastructure Layer* to provide via the serial port or CoAP protocol to the *Control Layer* real-time measurements on the protocols' and network performance (e.g., the round-trip time).
- The *Application Layer* offers high-level protocol configuration that matches different application types (i.e., for data collection, alerts & actions and data dissemination). The applications express their demands (e.g., the need for

P2P communication, or to support mobility) to the *Control Layer*. At the same time, the latter matches them with particular protocol configurations, i.e., employing either *Moderate* or *Deep RPL control*. Such matching is currently hard-coded, but we consider relevant, intelligent algorithms as future work. The applications' requirements are being communicated to the *Control Layer* over the *CORAL API* through JSON messages.

The control facility interacts with the user through the *CORAL Dashboard*, a flexible and modular Graphical User Interface (GUI), implemented in Node-RED (http://nodered.org) and depicted in Fig. 6. We can either configure RPL parameters, such as the I_{min} and $I_{doubling}$, or the OF to rely upon, along with setting the network nodes' appropriate coloring. Node-RED flows are wired with JSON messages which pass updated RPL's configuration to the *Control Layer*. Live visualization of the outcome illustrates the experiments' progress and the impact of particular SDN-inspired protocol strategies on the application and network performance.

An initial version of CORAL[1] focusing on the *Moderate RPL control* approach can be found at [19, 6]. We now elaborate on the proposed routing control strategies.

2.5.2 Moderate RPL Control

The first approach to SDN-inspired network control of RPL focuses on its dynamic protocol configuration based either on environmental conditions or node heterogeneity (e.g., whether they are mobile or not). This network control scheme considers the RPL as a black-box. It does not change its primary mechanisms. Still, it exploits the available protocol configuration options to adapt it to particular conditions, especially supporting mobility. This strategy aligns with the requirements of the urban environment scenario described in Section 2.4.1.

To further detail the *Moderate RPL control* approach, we now present how the RPL constructs the network topology. The sink launches the DODAG's construction based on the exchange of routing control messages, i.e., DODAG Information Object (DIO), Destination Advertisement Object (DAO), and DODAG Information Solicitation (DIS). The sink issues a first DIO message, and then plenty of them are sent in multicast by nodes getting connected to the graph. The DAO messages are used by all nodes, except the sink, to propagate reverse route information. Finally, DIS messages are sent by disconnected nodes to solicit DIO messages from their connected neighbors and join the graph.

The DODAG's maintenance is placed at the very core of the RPL's functionality, and hence, a dedicated algorithm–the trickle timer–synchronizes the propagation of DIO messages upon which the graph's convergence time is based. The critical aspect in DIO multicasting process is to achieve a short period of the graph's setup time and, thus, to reinforce the network's metrics, e.g., the packet delivery ratio, while restricting the control overhead towards lowering the node's power consumption [4]. To achieve the aforementioned trade-off, the DIO messages are sent periodically, but their interval ranges from I_{min} up to I_{max}, where $I_{max} = I_{min} * 2^{I_{doubling}}$. For example, the default RPL configuration specifies $I_{min} = 2^{12} = 4{,}096\ ms$ and $I_{doubling} = 8$ which entails $I_{max} = 2^{12+8} = 17.5\ min$. The timer's duration is doubled each time it fires. Any change in the DODAG, e.g., an unreachable parent or a new parent selection, resets the trickle timer to I_{min}.

[1]The relevant source code and video can be found at `https://github.com/SWNRG/wishful-coral`.

Table 2: The DODAG's setup time for different network settings as a function of the RPL's I_{min} parameter

#	**No. of nodes**	**Heterog.**	**Topology**	I_{min}	**Setup time (sec)**
1	10	Y	Fig 2a	8 12	11.4 42.3
2	15	N	chain	8 12	6.2 50.9
3	15	N	lambda (Λ) - sink on top	8 12	4.8 26.8
4	30	N	as in [4]	8 12	5.1 23.0
5	30	N	chain	8 12	11.3 107.4
6	50	N	random	8 12	10.2 27.3
7	100	N	random	8 12	32.4 68.1

Table 2 reports the impact of the RPL's I_{min} parameter on the graph's setup time for different network settings, i.e., the number of motes, heterogeneity in motes' behavior (fixed and/or mobile), and topology type. For example, we consider the topology of Fig 2a with 10 randomly positioned motes, including both fixed and mobile ones. If an accident close to mote 7 at the 20 *sec* (e.g., an isolated miner loses his senses), the default RPL configuration fails to route its signal since the mote is not connected yet to the graph. We notice that it is essential to begin with an "aggressive" graph setup policy to ensure that all motes are being connected to the graph (e.g., within less than 12 *sec*), and successfully report their data even at the cost of control overhead. Actually, this cost at the given time is not an issue for the successful data delivery since data cannot be collected until the routing graph is fully constructed.

A critical aspect hindering the applicability of RPL in challenging IoT use-cases is the following: the topology probe interval gradually increases and produces a delayed response to the topology changes caused by mobility [32]. This is attributed to the inherent focus of the RPL design on static networks with limited local adaptability [33], since the RPL specifications do not cover when and how DIS messages should be sent [34]. We argue here that RPL has the potential to improve its behavior in mobile environments by adjusting its main configuration parameters or mechanisms.

In the case of *Moderate RPL control*, a centralized control facility provides the options of dynamic and individual motes' configuration. More precisely, we start with a minimum I_{min} to set up the DODAG as quickly as possible and then continue with low I_{min} values for the fixed motes and high for the mobile ones, i.e., to alleviate the control overhead. To save precious time, the CORAL gives the option to enforce such strategies on-the-fly. Furthermore, live network monitoring enables early

detection of abnormal or inefficient routing behavior, new configuration decisions, and dynamic/individual enforcement. Relevant strategies can be straightforwardly supported from the CORAL facility, involving all the RPL configuration parameters (e.g., I_{min}, $I_{doubling}$, or OF parameters described afterward).

More details on the *Moderate RPL control* can be found in our conference paper [6]. Next, we introduce the *Deep RPL control* mechanism.

2.5.3 Deep RPL Control

In this second SDN-inspired routing control approach, the controller is more deeply involved in the protocol operation, but again consistent with the RPL standard. The *Deep RPL control* does not handle the RPL as a black-box, in the sense that it goes beyond its parameters' configuration. In practice, it enables the ability to change the OF upon which the DODAG is constructed. The *OF Deployer* changes the OF either pro-actively or re-actively, in response to communication requirements, and *RPL configurator* proceeds with appropriate customization in line with the OF deployed. The goal of this control mechanism is to improve the *P2P communication* of the RPL protocol in correspondence to the requirements of the mining scenario described in Section 2.4.2.

Before further detailing our proposal, we discuss the most common OFs used by the RPL and the way that the protocol handles the *P2P communication*. The RPL is a link-state routing protocol. Its topology (mapped to a DODAG) is constructed by the independent decisions of each node regarding their best parent among potential candidates, based on criteria defined in the particular OF used.

The RPL mainly uses the OF Zero (OF0) [35], and the Minimum Rank with Hysteresis OF (MRHOF) [36] (which can be found implemented in IoT environments, such as the Contiki OS [37]. The OF0 returns an output (set of selected parents) that optimizes (i.e., minimize) the number of hops to the sink node. It provides the network topology quickly consuming the minimum resources, although the solution is non-optimal regarding more sophisticated criteria. For example, the MRHOF selects node parents optimizing (i.e., minimizing) the Expected Transmission Count (*ETX*), which is the expected number of transmissions from a node to a destination to successfully deliver a packet [3]. The default formula is $ETX = 1/(Df * Dr)$, where Df is the probability for a packet to be received by a particular neighbor, and Dr is the probability that the acknowledgment packet is received successfully. The MRHOF allows the addition of new metrics and uses the hysteresis mechanism [36] to avoid frequent switching between parents due to minor metric changes. All parents' selections are being communicated up to the tree topology until the sink node, which is though aware of the full network graph.

This setup facilitates many-to-one communication since each node passes the packets to its parent until they reach the sink. However, parent information is not enough for communication towards a non-sink node since this may require communication down the routing graph. To address this issue, the RPL follows two alternative approaches: the *storing* and the *non-storing mode*. In the *storing mode*, each node maintains the portion of the DODAG starting from the its-own rank and towards the sink, whereas in the *non-storing mode*, only the sink holds the full topology information. These approaches tune the involved performance trade-offs differently: the *storing mode* trades memory state for lower communication delays, while the *non-storing* adopts the opposite strategy. However, both of them

Algorithm 1: Parent selection towards establishing a *P2P* path

Result: Establish a *P2P* path in RPL
Input : Color *node_color*, candidate parent $p1$, candidate parent $p2$
Output: Selected parent ($p1$ or $p2$) for a node colored *node_color*

```
1  begin
2      //if a receiver is found by a red node, make it a
        parent
3      if p1.node_color==orange then
4          if this.node_color==red then
5              return p1;
6      if p2.node_color==orange then
7          if this.node_color==red then
8              return p2;
9      switch this.node_color do
10         case orange do
11             //receiver chooses parent based on the ETX
12             return p1.ETX< p2.ETX? p1 : p2;
13         end
14         case purple do
15             //sender chooses a red parent
16             if p1.node_color==red then
17                 if p2.node_color==red then
18                     return p1.ETX< p2.ETX? p1 : p2;
19         end
20         case red do
21             //red nodes choose other red parents
22             if p1.node_color==red then
23                 if p2.node_color==red then
24                     return p1.IP< p2.IP? p1:p2;
25                 else
26                     return p1;
27                 end
28             else
29                 if p2.node_color==red then
30                     return p2;
31                 else
32                     return p1.ETX< p2.ETX? p1:p2;
33                 end
34             end
35         end
36         case white do
37             //white nodes are choosing parents based on the
                ETX
38             return p1.ETX< p2.ETX? p1:p2;
39         end
40     end
41 end
```

are inefficient concerning the need for a potential *P2P* path.

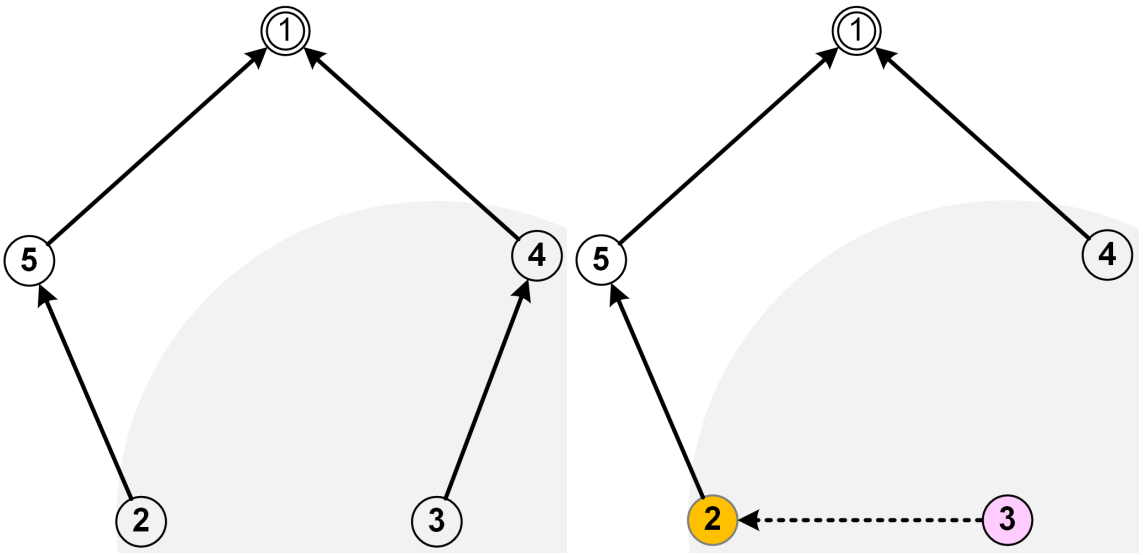
To enable such an option, we added a new mechanism in the CORAL controller, as part of the *RPL Configurator*, which calculates the direct path between any two nodes: the controller is running a second version of the DODAG, where the receiver acts as the sink, and stores the path created between the sender and the receiver. This path is now "implanted" into the existing DODAG with the least possible impact (performance-wise) on the rest of the network nodes' communication.

To enforce the desired *P2P* path in real-time, we utilize a new OF, namely the MRHOF-C(oloring). This OF exploits the Link Coloring (LC) feature of the RPL that can be implemented either as a metric or constraint, i.e., to attract or avoid specific links [38]. In practice, the *RPL Configurator*, which is part of the CORAL controller, is utilizing the UPIs to set the nodes' color, i.e., manipulating the LC along the desired *P2P* path. It utilizes four different sets of nodes/colors: sender/purple, receiver/orange, nodes-in-path/red, and nodes-non-in-path/white. Once the coloring phase is completed, the controller refreshes the topology in real-time either globally (i.e., a global repair takes place when the whole network resets the DODAG) or locally (i.e., local-repair enables a reset mechanism for a node and its sub-DODAG only). Global or local repairs in the topology are being handled by the Algorithm 1, which is executed in each node; some nodes should alter their parent selection to facilitate the establishment of the desired *P2P* path. The algorithm takes as input a node along with its color and two candidate parents (i.e., $p1$ and $p2$), and outputs the preferred parent according to the proposed MRHOF-C OF; in particular, it forces the purple and red nodes to select a red parent (red nodes lie along the path), if there is one available (lines $14-19, 20-35$), while it lets the orange and white nodes to go through the default *ETX* metric selection (lines $10-13, 36-39$). The orange (receiver) node is prioritized as a parent only by a red node (lines $3-5, 6-8$). Hence white nodes are not changing parents, minimizing the disturbance in the network. If multiple candidate parents exist, the Algorithm 1 is repeatedly executed by the RPL until the best among them is selected.

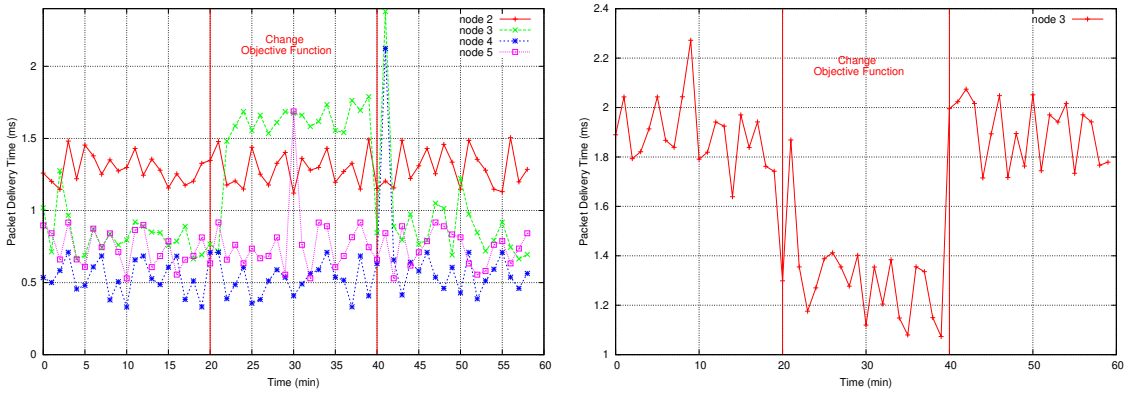
This centralized routing control interferes in the topology graph with few local adjustments: it improves the performance of a particular *P2P communication* changing the parents of a small subset of nodes. In addition, the performance overhead for the many-to-one communication of the rest nodes is minor since the new control mechanism is also based on a topology derived by a DODAG. Section 2.7 contrasts our approach to other relevant works handling the *P2P communication* when the RPL protocol is used.

2.5.4 Implementation details

To realize the *Deep RPL control*, our proposal enables the dynamic deployment of alternative OF(s) (*OF Deployer* component), along with the run-time configuration of such OFs by the controller mechanisms (*RPL Configurator*). The parameters of each OF are globally accessible through the UPIs. The dynamic OF deployment is an existing RPL protocol feature. In particular, the RFCs 6550 [3] and 6551 [38] state the need for RPL improvements and adaptations to certain conditions, although such feature was not yet part of a centralized control facility. Some proposals [32, 34] proceed with OF changes but only in compile-time and not dynamically. Our implementation exercise takes place in a Contiki-based experimentation environment. For the LC feature, we extended the DAG Metric Container



(a) Initial DODAG **(b)** Forced path in the DODAG



(c) PDT per node after path enforcement **(d)** PDT in $3-2$ *P2P communication*

Figure 7: Proof-of-concept experiment

(MC), defined in [3, 38], as the way that nodes use to report alternative or additional metrics along the DODAG. The LC parameter expressed as a 10-bit link constraint (assigned the value 8 by IANA) can be either adjusted statically or dynamically [38]. A new *node_color* parameter has been added in each node to be manipulated by the CORAL centralized controller through the UPIs. Such parameter is then propagated through the LC in the corresponding MC. To avoid loops, the nodes' id (reflected to the last number of the IP address) is decreasing from the sender to the receiver, so the Algorithm 1 chooses the smallest id if two red parents are found.

In cases where the MRHOF-C is employed re-actively, the CORAL platform triggers the global and local topology repair mechanisms, described in [3, 4]: each node nullifies all its connections and starts again sending solicitation messages looking for potential parents; nodes in the vicinity (i.e., within radio coverage) are evaluated through the OF as potential parents. We noticed that if we proceed with a Global-repair, it takes too long for the sender node to re-calculate routes with the new OF, a phenomenon well connected with the frequency of sending solicitation messages (DIS), as described in Section 2.5.2. A direct result of triggering such repairs is an anticipated increase of the control overhead momentarily.

In cases where the MRHOF-C is employed pro-actively, it can be initially used with no coloring customization (actually simulating the MRHOF). Once the need for *P2P* communication emerges, the *RPL Configurator* sets the nodes' colors, the Algorithm 1 outputs the new selected parents, and the desired path is established. This way, only Local-repairs are needed; hence converging time is smaller.

A proof-of-concept experiment, depicted in Fig. 7, is utilizing RPL and the default OF, i.e., MRHOF, to create the network topology shown in Fig. 7a. The packet delivery time (PDT) for each node is depicted in Fig. 7c. At 20 *min*, the OF changes to the MRHOF-C to improve $3 \rightarrow 2$ communication. A new DODAG with the forced path between the sender-node 3 and the receiver-node 2 is created and depicted in Fig. 7b. The delivery time for all nodes against the sink remains unchanged, except that of node 3, which is slightly increasing, since packets have to travel the longer path $3 \rightarrow 2 \rightarrow 5 \rightarrow 1$; the green line depicts this increase in Fig. 7c during the period $[20,40]$ *min* of OF change. On the contrary, the direct $3 \rightarrow 2$ path improves the delivery time for these nodes' communication as presented in Fig. 7d. For the last 20 *min* of the experiment, the MRHOF is reinstated, reverting the network to its initial behavior (Figs. 7c, 7d). The semantic of this experiment is twofold: firstly, our controller and its components, i.e., *OF Deployer* and *RPL Configurator*, can offer a point-to-point communication keeping consistency with the RPL standard; secondly, our solution is beneficial for the desired communication without causing delays on the rest network.

2.6 Performance Evaluation

In this section we evaluate the proposed routing control strategies, i.e., the *Moderate RPL* and the *Deep RPL control* against handling *mobility* and *P2P communication*, through a number of experimental setups. The CORAL platform acts as an enabler for the tough experimental task since it accommodates them and facilitates their deployment, configuration, execution, and real-time monitoring through the dashboard.

2.6.1 Experimental Methodology

More precisely, for the *mobility* issue we employ the *Moderate RPL* using real mobility traces derived by the MONROE H2020 EU project [39], which provides open access, flexible hardware and software platform for extracting measurements and carrying out custom experimentation on Mobile Broadband (MBB) networks across Europe. As such, the MONROE database includes vehicles' movement trace data (i.e., moving buses, trains, and tracks) from many European cities. We extracted real mobility traces from Stockholm buses, transformed the nodes' GPS coordinates to a 150 $m \times 150$ m canvas in the Cooja simulator, and removed the idle times. Our experiments involve 5 mobile and 16 fixed nodes (including the sink) and last 60 *min*.

Through the CORAL GUI, we employ the *Moderate RPL* mechanisms to extract results regarding two metrics: the packet delivery ratio (*PDR*) defined as the received UDP packets ($rUDP$) over the total number of UDPs being send ($sUDP$), i.e., $PDR = rUDP/sUDP$; and, the control *overhead (OH)* which expresses the ratio of the control packets (CP) to the total packets in the network, i.e., $OH = CP/(CP+sUDP)$. All scenarios use the same deterministic mobility model, so there is no need to confirm the results' statistical accuracy. Cooja TX/RX parameter (i.e., the rate

Table 3: Experimental setup

Layer Setting	Description	Notes
Transport	UDP	
Network	IPv6 / RPL	
Adaptation	6LoWPAN	
MAC	CSMA	
Physical	IEEE 802.15.4	
Cooja ver	3.0	GitHub master branch
TX / RX	100%	TX / INT Range same
Transmission Range	50/50 m	according to [34]
UDP size	60 B	
Node traffic load	60 UDP/h	
Mobility Model	real traces	MONROE Project [39]

of successfully Transmitting/Receiving a radio message) was set to 100 percent to eliminate randomness, and Transmission/Interference ranges were both set to 50 m according to [34]. All experimental setup parameters are depicted in Table 3. Our comparative analysis for the *Moderate RPL control* uses the standard RPL (indicated as "Default") as a baseline case.

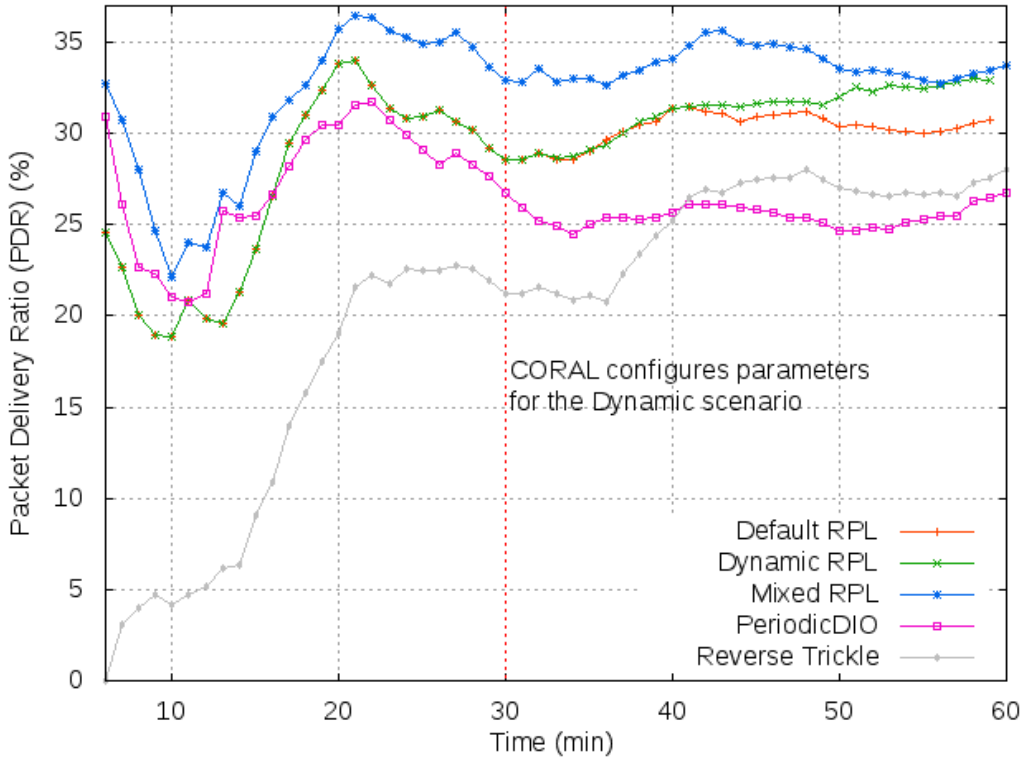
To handle *P2P communication*, we exploit the *Deep RPL control* and three distinct network illustrations, namely the Lambda (Λ), Neutral, and Random topologies. The first one is positively biased towards highlighting the advantage of forcing a routing path between two specific end-points; the second represents a deployment where our control mechanism does not significantly contribute since the existing paths can serve the desired *P2P Communication*; the last one depicts a random deployment to simulate a real-world case better.

The results in those cases regard two metrics. The packet delivery time (PDT) expresses the time that a UDP packet requires to travel between the two end-points of a *P2P* defined path. The PDT is further distinguished to the trip time (TT), if the one-way communication is considered, and the round-trip time (RTT) if both the forth and back routes are taken into account. The packet loss ratio (PLR) is defined as the ratio of packets lost over the total number of packets send, i.e., $PLR = (sUDP - rUDP)/sUDP$. The experimental analysis for the *Deep RPL control* contrasts the proposed OF MRHOF-C to the MRHOF, which is usually the default choice for the RPL protocol. Below, we present the detailed analysis for each control mechanism separately.

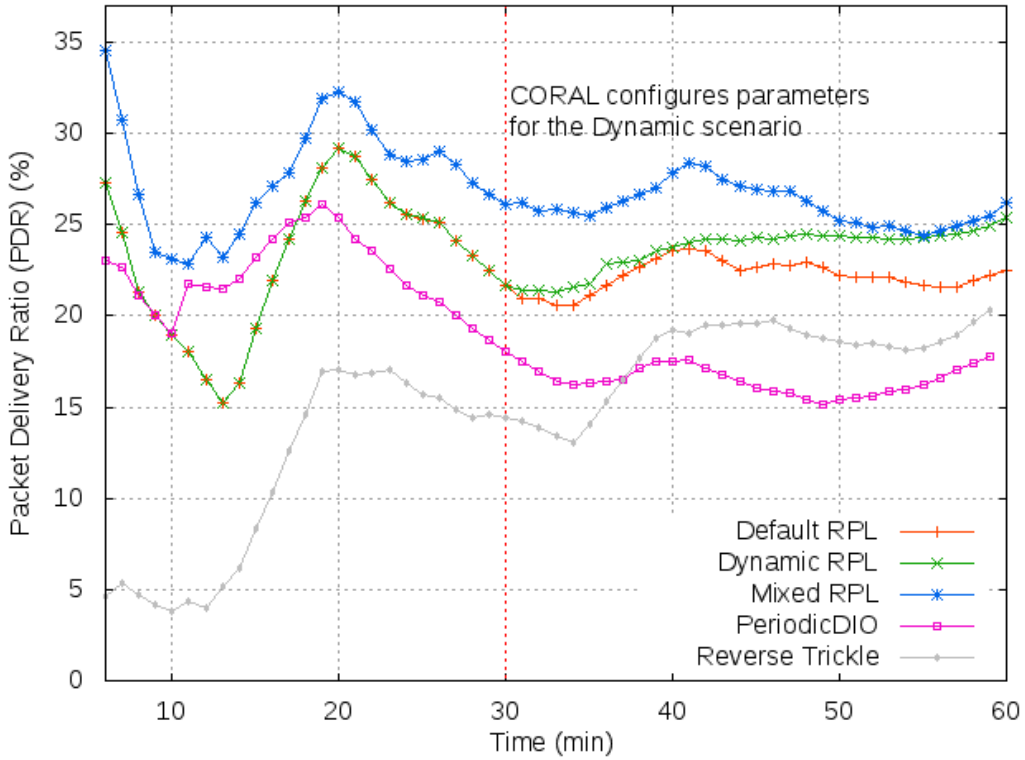
2.6.2 Moderate RPL Control Results

Although we experimented with different combinations of RPL parameters to evaluate the *Moderate RPL control*, here we focus on the I_{min} configuration since it is the most important RPL parameter for the context under study. Once the I_{min} value changes, the trickle timer is reset. We proceed with two different modification approaches, namely the *Mixed* (blue line) and *Dynamic* (green line), both compared to the Default RPL (orange line) in Fig. 8.

In the *Mixed* configuration, our platform configures the sink and the fixed nodes differently from the mobile ones at the very beginning of the experiment. The first have $I_{min} = 8$ and the latter are configured with $I_{min} = 12$ for the whole period of the



(a) The average PDR of all nodes as a function of time



(b) The average PDR of the mobile nodes' subset as a function of time

Figure 8: Cooja simulation with real mobility traces

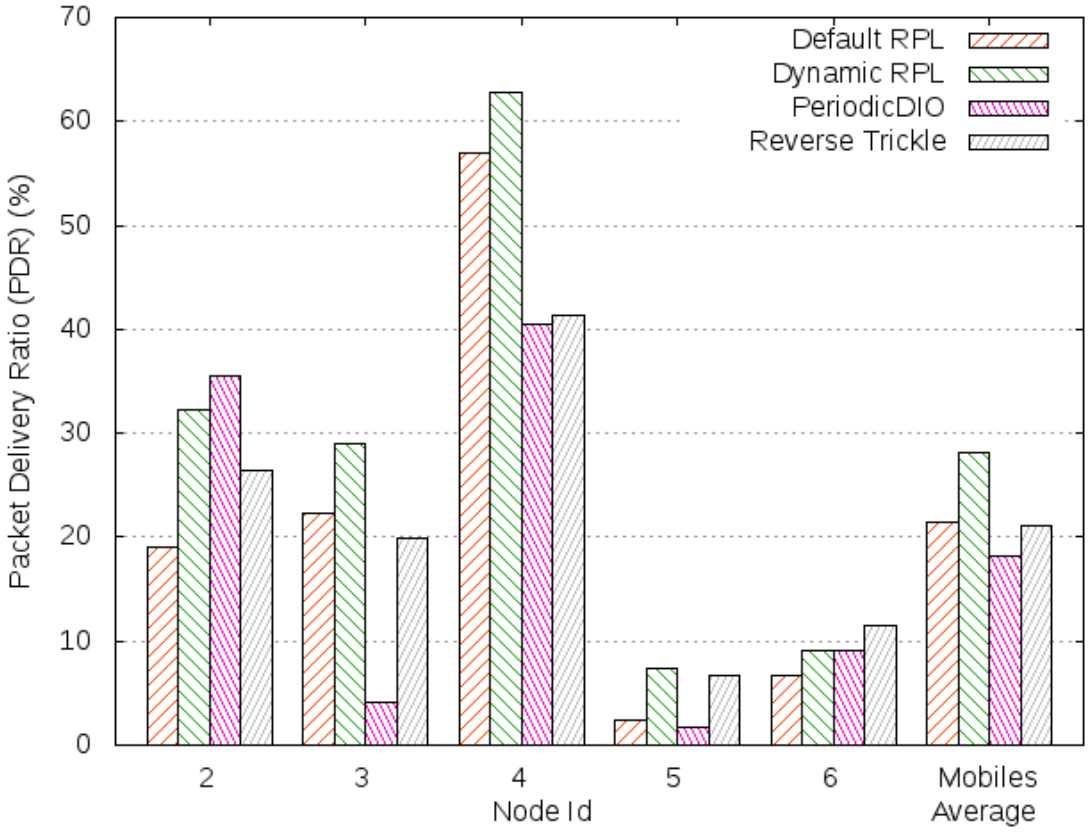


Figure 9: PDR per mobile node (2..6) in contrast to the total average values

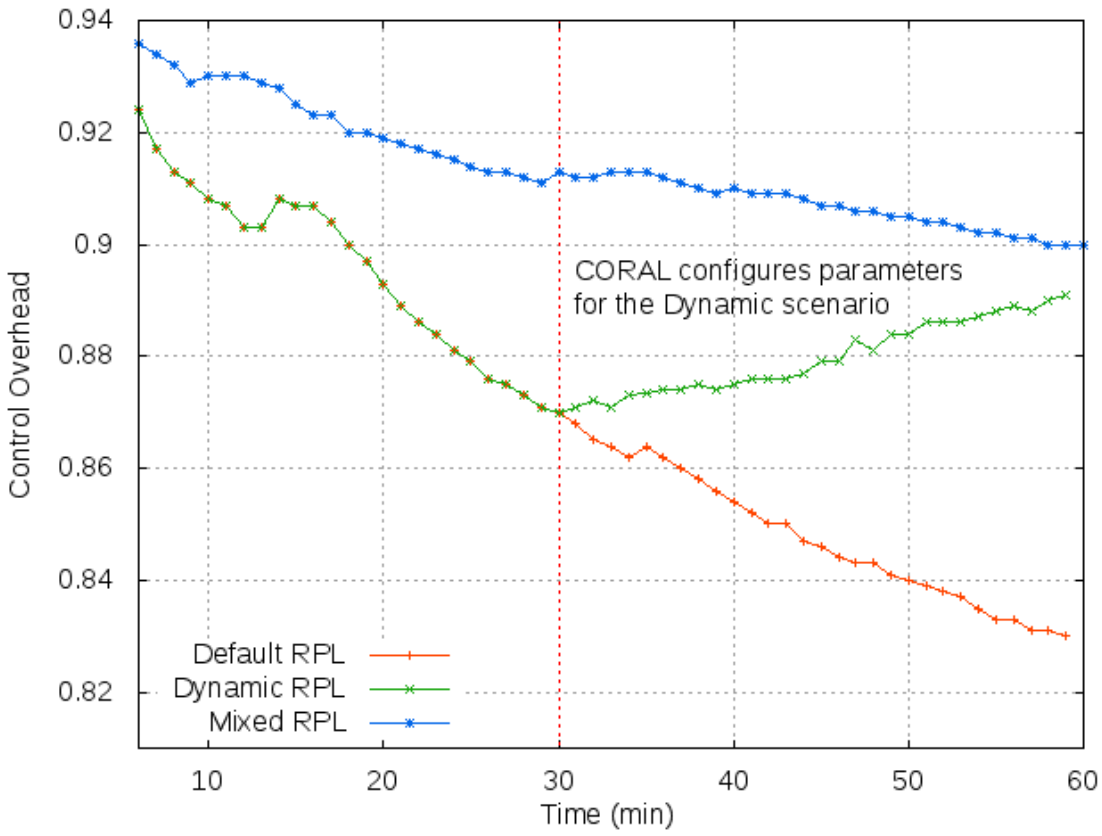
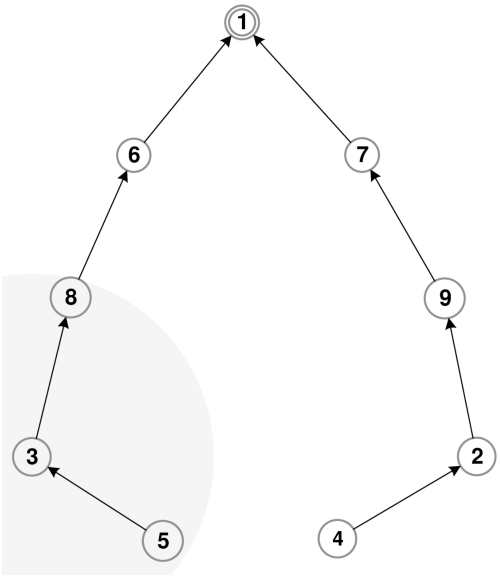
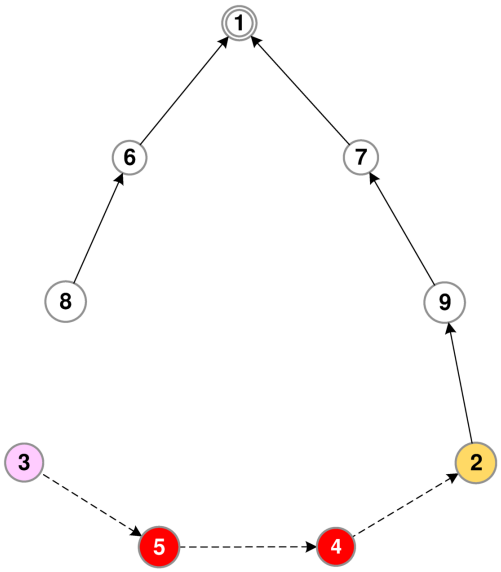


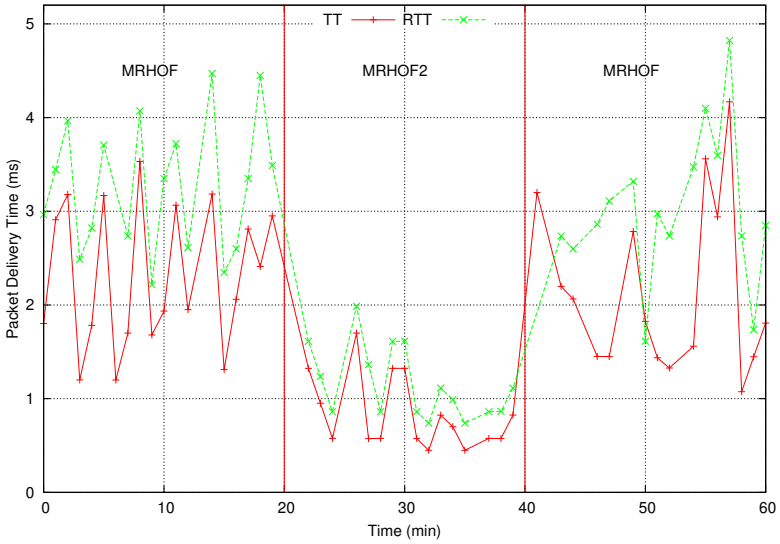
Figure 10: The network's control overhead



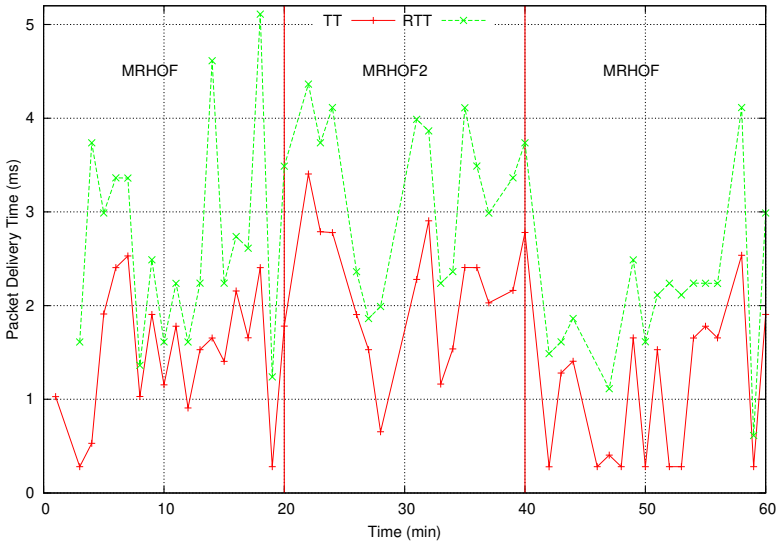
(a) Initial DODAG



(b) Forced path in the DODAG

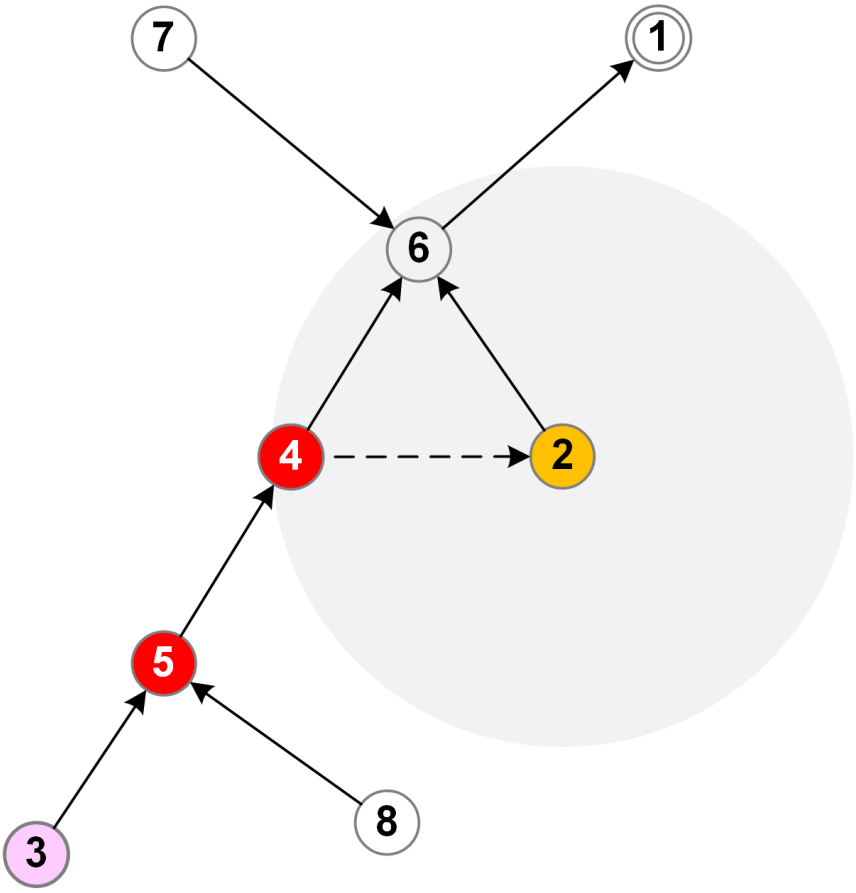


(c) PDT in $3-2$ *P2P communication*

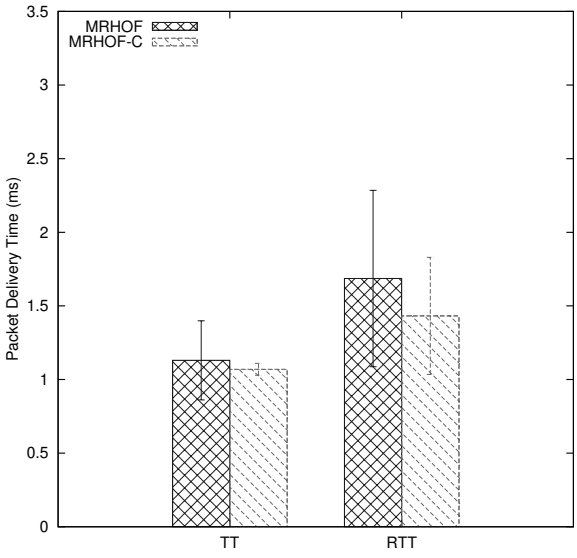


(d) PFT in $5-$ *sink* communication

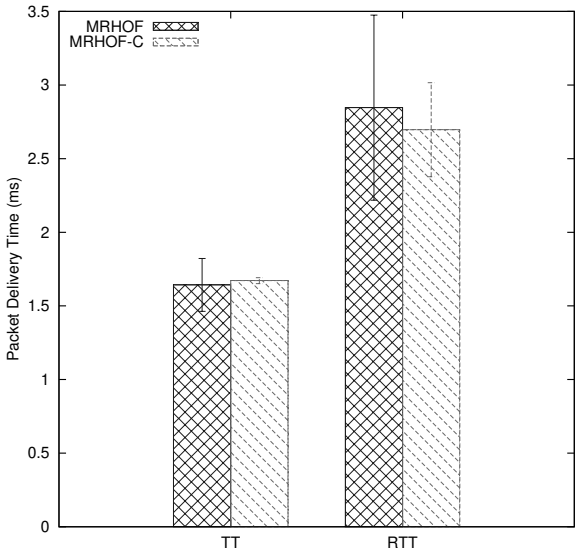
Figure 11: Lambda (Λ) topologies in the positively biased scenario



(a) The initial and final DODAG



(b) PDT in $3 \rightarrow 2$ *P2P communication*



(c) PDT in $8 \rightarrow$ *sink* communication

Figure 12: Topology of the negatively biased scenario

experiment, i.e., 60 *min* (x-axis in Fig. 8a, 8b and 10). This treatment contributes to a fast DODAG setup while preserving energy for the mobile nodes that send control messages with a lower frequency than the fixed ones. In the case of the *Dynamic* setup, the CORAL platform can inject configurations on-the-fly; thus, the protocol starts with the default RPL parameters, and after 30 *min*, the I_{min} parameter is dynamically changed to the value 8 only for the sink and fixed nodes. This value entails a higher frequency of the control messages, which in turn provides a higher probability for the mobile nodes' connectivity (compared to $I_{min} = 12$).

Besides the comparison with the Default RPL, our *Moderate RPL control* is also compared with two representative related works [32, 34]. In [32] the authors flatly configure the I_{min} to a fixed value and, thus, completely disable the trickle timer mechanism that proportionally increases the DIO interval time. Their *PeriodicDIO* approach deviates from the RFC specification [3] in that DIO messages are not sent subject to the expiry of the trickle timer and facilitates the adoption of an aggressive policy upon new parent selection. In our results, the *PeriodicDIO* approach is depicted with the pink line (Fig. 8) assuming that $I_{min} = 12$, since this is the default RPL configuration and one of the settings used by the authors in [32]. The *Reverse Trickle* [34] introduces the reverse rational in the timer mechanism, i.e., reduces to half instead of doubling the DIO interval time, assuming modifications in the DAO control messages, which entails non-compliance with the RPL standard. It begins with $I_{min} = 20$ and decreases it down to $I_{min} = 8$ each time a new DIO is sent due to an RPL incident. The idea is that once a mobile node connects to a new parent, it is likely that it remains connected to this parent for a long time. Then, over time, the node is more likely to move outside the parent's coverage. The gray line in Fig. 8 depicts the performance of the *Reverse Trickle* approach.

Fig. 8a and 8b demonstrate the performance advantages of the *Mixed configuration* in terms of PDR, the former for all network nodes, and the latter for the mobile ones exclusively. The *Moderate RPL* outperforms the default routing protocol regarding the PDR, especially in the case of the mobile nodes. More specifically, it shows an improvement of up to 21 percent for the whole network (Fig. 8a), which rises up to 33.3 percent for the mobile nodes (Fig. 8b). It is also superior to both the *PeriodicDIO* and *Reverse Trickle* as much as 30 percent and 10 percent, respectively, when all network nodes are considered (Fig. 8a), and as much as 18 and 11 percent, respectively, when only mobile nodes are taken into account (Fig. 8b). The *Reverse Trickle* begins with deficient performance due to the I_{min} configuration at a maximum value. Then it converges with the other solutions, offering nevertheless lower PDR compared to the proposed strategy. The *PeriodicDIO* exhibits stable and comparable to other solutions performance. Still, it also provides lower PDR since its control mechanism is not aware of the network environment. This is even more clear in Fig. 8b and 9 where we observe that the flat consideration in control messages' interval time is not beneficial for mobility.

Furthermore, in both figures, the performance of the *Dynamic configuration* of RPL is identical with the *Default* case (something that works as a proof-of-concept for our experimentation). It starts to converge with the performance of the *Mixed* configuration after the 30 *min*, i.e., once the configuration for the fixed nodes has been modified. Since the mobility pattern for the nodes $2-6$ is an emulation of real moving buses, there are quite long periods that the mobile nodes have no connectivity because of radio limitations. This explains the fact that the average

PDR does not exceed 30 percent in Fig. 8b and 9 for the mobile nodes. This outcome also highlights the benefits of offloading the control overhead to the fixed nodes.

Fig. 10 shows that PDR improvements come with an increase in control overhead, especially in the case of *Mixed configuration*. However, such an overhead increase may be traded for the PDR improvement in case of an emergency. These results highlight that the sooner the appropriate parameter setting, the better for the PDR. At the same time, this calls for further future improvements in the centralized platform accommodating the protocol's mechanisms, e.g., implementing rapid intelligent detection of the network conditions. We omit a straight overhead comparison with the *PeriodicDIO* and the *Reverse Trickle*, since these approaches deviate from the RPL specification [3] and, thus, they produce extra, non-RPL-standard control packets.

2.6.3 Deep RPL Control Results

In this section, we evaluate the Algorithm 1 and the *Deep RPL control* mechanism towards supporting *P2P communication*. As we previously explained (Section 2.6.1), we assume three distinct network topologies to show that the impact of the proposed algorithm is independent of the nodes' arrangement. Those topologies include one positively and one negatively biased, along with a random one, to highlight the degree of the advantage derived in each case.

To exhibit cases that can be most benefited from our mechanism, we start with the positively biased scenario (PBS) of a Lambda (Λ) topology depicted in Fig. 11a. Such a topology could be the case of a city's backbone road network, where two equal-length (in *Km*) branches accommodate monitoring equipment (e.g., temperature sensor nodes) that collect environmental data. Typically, in RPL, each node receives data from the node below, and it forwards them aggregated with its measurements to the node one-hop forward to the sink. The sink is located at the top, where the two branches end up (e.g., large traffic lanes leading to the city center exit). In such a topology, nodes at the bottom-end of each branch can communicate with each other only by traversing the one branch up to the sink and back down to the other branch. In Fig. 11a, using the MRHOF, the $3 \rightarrow 2$ *P2P communication* follows the path $3 \rightarrow 8 \rightarrow 6 \rightarrow 1 \rightarrow 7 \rightarrow 9 \rightarrow 2$, which entails slow and unreliable communication. Exploiting MRHOF-C, a small subset of links in the existent DODAG are re-arranged to facilitate the required connection, as shown in Fig. 11b. In practice, the *RPL Configurator* mechanism proceeds with coloring purple the sender 3, orange the receiver 2 and red the intermediate nodes $4, 5$. The rest of the network nodes remain white. This color information is exploited by the Algorithm 1 which results with the desired *P2P communications* path: $3 \rightarrow 5 \rightarrow 4 \rightarrow 2$.

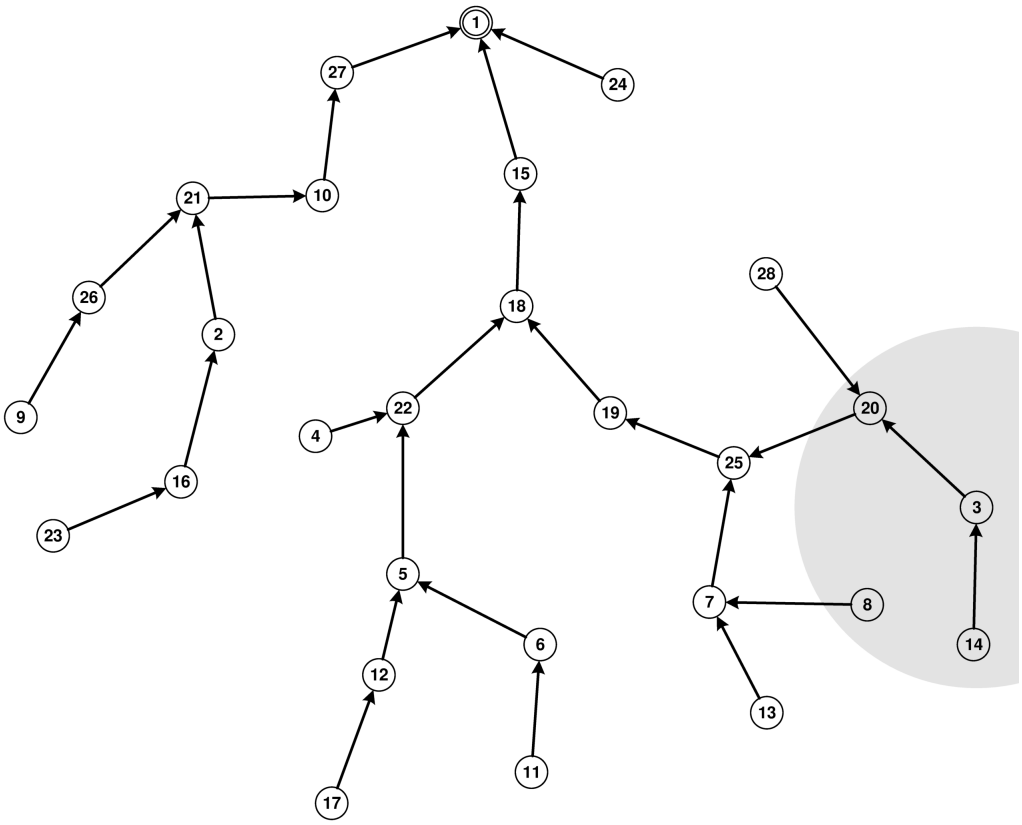
The impact of this change is depicted in Fig. 11c and 11d, where the CORAL starts with the default RPL OF (i.e., the MRHOF) in period $0-20$ *min*, transits to the proposed MRHOF-C in $21-40$ *min*, and concludes the experiment in period $41-60$ *min* with the MRHOF again. We use graphs with the time parameter on the x-axis only in this first experiment to demonstrate the ability of the CORAL to deploy a new OF dynamically using the *OF Deployer* component. During the whole period of 60 *min* we monitor and evaluate the TT (red line) and the RTT (green line) for the communication path of interest, i.e., $3 \rightarrow 2$ and we derive improvements up to 63.9 and 64.7 percent for TT and RTT respectively, which are depicted in Fig. 11c.

For the same period, Fig. 11d shows that the $5 \rightarrow 1$ (sink) communication is slightly affected in terms of TT and RTT. In both graphs, in each OF change, a period of roughly 2 *min* passes without reporting TT and RTT values due to the connectivity disruptions caused by the global DODAG's repair. Such an experiment shows that the CORAL and the *Deep RPL control* enable a *P2P* communication, improve the desired delivery time, and cause minor delays in the rest network, in a topology where the desired communication would be almost impossible in the *non-storing* mode of RPL.

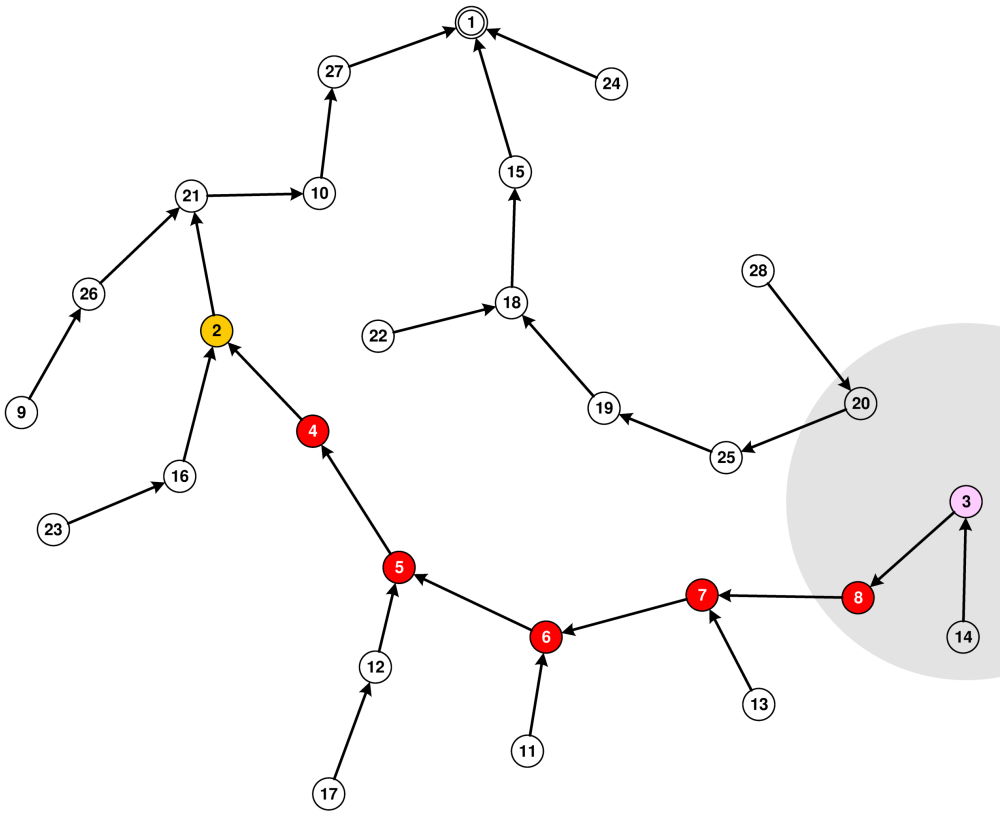
A second negatively biased scenario (NBS) indicates the case where the network topology is not significantly improved from the *Deep RPL control* since the existing nodes' arrangement can almost serve the desired *P2P* path. We use the topology of Fig. 12a, where the end-points of communication, i.e., the nodes 2 and 3, have a common "ancestor" (i.e., node 6), before their data reach the sink node, i.e., the node 1. This entails the minimum change of creating the $4 \rightarrow 2$ link (dotted line) instead of the existent $4 \rightarrow 6$ to speed up the $3 \rightarrow 2$ communication (when the MRHOF-C is used). The results of this experiment are also derived throughout 60 *min*. In this case, Fig.12b and 12c present the average TT and RTT values along with their standard deviation. We observe that the desired $3 \rightarrow 2$ communication is marginally improved (Fig.12b), mostly in terms of RTT, while the node 8, which is the most affected by the changed link in the DODAG, has a slight deterioration in respect to the TT, while it keeps improved RTT values (Fig.12c). An interesting outcome from both graphs is that MRHOF-C has better behavior in terms of RTT, indicating that the return path is specifically determined. Thus, the nodes do not waste time looking for the next hop. All in all, this scenario exhibits that even when a Local-repair does not end in a much different alternative path for the *P2P communication* in interest, it is still worth using the proposed MRHOF-C.

The last random topology scenario (RTS) can show the general applicability of our solution since it can better simulate a real-world case where randomly deployed nodes form a typical IoT network. The DODAG created by the RPL with the MRHOF is depicted in Fig. 13a. Assuming that an emerging network event triggers the need for $3 \rightarrow 2$ communication, the CORAL must find a path to serve them. Either pro-actively to save time or re-actively, it launches the new MRHOF-C, and the network nodes proceed with the DODAG's reconstruction. As soon as all repairs have been completed, a new DODAG is constructed as depicted in Figure 13b, and a direct path between the sender node 3 and the receiver node 2 is established. As expected, the communication between the nodes 2 and 3 is significantly benefited; we derive improvements of 32.7 and 42 percent for TT and RTT, respectively, as shown in Fig. 14.

With the default MRHOF, even in storing mode, the round-trip of a $3 \rightarrow 2 \rightarrow 3$ message is rarely successful—and only when there is no network traffic—because the returning packet goes through the sink node and most probably expires, given the RPL's known inadequacy for *P2P connections* [32, 6]. Fig. 15 provides the PLR for the two compared OFs both in storing and non-storing mode. At first, it is natural that packet loss in the round-trip would be higher than the one-way paths. The MRHOF has a bad performance of 70 percent loss in the one-way transmissions, which rises to 80 percent when a round-trip is considered. Such loss ratio makes it practically inappropriate for *P2P communication*. MRHOR-C limits the losses at 20 percent and 35 percent, respectively. Then, in non-storing



(a) Initial DODAG



(b) Forced path in the DODAG

Figure 13: Random topology of the neutral scenario

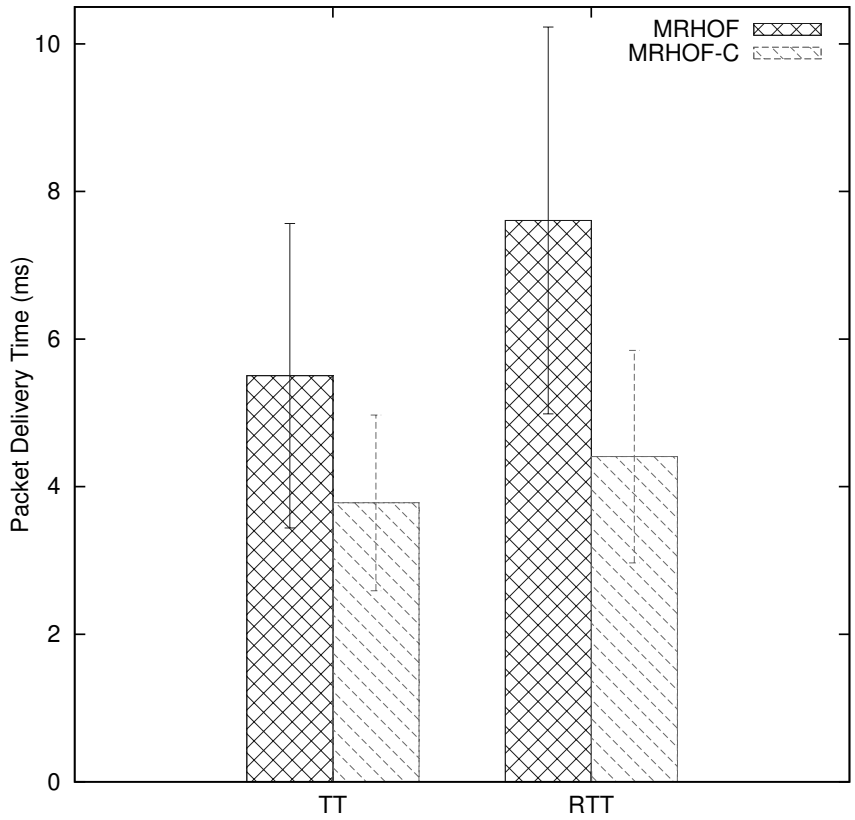


Figure 14: PDT in $3-2$ *P2P communication* for the RTS

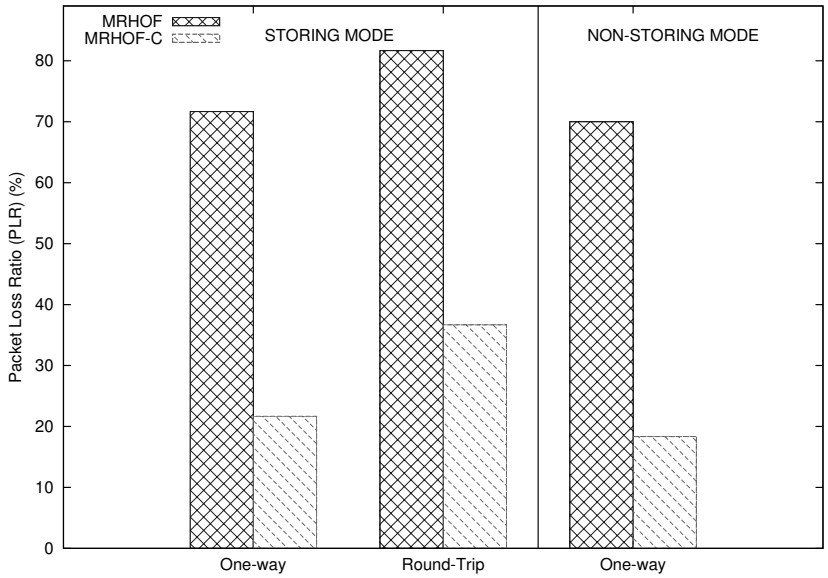


Figure 15: Packet loss in storing and non-storing mode for the RTS

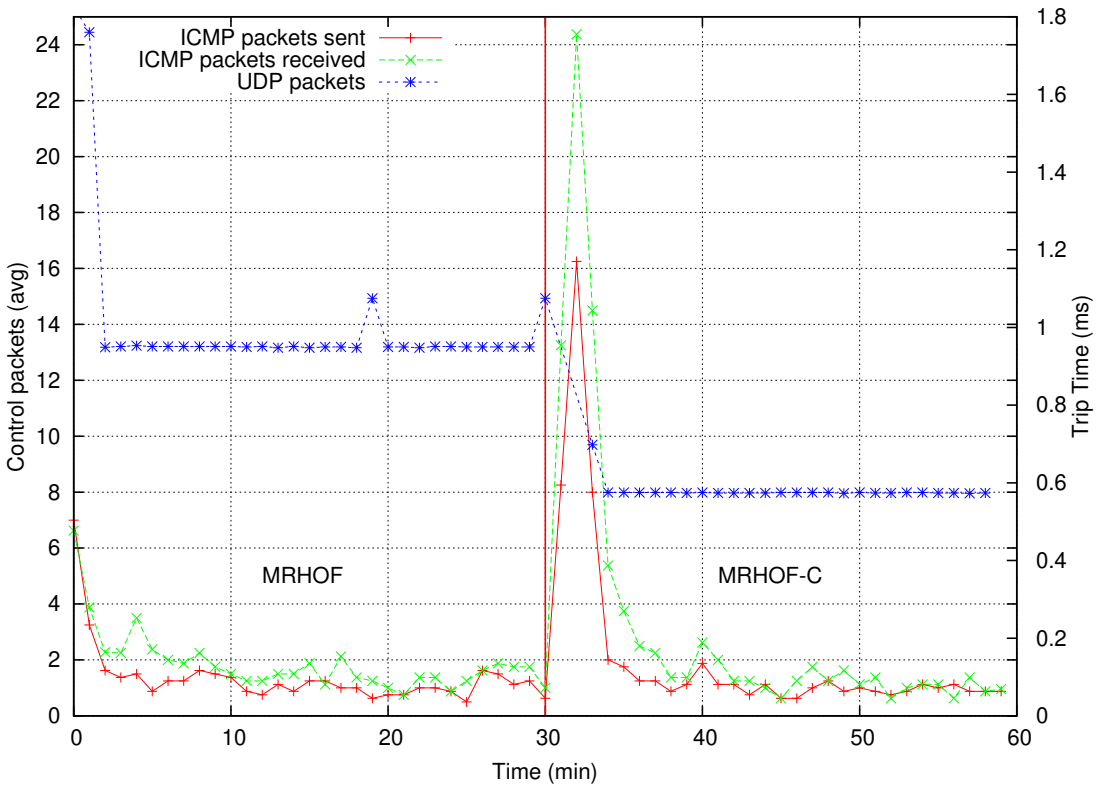


Figure 16: Control overhead and UDP delivery time in non-storing mode for the RTS

mode, only the loss in the one-way path can be measured and, thus, round-trip values are omitted. Figure 15 shows that MRHOF-C is more reliable than MRHOF in non-storing mode since it can reduce packet losses as much as 75 percent.

To demonstrate the effect of the DODAG's repairment once the *OF Deployer* proceeds with the dynamical deployment of the MFHOR-C, we run a last experiment over the RTS in the non-storing mode. The experiment lasts 60 min and starts with the MRHOF. Fig. 16 shows that at the midterm MRHOF-C is employed, and as it is anticipated, the ICMP control packets are temporarily multiplied in the network. The red and green lines correspond to the ICMP send and received packets between the end-nodes $2, 3$, and after a 2 *min* period of sharp increase, they revert to their previous levels. Thus, an exciting outcome is that the control overhead caused by the two OFs is at the same level. However, the valuable information is that Global and Local-repairs demand a $2-3$ *min* period to restore the control traffic. This could be a critical or non-critical period concerning the application requirements. In network environments with emergency events and strict safety requirements, a policy of pro-active MRHOF-C deployment is appropriate to save time towards serving a *P2P communication*. On the contrary, when applications can tolerate connectivity disruptions during the OF transition, a re-active policy offered by the CORAL mechanisms is a good asset. These disruptions are indicated in Fig. 16 by the frequency of UDP on 30 *min* (i.e., the star marks on the curves). Finally, to validate our previous findings, Fig. 16 shows that UDP packets delivery time drops significantly from 1 *ms* to 0.6 *ms*.

An overview of the results derived by the three experimental setups (i.e., PBS,

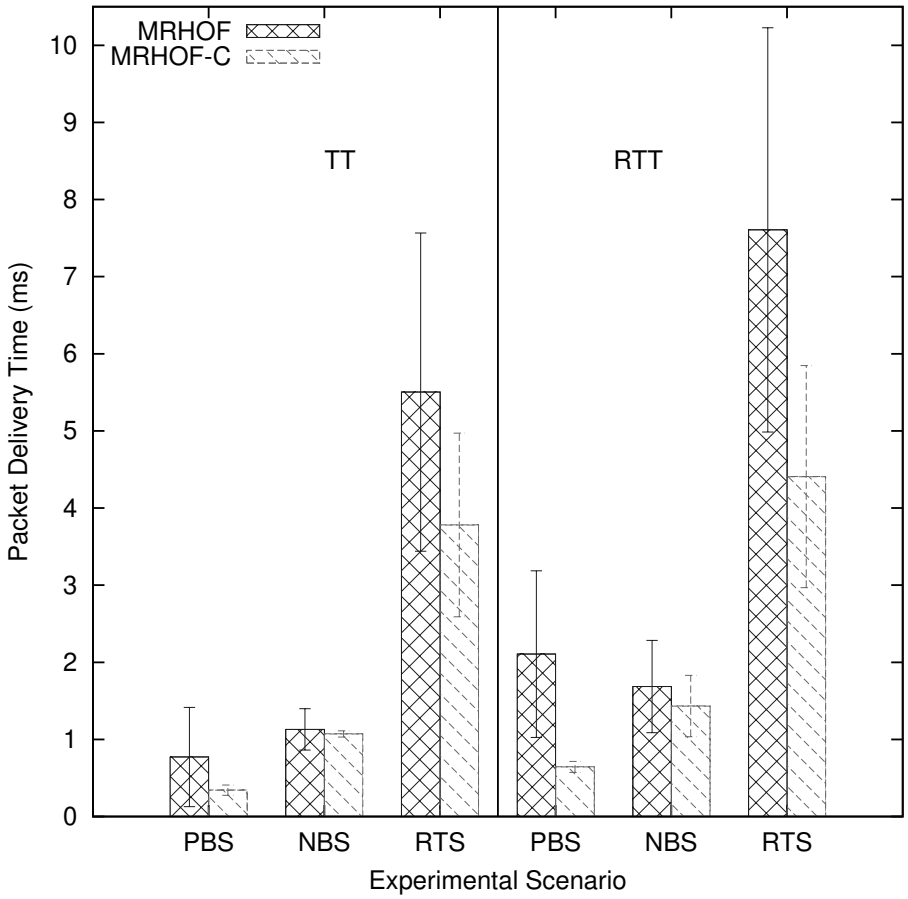


Figure 17: A comparison overview of the three experimental setups

NBS, and RTS), is provided in Fig. 17. In this summary, it is clear that MRHOF-C is superior compared to the MRHOF in all cases and for both TT and RTT. Apart from the reported average values, it is worth to mention the standard deviations indicate that MRHOF-C is (much) more reliable than its competitor OF.

2.7 Related Works

In this section, we contrast our proposal to the related works that mainly focus on solutions tackling the RPL's limitations for *mobility* and *P2P communication*, issues that are being addressed by the proposed *Moderate* and *Deep RPL control* strategies, respectively. Furthermore, we discuss other relevant SDN or SDN-inspired platforms.

Several works attempt to *improve RPL's behavior under mobility*, mainly through aligning the responsiveness of its topology discovery mechanism to the characteristics of the topology changes while considering the resource constraints of the devices. Although mobility can be handled through the MIPv6 standard [40], it is a resource-expensive approach not matching well the characteristics of WSNs and IoTs, especially for medium to large topologies.

Several RPL adaptations to tackle mobility have been proposed in the literature. In [32] the topology adaptation is based on immediately probing and evaluating the ETX value of a new neighbor, along with this node's parent ETX, and then altering the standard DIO message by stamping it with this neighbor's ID. To handle dynamic topologies, the authors in [34] set the I_{min} to a max value and

then reduce it to half after each new DIO. Both approaches are compared with our solution in Figs 8a, 8b and 9. We argue that the former solution is suitable for networks where the mobile nodes are mainly connected to the network, switching smoothly between parents, while the latter fits mobile nodes with a steady pace and predictable behavior.

More works within the same domain include: (i) the adjustment of the DIS transmission times depending on the nodes' status in terms of mobility [33, 41]; (ii) the suggestion that all mobile nodes should be set as leaves, i.e., this way they do not send DIO messages and, thus, they cannot be chosen as parents [42] (this insightful idea was exploited in our experiments in order to exclude mobile nodes from creating "legitimate" paths); and (iii) several scenario-specific solutions, such as the autonomous moving of the sink towards the mobile nodes to reduce the number of hops the information traverses [43].

Other proposals augment RPL with mobility-aware features. For example, the recent papers [44, 45] propose mobility-aware adaptations in the RPL protocol that include new OFs. Other RPL extensions introduce hand-off handling mechanisms, such as [46] which elaborate on a relevant, proactive mechanism called smart-HOP (i.e., lowering the trickle timer for wearable mobile nodes to improve their hand-off time), and [30] which propose the employment of additional proxy nodes to assist mobile communication. Furthermore, the work in [47] details a hand-off handling mechanism based on the average RSSI value, so the mobile nodes can immediately disconnect from the existing attachment points and connect to more suitable ones [46]. The latter functionality has been controlled by a management framework, underlining the advantages of such an approach. In addition, the following solutions employ mobility prediction mechanisms, such as: (i) adjusting the Trickle timer according to a prediction algorithm based on historical values [33]; (ii) introducing a fuzzy mobility estimator residing at an additional mobility-support layer [48]; or (iii) predicting nodes' mobility based on a Bayesian model [49]. Such a last idea is insightful and could be incorporated in our platform to enhance the mechanisms at the *Control Layer*. Our plans also include the utilization of UPIs to collect network parameters, such as the RSSI and LQI, which a decision-making mechanism can exploit once detecting nodes' mobility will correspond according to a predefined set of rules.

The above solutions, i.e., focusing on IoT mobility issues, can be categorized into two groups. The first proposes RPL adaptations (i.e., configuration tweaking or variations of its existing mechanisms, such as the Trickle timer). Still, it targets particular network characteristics or use-cases, while our solution remains generic and, as such, can benefit many diverse networks and topologies. The second implements RPL extensions with new architecture layers or functionalities (e.g., on mobility prediction or hand-off handling), but may lead to resource-expensive operation or protocols that are not compatible or consistent to the RPL standard, while our solution remains compliant with RPL, and as such can be easily and rapidly deployed without causing protocol malfunctions and inconsistencies.

Another set of RPL variations focus on *improving the P2P communication issues of RPL*, especially for downward routes [12, 27]. Even though RPL currently allows direct node communication through its storing or non-storing modes of operation, the storing mode is characterized by scalability and memory limitation issues. In contrast, the non-storing suffers from significant control overhead near the sink,

causing congestion. The DualMOP-RPL protocol [50] supports coexisting modes of operation (i.e., both storing and non-storing) in a single RPL network to improve the downward routing inefficiency. The work in [51] provides a performance analysis quantifying the routing cost difference between DAG-based *P2P* and the shortest potential direct routes, i.e., suggesting that RPL should be improved in terms of direct node communication. In this context, RFCs 6997 [29] and 6998 [52] propose a reactive approach establishing shorter *P2P* paths through defining temporary DODAGs that consider the destination node of the direct path as a sink. This process is regarded as a third mode of operation, called *P2P* route discovery. However, this approach is characterized by additional overhead for the maintenance of multiple, although temporary, DODAGs [12].

The leading relevant solutions are either protocols incompatible with RPL or hybrid versions of RPL, adopting alternative routing strategies for the *P2P communication* only. The AODV protocol [53] and its lightweight version LOADng [54] discover reactively bi-directional paths through communicating control packets, i.e., called Route Requests (RREQs). The 6TiSCH standardization initiative [55] proposes adopting hybrid protocols to improve node-to-node communication, such as the Asymmetric AODV-P2P-RPL in Low-Power and Lossy Networks (AODV-RPL) [56]. Another hybrid example is [57] which introduces a protocol combining the advantages of RPL with those of back-pressure routing protocol.

In our case, we are compliant with the RPL standard and allow dynamic changes in the existing protocol features to extend the applicability of RPL to novel use-cases. Additionally, we enable new functionalities to be added in a centralized controller instead of the resource-constraint nodes. For example, a mobility prediction mechanism is less accurate from a node's viewpoint than an algorithm residing at a centralized controller and taking decisions based on the global network view. Furthermore, our strategy avoids introducing additional overhead into the devices.

Relevant to our proposal control facilities and protocols include: (i) SDN-Wise [58], a logically-centralized IoT protocol and SDN controller; (ii) our proposal [59] also utilizing the WiSHFUL infrastructure in an OpenFlow-like SDN control environment; (iii) an on-top of SDN-Wise approach for topology discovery [60]; and (iv) a platform implementing basic SDN features, i.e., topology and device management over application, control, and infrastructure layers [61]. These control platforms and protocols bring OpenFlow-like solutions to IoT environments, but they do not preserve the advantages of RPL. In [62], the authors suggest the association of a mote with a particular DODAG to be guided from a centralized controller. At the same time, in [63] they discuss the synergy between TinySDN (an SDN protocol for IoT) with RPL and how they can assist each other. A recent Internet draft [55] suggests SDN-type centralized routing for time-sensitive flows and RPL for the rest of flows. Another relevant solution is [64], which introduces an SDN-like controller transmitting routing and control messages compatible to RPL to manipulate its operation. This approach works with legacy equipment. In another recent work [65], interoperability between protocol stacks is introduced to provide controller discovery and reduce control overhead.

RPL can cover a wide range of IoT deployments but with manual configurations and without obvious performance outcomes as a bottom line. Here, we argue that a centralized control facility can implement closed control loops, monitoring, decid-

ing, and configuring RPL parameters on-the-fly, depending on the mobility status of each node and the application requirements (e.g., requesting direct communication between nodes).

2.8 Conclusions and Future Work

This work presents two SDN-inspired routing control strategies for the IoT, namely the *Moderate RPL* and the *Deep RPL control*, which evolutionarily tackle the *mobility* and *P2P communication* issues of the RPL in the sense that they remain consistent to the protocol's standard. Our CORAL facility and its components, namely the *OF Deployer* and *RPL Configurator*, act as enablers to dynamically change the OF–along with the RPL operate to construct the DODAG–and appropriately configure either its parameters (e.g., I_{min}) or features (link coloring), both in real-time. The results confirm that the *Moderate RPL control* strategy can bring improvements in PDR of the order of 33 percent at the cost of increased control overhead. However, offloading this overhead to the fixed infrastructure can eliminate its impact and alleviate the network in emergency cases. On the other hand, the *Deep RPL control* and the newly introduced MRHOF-C Objective Function bring multiple advantages: enable a *P2P* communication both in storing and non-storing mode, improve the packet delivery time between the nodes of interest (up to 42 percent), significantly reduce the packet loss ratio (as much as 75 percent) and keep the rest network almost untouched from the local or global-repairs executed. The apparent cost of control overhead seems to temporarily disturb the network, and be eliminated if MRHOF-C is employed pro-actively.

The results of the proposed OF, offer insight for approaching the parents' selection process and the DODAG's construction as a multi-objective optimization problem. Finding an optimal solution for one OF may require accepting a poor solution for some other(s). Thus, different weights to each OF enable us to consider them in conjunction. Such a vision can ideally be supported by the CORAL architecture, whose planes can serve as place-holders for intelligent mechanisms detecting network conditions (e.g., nodes' failures due to connectivity or battery drain issues, or presence of malicious nodes) and tuning as a response a weighted OF or deploy the most appropriate automatically.

2.9 Chapter Summary

This chapter has presented mechanisms and methods that address the complex issues of peer-to-peer communication and mobility issues in IoT networks running the RPL protocol.

An evolutionary algorithm was also presented, tackling those issues, and several proof-of-concept experiments have shown that the proposed solutions have a significant positive impact on such networks.

An extensive, up-to-date, and bibliographic survey in Intrusion Detection Systems for RPL is presented in the next chapter.

3 Survey on RPL's Security Issues

Although the RPL is undoubtedly today one of the most used IoT protocols, it has specific open security issues. For example, the majority of RPL implementations assume the unsecured mode of operation. The RPL security features are characterized as optional [3] and, according to [12, 13], future versions of RPL will address issues such as authenticated security. However, some recent research efforts focus on a partial implementation of RPL's security features [10, 11]. IoT networks suffer from the inability or the difficulty to tackle certain attacks inherited from the distributed, atomic mode of operation they exercise by default to make things more complicated.

To systematically explore and record the above issues, this chapter is an up-to-date, systematic literature review on the RPL security issues and how Intrusion Detection Systems try to address those.

3.1 Introduction

The Internet of Things (IoT) is a broad field of technology and research, comprised of Low-power and Lossy Networks (LLNs). The nodes of such networks are susceptible to various restrictions and challenges, rendering the existing routing protocols inappropriate. The IPv6 Routing Protocol filled the gap for Low-Power and Lossy Networks (RPL), which has become the de facto standard for IoT routing, beyond initial expectations [3, 4]. RPL has been proven significantly mature to connect IPv6 devices, with reasonable control overhead and under challenging conditions, e.g., lossy links, heterogeneous and constraint devices, newfangled threats [5, 6].

Despite its advantages, RPL still has open issues, the most important of which are related to attacks that disrupt the IoT networks' operation [8]. RPL is unavoidably exposed to many attacks since it is based on the IPv6 open stack and uses mostly wireless media for the nodes' communication. In addition, by exploiting RPL's mechanisms, an intruder can gain access to the network and unleash attacks that originate from within the LLN. In such cases, encryption itself does not suffice to provide security [9]. On this front, the RPL standard specifies three modes of operation, i.e., unsecured mode, preinstalled mode, and authenticated mode [3], while it also defines mechanisms for data confidentially, data authenticity, and replay protection [10].

The most realistic approach to dealing with attacks is the Mitigation Methods and the Intrusion Detection Systems (IDSs). The former regard lightweight supplementary mechanisms to the standard RPL and deal with a limited number of attacks. The latter employ a combination of methods, allowing for a broader spectrum of attacks' treatment. A small number of surveys currently focus on the RPL aforementioned security issues and the IDSs confronting them. Mayzaud et al. [8] present a definite categorization of RPL attacks, where the IDSs are solely discussed in line with them, while a detailed taxonomy and evaluation of the attacks are missing. Furthermore, [8] includes only three of the new IDSs, available at the time of publication. Raoof et al. [14] discuss RPL attacks and their mitigation methods in general, leaving limited space for description and analysis of specific IDSs; only a list of those considered most influential by the authors are shortly

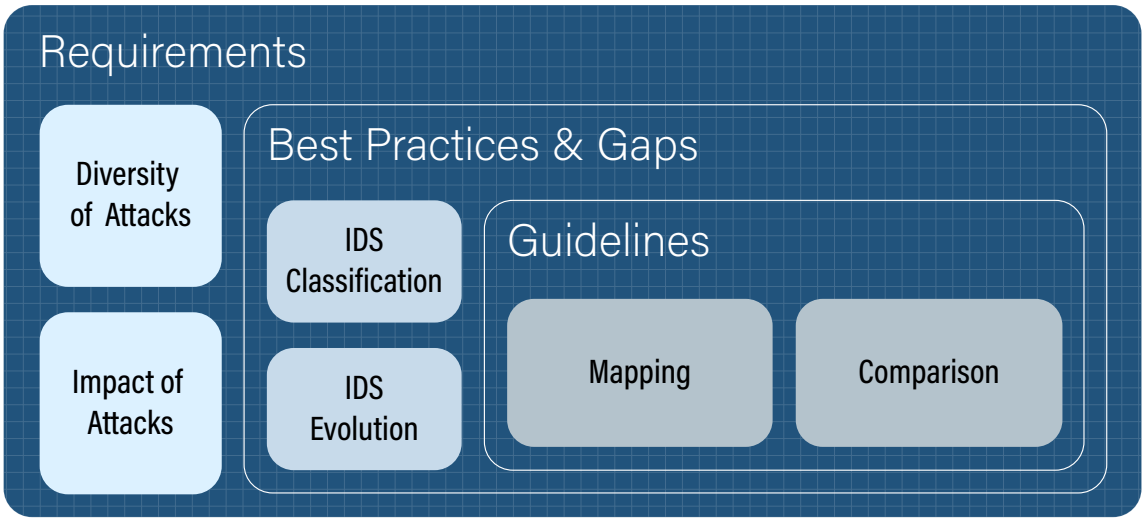


Figure 18: Conceptual framework of the analysis: an abstract representation.

described. In the recent work of Verma et al. [9], the authors also utilize the taxonomy of attacks from Mayzaud et al. [8], and they propose a comparison chart of the contemporary IDSs based on an extensive set of 26 categorization criteria. Despite being a detailed mapping with some potential of providing future insights, at this time, their comparison table is empty up to 92 percent, and, thus, it remains incomprehensible.

The above fact indicates that selecting criteria for analysis is a challenging issue since they should be primarily meant for the context they are proposed. Secondly, they should directly compare the subjects (the IDSs in our case) under investigation. To our mind, this can be achieved by a core of narrow and well-thought criteria.

In this context, this survey implements a coherent investigation of RPL-related IDSs according to a novel conceptual framework that defines a three-step methodology. It starts by investigating the diversity and impact of well-known attacks to determine essential design requirements for IDSs, based on both a literature review and illustrative simulations. The next step identifies best practices & gaps by studying the evolution of related IDS proposals. The last step involves mapping 22 selected IDSs to the attacks they encounter while contrasting them regarding the introduced requirements as comparison criteria. Our analysis concludes with essential design guidelines for future up-to-date IDSs.

3.2 Conceptual Framework & Methodology

This survey adheres to a novel conceptual framework, shown in Fig. 18, that provides the methodological basis of our investigation. It consists of three methodological steps, defined below.

The first one concerns the *requirements' definition* that a successful IDS should address. Our starting point is a better understanding of the problem IDSs tackle, i.e., the mitigation of attacks. For example, Wallgren *et al.* [66] identify the diversity of attacks as the main cause for attack detection accuracy issues in existing IDSs. Other papers, including surveys [8, 14] and IDS proposals [66, 67, 68, 69], do typically base their analysis on identifying the considered attacks' impact, e.g., increased control overhead or decreased packet delivery ratio (PDR). We conduct

a literature-based investigation of well-known RPL attacks from a new perspective for completeness: a combined study on attacks' diversity and impact.

More precisely, we elaborate on the RPL-related attacks, spanning from *resource depletion* attacks, that shorten the network's lifespan to *network topology* attacks, that degrade the paths created by RPL or isolate a subset of network's nodes, and *network traffic* attacks, that allow the analysis of packets to gain knowledge about the network. Several of them may not be harmful as standalone events. Still, they can be critically detrimental to the network (e.g., control overhead) or the applications (e.g., PDR) in conjunction with others. In this first step, we also provide illustrative simulation results, highlighting the primary outcomes of our combined investigation of attacks' diversity and impact. As an outcome, we define a set of seven design requirements for an RPL-related IDS that are directly connected with the protocol's standard.

Our next step identifies the *best practices & gaps* out of an extensive literature review in respect to the defined design requirements. Our goal is to realize the best approaches of existing works addressing the requirements, understand their evolution, as well as identify associated open issues. We investigate the 22 most recently introduced RPL-related IDSs in the literature ($2013 - 2020$). We firstly discuss their classification in respect to their detection method and their placement strategy. Then, we build up a timeline of their evolution stages along with their principle qualitative (i.e., detection method, placement strategy) and quantitative features (i.e., number of attacks). The adherence level to the requirements and classification criteria is discussed in the textual descriptions of each IDS.

Our last step involves a synthetic process producing our investigation's outcome, which is to *introduce design guidelines* for up-to-date IDSs. We consolidate the outputs of the steps mentioned above first, including mapping the IDSs to the attacks they tackle. Secondly, we provide a summarized comparison viewed under the design requirements we introduce. We consider both attack detection supported by simulations and those discussed conceptually only for the attacks' mapping. We are based on the respective authors' claims in the IDS' relevant articles for compliance with the requirements. Since the devised requirements are aligned to the RPL standard objectives, the vast majority of IDSs consider them, and hence, we ended up with a comprehensive comparison that produces and elaborates on four crucial design guidelines for future up-to-date IDSs.

The following section gives a brief overview of RPL, as an essential background for the analysis that follows afterward.

3.3 RPL Overview

RPL operates on the IP networking layer via the 6LoWPAN protocol stack, exploiting Destination Oriented Directed Acyclic Graphs (DODAGs) rooted at a single destination called sink [3]. In practice, the protocol builds a graph of logical paths upon physical network connections, which are directed towards the sink. Parents' selection on paths towards the DODAG root can be treated as a multi-objective optimization problem since a variety of metrics (e.g., link reliability, latency throughput) and constraints (e.g., nodes' energy, link color) can be exploited to evaluate the nodes' rank [4]. The specified Objective Function (OF) defines how the RPL nodes translate metrics and/or constraints into ranks, and select and optimize routing paths in a DODAG.

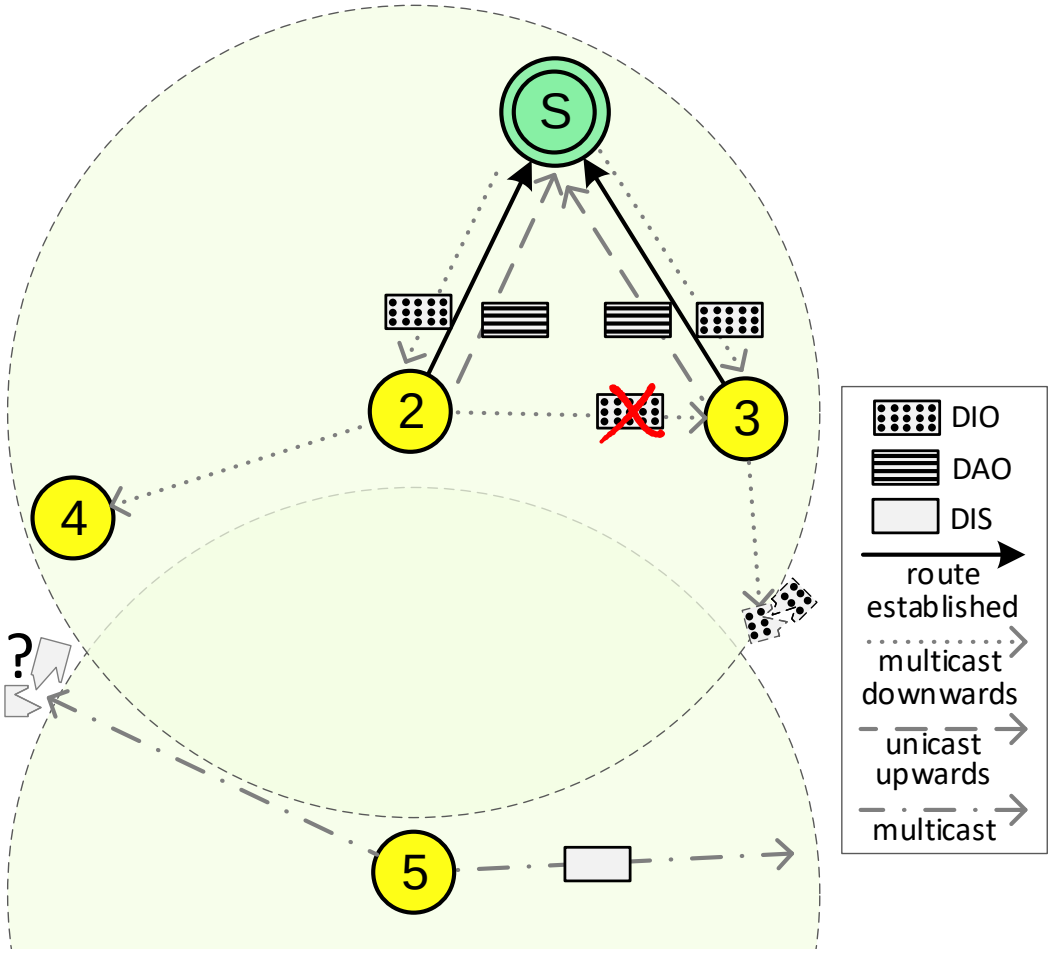


Figure 19: DODAG construction.

As depicted in Fig. 19, the sink-node launches the DODAG's (re)construction based on the exchange of routing control messages, i.e., DODAG Information Object (DIO), Destination Advertisement Object (DAO), DAO-ACK, and DODAG Information Solicitation (DIS). Once the first DIO message is multicasted by the sink, plenty of them are multicasted by nodes getting attached to the graph. DAO messages are used by all nodes, except to the sink, to propagate reverse route information; DIS messages are sent by the not connected (due to their isolated position) or disconnected (due to mobility) nodes in order to solicit DIO messages from other possible connected neighbors and join the graph. DIO messages are critical regarding the graph's construction since they contain the routing metrics and/or constraints and the OF used for the routing paths' establishment.

The DODAG's maintenance is a functionality placed at the very core of the RPL. Hence, a dedicated algorithm, namely the *Trickle timer*, synchronizes the propagation of DIO messages upon which the network's convergence time is based. The critical aspect in the DIO multicasting process is attaining a short network setup time and, thus, the reinforcement of the network's metrics, e.g., PDR, while restricting the control overhead towards lowering the node's power consumption. To achieve the aforementioned trade-off, the DIO messages are sent periodically; their interval ranges from I_{min} (Minimum Interval) up to I_{max} (Maximum Interval), where $I_{max} = I_{min} * 2^{I_{doubling}}$. For example, the default RPL configuration specifies

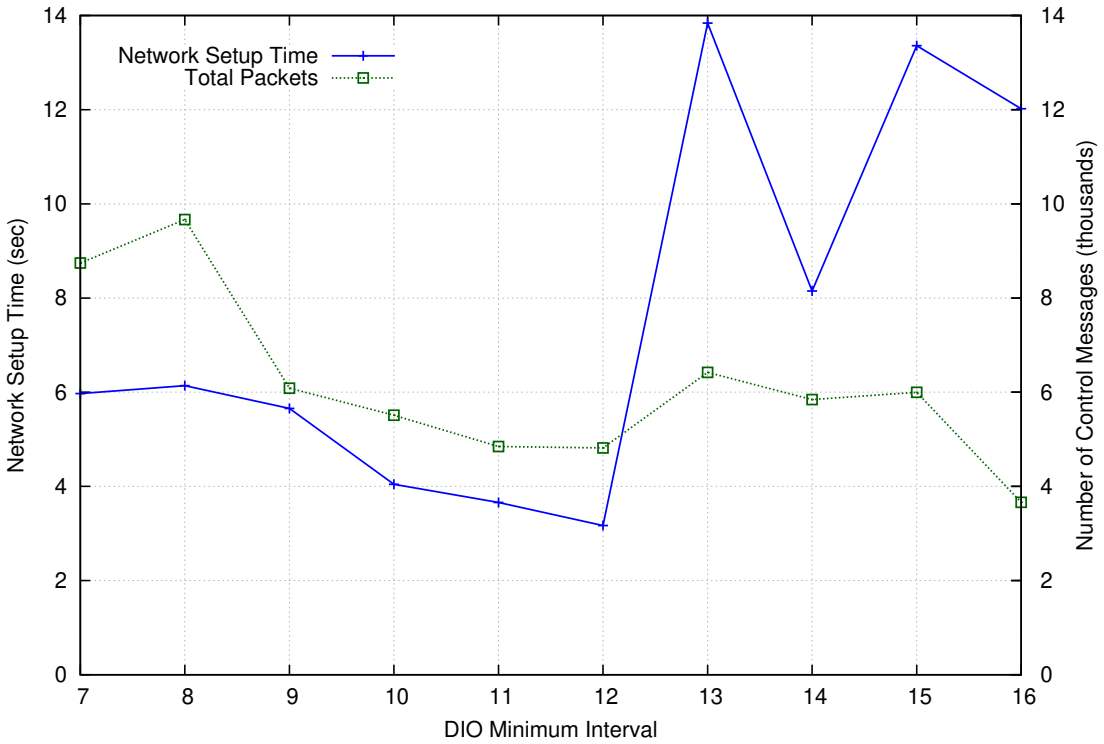


Figure 20: The network setup time and control overhead in respect to the DIO I_{min}.

$I_{min} = 2^{12} = 4.096\ ms$ and $I_{doubling} = 8$ which entails $I_{max} = 2^{12+8} = 17.5\ min$. The timer's duration is doubled each time it fires. Moreover, any change in the DODAG, e.g., an unreachable parent or a new parent selection, resets the *Trickle timer* to I_{min} [5]. According to the algorithm, DIO messages will be sent at a higher rate when the network is unstable and slower otherwise, i.e., to reduce protocol overhead and save energy.

The impact of DIO sending frequency in RPL is depicted in Fig. 20. We derive the graph by simulating a WSN in Cooja, which is embedded with Contiki OS [37]. Our explanatory simulation considers a network of one sink and 10 nodes that perform measurements' collection and forwarding them over multi-hop communication. Fig. 20 shows the impact of DIO I_{min} values on the network setup time (left axis - blue squared-dot curve) and the network control overhead measured in line with the total number of DIO, DAO, and DIS messages (right axis - green x-marked curve). According to the results, high values of I_{min}, i.e., infrequent DIO transmissions, cause delays in network setup time due to the nodes that have not yet received DIO messages and thus remain unconnected. On the opposite, frequent DIO messages entail lower setup time. I_{min} equal to 12, which is the default value in Contiki RPL implementation, provides the best performance concerning the setup time. Regarding control overhead, Fig. 20 validates that higher interval values produce less network traffic since the frequency of DIO messages is low. Fig. 20 is in compliance with our findings in [5].

Since the *Trickle timer* is the most responsible algorithm for the protocol's performance and along with the DODAG and the sink-node are fundamental parts of the RPL protocol, it is undoubtedly a profound target for a series of attacks.

In the following section, we give a taxonomy and describe such attacks, including those exploiting RPL mechanisms and/or weaknesses. We pay special attention to their impact, since in fact, several attacks may not cause severe damage by themselves. Still, they can have bothersome effects on the network (e.g., control overhead) or applications (e.g., PDR) when combined with others.

3.4 Attacks on RPL-based IoTs

Routing in the RPL-based networks is an incredibly challenging task basically due to the connected devices' power, storage, memory, and processing constraints. The RPL protocol offers several configuration parameters to satisfy diverse requirements regarding deployments of different scales, heterogeneity, and mobility [3, 7] as well as mechanisms to adapt to changes. However, such network contexts, including *resource-constraint nodes*, supporting *dynamic topologies*, and based on the *passive nature of the wireless medium*, do inevitably attract malicious actions, including but not limited to denial of service attacks (DoS), physical damages, and/or extraction of sensitive information, e.g., DODAG version, nodes' rank values, and IDs. In fact, some nodes can get compromised by exploiting the RPL mechanisms themselves; if the node happens to have a significant role in the network, e.g., the sink or parent nodes, then a combination of attacks can be applied with severe effects, spanning from resource-depletion of nodes, due to a sharp increase in the control overhead, to severe degradation of the protocol's performance in terms of data delivery.

Right afterward, a comprehensive list of the most common and disrupting attacks on the RPL protocol is presented. The network attacks that do not mainly target RPL are not included since they are not part of the work's scope, e.g., (Distributed) Denial of Service, (D)DoS attacks.

3.4.1 Diversity of Attacks

Reflected to the aforementioned characteristics of the RPL-based IoTs, i.e., *resource-constraint nodes*, *dynamic typologies* and *passive nature of the wireless medium*, the RPL-related attacks are rather divergent and classified into: *Resource depletion* attacks, *Network topology* attacks and *Network traffic* attacks [8]. Fig. 21 provides a panorama of them along with their classes and sub-classes.

More specifically, the *Resource depletion* attacks include malicious actions that intend to deplete nodes' computing, memory, or energy resources by creating a false impression of continuous operation. Given that the node's operation is inextricably linked to processing, memory, and energy assets' utilization, any overhead is equitable to excessive resource consumption. Consequences may be local or, even worse, affect the overall network availability and performance, leading to routing loops, unnecessary network traffic, and congestion [70, 71, 72].

Attacks against resources are distinguished into *Direct* and *Indirect*, according to the fashion of their execution. In direct attacks a malicious node overloads a subset of nodes-victims and affects their status or operation. Common examples are *Routing Table Overload* [70], and *Flooding Attacks* [14, 70]. On the other hand, indirect attacks manipulate intermediate nodes as a means of broadly affecting the network by, for example, causing unnecessary control traffic. *Local Repair* [71, 73], *DIS Message* [70, 67], *DODAG Inconsistency* [72], and *DODAG Version Number* [74, 75] attacks are typical examples of this sub-category.

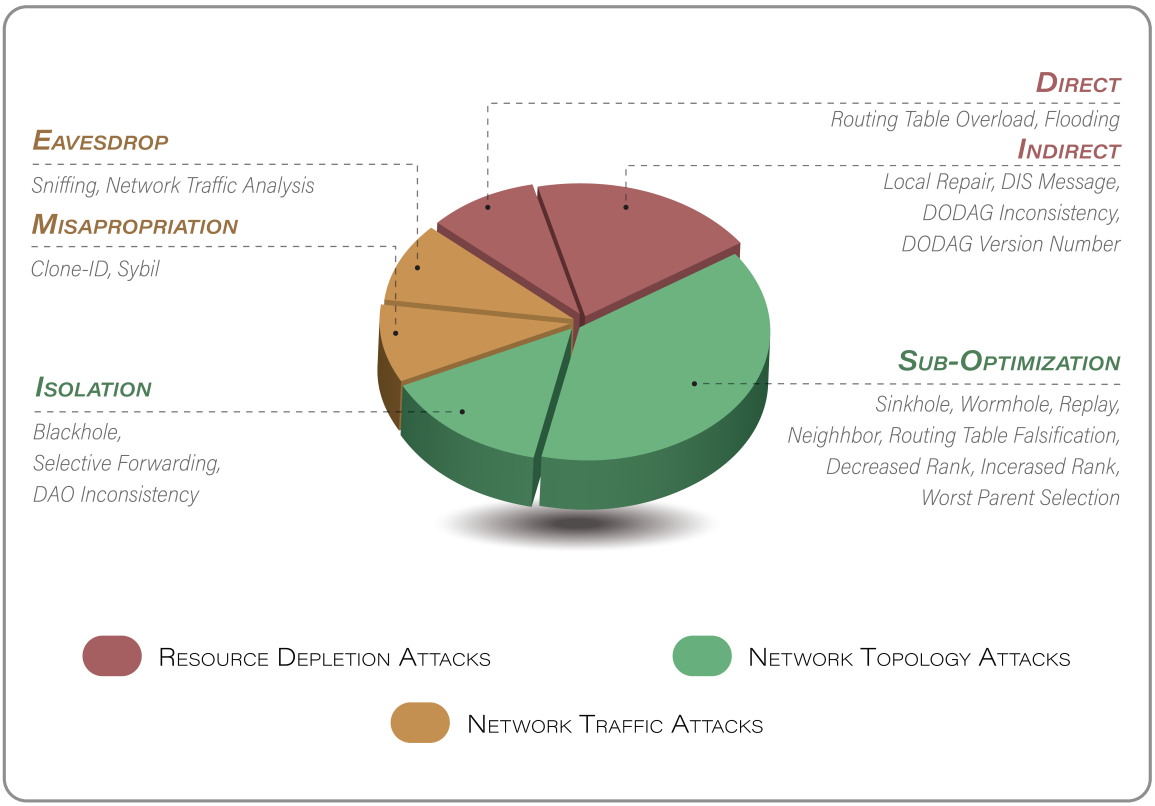


Figure 21: Classification of RPL attacks.

The *Network topology* attacks are divided into *Sub-Optimization* and *Isolation* attacks that disrupt the nodes' communication and DODAG's structure, respectively. In practice, the sub-optimization attacks impact the network's optimal convergence ability, i.e., they prevent the establishment of the optimal routes, and thus, affect the network traffic and degrade the network services. Some of the most common consequences include topology inconsistencies, significant packet losses, increased end-to-end delays, network congestion and nodes resources' depletion. The aforementioned effects can be particularly detrimental to dynamic networks due to the nodes' mobility. *Sinkhole* [66], *Wormhole* [76, 77], *Replay* [78, 79], *Neighbor* [70], *Routing Table Falsification* [69], *Decreased Rank* [73], *Increased Rank* [69, 80], and *Worst Parent Selection* [80] attacks are well-known sub-optimization attacks.

Isolation Attacks exploit the tree topology of the RPL network; they aim at cutting off part(s) of the network by interrupting the nodes' communication with either their parent- or sink-node. Amongst their effects are loss of network traffic, end-to-end delay increase, significant service quality deterioration (e.g., PDR), and isolation of sub-graph parts along with starvation of their participating nodes. The most common isolation attacks are *Blackhole* [71, 81, 82], *Selective Forwarding* or *Greyhole* [71, 66, 81, 82], and *DAO Inconsistency attacks* [8, 14]. These attacks can be severe when combined with others, e.g., decreased rank and blackhole attacks.

The *Network traffic* attacks intercept and monitor the network traffic to acquire or deduce information, e.g., DODAG version or rank value, which can be exploited by attacks launched later on. Depending on how the traffic is affected, they are

classified into *Eavesdropping* and *Misappropriation* attacks. In the first case, the intruder monitors the network's transmissions and analyzes the packets through a breached node or by directly "listening" to the wirelessly transmitted packets. This way, he/she gains access to the topology and routing-related information or even to the actual content of the transmitted packets. The most known eavesdropping attacks include *Sniffing* [8] and *Network Traffic Analysis* [8].

In the latter case, the attacker impersonates other network nodes to extract information about the network topology or gain knowledge of other parameters. The node with the greatest interest in such attacks is the sink due to its crucial role. Appropriating a network node's identity negatively affects the routing service. It also confuses the rest nodes leading to potential incorrect messages' forwarding since, for example, instead of reaching their legitimate destination are delivered to the attacker. *Clone-ID* [8, 14, 66] falls in this category and can be the first stage of further hostile actions causing serious troubles in the network; *Sybil attacks* [66, 83, 84] are an escalated type of Clone-ID attacks which eventually can cause increased network control traffic, high energy consumption and degradation in PDR.

Diversity and/or combination of attacks may affect different aspects of an RPL-based IoT network. The next section provides some indicative examples through simulation.

3.4.2 Impact of Attacks

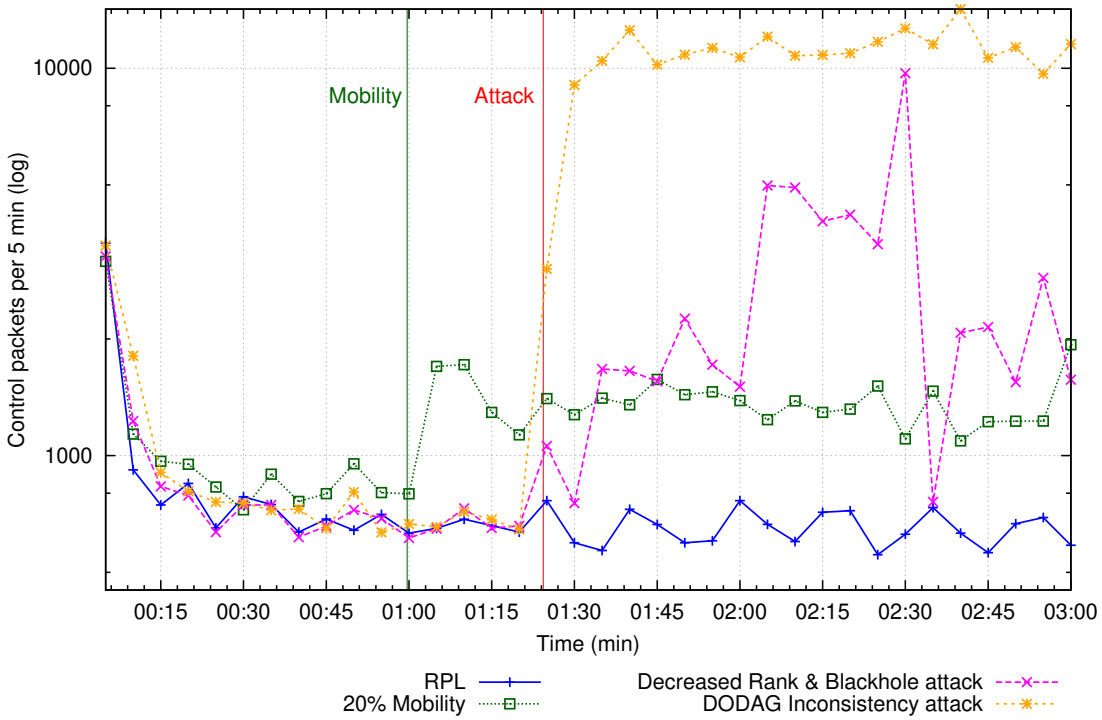


Figure 22: Control overhead under attack and mobility over time.

To indicatively illustrate the impact of attacks on an RPL network, we simulate (in Contiki Cooja [37]) a multi-hop network with one sink and 50 nodes randomly placed around it; the outcome is shown in Fig. 22 and Fig. 23. In practice, we run the simulation for three hours (x-axis) and consider that 20 percent of nodes

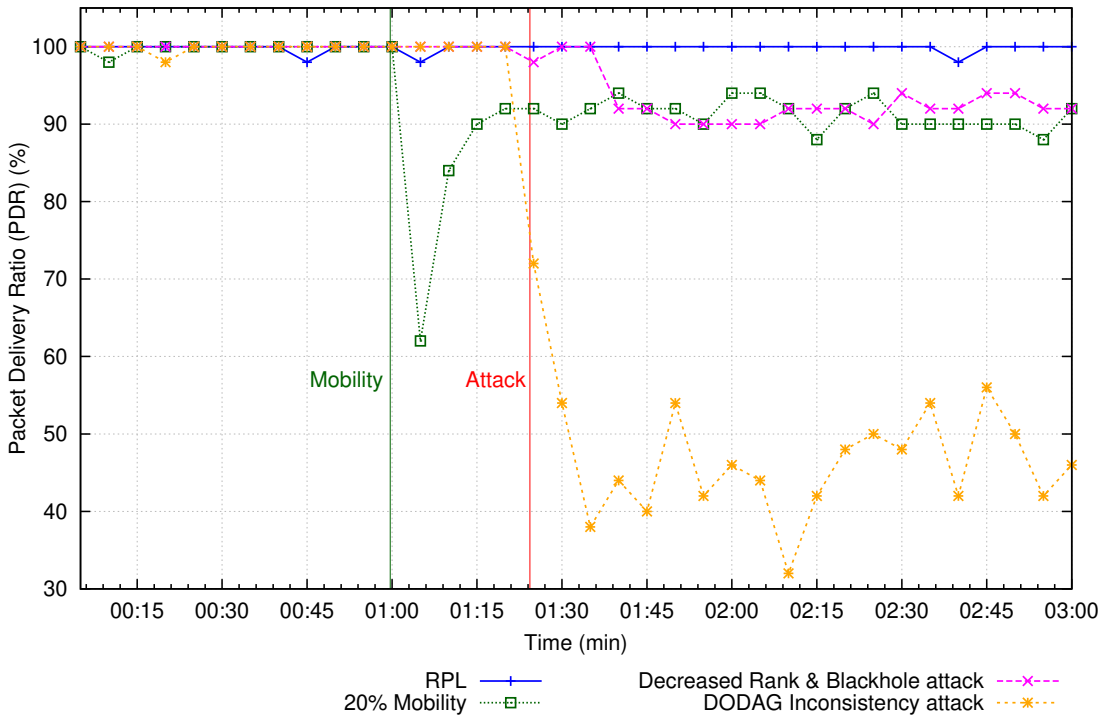


Figure 23: PDR under attack and mobility over time.

become mobile at 01:00 hour (vertical green line). Regarding attacks, we select one from the resource depletion class, i.e., DODAG inconsistency (yellow curve), and a combination of attacks from the network topology class, i.e., decreased rank and blackhole attack (purple curve). Attacks start at 01:20 hour (vertical red line), for visualization clarity reasons.

Fig. 22 shows the impact of attacks on the network concerning the *control overhead* which is calculated in line with the total number of ICMP packets. The RPL standard operation (blue curve) expresses the ground-truth performance which is contrasted with the performance under attacks' scenario. In our simulation, we notice a heavy impact on control overhead in case of DODAG inconsistency attack, i.e., 750 percent (on average), since a big part of the network is isolated and many nodes are forced to constantly update and recalculate ranks and paths to find routes to the sink. Significant deterioration, i.e., 153 percent (on average), is also caused by the decreased rank and blackhole attacks launched in combination. This deterioration happens because the attacker advertises a lower rank value compared to all other legitimate nodes in a network's neighborhood, causing the affected nodes to send an excessive number of ICMP packets in their try to find paths to the sink.

Our previous experience with nodes' mobility [5, 6] urges us to investigate further the attacks' impact in comparison to the effects of mobility. The graph confirms our intuition, i.e., mobile nodes trying to get attached to the graph after being disconnected, can create control overhead easily misinterpreted as the effect of an attack, depending on the observation's time-window, e.g., the green and purple curves on the period 01:30 - 02:00.

Apart from the network, attacks also affect the application, e.g., by aggravating data packets' delivery rate. Fig. 23 shows the impact on the *PDR* which is defined

Table 4: Design requirements.

i.	RPL Specification Compliance
ii.	Low Overhead
iii.	Scalability
iv.	Robustness
v.	Extendability
vi.	Low False Positives/Negatives
vii.	Mobility Support

as the received UDP packets (rUDP) over the total number of packets being send (sUDP), i.e., $PDR = rUDP/sUDP$ [5]. While RPL rarely fails to deliver a UDP packet, e.g., 100 percent PDR in the graph, its performance drops to 49 percent on average and 38 percent on the worst-case under DODAG inconsistency attack, since there are no paths to deliver the packets of nodes that are being detached from the DODAG due to the attack. A mild impact, but again very similar to the mobility case, is caused by the rank and blackhole attacks. The intruder attracts many neighboring nodes as a parent only to drop their data packets once received.

All the above make clear that RPL-based networks must integrate adequate security mechanisms, which will detect and mitigate the attacks of along-coming intruders. According to the literature [8, 14, 12], IDSs are a suitable approach to encounter malicious activities since they aim at detecting several attacks at once and ideally can be extended to deal with attacks that are not initially included in their design goals. However, the design of an RPL-related IDS has further requirements derived from the protocol itself and the impact of its related attacks. The following section elaborates on such design requirements.

3.4.3 Design Requirements of an RPL-related IDS

The design of an IDS that aims to shield an RPL network is a challenging task since it should consider the issues of LLNs, the objectives described in RFC 7416 [85], and the heterogeneity of IoT devices, combining them with its principal mission. In that regard, Table 4 presents a set of seven design requirements of an RPL-related IDS whose selection is justified right afterward.

i. *RPL specification compliance*: In fact, an RPL-related IDS should be primarily compliant with the RPL standard [3], i.e., the fundamental way in which the protocol operates. This includes, among others, the DODAG's construction, the rationale of control messages' exchange, the *Trickle timer* algorithm. The advantages of compliance are twofold: firstly, the IDS exploits data that are meaningful in the context of the protocol itself, i.e., the rank value, the number of nodes attached to a single parent-node, which may prevent false positives due to misinterpretations, e.g., the attack instead of mobility, as we saw on the previous section. Secondly, it preserves the protocol's efficiency, for example, in terms of time needed for the graph's convergence, packet delay, and resource consumption, which is essential in constrained environments.

ii. *Low overhead*: Any security solution should take into consideration resources' availability, let alone when the solution is intended for LLNs. Fig. 22 indicates that a "low budget" approach should take care of control messages

exchanged and aim at exploiting the standardized ones to train the system and detect any abnormal event. Keeping the control overhead at regular levels entails energy preservation in transceivers, which are the significant consumers of constraint devices. In addition, components that serve to monitor the network, collect and/or analyze data or perform more sophisticated tasks should be hosted by the nodes with the corresponding processing, memory, storage, and power capabilities.

iii. *Scalability*: In [6] we argue that RPL can cover a wide range of IoT deployments. Once the LLNs and their routing approaches inherit IoT characteristics, such as large-scale deployment, it is reasonable to evaluate an IDS in terms of its ability to shield the protocol even when the network's size, in terms of connected devices, is significantly increased. Obviously, satisfying scalability should not jeopardize the low overhead requirement.

iv. *Robustness*: The diversity of attacks previously described entails the necessity of an IDS that can detect a range of attacks. If an IDS does not protect the network against different types of attacks, the adversary can compromise a node, in the worst case a central one, and affect both the network and applications, as we saw in Fig. 22 and Fig. 23.

v. *Extendability*: Apart from their primary performance with respect to the attacks they cope with, many IDSs can be extended to encounter additional cases. Some systems exhibit a "static", binary rationale that recognizes a known threat pattern or not and proceeds accordingly with the decision. However, new attacks and security issues emerge following the progress of research and development on the IoT. Systems should exploit all current technology assets to remain up-to-date and deal with threats that might be currently unknown. To our mind, an IDS can be extendable once its detection method becomes intelligent and its placement is sophisticated.

vi. *Low false (positive or negative) detections*: The effectiveness and detection accuracy of a system is associated with the number of false positives and/or negatives. Thus, beyond being robust and extendable, an IDS should exhibit a high accuracy rate; this means that the system sends alarms for precise attacks while minimizing the cases that attacks are overtaken. To satisfy this requirement, it is necessary to monitor different aspects of the network's operation, e.g., the control overhead combined with the number of times a node changes its parent or the PDR in line with the local repairs triggered by the RPL itself. This enables more accurate decisions, including differentiating regular but unexpected operations from attacks.

vii. *Mobility support*: Many applications with mobile IoT devices have emerged over the last decade, and RPL operation under mobility is the leading research challenge since it entails connectivity hand-overs and additional control overhead to maintain the topology [5, 6, 21]. Thus, we should not underestimate or surpass the mobility issue when it comes to the IDS design. Security mechanisms, similar to the basic ones, should consider both fixed and mobile nodes, and the literature has shown, so far, that there are no straightforward solutions.

Apparently, designing an IDS able to satisfy all the above requirements is a great challenge. In the next section, we provide classification and present the evolution of most recent IDSs in the literature to identify best practices and possible gaps in

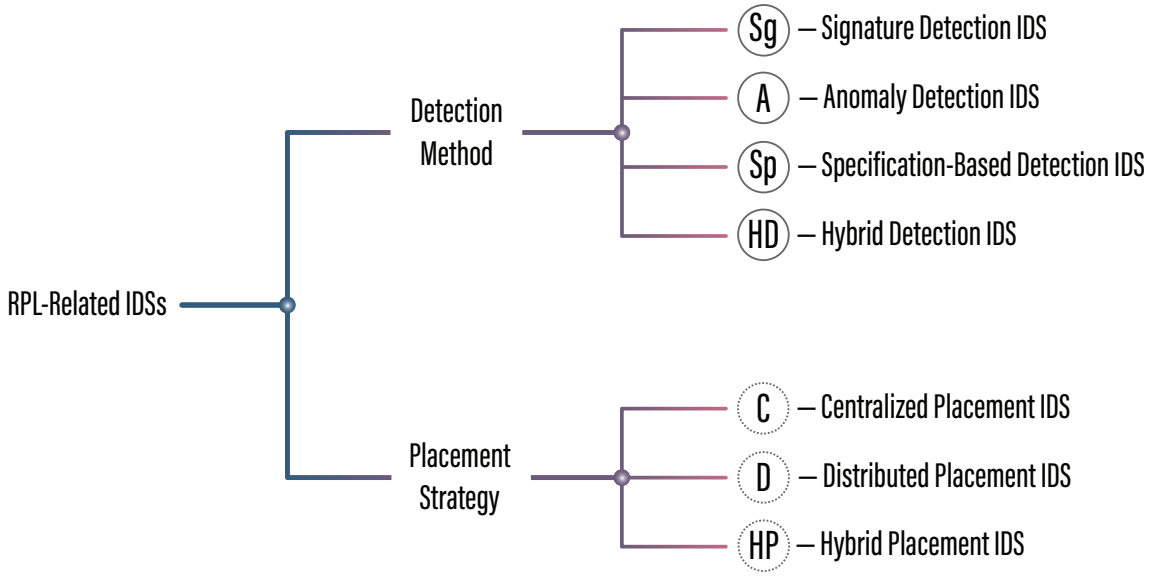


Figure 24: Classification of IDSs in respect to their detection method and their placement strategy.

the so-far related research.

3.5 RPL-related IDSs

3.5.1 Classification of RPL-related IDSs

The RPL-related IDSs in the literature are classified according to two main criteria [86]: (i) the detection method they employ and (ii) their network placement, as depicted in Fig. 24. Based on the detection method, the IDSs fall into one of the four distinct categories that follow [14, 67, 86]:

1. The *Signature Detection* (S_g) IDSs identify specific patterns in the network traffic that signify a particular attack [87]. They usually rely on databases [71], which contain known malicious signatures. While these systems consume limited resources, they are not effective against unknown threats [14] since their effectiveness depends on threat awareness.
2. The *Anomaly Detection (A)* IDSs rely on network traffic monitoring and machine-learning or statistical analysis. They develop a healthy network behavior profile and then compare it to any future network state, intending to recognize possible discrepancies that signal malicious activity. They can detect events that correspond to known or unknown threats at the expense of having high false detection rates [14, 67, 88].
3. The *RPL Specification-based* (S_p) IDSs are similar to the previous ones in the sense that they detect attacks based on divergent network behaviors' observation. However, they build healthy network models by monitoring RPL-related data specified under the security goals [14, 71, 67, 86]. This category's IDSs present high efficiency and low false detection rates while requiring less training time than the *Anomaly Detection* IDSs. Though, in the case of regularly changing environments, their manual configuration reduces their effectiveness.

4. The *Hybrid Detection (HD)* IDSs are a combination of at least two of the categories mentioned above. They tend to inherit the advantages of the combined categories while minimizing their drawbacks [86]. The prevailing hybrid scheme, at this time, is signature along with anomaly detection; to the best of our knowledge, currently, there are five *HD* systems [89, 90, 91, 92, 93], spanned across the time evolution of IDSs, and three of them, i.e., [89, 90, 91], employ signature and anomaly detection. Signature-based techniques are simple [92] and can be executed very quickly and efficiently [94] because they rely on pattern matching. Hence, they are a favored choice of combination to detect the known attacks effectively. In contrast, the unknown ones are left to be caught by the mechanism which is combined with, e.g., anomaly detection [89, 90, 91] or specification-based detection [93].

Regarding their placement strategy, the RPL-related IDSs are classified into three categories [86]:

1. *Centralized (C) IDSs* are installed and operate at the root-node of DODAG or a subset of network nodes [14, 86] assuming that resource-intensive processes are being handled by nodes that are sufficiently equipped [14]. Due to the centralized strategy, these systems cannot detect simultaneous malicious activities in different network locations, e.g., in broad networks. Additionally, such IDSs could render the network exposed in failures at the single point of defense, e.g., the sink-node [95, 96].
2. *Distributed (D) IDSs* on the opposite side are decentralized and fully implemented in every node of the network. They usually require cooperation between the network nodes [14], whose availability may be highly fluctuated [96]. Detection mechanisms are usually implemented in specific node-guards distributed across the network and are responsible for monitoring, whereas the attack mitigation functions are implemented at each node. The benefit of these systems is that threat mitigation is performed from within, as all the nodes are involved in protecting the network [14]. In this manner, the network's scalability and adaptability with a high-security level can be achieved [96]. Nonetheless, the resource consumption of these IDSs remains a significant issue.
3. *Hybrid Placement IDSs (HP)* combine the two previous categories as a means of balancing the pros and cons [14, 71, 66, 86]. In practice, they delegate the resource-demanding processes, such as monitoring, analysis, and decision-making, to the central nodes while assigning the lightweight tasks to the rest. Nevertheless, the IDSs of this category require continuous optimization; the central nodes' deployment should be done wisely and may variate for each RPL network [14].

Remarks: As an outcome, we notice that *Signature Detection* IDSs' primary weakness is their ineffectiveness against unknown threats. In contrast, the *Anomaly Detection* ones can detect even unknown threats, but they suffer from high false positives' rates. Exploiting data related to the protocol seems promising, and thus, the relevant systems dominate the detection method. However, it is interesting that only two out of five *Hybrid Detection* systems employ them in combination with either signature [92] or anomaly detection methods [93]. This leaves room for investigating the potentiality of hybrid systems that indeed contains RPL

specification-based methods.

Apart from the attack detection approach, the design of modern IDSs demands an energy-aware efficient placement strategy due to the resources' limitations of the IoT devices. The decision to place the IDS at the root-node (i.e., *Centralized*) keeps the computationally intensive tasks away from the constrained devices; however, it endows the disadvantages of the single point of failure solutions, i.e., the root-node can be compromised or cut-off. *Distributed IDSs* do not face this problem, plus they can be scaled easily but require some tasks to be executed by the constrained nodes. *Hybrid Placement* logic attempts to blend the above two approaches by keeping the "heavy" tasks for the root-node and delegating the lightweight ones to the rest. Nowadays, there is a trend towards this category, since it seems to bring satisfactory results. Our experience advocates that this trend can be further enhanced by the emergence of the softwarization paradigm [5, 6, 21]; we discuss this challenge later in this work.

We now summarize the most recently proposed IDSs based on the above taxonomy, along with a timeline highlighting their evolution.

3.5.2 The evolution of RPL-related IDSs

The research field of IDSs is vast, but only a restricted subset is appropriate for LLNs [71], i.e., considering the resource-constraints and lossy nature of the latter. In this survey, we identified 22 relevant works that have been proposed in the literature over the last seven years, i.e., from 2013 to 2020. We summarize these RPL-related IDSs in Fig. 25, which illustrates their time evolution along with their qualitative features, i.e., the incorporated detection method and the placement strategy, as well as their quantitative feature, i.e., the number of attacks they encounter.

DEMO [97] is an adaptation of "Suricata", an open-source IDS, developed in the context of the "EBBITS" European project and deals with flooding attacks. DEMO includes a frequency agility manager (FAM) and security information and event management system (SIEM). At the same time, it defines two particular non-RPL node types: the IDS node, which is responsible for the attack detection, and the monitoring nodes that monitor the network traffic and send the relevant data via a wired connection (to prevent jamming) to the IDS node for further analysis. The system is scalable and effective in detecting attacks. Regarding its extendability, the authors propose hosting the Simple Network Management Protocol (SNMP) along with special modules into the system to detect additional attacks and combine DEMO with SVELTE [92] to create a hybrid solution. Overall, exploiting non-RPL nodes and wired connectivity incurs no overhead to the RPL network but also entails a solution that is not totally RPL-compliant.

3.5.3 Signature detection IDSs

Authors in [76, 97, 98, 99, 68, 100, 101] introduce signature detection systems. Regarding their placement, the majority of them [76, 99, 68, 100, 101] are hybrid schemes, while *DEMO* [97] is a distributed and *ELNIDS* [98] is a centralized approach.

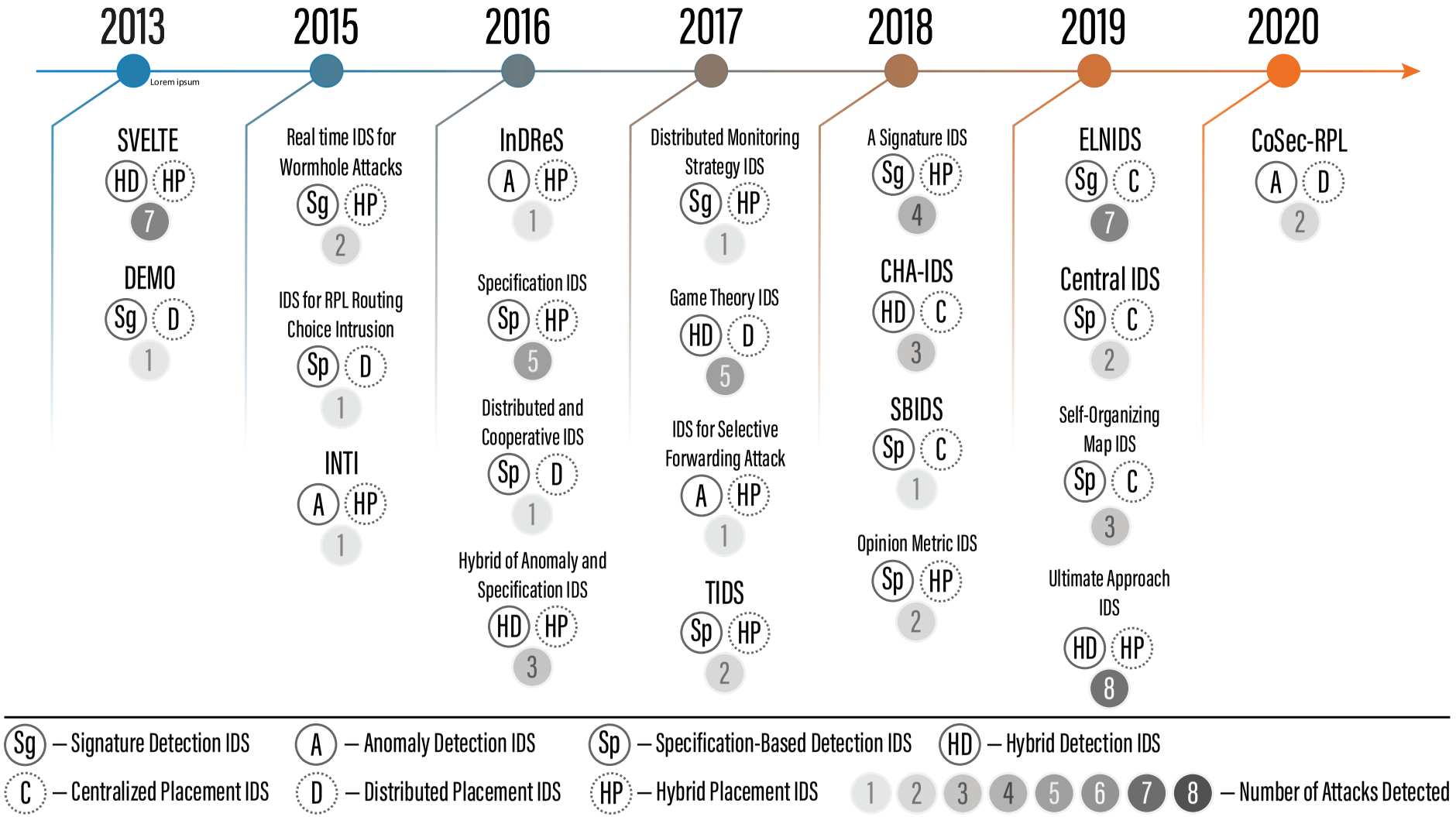


Figure 25: The RPL-related IDSs in a timeline.

Compliant with the RPL specification and hybrid regarding its placement, the *Real time IDS for wormhole attacks* [76, 100] exploits measurements regarding the nodes' Received Signal Strength Indicator (RSSI) as a means of cross-checking the network's topology. It deals with two types of wormhole attacks, i.e., by packet encapsulation and by packet relay, as well as with neighbor attacks. More specifically, during the network setup, the root-node records topology-related data and receives their neighbors' RSSI values from the rest nodes. Then, it exploits such information to estimate the distances between the nodes and compare them to the pre-saved topology data to detect discrepancies that indicate an attack. The system demands low resources and has low false detection rates. It can be extended to detect more attacks, such as clone-id, sybil, DODAG version number, and local repair attacks. However, it bases its operation on static topology information ignoring mobility issues that networks usually face.

Distributed monitoring strategy IDS for the detection of version number attacks [68] is also a hybrid placement IDS that focuses on DIO, DODAG version, and nodes' rank monitoring. The IDS defines several monitoring nodes responsible for identifying and sending to the DODAG root a list of malicious nodes detected by tracking the RPL's specification parameters. Once the root receives and merges all the incoming lists, it notifies the network nodes to interrupt further contact with the adversaries. The system behaves effectively in small and medium-scale networks, but its performance deteriorates in high false positives/negatives rates in large networks. An idea to overcome this disadvantage is to cross-monitor each node by at least two other ones.

Another hybrid placement system proposed in 2018 is the *Signature-based IDS for the IoT* [99, 101], which is designed to detect sinkhole, selective forwarding, and clone-ID attacks. It assigns the central role to the IDS router and defines a subset of nodes as IDS detectors. The router serves both as a network traffic monitoring node and a firewall and is capable of accessing the required resources. The detectors narrow the monitoring operation in their neighborhood and forward any useful information derived by a local, lightweight decision-making algorithm. Among the parameters that the IDS monitors are the RSSI and the packet drop rate. A security scheme is used for wireless communications protection; however, the authors suggest the IDS nodes are wire-connected to avoid signal jamming and eavesdropping. The system is extended [99] to also detect the DIS message attacks by monitoring the DIS sending rate and comparing it to a pre-defined threshold. The evaluation shows high accuracy and low false positives even in large networks [99]; concerning the trade-off between performance and overhead, the authors conclude that three to eight detectors should be deployed.

The most recent signature detection system is *ELNIDS* [98] that utilizes artificial intelligence and machine-learning mechanisms on central premises. It is based on ensemble learning to encounter sinkhole, blackhole, selective forwarding, sybil, clone-ID, flooding, and local repair attacks. The IDS relies on the following modules: the sniffer, the sensor events/traffic repository, a feature extraction module, the analysis engine, the signature database, and the alarm/attack notification manager. The sniffer module monitors the network traffic and records the information in the storage unit. The feature extraction module distinguishes the network traffic characteristics that aid in a later classification performed by the analyzer using ensemble models. An event is classified as an attack if any database known

signature is detected. According to its evaluation, ELNIDS exhibits high accuracy; however, similarly to the other S_g IDSs discussed, it does not consider nodes' mobility.

Remarks: We can notice that early signature detection systems [68, 76, 97, 100] aim at a special attack by design and operate deterministically. On the contrary, the latest systems of this category [98, 99, 101] expand their impact to a broad range of attacks either by adopting a hybrid placement strategy [99, 101] or by employing centralized machine-learning mechanisms [98], e.g., ensemble learning.

3.5.4 Anomaly detection IDSs

Anomaly detection systems are proposed in [102, 103, 104, 105]; most of them are hybrid regarding their placement [103, 104, 105], while *CoSec-RPL* [102] is the most recent one (published on May 2020) and adopts distributed placement logic. Both *CoSec-RPL* [102] and *INTI* [104] belong to the IDSs' minority which supports mobility.

Anomaly detection in *INTI* [104] relies on separating the network into clusters (i.e., group of nodes). Each cluster consists of a leader-node, at least one associated-node, and the member nodes. The system bases its functionality on trust estimation, using the nodes' ranks and statistics. The attack detection and the malicious nodes' isolation is performed using the Dempster-Shafer evidence theory [106]. Evaluations [14, 104, 86] showed that the system mitigates sinkhole attacks at the cost, however, of high computational processing requirements. According to the authors [104], INTI is an extendable IDS and takes into account nodes' mobility.

InDReS [103] is an improvement of *INTI* [104] that keeps the main principles of functionality while limiting the computational overhead, thus preserving resources which is critical for LLNs. Once the system identifies malicious nodes, it reconstructs the network's topology, excluding them. However, compared to its predecessor, InDReS' performance was not evaluated in terms of false positives/negatives and mobility support.

The *IDS for selective forwarding attack* [105] was proposed in 2017 and uses the Sequential Probability Ratio Test (SPRT) combined with an adaptive threshold. Its mechanism relies on two modules: the first is responsible for decision making and is implemented at the root-node. The second, used for incoming and outgoing packet monitoring, operates on the rest routing nodes. The monitoring nodes send information to the root via randomly selected paths. The root analyzes the data it receives using the SPRT and assigns every node with a probability of being malicious. The decision-making is based on a threshold above which a node is classified as malicious. Then, the root notifies the non-malicious nodes about the adversaries' presence and initiates a DODAG global repair to isolate the possible intruders. The system's evaluation indicates its effectiveness, which comes at the cost of being resource-intensive. Due to the high resource requirements, the IDS is not scalable.

CoSec-RPL [102] has been lately introduced and deals with a combination of flooding and replay attacks, namely "copycat attacks". To detect anomalies and analyze the statistical data, the system relies on a modified version of the Interquartile Range (IQR) Outlier Detection (OD) method [107], which uses the median instead of the mean value and entails less implementation complexity. The idea behind

CoSec-RPL is to identify the nodes with significantly diverse behavior. The authors tune the IDS's thresholds appropriately via multiple experiments. CoSec-RPL is triggered whenever a DIO message is received from any neighbor and monitors the time difference between consecutive DIO messages. When measurements surpass certain thresholds, a node is initially considered suspicious, and its state is characterized accordingly as "suspected". In this state, communication with the node is still allowed; however, when a second threshold is reached, the node is considered malicious, and its state becomes "blocked"; in this case, no further communication with it is permitted. Even though the system's memory requirements are not negligible, since it demands a neighboring table in every node to store relative information, they are not prohibitive for IoT devices. Thus it does fit inside a Z1 mote. *CoSec-RPL* is evaluated under both static and mobile network scenarios and is proved to be very useful. However, it performs better in fixed topologies (since mobility affects DIO message transmissions intervals). It can be extended to detect more attacks, particularly DIS flooding, DAO insider, wormhole, and spoofed copycat attacks.

Remarks: The anomaly detection IDSs are a minority of the systems under analysis (four out of 22), probably because anomaly detection is, by definition, a general method loosely coupled with the RPL itself. So far, most systems [103, 104, 105] have been exercised with only one attack type, but they can potentially detect unknown attacks. Such a feature relates to the anomaly detection mission, which identifies unusual or even unknown "behavior" and attributes it to an attack. They mainly exploit intelligent mechanisms, e.g., clustering, probability theory, and statistical parametric or non-parametric tests, along with appropriately defined thresholds. Of course, thresholds' tuning is an essential issue since it may result in either high false positives or negatives. As we will see later in this section, combining the advantages of anomaly detection with other detection methods brings very positive results [89, 90, 91, 93]. It is indicative, for example, that they dominate as a component of the *Hybrid Detection (HD)* systems.

3.5.5 Specification-based detection IDSs

IDSs of this category [67, 95, 108, 109, 110, 111, 112, 113] share the feature of taking into account RPL-related information, e.g., control messages, rank value, DODAG information and try to identify an attack exploiting such knowledge. Regarding their placement, there is a shared trend.

IDS for RPL routing choice intrusion [111] is a distributed placement system that relies on monitoring DIO messages' fields, nodes' parents, and rank values, as well as the number of nodes connected to a single parent to detect decreased rank attacks. The idea is that a low-rank value advertised by a node that presents an increased number of nodes attached to it indicates that this node is probably malicious. Energy requirements were taken into account, and the IDS can operate in large networks.

The IDS proposed in [67] is a hybrid placement system that, similarly to the *INTI* [104], divides the network into clusters and uses specification-based detection to mitigate the attacks. It is designed to repel sinkhole, worst parent selection, local repair, neighbor, and DIS message attacks. The system is effective, it presents low false detection rates, and due to its low energy demands, it is scalable. It can be extended to detect a broader range of attacks; however, it does not address mobility

issues.

The *Distributed and Cooperative Verification IDS to defend against DODAG version number attack* [113] suggests that when the nodes receive a DIO message containing an increased DODAG version, the message should be accepted once it is confirmed. If the sender is the root-node, the receiver will accept the message; otherwise, the receiver requests the DODAG version number from its two-hops-distant neighboring nodes. This functionality demands two additional message types, the "CVQReq" for the request and "CVQRep" for the reply. Evaluation results show that the IDS is effective against the DODAG version attack; however, the false detection rate increases in proportion to the attacking nodes' number. Furthermore, the control overhead is significantly low.

TIDS: Trust-based IDS [110] is a hybrid placement system that mitigates sinkhole and selective forwarding attacks using the notion of trust. TIDS relies on Subjective Logic [114], incorporating variables both for trust and uncertainty, and considers a node as malicious when its disbelief value is higher than its belief value. Trust values are calculated based on the level of nodes' good cooperation and conformity with the RPL specification. Each node observes its neighbors and forwards the recorded data to the root-node using a new control packet, namely "Trust Information (TRU)". The root-node has the required resources for the purpose and calculates the trust values. The system was evaluated and found to successfully detect sinkhole attacks even in large topologies (at the expense of high energy demands on the root-node). In contrast, selective forwarding attack was discussed only in a theoretical context. According to the author, TIDS is useful in topologies comprised solely of static nodes, and it can be extended to mitigate version number attacks additionally.

SBIDS: Sink-based Intrusion Detection System [112] is a centralized system designed to detect decreased rank attacks in non-storing RPL networks. The root-node, which is considered trusted by default, marks a node as malicious by monitoring the rank changes and defining thresholds accordingly, i.e., it records the previous and current ranks of parent-nodes, and establishes a threshold for parent switching. SBIDS considers both static and mobile nodes. Its evaluation revealed high accuracy in large networks in both cases; however, its performance degrades as the number of attacking nodes increases, especially when mobility is considered. The IDS incurs an overhead of around 20 percent compared to the unprotected network consumption concerning the power consumption. Finally, SBIDS can be extended to accommodate more routing metrics and, thus, repel additional attacks.

Opinion Metric based Intrusion Detection System for RPL Protocol in IoT [109] is a hybrid placement IDS, able to mitigate sybil and flooding attacks, utilizing an opinion metric-based mechanism which is based on subjective logic [114]. The nodes monitor their neighbors' transmissions and rate them according to their compliance with the RPL specification. Nodes that behave as per specification principles are rated positively, whereas the diverging ones are rated negatively. The ratings are later aggregated to the root-node, where the subjective logic (the "Θ" consensus operator) is employed for the malicious nodes' detection. A node is considered malicious when the aggregated degree of disbelief exceeds a threshold. The system is solely evaluated in detection performance, and a considerable number of false detections were recorded. Nevertheless, the authors plan to extend their work and

consider additional routing attacks using a neural network trust model.

A Central IDS able to mitigate flooding and DODAG version number attacks was proposed in [95]. The system is implemented at the root-node and uses genetic programming to generate the IDS's algorithm automatically. The root continuously analyzes the network traffic and extracts 50 features, which are later used for the constitution of the genetic programming trees. The last generation's best individual (tree) is evaluated for both flooding and DODAG version number attacks, and two corresponding detection algorithms are obtained. In its current version, a central logic is adapted. The root-node executes the resource-demanding tasks; the authors also suggest a decentralized fashion of operation, but this entails further challenges to be addressed. The system is highly effective, probably due to centralized monitoring, which provides a global network view. Aspects such as resource requirements, scalability, extendability, and mobility support, were left out of the system's evaluation.

Self-Organizing Map IDS for RPL Protocol Attacks [108] exploits machine-learning and more precisely Self-Organizing Maps (SOM), built centrally to the RPL network, to detect flooding, sinkhole, and DODAG version number attacks. The authors elaborate on the way that several modules collaborate to generate the maps. Initially, synthetic data from numerous simulations of different real-life scenarios were produced and used as input to the "aggregator" module. This module utilizes six packet fields (i.e., message type – DIO/DIS/DAO, IP addresses of the sender and destination nodes, current DODAG version, current sender node rank, Unix timestamp), pre-processes the input data and provides as an output six features (i.e., DIS, DIO, DAO, DODAG version changes, rank changes to total messages ratios in the timeframe, average power consumption on the destination node in the timeframe). These features are getting normalized by the "normalizer" module, to be used by the "trainer" module to generate the maps. Simulations run by the authors indicate that the IDS can identify the attacks.

Remarks: Not surprisingly, eight out of 22 systems (36.4 percent), according to Fig. 25, fall in this category. Either intuition or experience leads the researchers to exploit the cardinal RPL data structure, i.e., the graph, and its relevant information, e.g., control messages and *Trickle timer* algorithm, in IDS design. However, judging by the outcome, the specification-based detection, either as a single detection method or in combination with others, performs moderately regarding the number of attacks. In the worst-case, systems detect one attack [112, 111, 113], while it is remarkable that they perform better once the hybrid placement strategy is adopted [67, 109, 110], or when RPL-related information is processed by machine-learning mechanisms [108, 109]. Indeed the specification-based systems that exploit clustering, trust schemes, genetic programming, and artificial neural networks to process the RPL-monitoring parameters outperform those that take these parameters into account without any kind of intelligence.

Here, the aftermath is that tight coupling with the protocol itself is not sufficient but rather a starting step. Mixing techniques can help to develop robust systems that do not jeopardize performance and cost.

3.5.6 Hybrid detection IDSs

SVELTE [92] is one of the oldest RPL-related IDSs. It is a hybrid placement system that consists of three modules: (i) the 6LoWPAN Mapper (6Mapper), implemented

at the root-node, maps and keeps track of the DODAG along with the parent and neighboring information of each node; (ii) the intrusion detection module, which is also executed centrally, relies on the RPL specification, signature and anomaly detection to specify the attacks, and; (iii) the distributed firewall and response module that prevents the out-of-network attacks and is implemented in every node. SVELTE combines all three detection methods and tries to achieve a trade-off between the storage cost of S_g and the computing cost of anomaly detection techniques. The system's evaluation revealed its effectiveness against blackhole, selective forwarding, sinkhole, and DODAG inconsistency attacks. However, since SVELTE uses a rank threshold to detect anomalies, it suffers from high rates of false positives/negatives [67, 92, 103, 86]. In addition, it has significant resource requirements and does not take into account mobility issues. Improvements of SVELTE [115, 116] reduce false detections and add geographical hints of the malicious nodes, increasing the IDS's robustness by allowing it to discover clone-ID, sybil, and wormhole attacks additionally.

Hybrid of Anomaly-Based and Specification-Based IDS for IoTs Using Unsupervised OPF Based on MapReduce Approach [93] is a full hybrid approach that encounters selective forwarding, sinkhole, and wormhole attacks. The system combines an Anomaly Agent-Based IDS (AA-IDS) with several Specification Agent-Based IDSs (SA-IDSs) and considers the leaf-nodes traffic solely to the root. The SA-IDSs, implemented at the router-node(s), are used for traffic monitoring and the identification of malicious nodes. Once traffic is analyzed, the output data are embedded into data packets and forwarded to the root-node, where the AA-IDS resides. AA-IDS employs the unsupervised Optimum-Path Forest (OPF) algorithm [117] to cluster the collected data and proceed with the anomaly detection. The decision that classifies a node as malicious is based on a voting mechanism that considers both local results of SA-IDS agents and the global analysis of the AA-IDS. The system can also be extended to mitigate blackhole, and decreased rank attacks.

The authors developed a dedicated RPL WSN simulator for their evaluation analysis and provided high accuracy rates regardless of the network size, justifying their system's scalability; their evaluation, however, considers only a static topology. Regarding the energy requirements, abundance was taken for granted for all kinds of nodes. Still later in a theoretical context, it was concluded that the IDS could be used in real-world IoT applications by offloading the resource-intensive tasks from the root-node to an external device; obviously, such assumptions leave space for improvements.

Game Theory IDS [89] is a distributed placement IDS that combines signature detection for the known attack patterns and anomaly detection for the unknown ones. In this way, the system is proved to encounter a considerable number of attacks, i.e., flooding, sinkhole, blackhole, sybil, and wormhole attacks. The Nash Equilibrium Game Theory is used to set a game between the IDS entities and the attackers; when the system detects a traffic pattern that reaches a threshold, it considers it an anomaly. To reduce false detections, the authors combine the IDS with a reputation system. The evaluation of the IDS assumes both fixed and mobile nodes and reveals low requirements on resources.

CHA{IDS [90] is a centralized system that elaborates on the IPv6 compressed header's analysis using machine learning. The root-node extracts data from the

Hybrid Detection IDS	Detection Method			Placement Strategy		Number of Attacks	Attacker Localization
	Signature Detection	Anomaly Detection	Specification Detection	Centralized Placement	Distributed Placement		
SVELTE	✓	✓	✓	✓	✓	7	✓
Hybrid of Anomaly and Specification IDS		✓	✓	✓	✓	3	✓
Game Theory IDS	✓	✓			✓	5	
CHA-IDS	✓	✓		✓		3	
Ultimate Approach IDS	✓	✓		✓	✓	8	✓

Figure 26: Overview of the Hybrid Detection IDSs.

network traffic, which are later used as an input to the "J48" algorithm [118] for the attacks' detection. In this way, it detects flooding, sinkhole, and wormhole attacks, taking place either individually or in combination, with high accuracy. According to the authors, the system exhibits a good performance regarding the trade-off between performance and overhead. However, in its current version, it does not succeed in locating the attacker's position; future extensions and possible combinations with other distributed placement schemes could offer this capability. Furthermore, extensions could improve the system to mitigate sybil, clone-ID, DODAG version number, and local repair attacks.

Lastly, the *Ultimate Approach IDS of Mitigating Attacks in RPL Based Low Power Lossy Networks* [91] follows a holistic approach, is full hybrid regarding its design, and encounters the maximum number of attacks, i.e., eight. More specifically, the system encounters sinkhole, DODAG version number, flooding, neighbor, wormhole, decreased rank, clone-ID, and sniffing attacks and can detect events that originate both inside and outside the network. The IDS incorporates many non-mobile sink/sub-DODAG parent-nodes that can detect both known signatures and anomalies. The system uses blockchain and calculates trust values to detect the attacks and isolate the adversaries. The author presents a conceptual framework of their approach, stating its effectiveness along with low resource requirements and its ability to be extended. The system seems to support mobile nodes since only partially the root, and the sub-DODAG parents are considered to be fixed-positioned.

Remarks: The time evolution of IDSs (Fig. 25) shows that hybrid detection systems span across the whole investigation period, i.e., 2013 – 2020, indicating that even in the early systems, such as SVELTE [92], the researchers pinpointed that combining the attacks' detection methods brings advantages to the process. The basic and, probably, the apparent benefit is quantitative and regards the number of attacks that the system can encounter; this ranges from three to eight as depicted in Fig. 26.

Further benefits include the ability of some systems to localize the adversary [91, 92, 93], as well as the detection accuracy rate in conjunction with low

resource overhead, especially when the developed mechanisms are appropriately located both in central and distributed nodes. In particular, appropriately tuning the parameters of *SVELTE* [92] can offer as much as 100 percent of detection accuracy and zero false positives. In comparison, solution [93] shows an average of 93.3 percent accuracy with less than 3.3 false positives for multiple runs. *Game Theory IDS* [89] reports an average of 98.6 percent accuracy and less than 2.5 percent of false positives for a variety of setups, while *CHA{IDS* [90] shows an accuracy within $85.2 - 100$ percent and up to 0.058 percent false positives, in the worst case.

Evaluating these numbers in real-world environments is a challenging issue that certainly deserves further investigation, e.g., whether they allow a realistic operation of the particular IDSs. This angle of investigation is associated with: (i) the considered use-case in terms of required security level and affordable control overhead or processing cost; and (ii) the type of involved mitigation action and its impact, since this determines the communication or performance issues a false positive causes.

Most of these hybrid systems use machine-learning, i.e., *Game Theory IDS* [89], *CHA{IDS* [90] and [93] employ Nash equilibrium game theory, the "J48" algorithm, and unsupervised data mining, respectively. We omitted a more in-depth discussion and comparative analysis on the involved algorithms in the IDSs at this point of the investigation since we mainly focus on their systemic aspects. Such investigation requires comparisons between different approaches (e.g., machine-learning vs statistics-based) under a given environment or theoretical investigation on their impact on the computational burden, as an example. From our point of view, this exercise diverges from the given scope of this work. However, this issue is important and complex enough to deserve an independent study. Consequently, it is considered future work.

Next, we provide a summary that compacts the individual remarks into a set of best practices and identified gaps in IDS design.

3.5.7 Best Practices & Gaps

The so far research, reflected on the IDSs under analysis, reveals best practices in the design of RPL-related IDSs. The most important is that utilizing detection methods in conjunction can bring a high score regarding the number of attacks detected. In particular, anomaly detection contributes as a general method to detect known and unknown threats and performs excellent with either signature or specification-based methods, which provide some "knowledge" to the process, i.e., patterns or threshold crossings RPL-related parameters. Another best practice is to exploit both distributed and centralized mechanisms to achieve optimal placement in the detection mechanisms. This includes coarse-grained, lightweight monitoring at every node, which conditionally triggers fine-grained, resource-demanding processes executing at central premises, e.g., machine-learning. The third point is that own-detection narrows the IDSs' mission; some systems [91, 92, 93] go beyond it by identifying the attacker(s) and mitigating the threats using information relevant to the RPL protocol.

This observation, combined with the summary of the most robust systems – Fig. 26 – reveals that eventually, a minority of IDSs follow a holistic approach that deals with the threefold mission of detection, identification, and mitigation. Thus, there are several gaps in the literature regarding methods: to identify and

then mitigate the intruder, to detect multiple attacks, to deal with false positives decisions, e.g., how and when a blacklisted node comes back to the network and which are the coincidences of its isolation. Our analysis also finds the lack of an architecture beyond a hybrid-wise fashion of combination and builds up a "polymorphic" system able to adapt in dynamic conditions.

Finally, we notice a lack of IDS evaluation in real environments, i.e., test-beds, since most systems in our analysis are evaluated using simulations. More specifically, 16 out of 22 IDSs utilize Contiki Cooja [37], while NS-2, Matlab and TOSSIM simulators are also used for evaluation in [103], [89] and [98], respectively. Only authors in CHA{IDS [90] document utilizing Cooja in combination with a test-bed facility, however, without providing the details of the latter. Our previous experience with test-beds participating in the FED4FIRE [119] and GENI [120] federations, in the context of 5G network slicing research [121, 122, 19], shows that it would be interesting, but also very challenging, to deploy complete IDSs in test-beds for evaluation reasons and address possible issues that arise. Currently, the Sharing Artifacts in a Cybersecurity Community Hub (SEARCCH) project [123] offers a facility that provides validation, repeatable sharing, and reuse of security-related research results. A relevant initiative for IoT security could establish a common framework where open-source IDS code could be released and comparatively evaluated, e.g., in a common environment with the same methodology and evaluation scenarios.

The section that follows proceeds with a comparative analysis of the IDSs under investigation that includes: (i) a complete mapping of IDSs to the type of attacks they encounter; and (ii) their comparison in the light of the design requirements we introduce. The ultimate goal is a list of four guidelines that, to our mind, modern IDSs should follow.

3.6 Comparative Analysis & Insights

3.6.1 Mapping IDSs to Attacks

We start our comparative analysis by assigning each of the 22 most recently introduced IDSs under discussion to the RPL-related attacks they tackle. This is a challenging and not straightforward task since it depends on how an IDS covers the addressed attack(s). To this point, our literature study reveals that different approaches are spanning from simulating all or some of the attacks to conceptually supporting coverage for all or subset of the attacks under study. In the case of simulation approaches, differences also concern the simulation environments and the metrics used to evaluate the IDSs' performance.

To proceed with our mapping, we listed the attacks concerning the classes they belong to and are illustrated in Fig. 21. Next, to highlight the aforementioned differences, we mark in bold the IDSs in a row when they are evaluated through simulation (e.g., based on Contiki Cooja, NS-2, Matlab, or TOSSIM) for the attack on the same row on Table 5. Regular fonts indicate that no simulation is carried out. Regular fonts with the star mark refer to the IDSs that can be extended to tackle an attack, according to the corresponding authors. The outcome is summarized in Table 5 which synthesizes the knowledge gained from Sections 3.4 and 3.5.

To better highlight the mapping process, we give two indicative examples. The authors in [67] utilize Contiki Cooja [37] and evaluate their IDS against sinkhole, worst parent selection, local repair, neighbor, and DIS message attacks; their simu-

Table 5: Mapping the IDSs to the type of mitigated attacks.

		Attacks		IDS
RESOURCE DEPLETION ATTACKS	DIRECT	Routing Table Overload		-
		Flooding		**[89]**, **[90]**, [91], **[95]**, **[97]**, **[98]**, [102], **[108]**, **[109]**
	INDIRECT	Local Repair		**[67]**, **[98]**, [76]*, [90]*
		DIS Message		**[67]**, **[99]**, [102]*
		DODAG Inconsistency		[92]
		DODAG Version Number		**[68]**, [91], **[95]**, **[108]**, **[113]**, [76]*, [90]*, [110]*
NETWORK TOPOLOGY ATTACKS	SUB-OPTIMIZATION	Sinkhole		**[67]**, [89], **[90]**, [91], **[92]**, **[93]**, **[98]**, [99], **[103]**, **[104]**, **[108]**, **[110]**
		Wormhole		**[76]**, [89], **[90]**, [91], [92] (D. Shreenivas' version [116]), **[93]**, [102]*
		Replay		**[102]**
		Neighbor		**[67]**, [76], [91]
		Routing Table Falsification		-
		Rank Attacks	Decreased Rank	[91], **[111]**, **[112]**, [93]*
			Increased Rank	-
			Worst Parent Selection	**[67]**
	ISOLATION	Blackhole		[89], [92], **[98]**, [93]*
		Selective Forwarding		**[92]**, **[93]**, **[98]**, [99], **[105]**, **[110]**
		DAO Inconsistency		[102]*
NETWORK TRAFFIC ATTACKS	EAVES-DROP	Sniffing		[91]
		Network Traffic Analysis		-
	MISAPPR-OPRIATION	Clone-ID		[91], [92] (D. Shreenivas' version [116]), **[98]**, [99], [76]*, [90]*
		Sybil		[89], **[98]**, [92] (D. Shreenivas' version [116]), **[109]**, [76]*, [90]*

– IDSs in **[bold]** are evaluated through simulations for the corresponding attack.

– IDSs with the star mark (*) can be extended to encounter the corresponding attack according to the authors' declaration in the relevant publication.

– The rest IDSs are mapped to the corresponding attack according to the authors' declaration in the relevant publication.

lation results include true positives/negatives, false positive/negatives, and energy consumption. For this reason, the reference [67] appears in bold in rows: $3, 4, 7, 10$ and 14 that refer to the aforementioned attacks. On the other hand, *SVELTE* [92] is an example for which the authors declare its effectiveness against selective forwarding, sinkhole, blackhole, and DODAG inconsistency attacks. However, they evaluate it only for the first two attacks using the metrics of true positive rate, energy, and memory consumption in Contiki Cooja [37]. Thus, it appears in bold only in rows 7 and 16; the rest of the table is with regular fonts. The same applies to SVELTE's improvement [116] where the corresponding authors claim effectiveness against clone-ID, sybil, and wormhole attacks due to additions considering the malicious nodes' geographical position. However, relevant to these new attacks, results are not provided. The only simulation results refer to reducing false detection rates for the initial attacks having already been evaluated, i.e., selective forwarding and sinkhole.

Mapping of Table 5 reveals that the vast majority of the RPL-related IDSs (73 percent) deal with network topology attacks; this is expected since the DODAG and its related mechanisms, i.e., the *Trickle timer* algorithm, and parameters, i.e., DODAG ID and rank values, play a cardinal role on the RPL networks. An even

more interesting fact is that as much as 54.5 percent of the IDSs focus on the *Sinkhole* attacks, indicating the sink-node's significant role in such networks. On the contrary, network traffic attacks do not attract significant attention, probably due to the passive nature of eavesdropping attacks, which are difficult to be detected. To our mind, energy-awareness, in conjunction with resources' limitations on IoT networks, both create an emerging field of research regarding resource depletion attacks and the corresponding IDSs.

Table 5 also shows that some IDSs [67, 91, 98] are more robust than others since they encounter a greater number of attacks; in fact, they repel different attacks that expand to all three categories, i.e., resource depletion, network topology, and network traffic attacks. Among them, the *Ultimate Approach* [91] introduces a full-hybrid, conceptual framework where the authors discuss but do not evaluate their IDS concerning the attacks encountered. On the contrary, the Specification-Based IDS [67] and ELNIDS [98] tackle five and seven attacks, respectively, for which simulation analysis and results are provided. SVELTE [92] addresses seven different types of attacks, evaluates a subset of them through simulation, and gives an indication towards the potentiality of full-hybrid IDSs to deal with a broad spectrum of attacks. Overall, most works (17) proceed with comprehensive simulation approaches in the sense that they evaluate all the attacks the corresponding authors claim tackling. A small subset of works [76, 89, 92, 99] evaluate through simulation a portion of attacks they investigate, while Kaur [91] introduces a conceptual work that misses simulation results.

In the following section, we elaborate on comparing those RPL-related IDSs in light of the design requirements we introduced.

3.6.2 IDSs' Comparison

Table 6 presents the comparative overview of the 22 IDSs under analysis (their order is consistent with their time evolution on Fig. 25) in respect to the seven design requirements introduced and discussed in Section 3.4.3. The comparison shows if a system satisfies (✓) or not (✗) each of the requirements, while a dash (−) denotes that no information is available. We are essentially based on the respective authors' claims in the relevant articles, and, in some cases, we exploit feedback from them for clarifications. This way, we manage to build a table completed as much as 80.5 percent, which indicates that both the design requirements and the comparison itself are meaningful.

Elaborating on RPL-related systems, it is expected that the majority of them are compliant with the protocol. However, even if they are designed for LLNs only one-third of them presents low overhead; the rest are either high-cost solutions or do not clarify their trade-offs in terms of performance and cost. Half of the systems are scalable, and the rest are not evaluated for large-scale deployments.

Regarding robustness, most of the systems deal with up to four attacks, while almost 37 percent of the IDSs are single-attack solutions (Fig. 25). As a result, 22.7 percent of them appear to be robust since they claim to cope with five or more attacks; among them, only the Specification-Based IDS [67] and ELNIDS [98] are evaluated for all the attacks they investigate. Despite these relatively low scores, a significant number of IDSs (almost 73 percent) claim that they are extendable and able to detect and mitigate more attacks once they are modified. Unexpectedly, we notice that robustness is not necessarily associated with a low overhead cost,

Table 6: Comparative overview of RPL-related IDSs.

IDS	Criteria						
	i	**ii**	**iii**	**iv**	**v**	**vi**	**vii**
SVELTE [92] [116]	✗	✗	{	✓	✓	✗	✗
DEMO [97]	✗	{	✓	✗	✓	{	✗
Real time IDS for Wormhole Attacks [76]	✓	✓	{	✗	✓	✓	✗
IDS for RPL Routing Choice Intrusion [111]	✓	✓*	✓	✗	{	{	✗
INTI [104]	✓	✗	✓	✗	✓	✓	✓
InDReS [103]	✓	✓	{	✗	✓	{	✗
Specification-Based IDS [67]	✓	✓	✓	✓	✓	✓	✗
Distributed and Cooperative Verification IDS [113]	✗	✓	{	✗	{	✓*	✗
Hybrid of Anomaly and Specification Based IDS [93]	✓*	✗	✓	✗	✓	✓	✗
Distributed Monitoring Strategy IDS [68]	✓	{	✓	✗	{	✓*	✗
Game Theory IDS [89]	✓	✓	✓	✓	{	✓	✓
IDS for Selective Forwarding Attack [105]	✓	✗	✗	✗	{	{	✗
TIDS: Trust based IDS [110]	✗	✗	✓	✗	✓	✗	✗
Signature IDS [99]	✓	✗	✓	✗	✓	✓	✗
CHA { IDS [90]	✓	✗	{	✗	✓	✓	✗
SBIDS: Sink-based IDS [112]	✓	✗	✓	✗	✓	✓	✓
Opinion Metric based IDS [109]	✓	{	{	✗	✓	✗	✗
ELNIDS [98]	✓	{	✓	✓	✓	✓	✗
Central IDS [95]	✓	{	{	✗	{	{	✗
Self-Organizing Map IDS [108]	✓	{	{	✗	✓	{	✗
Ultimate Approach IDS [91]	✓	✓*	{	✓	✓	{	✓*
CoSec-RPL [102]	✓	✗	{	✗	✓	✓	✓

Design requirements:
i = RPL specification compliance
ii = Low overhead
iii = Scalability
iv = Robustness
v = Extendability
vi = Low false positives
vii = Mobility support

* = Under certain conditions or estimated but not evaluated
✓ = Satisfied
✗ = Not Satisfied
– = No Information Available

i.e., three out of five robust systems present low overhead [67, 89, 91], while two of them [67, 89] also combine robustness with low false detection. These findings indicate that research towards balancing the trade-off among security (expressed with robustness and extendability), performance (in terms of low false positives, scalability, and RPL compliance), and cost (associated with low overhead) can bring fruitful results.

Finally, an insightful outcome of Table 6 is that 77 percent of IDSs do not consider the mobility issue, probably due to the difficulties that it entails. We demonstrate, for example, on Fig. 22 and 23 that nodes' mobility causes control overhead comparable to some attacks, e.g., decreased rank and blackhole attack; this could mislead the decision-making of an IDS with impact on false positives rate. Indeed, IDSs that deal with sinkhole [67, 89, 90, 92, 93, 98, 99, 103, 104, 108, 110], wormhole [76, 89, 90, 92, 93] and rank attacks [67, 93, 111], mishandle nodes' mobility and interpret it as an attack pattern (since, for example, mobile nodes send control

messages from different network places and in irregular intervals compared to the fixed ones). In addition, mobility patterns can be known a priori (e.g., a city-bus, with IoT nodes on it, follows the same route every day) or completely random; in the latter case, even probabilistic or machine-learning models face accuracy issues in predicting nodes' status and, thus, providing appropriate input to an IDS.

These observations make clear that an IDS should monitor and evaluate several parameters in conjunction with each other to combine high accuracy with low false positives.

3.6.3 Guidelines

So far, it is clear that there is no one-for-all solution that mitigates a great portion of the RPL-related attacks and, at the same time, meets all the design requirements we introduced. As an aftermath, we present here some basic guidelines for an up-to-date IDS.

- *Trade-off between security and performance*: This notice reflects the need for robust and extendable systems while simultaneously presenting high accuracy and ability to operate regardless of the network's scale and be compliant with the RPL to preserve the protocol's native performance. Table 6 shows that only [67, 89] are robust systems and at the same time satisfy the criteria i, ii and vi. Thus, there is room for research and improvements, especially if we consider that out of 21 different RPL-related attacks, a critical portion of the IDSs, 77 percent, deal with up to only four of them. Furthermore, current literature lacks proposals that cope with certain attacks, such as routing table overload and falsification, increased rank, and worst parent selection. Simultaneously, the built-in security mechanisms of RPL have not been thoroughly investigated and are considered optional features in the RPL specification. Their implementation and further research on their effectiveness against the various attacks may bring positive results for the trade-off between security and performance.
- *Trade-off between security and cost*: Designing security systems for LLNs should take the cost as a primary concern. The fact that 63 percent of IDSs do not satisfy the low overhead and robustness criteria simultaneously and 27 percent do not provide any cost-related results indicates that current research underestimates this issue. Of course, a high level of security entails cost barriers. However, three systems [67, 89, 91] are robust and entail low overhead simultaneously, while [67] exhibits the best behavior in respect to all the requirements defined. Probably the last seven years are a trial period during which many ideas and approaches are under investigation. Fortunately, the above IDSs provide evidence that we gain knowledge and invest in holistic solutions that combine security, performance, and cost.
- *Mobility support*: Mobility is a trend of modern IoT networks and, among others, contributes to widening the networks' range deployment. Current IDSs' literature is not mature enough to provide solutions that deal with this issue efficiently, i.e., to combine it with robustness and low false positives rates. Mobility is the least satisfied among our defined requirements. Previously in this section, we justified this weakness, which provides room for research, especially in the light of results and solutions regarding the RPL under mobility [5, 6, 21]. Both from our previous experience [5, 6, 21] and from the

systems that support mobility [89, 91, 104], we conclude that hybrid solutions regarding the detection method and/or the placement strategy could efficiently contribute to building efficient IDSs.

- *Alignment to the IoT evolution*: IoT advances towards supporting applications with diverse, challenging requirements, e.g., ultra-low delays, mobility, or high capacity of nodes, through exploiting Edge Cloud Computing, Software-Defined Networks (SDN), and 5G or Beyond Networks. New critical IoT installations (e.g., Industry 4.0 or Smart-city) come together with new sophisticated attacks in this complex ecosystem. Consequently, an up-to-date IDS should be extendable, tune security/cost and security/performance trade-offs to particular IoT applications, and benefit from such advanced networking, processing, and storage capabilities. For example, Edge Clouds' incorporation brings significant processing and storage resources that can support Artificial Intelligence/Machine-Learning (AI/ML) capabilities, e.g., data analysis, clustering, or prediction. Such features perfectly match with RPL extensions inspired by the SDN paradigm [5, 6, 21] that enables modularity, adaptation, and dynamicity; e.g., to jointly recognize mobility patterns, detect, and mitigate unknown attacks.

  The hybrid approaches are consistent with the above direction since their centralized mechanisms can be driven by intelligent mechanisms deployed at Edge Clouds, their decisions enforced by SDN controllers. Simultaneously, the nodes are assigned lightweight tasks, such as local monitoring and/or low-complexity algorithms, i.e., for instantaneous reporting or acting upon attacks.

3.7 Conclusion

The RPL routing protocol is a relatively mature technology that allows IPv6 routing in LLNs. By investigating RPL attacks with special attention on their impact in terms of control overhead and application performance and evaluating the related IDSs in the literature, we conclude that there is room for research regarding holistic solutions with specific tailored-made characteristics, such as: monitoring and exploiting several features in conjunction, e.g., network conditions and protocols' mechanisms, handling mobility, respecting resource constraints, while at the same time providing a high level of security reflected in robustness and low false positives. We introduce seven design requirements that a modern RPL-related IDS should satisfy. Moreover, we provide a list of four concrete guidelines that, according to our experience, future approaches should consider. In fact, we are currently working on an SDN-inspired, machine-learning-based polymorphic IDS that exploits our findings and brings promising results.

3.8 Chapter Summary

In this chapter, the main limitations faced by the current state-of-the-art IDS solutions are highlighted. Those can be summarized as following:

- RPL security front has several issues to be addressed
- There are open research questions regarding centralized, intelligent, SDN-like IDSs for IoT and for RPL in particular

The next chapter presents a state-of-the-art solution on the front of SDN-like

Intrusion Detection Systems for IoT and RPL in particular based on the above-gained knowledge.

4 IoT Security

Among the challenges described in Chapter 1, IoT security is a major one. Moreover, RPL has several issues not only because of not implementing encrypted communications by default, but also because several features of the protocol (e.g., Local /Global repairs, Rank of nodes), and technologies (e.g., DODAG construction) can be exploited to create attacks and malicious activities disrupting or even ceasing the proper operation of the protocol. Along comes the need for a state-of-the-art, SDN-like, centralized infrastructure, tackling the above challenges. Such an IDS, utilizing machine learning, artificial intelligence, and several mechanisms from other areas, is presented in this chapter.

4.1 Introduction

In the aftermath of the security issues stemming from the attacks described in Chapter 3, along with the diversity of those attacks, the particularity of malicious nodes' placement in the network, and the detrimental effects of combining simple attacks, are just a few of the challenges need to be faced by such an IDS. Since many of the attacks share common features regarding their origin, e.g., local repair self-healing mechanism exploitation, or their impact, e.g., irregularities in the data and/or control packets rates of the affected nodes, the proposal invests in this observation. Thus, *ASSET* accommodates a minimum set of mechanisms that succeed in the three-fold mission of an IDS and operate in conjunction without refuting each other.

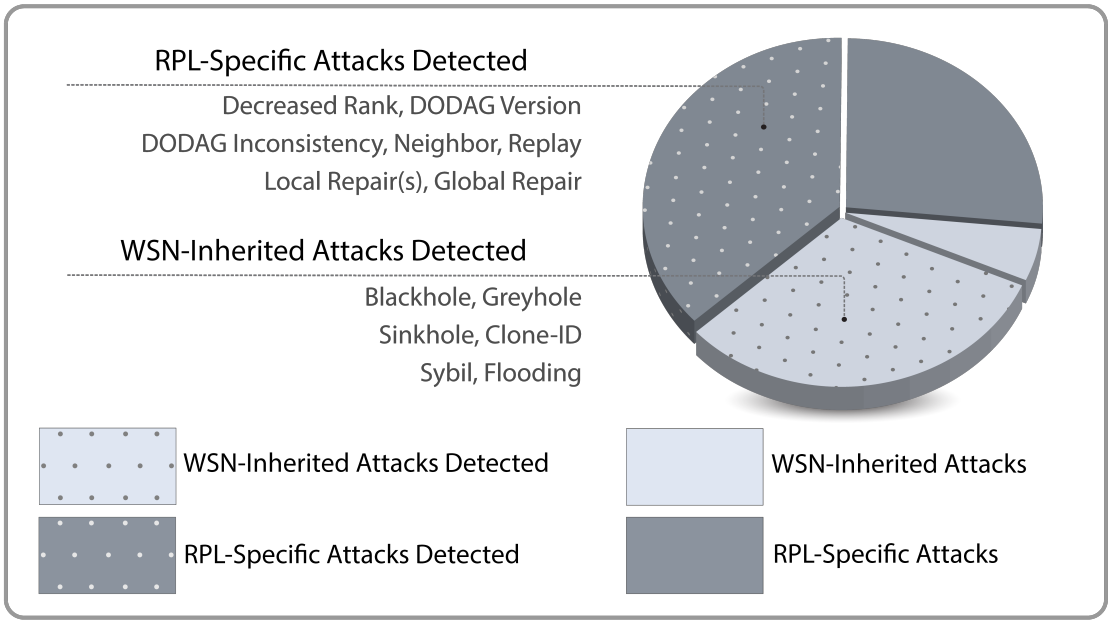


Figure 27: Attacks Detected by *ASSET*.

Figure 27 illustrates that *ASSET* handles above 50 percent of both WSN-inherited and RPL-specific attacks. Next, *ASSET*'s details are presented and dis-

cussed. Moreover, some notes also about RPL's weaknesses and issues are essentially described below.

4.2 RPL Weaknesses & Intrusion Detection Systems

In RPL's storing mode, the sink knows only the direct descendants and via which descendant each network node can be reached, not the exact full path to each node. In other words, for each node, there is a record of type {Destination[IPv6] *via* [IPv6]} in its routing table, since a DAO will travel upwards until reaching the sink. Each node will add into the routing table the information about the descendant via which it can contact the DAO originating node. When the DAO message reaches the sink, it indicates the originating node and the direct descendant of the sink it came through, but not all the node(s) it traveled through to reach the sink. Moreover, for both storing/non-storing modes, nodes can casually participate in the network at any given time since there is no central monitoring mechanism. Because of this, it is rather difficult to distinguish the traffic-originating node or even nodes (un)intentionally altering their behavior.

As the fundamental pillar of RRL, the DODAG needs to be updated and maintained frequently. A dedicated algorithm—the Trickle Timer—handles the frequency of DIO messages, upon which the graph's convergence time is based. The Trickle Timer must balance between preserving the node's power consumption and keeping the network information up-to-date and trustworthy. To achieve this trade-off, DIO message dispatching frequency varies from a few seconds up to 17.5 min. In detail, $I_{max} = I_{min} * 2^{I_{doubling}}$ (default RPL configuration specifies $I_{min} = 2^{12} = 4096\ ms$ and $I_{doubling} = 8$ which entails $I_{max} = 2^{12+8} = 17.5\ min$). The timer's duration is doubled each time it fires [6]. Any change in the DODAG, e.g., unreachable parent, DIO or DAO mismatch, or new parent selection, resets the Trickle Timer of the particular node to the initial default. Hence, DIO messages will be dispatched in a higher rate when the network is unstable, and in a lower rate otherwise, preserving energy and reducing network traffic.

According to the literature research conducted in 3, RPL is open to several attacks because it is based on the IP(v6) stack and because of the wireless media usage for transmitting the information. More specifically, those attacks are classified into *resource depletion attacks*, *network topology attacks* and *network traffic attacks*.

Intrusion Detection Systems (IDSs) are a suitable approach to encounter malicious activities [8, 12, 14]. IDSs refer to a set of methods designed toward: (i) *detecting an attack*, (ii) *identifying the attacker*, and (iii) *mitigating* the event. They actually aim at detecting several attacks concurrently and ideally can be extended to deal with attacks that are not originally included in their design goals. Compared to the standalone mechanisms, they usually require some degree of collaboration among the network's nodes [14].

Regarding the RPL security case, the design, development, and evaluation of an IDS should satisfy a set of requirements that reflect the solution's *width* and *depth*. The metrics of *robustness* and *extendability* are defined for quantitative evaluation, referring to the range over which the impact of an IDS can be spread (*width*), in respect to the number of attacks detected. Suppose an IDS is a single or a limited case(s) solution. In that case, it is characterized by low *robustness*, since it cannot protect the network against different types of attacks, i.e., an adversary can

compromise a subset of nodes, in the worst case a big part of the network, and affect the network's mission. Furthermore, given that new attacks and security issues emerge following the progress on the IoT research, IDSs should be developed as a set of software components to be quickly and on-the-fly modifiable to encounter attacks beyond their initial scope, i.e., to be *extendable*.

Finally, the metrics of *accuracy* and *mitigation time* for qualitative evaluation (*depth*) are defined. In fact, beyond detection itself, an IDS should exhibit a high accuracy rate regarding both the event and the adversary; this means that the system does not misinterpret abnormal events or nodes' behavior as attacks or attackers, respectively (i.e., it exhibits low false positives), while minimizing the cases that attacks or intruders are overtaken (i.e., it exhibits low false negatives). Ideally, once an attack/attacker has been detected successfully, a mitigation strategy should be employed to rapidly exclude the malicious nodes from the RPL network and restore the rest of nodes' communication.

The research field of IDSs in the IoT domain is generally vast [124]. Still, only a restricted subset of them is appropriate for Low-power and Lossy Networks (LLNs) [15, 71], i.e., they take into consideration limitations in respect to their lossy links, heterogeneous and resource-constraint devices. In fact, most of them have been proposed in the recent bibliography, i.e., from 2013 to 2020 [8, 14, 15]. An overview of these works makes clear that there is no one-for-all solution that succeeds in all three axes, i.e., to *detect* a number of attacks at once, to *identify* the intruder, and to *mitigate* the event, and at the same time, meet the aforementioned requirements of *robustness*, *extendability*, *high accuracy* and *rapid mitigation*.

With this, *ASSET*, a softwarized intrusion detection system that offers a holistic approach to shield an RPL-based IoT network against different types of attacks, is introduced. The SDN paradigm inspires the system, i.e., it transfers functionality from the constraint end-nodes to central premises, e.g., the controller, offloading both computational and communication overhead. At the same time, it follows a modular architecture that allows adaptations. More specifically, *ASSET* offers:

1. *A novel workflow* hosting well-known mechanisms for data analysis, e.g., the K-Means algorithm, able to efficiently collaborate in data exchange toward accomplishing the threefold mission of detecting an attack along with the attacker, as well as of rapidly mitigating the intrusion. The challenging point is synthesizing a framework of independent components that are not merely put one next to the other but work as an integrated whole. Moreover, *ASSET's* workflow gives promises for further enhancements and extensions regarding either detection or mitigation.
2. *A set of mechanisms* for: i) attacks' detection, ii) attackers' identification, and iii) malicious activities' mitigation. An overview of the recent bibliography shows that combining detection methods as well as placement strategies brings advantages related to the number of attacks that an IDS can encounter, the ability of the system to localize the adversary, the accuracy rate, and the resource overhead [15]. Exploiting such outcomes, *ASSET* employs the anomaly detection either (or both) on a node- or controller-level, as a matter of providing the alternatives of a lightweight and a computationally-intensive solution. In addition, it hosts mechanisms for specification-based detection that gain knowledge based on RPL-related data, such as the DODAG and the RPL control messages. Overall, it utilizes three anomaly de-

tection and four specification-based, contributing to both *width* and *depth* of attack detection. Judging by the evaluation results, *ASSET* remains tightly coupled with the RPL protocol, and it is a robust system detecting as many as 13 different types of RPL-related attacks with high accuracy and moderated cost due to the ability to take alternative decisions according to the network's conditions.

3. *An adaptable control & monitoring protocol* that enables centralized network supervision. In practice, the protocol offers: i) monitoring of RPL-related data, raw UDP packets, or ICMP statistics in an adaptable fashion, i.e., trading the amount of communicating information for control overhead in respect to the network's conditions; ii) configuring RPL parameters on-the-fly as a means of enforcing central decisions to the network nodes once a mitigation action should be taken; and iii) communicating node-level anomaly detection events that should trigger further investigation centrally, e.g., detailed monitoring by the controller. To achieve adaptability, three modes of the protocol's operation are defined, i.e., *slim-mode* that offers "baseline" monitoring at regular periods of network's operation, *essential-mode* that indicates a first level of surveillance due to detected anomalies more than two nodes, and *full-function-mode* that denotes the need of intensive surveillance due to detected anomalies that require detailed data from several nodes to be compared.

4.3 Proposed System

In this section, the design artifacts of *ASSET* platform are presented, including its high-level architecture and details of the control channel interface. Furthermore, the basic workflows for *attack(s) detection*, *intruder identification*, and *attack mitigation* are described, along with the relevant incorporated mechanisms.

ASSET can mitigate a large number of attacks with excellent accuracy since it exploits the softwarization paradigm in computer networks that allows: (i) centralized monitoring and control of the network; (ii) co-existence of multiple mechanisms while being extendable to support new algorithms; and (iii) consideration of both global and local view-points of the IoT network. For example, anomaly detection at the node (or a central) level may trigger other specification-based detection mechanisms. At a functional level, *ASSET* mainly consists of a network *controller* with attack detection, attacker identification and mitigation algorithms, a *control channel interface* with adaptable control overhead, and *node-level features* for anomaly detection, and network control and monitoring.

The *controller* implements centralized intrusion detection capabilities through a global view and control of the network operation. At the same time, it also hosts the majority of security mechanisms to offload the resource-demanding processes from the hardware-constraint devices to the centralized infrastructure. It is straightforward to extend its functionality with new mechanisms or cooperation with external tools and libraries, e.g., the machine-learning library Weka [125] was utilized. The *controller* collects information both passively and actively from different layers, e.g., ICMP messages from the network layer and data link quality metrics from the link-layer; currently utilized network-layer and application-layer data. Such a cross-layer approach helps the *controller* to maintain a detailed network view towards accurate decision-making. Apart from detection, the *controller*

is also capable of attacks' mitigation by mandating changes of RPL parameters in real-time, e.g., like in [21, 19]. In practice, it provides a front-end to the network administrator and supports a number of mechanisms to detect attacks and attackers and mitigate the threats.

The *controller* communicates with the nodes through the *Southbound Interface*, which utilizes a lightweight protocol to lookup or configure particular RPL parameters on-the-fly, monitor the network in an adaptable fashion, i.e., trading information accuracy for control overhead, as well as communicate anomaly detection events from the data communication plane to the application plane. Such information is derived by lightweight monitoring and anomaly detection within the devices to reduce communication overhead with the controller and enable fast detection at node-level, which acts complementary to the controller-level IDS mechanisms.

The detail of the IDS architecture and its primary interfaces are presented below.

4.3.1 Architecture and Interfaces

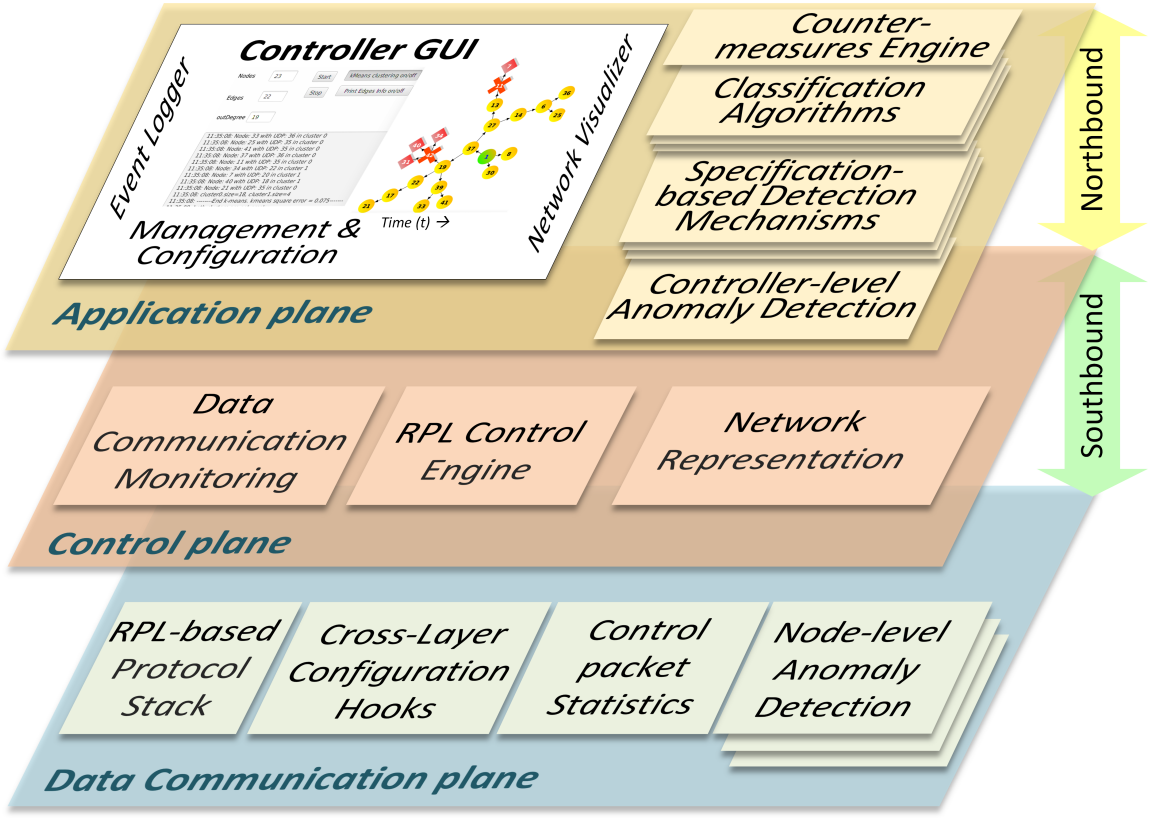


Figure 28: The architecture of *ASSET* IDS.

The *ASSET* IDS adopts a three-tier architecture, aligned to the SDN paradigm [126]. This is relevant to the three-layers approach of several other IoT proposals [12, 127, 128], i.e., usually termed as *Perception*, *Network* and *Application planes*. In Fig. 28, the *Data Communication*, *Control*. and *Application Planes* and their main components are depicted, which are detailed below.

The *Data Communication Plane* concerns the IoT infrastructure, including the *RPL-based protocol stack* of the corresponding nodes. Afterward comes the enabling *cross-layer configuration hooks* to the protocol stack [21, 19] allowing the *con-*

troller to read or apply cross-layer configuration settings, e.g., to enforce changes to RPL operation to mitigate attacks instantly. Furthermore, the nodes support *data packet statistics* being either processed locally, i.e., by manifesting *per-node anomaly detection* capabilities or communicated to the *controller*. The *Data Communication Plane* interacts with the *Control Plane* through the *Southbound Interface*, carrying either *data packet statistics* from the *nodes* to the *controller* or configuration actions towards the opposite direction. For example, a detected anomaly in the exchanged data of a particular node is being communicated to the *Control Plane* to signal further actions from the IDS.

The other two layers, i.e., the *Control* and *Application Planes*, reside at the *controller* and interact between each other through the *Northbound Interface*, which is REST-based. The *Control Plane* is responsible for the network control aspects, while the *Application Plane* for the data analysis and GUI features of the IDS.

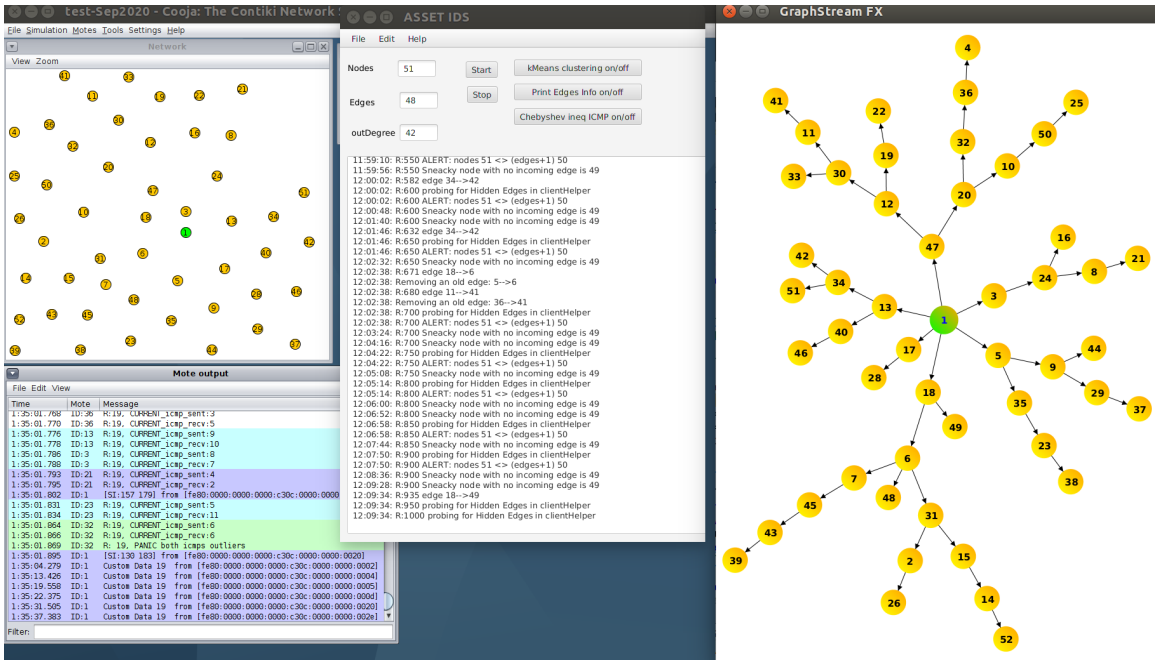


Figure 29: Controller in action: At the top left is the Cooja-based emulation of a 50-nodes RPL network. In the middle is the controller's continuous output, and to the right, the dynamic real-time representation of the network under surveillance.

The *Control Plane* is attached to the sink node, employing both passive and active *data communication monitoring* of the nodes, i.e., retrieving data communication statistics from the sink or the nodes, respectively. The *RPL control engine* is responsible for enforcing particular RPL configuration processes, as well as receiving node-level anomaly detection events from the nodes. The data communication statistics and the anomaly detection events are being communicated to the *Application Plane* for further actions through the *Northbound Interface*. Furthermore, the *Control Plane* maintains a real-time *network representation* based on the Graphstream Java library for modeling and analysis of dynamic graphs [129, 130]. Also, the spanning tree algorithm is applied to detect loops and actual nodes that do not appear in the depicted network graph.

The *Application Plane* provides the GUI and configuration aspects of the IDS, i.e., as shown in Fig. 29. It supports a real-time visualization of the IoT topology, which

also designates potential IoT nodes acting as attackers. Furthermore, it provides handles to the administrator for *management and configuration* aspects of the IoT network and the intrusion detection process. Finally, it is responsible for all data analysis processes of the *controller*, including *controller-level anomaly detection* algorithms, *attack-specific detection mechanisms*, *classification algorithms* for the attacker identification, as well as a *counter-measures engine*, being responsible to trigger attack mitigation processes as a result to the data analysis.

The proposed IDS has been implemented in Contiki OS [37] and Java, also utilizing the Weka [125], and Graphstream [130] libraries. The source code for the IDS is released as an open-source[2], under GPLv3.0.

Since the *Northbound Interface* is an internal interface of the *controller*, the main focus is on the *Southbound Interface*, which is essential for the performance of the IDS, especially towards reducing the involved control overhead. The details of the *Southbound Interface* follow.

4.3.2 The Southbound Interface

The *Southbound Interface* utilizes a lightweight application-level protocol that allows the *controller* to communicate with the nodes via the sink. The protocol maintains compatibility with the RPL standard while being flexible to incorporate new features. It supports either pulling of information, i.e., the *controller* retrieving monitoring information or configuration parameters from nodes, or pushing information, i.e., nodes notify the *controller* regarding their monitoring data periodically. The implemented protocol configuration hooks [6, 19, 5], based on the relevant interfaces implemented in the context of the WiSHFUL project (i.e., the UPIs), enable the *controller* to act as a centralized network control facility, especially for enforcing attack mitigation measures.

The *Southbound Interface* is responsible for the following aspects: (i) monitoring statistics of nodes' data exchanged, with different levels of accuracy and communication overhead, depending on the criticality of network conditions; (ii) mandating nodes to enforce changes in RPL protocol behavior to mitigate an attack, such as disabling Trickle Timer reset in cases of a DODAG version attack; and (iii) communicating node-level anomaly detection events—from the nodes to the *controller*—to trigger further actions, e.g., enable more detailed monitoring of nodes and controller-level anomaly detection.

The *Southbound Interface* of *ASSET* tunes the monitoring overhead and accuracy of attack detection trade-off, depending on the network conditions. For example, whenever a particular node detects an anomaly, the *controller* mandates additional information to be monitored, so an attack cannot be missed. In practical terms, the interface operates in three different modes, i.e., *slim-mode*, *essential-mode*, and *full-function-mode*, described as follows:

- In *slim-mode*, *ASSET* operates with the minimum number of monitoring messages, being essential to construct the complete graph of the network centrally. Either the *controller* may request the parent of a node, or the nodes can report all parent changes. This mode is in place in networks without indications of attack, i.e., detected through lightweight node-level anomaly detection.

[2] https://github.com/SWNRG/ASSET

- In *essential-mode*, the nodes transmit to the *controller*—besides the parent-change notifications—periodic ICMP statistics, which enable controller-level anomaly detection. The *essential-mode* is enabled when a node detects an attack through its node-level anomaly detection process, i.e., raising the criticality level.
- In *full-function-mode*, the nodes complement the parent-change and ICMP-related monitoring data with additional information, e.g., data dispatching period, the rank of node, and node's neighbors and their rank. This way, *ASSET* can detect—among others—Rank attacks and Sybil attacks with higher precision. The *ASSET* administrator can configure and set this mode to be enabled when at least more than two nodes detect an anomaly in their vicinity.

Table 7: Format and description of basic messages exchanged between the *controller* and the nodes.

	ID	**MESSAGE FORMAT**	**DESCRIPTION**
From Nodes	NP	[IPv6-NODE][IPv6-PARENT][int]	Communicates the parent of a node and its rank
	IS	[IPv6-NODE][int]	Communicates ICMP statistics
	UD	[IPv6-NODE][UDP Data][packets]	Communicates UDP data
	NR	[IPv6-NODE][int]	Communicates current rank of nodes
	NN	[IPv6-NODE][IPv6 node's neighbors][list]	Communicates available neighbors
	AD	[IPv6-NODE][boolean]	Anomaly detection notification
	VN	[IPv6-NODE][boolean]	Version attack notification
	RN	[IPv6-NODE][boolean]	Local repair attack notification
From Controller	SP	[IPv6-NODE][int]	Controller requests the parent of a node
	SN	[IPv6-NODE][list]	Controller solicits the neighbors of a node
	EI	[IPv6-NODE or multicast][boolean]	Enable/Disable ICMP statistics notifications
	ED	[IPv6-NODE or multicast][boolean]	Enable/Disable UDP data notifications
	NL	[IPv6-NODE or multicast][boolean]	Enable/Disable neighbors list notifications
	TT	[IPv6-NODE or multicast][boolean]	Enable/Disable Trickle Timer resetting
	BL	[IPv6-NODE][boolean]	Node is (not) blacklisted, i.e., Node can (not) be a parent
	LR	[IPv6-NODE or multicast][boolean]	Enable/Disable local repair feature
	GR	[IPv6-NODE or multicast][boolean]	Enable/Disable global repair feature

We now describe in detail the messages exchanged between the *controller* and the nodes. In Table 7, all messages are enlisted, and their design primitives, supported by the *Southbound Interface* and its corresponding network control and monitoring protocol.

In RPL, nodes collect information about their neighbors (i.e., nodes within the wireless radio coverage) and nominate a preferred parent within time instances specified by the Trickle Timer algorithm. This way, a network graph, i.e., the DODAG, is constructed in a distributed manner. Since this information is local, a notification feature was implemented in every node, triggered by any parent-change event. In such a case, the node transmits a message to the *controller* indicating the latest chosen parent, i.e., a [NP] message. Consequently, the *controller* is aware of the (current) parent of all nodes and can form the topology graph. Alternatively, the *controller* may proactively request the node's parent information, in case such information is missing, through a [SP] message. The *slim-mode* uses these two messages only.

Other messages from nodes to the *controller* include the [IS], [UD], [NR], and [NN], communicating ICMP statistics (e.g., total sent and received messages), UDP

data, node's current rank, and available neighbors respectively. Whenever a node detects an anomaly through its lightweight anomaly detection feature, e.g., an outlier in its ICMP statistics, it notifies the *controller* for this event with a [AD] message. Furthermore, the [VN] and [RN] messages inform the *controller* for a DODAG inconsistency or Local Repair attack detected by a node, respectively.

The *controller* uses designated messages to: (i) solicit missing node's parent or node's neighbors' information with [SP] and [SN] messages, respectively; (ii) enable or disable ICMP statistics, UDP data, and neighbor list notifications with [EI], [ED] and [NL] messages, respectively; and (iii) implement actions to mitigate attacks, including disabling Trickle Timer resetting with [TT], blacklist a node from becoming a parent with [BL], and disable Local and Global Repair features with [LR] and [GR] messages, respectively.

The *controller* also monitors the transmitted messages from the nodes for discrepancies. If the received messages are significantly different for one or more nodes (e.g., the number of data packets sent by one node are much less than those sent by its neighbors), this is a sign of a Blackhole/Grayhole attack. Moreover, based on the last received message, *ASSET* stamps the previous time each node was active. In case a node is "quiet" for a particular timeout period, it is first marked as "inactive" and eventually removed from the network graph maintained at the *controller's* side. The IDS will also identify attacks from the Clone-ID/Sybil family if two identical nodes co-exist in the network (e.g., same IPv6 address).

The *Southbound* interface gives the flexibility to the *ASSET* platform to balance the control overhead to the current network conditions, as well as implement a large number of intrusion detection features.

In the following subsections, there is an elaboration on the basic intrusion detection workflow of *ASSET* and its corresponding mechanisms for attack detection, attacker identification, and attack mitigation.

4.4 Intrusion Detection Workflow

ASSET is a fully-featured IDS that performs intrusion detection for a wide range of attacks, attacker identification even for multiple malicious nodes, as well as bespoke mitigation strategies proportional to the type of attack. Such features are implemented through its novel intrusion detection workflow, offloading processes traditionally handled by the nodes to a centralized *controller* for a better intrusion detection accuracy and resource efficiency. The workflow can be extended with new attacks straightforwardly since it is implemented in Java rather than on device firmware coding.

The IDS workflow operates on top of the IoT nodes, including the sink and the *ASSET* controller. When the network runs stably, in terms of ICMP and data traffic behavior, the *controller* collects information on the active topological structure only. Practically, the IDS operates in slim-mode, i.e., the nodes communicate any parent-changes to the *controller*, an action related to the minimum topological information. In parallel, the nodes perform anomaly detection based on their own ICMP measured statistics. In case they detect one or more outliers, they enable the *essential-mode* of the *Southbound Interface*, i.e., start communicating the ICMP statistics to the *controller*. The nodes also implement local RPL specification-based attack detection for exceptional cases like monitoring the number of recent local topology repairs and DODAG inconsistencies. In case they exceed particular

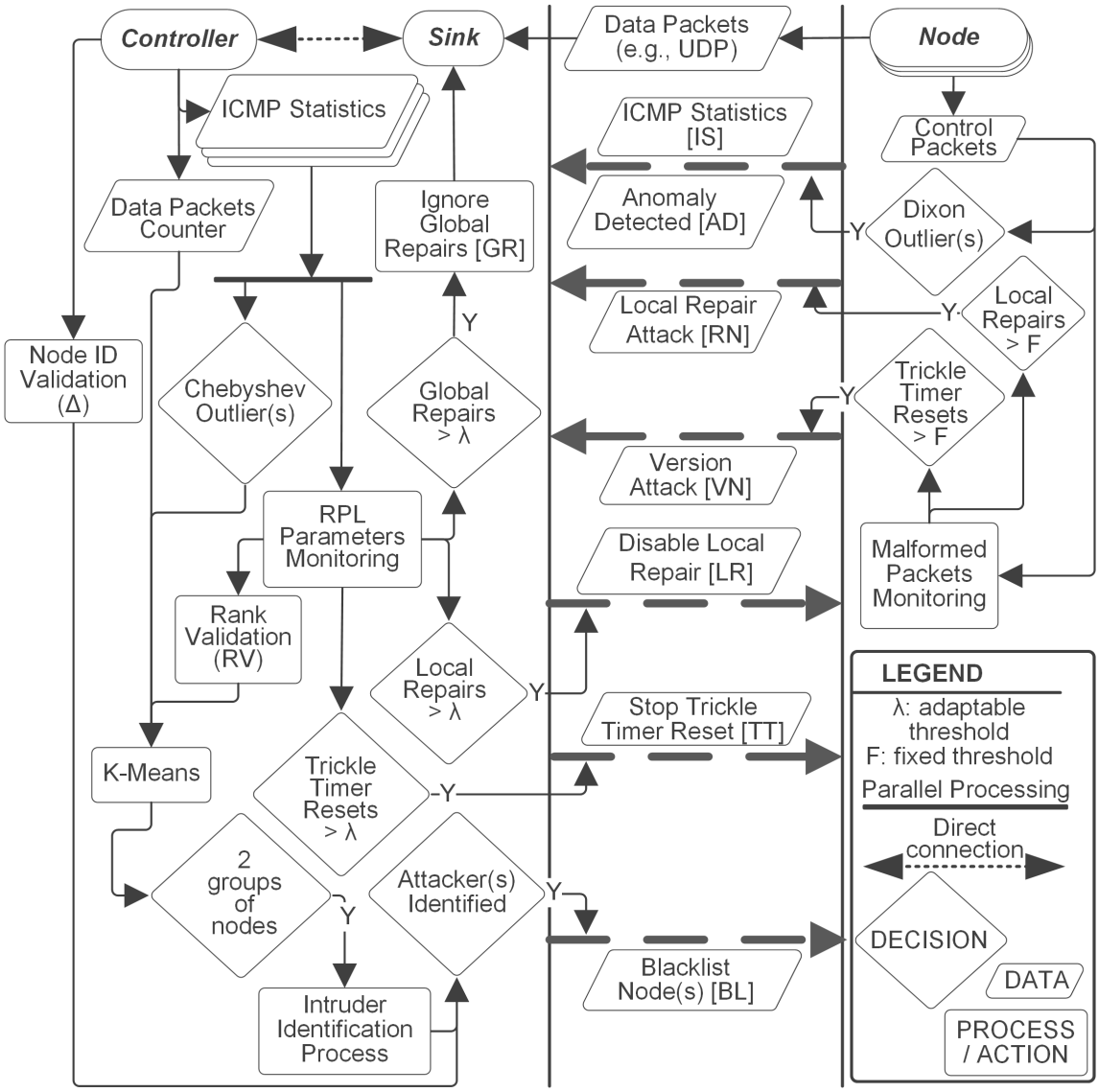


Figure 30: Flowchart of basic system functions, in a perpetual loop.

thresholds, they disable the Local Repair, and Trickle Timer reset features to overcome the Local Repair and DODAG Inconsistency and version attacks, respectively.

When one or more nodes detect an anomaly, i.e., the *essential-mode* is enabled, the attack detection process is being delegated to the *controller*, implementing anomaly detection based on the global network view. If one or more outliers are detected, several attack-specific detection mechanisms are being executed, including for DODAG Version or Inconsistency, and Global/Local repair attacks.

Since more sophisticated attacks require information beyond the ICMP statistics, the *controller* follows passively the communicated data statistics from the sink, which is straightforward because it is connected to the latter. For example, it performs anomaly detection on data statistics to detect Blackhole and Grayhole attacks. Furthermore, it may utilize the *full-function-mode* to request additional information, such as the node's rank and neighbors, to detect a Decreased Rank attack by comparing the rank declared by each node with the one reported by all neighboring nodes. The current version of the workflow also supports the detection of Flooding and Replay/Neighbor by detecting the ICMP anomalies created and

Clone-ID attacks by continuously comparing all nodes' IDs reported.

Depending on the type of attack detected, the workflow implements an attacker(s)' identification process and several attack-mitigation processes concerning the located malicious nodes, including node blacklisting, suspension of global / local topology repairs, or Trickle Timer resets. In specific cases, an administrator approval may be requested, e.g., for the Clone-ID attack.

We now move on to elaborate on the particular attack detection, attacker identification, and attack mitigation mechanisms implemented by the *ASSET* IDS workflow.

4.5 Attack Detection Mechanisms

ASSET implements both anomaly detection and RPL specification-based mechanisms to improve its capability to tackle a large number of attacks and the accuracy of the relevant detection processes. The detected anomaly at a node-level activates the less lightweight but more accurate controller-level anomaly detection process, utilizing several attack-specific detection mechanisms. For example, the IDS benefits from both advantages of distributed and centralized approaches to anomaly detection and the comprehensive detection capabilities of RPL specification-based strategies.

The following subsections detail both anomaly detection processes and the attack-specific detection mechanisms, supported by *ASSET*.

4.5.1 Anomaly Detection

ASSET supports anomaly detection features at both node- and *controller*-level. Regarding the node-level feature, nodes detect outliers in their measurements and trigger an "orange" alert whenever an anomaly in nodes' communication appears while producing light overhead in the regular operation of the IoT network. Anomaly detection at a node level is considered rapid and efficient [92, 90], because of the locality of detected attacks. Furthermore, relevant mechanisms should be lightweight, i.e., consider the resource-constraint nature of IoT devices.

ASSET employs for its node-level anomaly detection feature a low-complexity and the memory-efficient mechanism that detects irregularities, i.e,. Dixon's or Dixon-Q Test. The same method was successfully used for detecting malicious users in a cognitive radio networks setting, outperforming Grubb's and boxplot tests [131], with the limitation of considering one malicious user only. Since the Dixon-Q test runs in every node and communicates the possible outlier to the controller, *ASSET* can employ Dixon-Q to detect multiple concurrent intruders. Dixon-Q is also widely used in other scientific disciplines, for example, as a method for rejecting grossly deviant (outlying) values of data sets [132, 133].

More specifically, the test assumes a normal (Gaussian) distribution of data, a typical assumption of significance tests, and is based on the calculation of a Q-value defined as the ratio given by the distance of the value to be tested from its nearest neighbor, divided by the range of values. If it exceeds the tabulated critical Q-test value (i.e., called Q_{crit}) for a given Confidence Level (CL) and number of samples N, then this value can be rejected with a probability of erroneous rejection (type I error) that is a function of the selected confidence level. For example, probabilities p = 0.01, 0.05, and 0.10, correspond to CLs of 99, 95 and 90 percent, since $CL = (1-p) * 100$, named as confidence values q99, q95, q90, respectively.

Consequently, Dixon-Q test's sensitivity can be adjusted by altering the size N of data (i.e., *window-size*), along with the probability p of Type I error (or confidence level, CL). Dixon-Q test is lightweight and easy to implement for resource-constraint devices since it only needs a couple of subtractions and one division with every two newly arrived samples. Each time an outlier is detected, it is communicated to the *controller* through the *Southbound Interface* to trigger further intrusion detection actions, such as a controller-level anomaly detection process.

The controller-level anomaly detection represents the second line of defense of *ASSET* IDS. It is enabled whenever an anomaly in ICMP statistics is detected in the neighborhood of one or more nodes. It attempts to identify outliers in either ICMP statistics or relevant data-plane packet counters, matching the requirements for diverse types of attacks. The *controller* can implement more resource-consuming attack detection approaches, compared to Dixon-Q mechanism utilized in node-level anomaly detection, as well as have a global view of the network environment, however with more control overhead, i.e., the IDS switches to essential-mode.

At this point of the investigation, *ASSET* IDS employs Chebyshev's inequality [134] in its controller-level anomaly detection mechanism, acting as a more accurate but also complex example, compared to Dixon-Q. Applying more sophisticated approaches, including based on machine-learning or change-point analysis [135], is a subject of future work.

When the data distribution is unknown, Chebyshev's inequality theorem guarantees that at least $1 - \frac{1}{K^2}$ of data from a sample fall within K standard deviations from the mean. This can be the basis of an outlier detection method [134] by calculating relevant lower or upper outlier detection value (ODV) limits. Any data value outside these limits is considered to be an outlier.

To calculate the above ODV limits, there is a need to define a p_1 threshold, trimming a small percentage of extreme values at the beginning of the outlier detection process, so outliers do not bias the standard deviation calculation. Indicative p_1 values are 0.01, 0.05, or 0.10. Additionally, a second p_2 threshold represents the expected probability of an outlier appearance. The p_2 threshold is used to determine outliers, and is usually lower than p_1, taking values like 0.01, 0.001, or 0.0001. Both p_1 and p_2 control the outlier detection process's sensitivity and determine the k values for the outlier pre-filtering (first phase) and actual outlier detection (second phase) processes, respectively.

For the family of Blackhole/Greyhole Attacks, data packet reception is monitored by K-Means for discrepancies. K-Means, in summary, is: given n measurements of nodes to be clustered, a distance measure d to capture their dissimilarity and the number of clusters to be created (i.e., $k = 2$ in the current implementation), the algorithm initially selects k random points as the clusters' centers and assigns the rest of the $n - k$ points to the closest cluster center (according to d). Then, within each of these k clusters, the cluster representative (also known as centroid or mean) is computed. The process continues iteratively with these representatives as the new clusters' centers until convergence. Although this is an NP-hard problem, it is simplified by heuristic algorithms being able to converge to a local optimum [136]. The implementation of K-Means in Weka library [125] was utilized.

Furthermore, the controller-level anomaly detection triggers specification-based mechanisms for a more thorough investigation of the attack. This happens in the case of Decreased Rank (or Sinkhole), DODAG version, DODAG inconsistency,

Global and Local Repair(s), and Clone-ID (or Sybil) Attacks.

RPL specification-based features of *ASSET* are presented next.

4.5.2 Specification-based Detection

The softwarization approach of *ASSET* and its novel IDS workflow allow the co-existence of general-purpose anomaly detection mechanisms at both node and *controller* level, along with RPL specification-based detection algorithms. Here, the latter is described in detail and how they were implemented to realize the detection of the attacks supported at this point of investigation. We capitalize on the benefits of centralized detection while also preserving network overhead through the involvement of node-based detection mechanisms. For example, as a common strategy, there is an offloading of node-based mechanisms to the controller, i.e., to save processing cost and support more informed decisions while mitigating the involved control overhead. Keeping many attack-specific mechanisms in a single IDS workflow also allows the mitigation of multiple co-existing attacks, usually causing more severe problems in the IoT network. However, such an aspect deserves further investigation.

To highlight the *extendability* benefits of *ASSET*, basic building blocks are introduced that can form alternative RPL specification-based detection methods. Their brief description follows:

1. *Rank & ID validation:*
   The *ASSET controller* is able to monitor particular RPL subsystems or parameters for the whole topology through the *Southbound* interface. For example, as shown in Table 7, nodes may communicate their current rank and their parents' (advertised) rank to the controller by dispatching *NR* and *NP* messages, respectively. In case of inconsistencies between the communicated ranks, the *controller* can identify particular attacks, e.g., a Rank Attack. In practical terms, a dedicated mechanism named *Rank Validation* (RV) is adapted at the controller-level from the node-based algorithm introduced in [67]. According to this algorithm, if a node's rank plus the RPL stabilizing parameter *MinHopRankIncrease* [3] is lower than its parent's rank, then the latter is considered as an attacker. As a meta-step, all advertised ranks are monitored to be bigger than the sink's rank plus the *MinHopRankIncrease*. Furthermore, two nodes with the same ID can be detected by the *controller* straightforwardly, i.e., detect a Clone-ID attack.
   ASSET also monitors essential parameters at the node level, including the number of triggered local and global repairs, and Trickle Timer resets or faced DODAG inconsistencies. Whenever they exceed particular thresholds, the *controller* is notified for further attack detection actions.
2. *Adaptable and fixed thresholds:* At this point of the investigation, *ASSET* uses an adaptable λ threshold and a configurable fixed threshold (F) to monitor critical parameters at the node level, including the number of triggered local and global repairs and Trickle Timer resets; whenever they exceed particular thresholds, the *controller* is notified for further attack detection actions. The adaptable threshold λ is described afterward. Several attacks relate to fabricated control messages causing RPL performance issues. For example, the sink-node avoids routing loops and topology inconsistencies by increasing the DODAG version whenever a global topology repair occurs.

Intruders can inject continuously increasing DODAG versions into DIO messages they dispatch, causing the receiving nodes to reset their Trickle Timer and implement local topology repairs, consequently facing increased communication overhead. RPL reduces the effects of such attacks by limiting Trickle Timer resets based on a fixed threshold with a value of 20. Any malformed packets, i.e., with the 'R' flag IPv6 header option set, are being dropped by the receiving node upon reaching this threshold without triggering Trickle Timer resets.
Here, the adaptable $\lambda(r)$ threshold function introduced in [72] was utilized, which is an adaptive threshold function $\lambda(r)$ which is more effective than RPL's fixed threshold in terms of reacting to varying attack patterns. In practice, a fixed threshold node-level was used, while introducing a centralized variation of the above algorithm on a controller-level. The latter is defined as follows:

$$\lambda(r) = [\alpha + \beta \cdot e^{1-\gamma \cdot r}] \tag{1}$$

where,

$$r = \frac{\sum_{i=1}^{n} E_{pkts}^{i}}{\sum_{i=1}^{n} D_{pkts}^{i}}, \quad \alpha = 5, \quad \beta = \frac{15}{e}$$

n is the number of nodes communicating packets, E_{pkts} the number of received packets with 'R' flag set true, D_{pkts} the total number of packets received. The β is chosen to lead to a default $\lambda(r)$ value of 20 (i.e., as suggested by RPL RFC [3]) and α ensures that $\lambda(r)$ cannot be zero. The value of γ, according to the authors, should be $20 < \gamma < 25$, i.e., the value 22 was set here. Such centralized variation brings the advantage of having a λ value characterizing the whole topology, so a local attack incident leads to corresponding protection of all nodes in the network. As a result of this, adaptable threshold λ appears more conservative compared to the one introduced in [72], since the r value reduces with the topology size. However, it produces very good results in the particular experiments carried out. A possible improvement could be a normalization of the equation concerning the number of nodes.
Furthermore, *controller-configurable* fixed thresholds (F), beyond the fixed *lambda* already supported by RPL, include one for the number of nodes with detected anomalies at the same time, frequency of Trickle Timer resets, number of local repairs, etc.

4.6 Algorithms Incorporated

For *ASSET* to accomplish the above, it incorporates several technologies and adapted algorithms, as they are briefly presented below.

4.6.1 Rank Attack Detection Algorithm

The Rank Attack detection Algorithm 2 which is adapted from [67], is placed in the *controller* instead of each node as the original. The algorithm has a considerable accuracy rate, according to the authors [67].

Algorithm 2: Detecting Rank Attack at the *controller*.

```
Result: Detect node launching rank attack
Input  : Each node's current rank
Output: Boolean: Node has launched Rank Attack
begin
    for each (node!=sink) do
        if node.rank()+MinHopRankIncrease <node.parent.rank() then
            alarm "Rank Attack";
            return True;
    end
end
```

4.6.2 Kosaraju's Algorithm

The algorithm can find strongly connected components of a directed graph $G = (V,E)$ in linear time (i.e., $\Theta(V+E)$-time) [1]. In detail, a strongly connected component of a directed graph $G = (V,E)$ is the full set of vertices $C \subseteq V$ such that for every pair of vertices (u, v) in C, those vertices u and v are reachable from each other. The pseudocode of a Depth First Search (DFS) recursive such algorithm is depicted in Algorithm 3.

the Kosaraju's Algorithm can be used to identify multiple intruders in an IoT network graph, given that the following conditions stand:

If there is more than one intruder simultaneously in an IoT network, those intruders will disrupt the "normal" operation in several neighborhoods of the network. The nodes around it will alter their behavior for each intruder compared to other nodes not reachable (or uninterrupted by the intruder). Since the intruder operates under the limitations of physical media and protocol, it will only affect and disrupt nodes which are covered by the wireless media. Hence, all interrupted nodes along with the intruder will form a strongly connected sub-graph. Consequently, an algorithm identifying the strongly connected sub-graphs in a graph can clearly distinguish the different neighborhoods of nodes exhibiting the same "disrupted" (not typical) behavior identified earlier by algorithms like K-Means.

After the Kosaraju algorithm has identified one or more "attacked" neighborhoods or groups of nodes, the next step is to find the attacker node, which, according to previous findings, is part of this group of nodes. Not only the attacking node is within this group of nodes, but it is the root of this sub-network's graph. One has to think that if a node does not belong to a sub-graph where the root is the attacker, this node is not under attack since it regularly communicates with the (legitimate) root. If, on the other hand, the node belongs to a branch of the attacked neighborhood, no matter what its depth is, it will have the attacker as the root of its (sub-)network.

Hence, the next step is to identify the root of each such sub-graph. By definition, this happens by finding the mother vertex of the graph. The mother vertex of a (strongly connected) graph $G = (V,E)$, is a vertex v such that a path from v can reach all other vertices in G. It has to be pointed out that if a mother vertex exists, it was already identified as the last finished vertex in the DFS. To determine the mother vertex, the algorithm has to check if v is a mother vertex by doing DFS again, and hence, the complexity of the algorithm's complexity is $\Theta(V+E)$+$\Theta(V+E)$=$\Theta(V+E)$.

Algorithm 3: Kosaraju's Algorithm adapted from [1].

```
Result: Strongly Connected Graph(s)
Input  : Graph G = (V, E)
Output: Graph(s) G_i

/* A DFS recursive function starting from v */
Function DFSUtil(int v, boolean visited [ ]) :
    // Mark the current node as visited visited[v] = true;
    // Recur for all the vertices adjacent to this
     vertex while adj[v] do
        if (not visited[adj[v]]) then
            DFSUtil(adj[v],visited);
        end
    end
    return;
End Function

/* Returns transpose of this graph */
Function getTranspose() :
    for i = 0; i < 10; i = i+2 do
        for v = 0; V = V; v++; do
            // Recur for all the vertices adjacent to this
             vertex
            while adj[v] do
                transposed_graph.adj[adj[v]].add(v);
            end
        end
        return transposed_graph;
    end
End Function

Function fillOrder(int v, boolean visited[], Stack stack) :
    // Mark the current node as visited
    visited[v] = true;
    // Recur for all the vertices adjacent to this
     vertex
    while (adj[v]) do
        if (not visited[adj[v]]) then
            fillOrder(adj[v], visited, stack);
        end
    end
    /* All vertices reachable from v are
    processed by now, push v to Stack
    */
    stack.push(v);
    return;
End Function
```

Algorithm 4: Mother vertex finding algorithm.

Result: Graph's Mother Vertex as v, or -1
Input : adj[int v]
Output: v

```
// Returns a mother vertex if exists.  Otherwise
 returns -1
Function findMother():
    // DFS will use visited[].  All vertices are
     initialized as not visited
    boolean visited(V, false);
    // To store last finished vertex (or mother vertex)
    int v = 0;
    // Do a DFS traversal and find the finishing vertex
    for (i = 0; i < V; i++) do
        if (not visited[i]) then
            DFSUtil(i, visited); v = i;
        end
    end
    /* Reset all values in visited[] as false.  Do DFS
     to check if all vertices are reachable from v.  If
     they are, v is a mother vertex */
    fill(visited.begin(), visited.end(), false);
    DFSUtil(v, visited);
    for i = 0; i < V; i++; do
        if (not visited[i]) then
            return -1;
        end
    end
    return v;
End Function
```

The algorithm's pseudocode is depicted in Algorithm 4. Notice that the function *DFSUtil* is derived from Algorithm 3.

Right below, a relevant attacker identification mechanism based on the above is presented.

4.7 Attacker Identification

Several attacks require identifying the intruder(s), before their mitigation, e.g., blacklisting a node causing flooding. In specific cases, intruder detection may be straightforward. For example, the controller can detect a Clone-ID attack whenever a duplicate id is received from two nodes. Then, the node that appeared second is considered to be an attacker. A novel intruder identification process is proposed for all other cases that can handle multiple co-existing attacks with high accuracy. Example usage of the *ASSET* platform and its GUI locating two intruders (marked with red X's), as well as the affected nodes (marked as red diamonds) is shown in Fig. 31.

In Fig. 32, the corresponding workflow for the proposed attacker identification

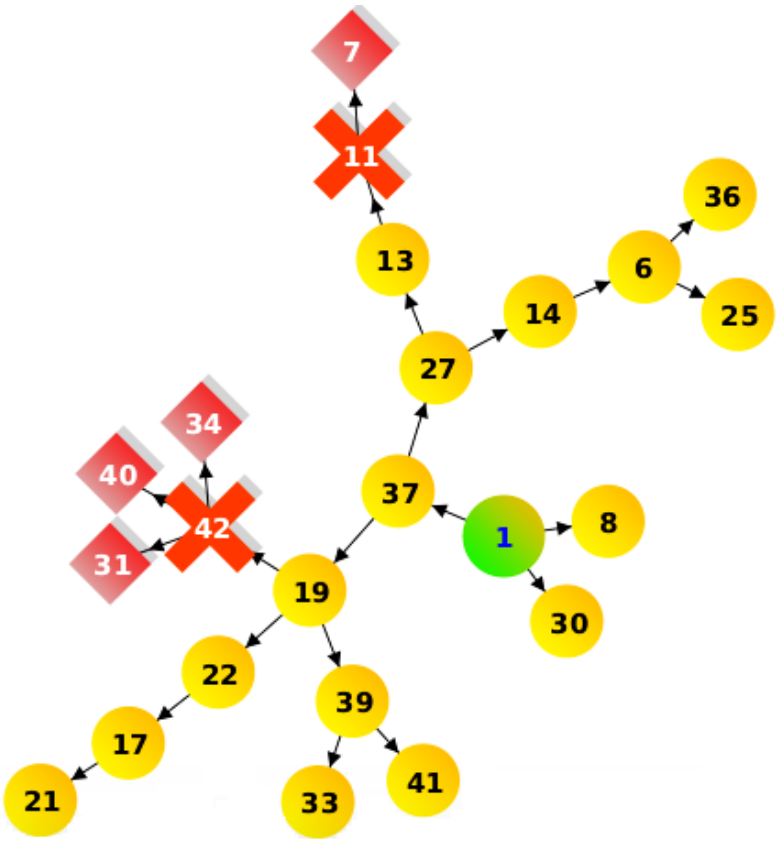


Figure 31: *ASSET* identifies two concurrent intruders.

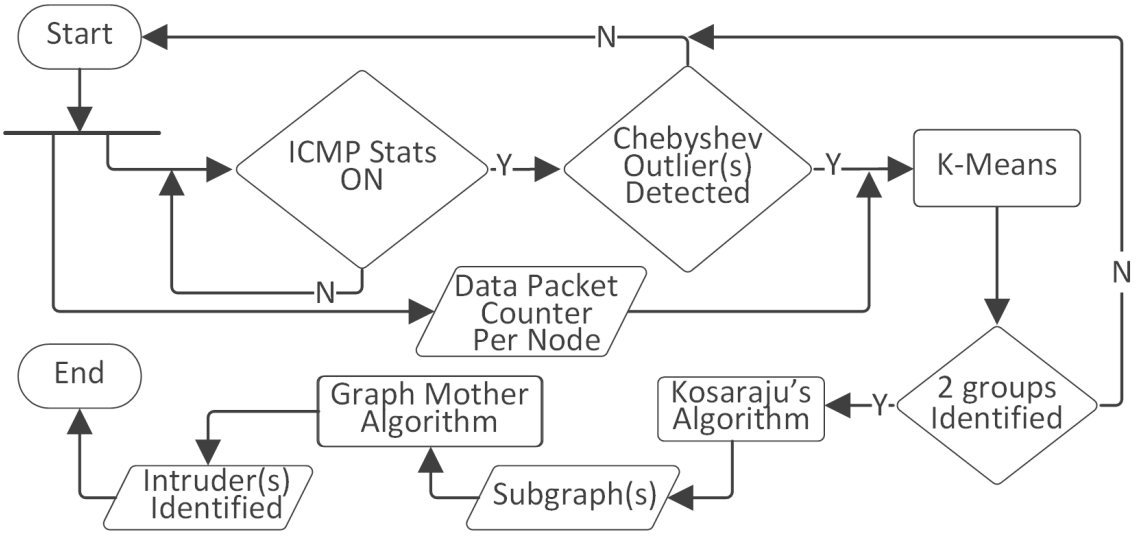


Figure 32: *ASSET's* intruder identification process.

process is depicted. In particular, such a process is being triggered by detecting an anomaly at the controller-level, i.e., by Chebyshev's inequality approach. This is based on information related to the implemented monitoring mode, e.g., ICMP statistics in the case of *essential-mode* or data packet counters and discrepancies in the case of *full-function mode*). *ASSET* implements its attacker identification process in three different phases, as depicted in Fig. 32:

- *Network clustering:* The first phase implements network clustering into two

groups, i.e., the healthy group and one affected by the attack, based again on K-Means algorithm [137]. As soon as K-Means divides the network nodes into two groups with high confidence, the smallest group will be considered next.

- *Subgraph(s) division:* The division of the affected nodes' group into one or more connected subgraph(s) representing multiple co-existing attacks, i.e., defined as a clique. Here, Kosaraju's algorithm is applied (Algorithm 3), accredited to Kosaraju and Sharir [138]. The algorithm locates strongly connected components a directed graph $G = (V, E)$ in linear time (i.e., $\Theta(V + E)$ time) [1]. In particular, the Depth First Search (DFS) recursive algorithm from [1] is utilized. The main assumption is the following. In case of multiple intruders, the network faces several neighborhoods with disrupted regular operation. Hence, all affected nodes along with the equivalent intruders form strongly connected sub-graphs.
- *Root nodes identification:* The next step is to identify the root of each such sub-graph, i.e., representing the attackers. The roots are defined as mother-vertices and located through applying the mother-vertex algorithm. The mother-vertex of a (strongly connected) graph $G = (V, E)$ is a vertex v such that a path from v can reach all other vertices in G. The algorithm has to check if v is a mother-vertex by executing DFS one more time to identify the mother-vertex. Consequently, the complexity of the algorithm is $\Theta(V + E)$+$\Theta(V + E)$=$\Theta(V + E)$.

As soon as one or more intruders are identified, *ASSET* implements a mitigation method that corresponds to the considered type of attack. For example, a blacklisting process may be initiated, blocking the attacker(s) from being part of the RPL DODAG. In the following subsection, the mitigation features supported by *ASSET* are discussed.

4.8 Attack Mitigation

The final step of *ASSET* intrusion detection workflow concerns the attack mitigation. The selection of the appropriate mitigation method to enforce depends on the detection algorithm that precedes, i.e., corresponding to particular types of attacks. In this context, *ASSET* supports the following mitigation methods:

- *Blacklist Intruder:* A large number of attacks can be mitigated by excluding the intruder(s) from being considered as a parent by all nodes in the network. In order to preserve full compatibility with the RPL standard, a node blacklisting mechanism was implemented (described in Algorithm 5) as an extension of the default MRHOF OF [36], briefly described in Section 3.3. In detail, each node maintains a local blacklist array, which is updated by [BL] messages received by the controller. The nodes exclude blacklisted nodes from being selected as parents, even if they appear as more suitable options, as shown in Algorithm 5. In practical terms, the *controller* communicates a [BL] message to all nodes, specifying the node to be blacklisted, which is no more an option as a parent-node.
- *Ignore Global Repairs and Stop Local Repairs:* Since global and local repair processes may consume significant resources, as a result of an attack (e.g., DODAG inconsistency attack), the *ASSET* IDS may decide to suspend one or both of them, the former at the sink, and the latter at the concerning nodes.

Algorithm 5: Parent selection considering blacklisted nodes.

```
Result: Selects the best parent that is safe
Input  : Candidate parents p1 and p2
Output: Safe parent
1  begin
2      if p1 in blacklist[] then
3          if p2 in blacklist[] then
4              return null;
5          else
6              return p2;
7          end
8      else
9          if p2 in blacklist[] then
10             return p1;
11         else
12             // Choose parent process of standard RPL-MRHOF
                   objective function
13             return p1.ETX< p2.ETX? p1 : p2;
14         end
15     end
16 end
```

This also comes together with the suspension of exchanging corresponding DIO packets. The Ignore Global Repair mitigation method is being triggered through the [GR] message transmitted from the *controller* to the sink. The Stop Local Repair mitigation method is being triggered either locally or through the [LR] message sent from the *controller* to the corresponding node that should suspend its local repairs.

- *Stop Trickle Timer Resets:* In an equivalent manner, the Trickle Timer resets consume significant overhead since RPL control messages are being exchanged more frequently. Suppose the Stop Trickle Timer Resets mitigation method is triggered locally. In that case, the node ignores all regular RPL operations that start a Trickle Timer Reset for a particular period. Alternatively, if the *controller* detects an attack that requires the suspension of the Trickle Timer Resets in specific nodes, it transmits a [TT] message to all of them.

4.9 The proposed IDS in summary

In Table 8, there is a summarization of how all the above IDS features are associated with all handled attacks. More specifically, enlisted for all attacks: (i) the detection method applied (i.e., whether it is anomaly detection or specification based) as well as the specific detection features utilized; (ii) the placement of the detection method (i.e., at the controller, the nodes or hybrid); (iii) the required data input for the particular detection method; (iv) whether the identification of an attacker is needed for its mitigation; (v) the mitigation method which is appropriate to this type of attack; and (vi) the source of the corresponding detection algorithms, including the relevant citations.

Table 8: Attacks and designated actions supported by the IDS.

ATTACK /ANOMALY	DM	DP	DI	MS	IA	MM
SINK /CONTROLLER						
Chebyshev's Inequality	Di/Ch*	C	I,U	A[134]	-	-
Blackhole /Greyhole	K	C	U	O	Y	B
Decreased Rank /Sinkhole	Ch,RV	C	I,R	O,Co[67, 139]	Y	B
DODAG Version	λ(C,n)	C	T,R	A[72]	N	G,L,P
DODAG Inconsistency	λ(C,n)	H	T,R	A[72]	N	G,L,P
Global Repair	λ(C)	C	R	A[72]	N	G
Local Repair	λ(C),F(n)	H	R	A[72]	N	L,P
Flooding	Ch	C	I,U,R	A[134]	N	**
Replay / Neighbor	Ch	C	I,R	O,A[134]	N	**
Clone-ID / Sybil	Δ	C	I,R	O	Y	B
NODE						
Dixon-Q Test	-	n	I	A[132]	-	-
Local Repair	F(n)	H	T	O	N	L,P
DODAG Inconsistency	λ(n)	H	T,R	A[72]	N	L,P

Legend

(C)ontroller, (n)ode.
DM: Detection Method: Anomaly Detection [(Di)xon, (Ch)ebyshev, (K)-Means], Specification Based [λ(): Adaptable Threshold, F(): Fixed Threshold, RV: Rank Validation, Δ: Node ID Validation].
DP: Detection Method's Placement: (C)ontroller, (n)ode, (H)ybrid.
DI: Data Input for Detection Method: (I)CMP Statistics, (U)DP Statistics, (T)rickle Timer Resets Counter, (R)PL Control Messages.
MS: Detection Method's Source: (O)riginal, (Co)mposition [ref1..refX], (A)dapted from [ref].
IA: Identification of Attacker needed: Y/N.
MM: Mitigation Method: (B)lacklist Node, I(G)nore Global Repairs, Stop (L)ocal Repairs, Sto(P) Trickle Timer Resets.
* Chebyshev's algorithm is initiated upon Dixon-Q test outliers detection per node.
** Any subsequent attack deriving from the replayed packets will be addressed by the specific mechanism.

The table highlights that *ASSET* handles diverse types of attacks through different combinations among the supported IDS features. It is noted that the anomaly detection capabilities can even detect unknown attacks causing communication disruptions. Furthermore, new specification-based building blocks can be integrated to increase its supported number of attacks further. Although the IDS could be implemented with different relevant algorithms performing even better, the current selection performed decently in the experimentation exercise below and enough to validate the main *ASSET* novelties.

In the next section, the experimentation analysis is presented.

4.10 Evaluation Results

Here, *ASSET* is evaluated in line with *robustness* and *extendability* that reflect the *width* of the proposed solution, as well as *accuracy* and *mitigation-time* that express its *depth*. More specifically, there is a discussion on the evaluation methodology and, then, the following: (i) proof-of-concept simulation results that demonstrate two attack incidents, along with *ASSET's* response in terms of detection and mitigation, as well as the impact of the attack on network control packets' overhead; and (ii) the *ASSET's* robustness with an evaluation of its operation under a large number of attacks triggering all discussed mechanisms.

4.10.1 Evaluation Methodology

We utilize the Cooja emulator in Contiki OS [37] for the *ASSET's* performance evaluation. The simulations were carried out considering one sink node, a set of legitimate nodes, and one attacker node. Although *ASSET* can potentially mitigate attacks caused by multiple malicious nodes, the relevant experimentation is considered as future work. The network setup parameters are described in detail in Table 9.

Table 9: Network setup parameters.

Parameter	**Value**	**Notes**
Network Layer	RPL	Storing mode
MAC Layer	802.15.4	
Implementation	Contiki 3.0 - Cooja	
Sink Node(s)	1	Serial Port Connection
Mote Type	Zolertia Z1	
Nodes Placement	Random	
Number of nodes	25 or 50	
Area	800 m $\times$ 800 m	
Simulated Time	3 hr	10,800,00 ms
Data (UDP) Transmission Period (P)	5 min	Unless otherwise stated
ICMP Probing Frequency	5 min	Need to avoid zero probings
Packet Size	70 B	Average size
TX Range	50 m	
Interference Range	50 m	
TX/RX Success Ratio	100%	
Trickle Timer Duration	4 ms-17.5 min	Default values of Contiki RPL

We assume that the attacker runs the same firmware as the legitimate nodes and responds promptly to the controller's solicitation messages, e.g., it would be rather trivial for an IDS with centralized components to detect and, consequently, blacklist as possible intruder a node not responding to such messages. Once being blacklisted, the intruder cannot be chosen as a parent-node, and hence, it cannot successfully launch most of the RPL attacks described in Chapter 3. In practice, the attacker node(s) are considered to run multiple modified Contiki OS versions[3] (also available under GPLv3.0) in order to execute one or more attacks in conjunction.

Right afterward, proof-of-concept results are presented, demonstrating *ASSET's* operation under various attacks.

[3]https://github.com/SWNRG/contiki-malicious

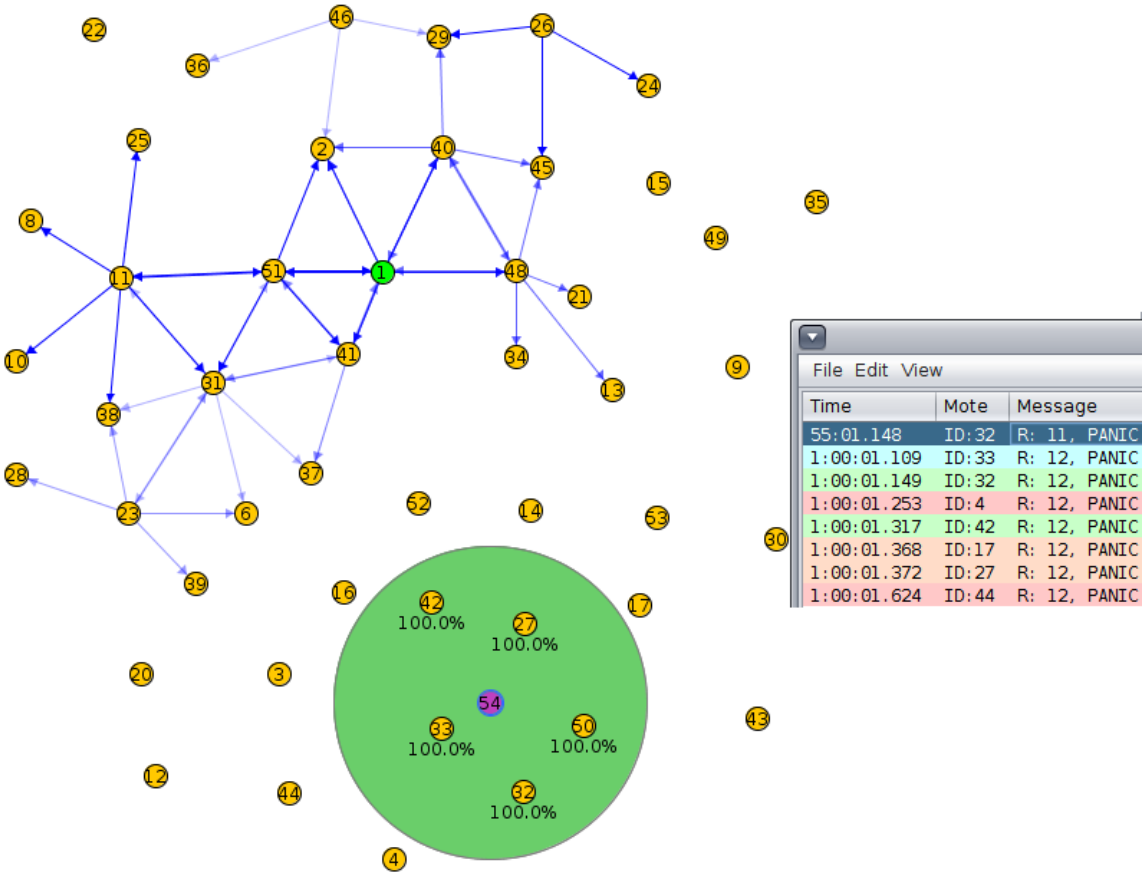


Figure 33: An RPL-based network with 50 nodes under rank attack.

4.10.2 Proof-of-concept Results

The first proof-of-concept simulation is associated with anomaly detection mechanisms of *ASSET*. As illustrated in Fig. 33, there is a network with 50 nodes (marked with yellow) randomly placed around the sink-node (the green one), while an intruder, indicated by the purple circle (the node with the $ID = 54$), compromises the network. The intruder unleashes a Decreased Rank attack by advertising a lower rank value than all other legitimate nodes in its wireless coverage (i.e., the green range). As a result, most of the nodes within range, i.e., nodes with ID $27, 32, 33, 42$ and some others around it, i.e., nodes with ID $4, 17, 44$, increase the number of ICMP packets exchanged, in their effort to recalculate paths to the sink.

The Dixon-Q test mechanism, which takes place in every node, detects this anomaly in the number of ICMP messages sent and received, as shown by the *PANIC* entries in the log file illustrated in the right-hand window in Fig. 33. In the simulation used, the Dixon-Q window-size was set to 7. Table 10 shows for each of the above nodes, the latest of the seven values, regarding both the incoming (RECV) and outgoing (SEND) ICMP packets is an outlier. These values at t_0 trigger the [AD] messages in seven nodes that notify the *controller* of the detected anomaly (we distinguish the nodes within the attacker's range with gray background in Table 10). Indeed, since the number of nodes sending a [AD] message exceeds the threshold of three, *ASSET* activates controller-level anomaly detection by Chebyshev's inequality mechanism for further investigation of the attack instance, i.e.,

Table 10: Node-level anomaly detection: Dixon-Q test using $window - size = 7$.

ICMP	NODE	t_6	t_5	t_4	t_3	t_2	t_1	t_0
SEND	4	4	4	4	5	4	4	18
	17	5	2	5	3	3	4	15
	27	5	3	6	4	4	5	19
	32	4	4	4	3	6	4	19
	33	7	4	6	5	7	7	17
	42	8	7	6	6	9	8	13
	44	3	5	3	3	4	5	8
RECV	4	3	4	3	1	5	4	39
	17	12	5	4	5	5	4	42
	27	10	6	5	4	4	6	82
	32	9	4	2	3	3	3	64
	33	11	6	5	5	7	6	91
	42	6	6	5	5	9	8	58
	44	4	3	3	7	3	3	20

attacker's detection and mitigation.

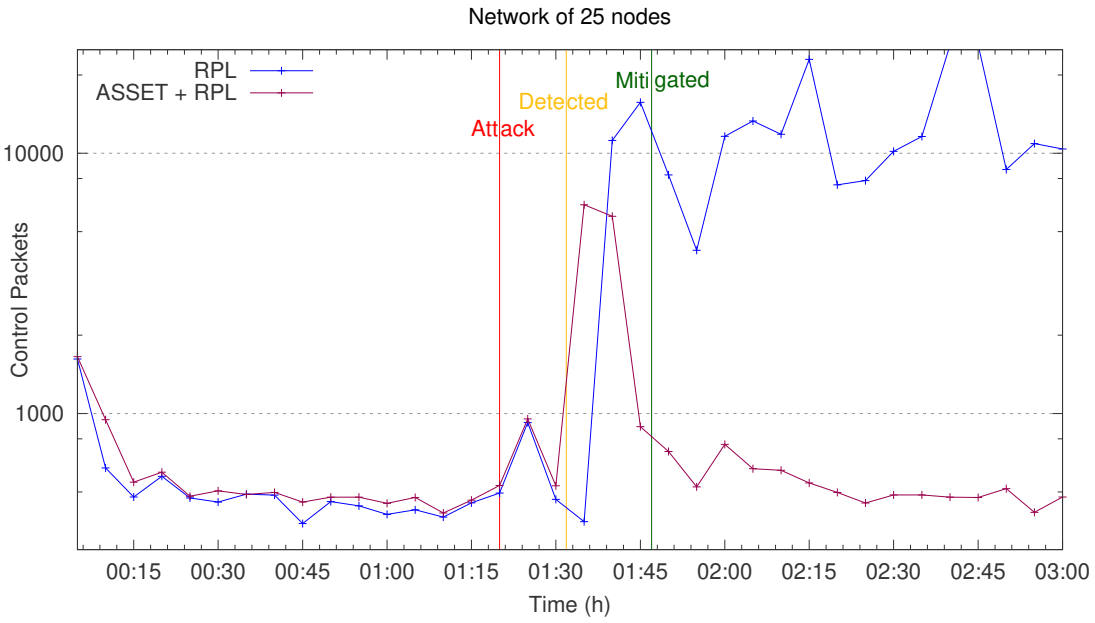


Figure 34: Control overhead over time: standard RPL operation versus RPL with the *ASSET* functionality in case of a combined Decreased Rank and Blackhole attack.

This holistic approach provided by *ASSET* is illustrated in Fig. 34 which is the outcome of the second proof-of-concept simulation. In practice, the simulation lasted three hours (x-axis) a multi-hop network with one sink and 25 nodes randomly placed around it, considering a combination of attacks, i.e., Decreased Rank and Blackhole attack, and the network's control overhead was observed, to validate the initial intuition regarding the impact of attacks over it. Fig. 34 shows that attacks are launched at 01:20 hour (vertical red line), detected at 01:32 hour (vertical yellow line), and mitigated at 01:47 hour (vertical green line).

More precisely, a common combination of attacks was chosen. The intruder discards data packets, e.g., UDP, once it successfully deceives several nodes that choose it as a routing node for their packets. Fig. 34 does imprint the impact of the Decreased Rank attack, which precedes the Blackhole one. Once the attack has taken place, Dixon-Q test detects outliers in control packets on six nodes at 01:25 hour and three more nodes at 01:30. These nodes notify the *controller* with [AD] messages, which process activates Chebyshev's inequality mechanism and proceeds with a more fine-grained detection. For this purpose, apart from a [NP] message, nodes need to send further control information to the sink for their latest chosen parent-node; this extra input includes ICMP statistics ([IS] messages), UDP data ([UD] messages), node's current rank ([NR] messages), and available neighbors ([NN] messages), assisting the *controller* in identifying the intruder. Once the intruder is identified, at 01:32 of the simulation, the *controller* dispatches a [BL] message to all nodes as a mitigation action. Fig. 34 provides evidence that, at 01:47 hour, the network graph is concise again, i.e., network nodes have selected legitimate parents after excluding the attacker as a candidate one.

From Fig. 34, the following conclusions can be drawn:

- The slim-mode operation of *ASSET* does not overload the network. In the period from the beginning of the simulation until the attacks (vertical red line), *ASSET* operates with the minimum number of monitoring messages, i.e., [NP] messages from nodes to report parents' changes and/or [SP] messages from the *controller* to the nodes, requesting missing information regarding their parents. It is clear that the purple curve, corresponding to the RPL network with the IDS functionality, is only slightly higher, i.e., 6.28 percent on average in the simulation, compared to the blue line, representing the standard RPL operation.
- The essential and full-mode operations of *ASSET* succeed in attacker's identification and mitigation, at the cost of increased control overhead. However, this overhead remains lower compared to that when the RPL protocol is left unshielded, which is as much as 49.87 percent on average. Indeed, within the time frame between the red and green verticals, node and controller-level anomaly detection are taking place, additional information is sent to the *controller* (e.g., [IS], [UD], [NR], and [NN] messages). At the same time, the *controller* also activates the three steps described in Section 4.7 to identify the attacker. However, despite these demanding processes, *ASSET* controls network topology disruptions and updates moderating Local and Global Repair ([LR] and [GR] messages) and, thus, holding the peak in purple curve lower.
- Mitigating the attack brings as much as 95.96 percent benefit to the network in terms of control overhead. In the period from the attacks' mitigation (vertical green line) until the end of the simulation, *ASSET* manages to establish a new DODAG consisted of legitimate nodes while allowing the network to continue its mission, i.e., data gathering.

In another similar series of experiments, the power consumption of the nodes under the operation of *ASSET* is measured.

In Fig. 36 we show the results of a *DODAG Inconsistency* attack against a network of 25 randomly placed nodes. The attack happens again at 01:20 hour, where both the power consumption (y1 axis) and control overhead (y2 axis) do

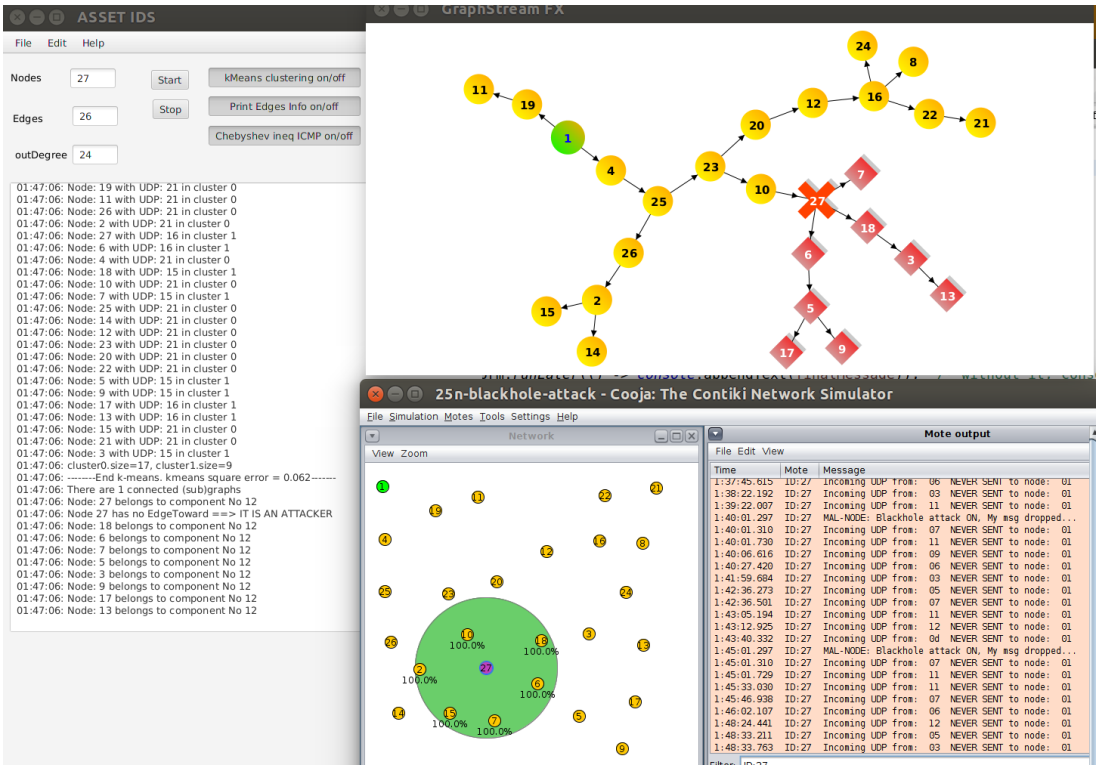


Figure 35: An RPL-based network with 25 nodes under blackhole attack.

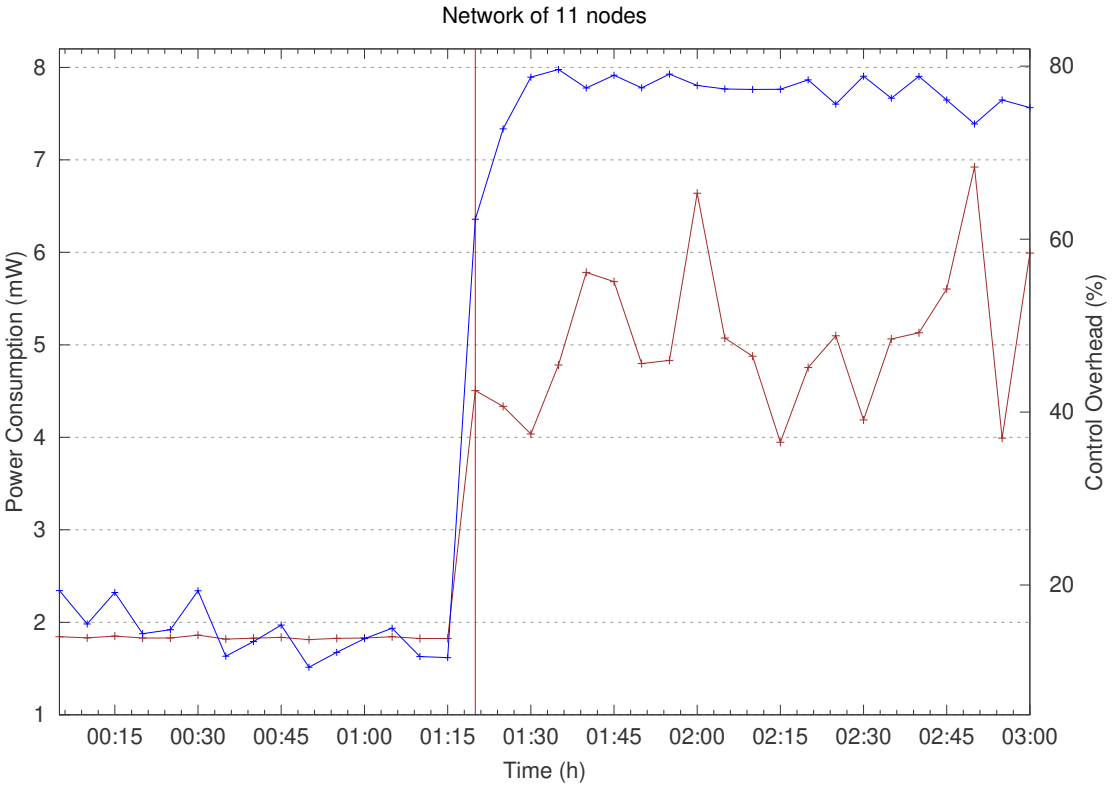


Figure 36: Power consumption of nodes and control overhead over time in case of a *DODAG Inconsistency* attack.

significantly increase. Especially this experiment, could be directly conducted in a testbed and compare the power consumption outcomes.

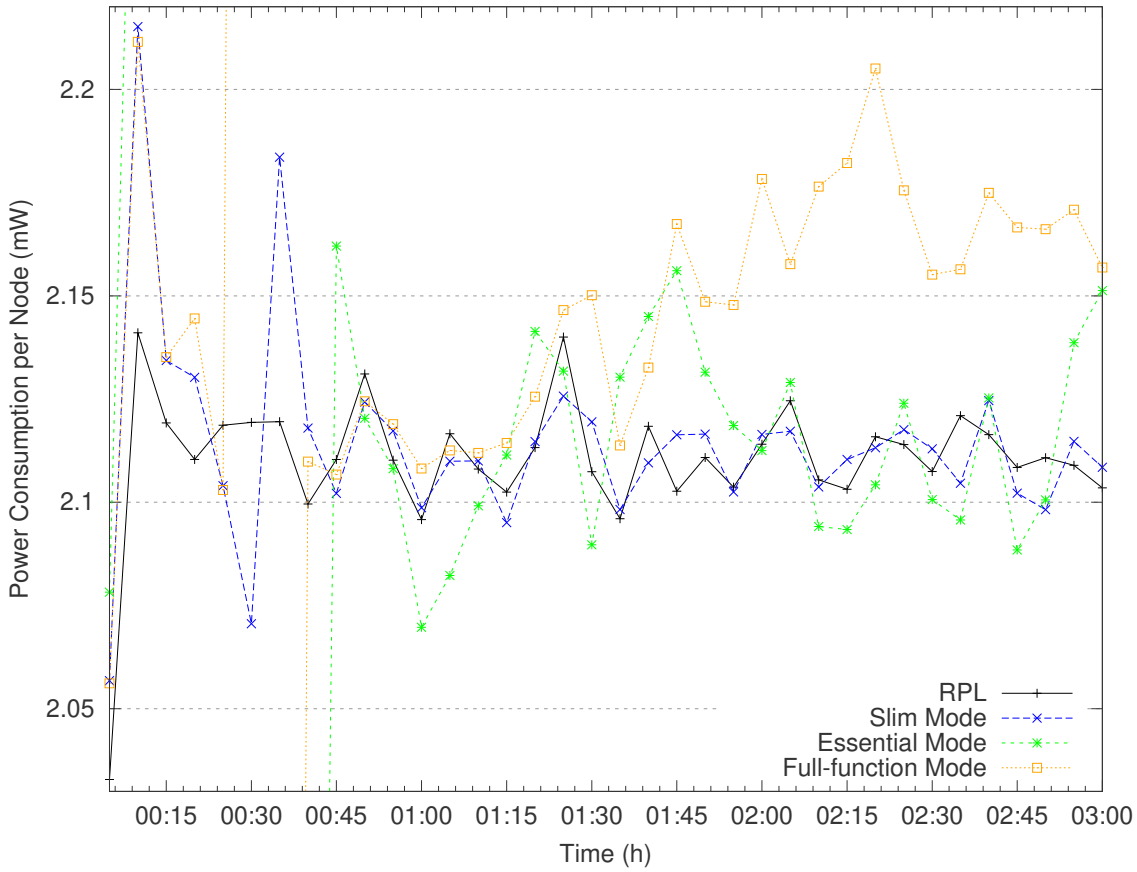


Figure 37: Average power consumption of nodes under *ASSET's* different modes of operation.

Regarding power consumption under the different modes of *ASSET*, we conducted the same experiment under four different modes of operation, i.e., standard RPL compared with the three operation modes (i.e., slim-, essential-, full-function-modes). The results are presented in Figure 37, where after the initially anticipated power "spikes" until the network settles down, the power consumption is minimal, with only full-function mode consuming slightly more energy. In total, compared with RPL, the slim-mode consumes 0.18 percent more power per node, the essential mode consumes 0.71 percent, while the full-function mode consumes 1.54 percent more energy. Compared to other similar solutions, SVELTE [92] has a 30 percent overhead compared to RPL.

The last proof-of-concept outcome elaborates on the attacker's identification mechanism. In a three-hour run, there is another random, multi-hop topology, where 25 nodes (the yellow ones) are under Blackhole attack by the node with ID=27 (marked with purple) while they route their data packets to the sink (green node). Network setup is illustrated on the bottom part of Fig. 35. The intruder is placed within the direct reach of six nodes (those with ID $2, 6, 7, 10, 15, 18$) and presents a legitimate behavior until 01:20 hour. Then, the intruder starts the attack by dropping all received data packets in their routing to the sink (including their own ones to make the scenario more challenging).

Attack detection triggers the K-Means algorithm that executes at the controller to identify the malicious node. The algorithm accepts as an input the number of UDP packets arrived at the sink from each node and the parameter $cl = 2$ that expresses the number of groups (clusters) to be created. The rationale is that in a network with scheduled UDPs and a pre-defined dispatching period, the impact of a Blackhole attack is to differentiate affected by non-affected nodes in terms of the UDP packet number arrived at the sink. Indeed, depicted in the left part of Fig. 35, K-Means has successfully divided the network into two distinct groups, i.e., clusters 0 and 1. These two clusters are depicted in the up-right part of Fig. 35, i.e., cluster 0 contains the yellow nodes along with the sink (non-affected as indicated by the high number of UDP packets), while cluster 1 consists of the affected nodes shown in red (due to the low number of UDP packets).

Looking carefully at the affected sub-graph, it is interesting that only nodes $6, 7$ and 18 within the intruder's coverage are affected by the attack, while the other three ones, i.e., $2, 10$ and 15, are not affected, because they do not select the intruder as a parent (indeed, the parent of nodes $2, 15$ is node 26, while the parent of node 10 is node 23). At the same time, nodes $3, 13$ and $5, 9, 17$ select as a parent the affected nodes 18 and 6, respectively, and consequently are also influenced by the Blackhole attack, although they are not within intruder's coverage.

At this step, it is crucial to distinguish among cluster members to identify the malicious one. Thus, once the K-means has been completed, it feeds the Kosaraju's algorithm with the red sub-graph. Kosaraju defines that there is one sub-graph (in case of multiple attacks, there would be more) and passes the graph to the mother node algorithm. The algorithm recognizes node 27 as the "root" of this sub-graph, identifying this ID as the malicious node. In the simulation used, the attack begins at 01:20, and the system identifies the attacker at 01:47. Right afterward, the *controller* blacklists this node in order not to be selected as a parent node.

In this scenario, leaving unmitigated such an attack reduces the PDR by 82 percent. The proposed system improves data packet delivery offering a 94 percent ratio. Next, there is a discussion on the robustness of *ASSET*.

4.10.3 Robustness Results

The diversity of RPL-related attacks, described in Chapter 3, entails the necessity for an IDS to be able to detect a variety of attacks. If an IDS does not protect the network against different attacks, the adversary can compromise one or more nodes and affect the network's operation. Table 6 shows that the most robust systems in the literature can detect up to 8 attacks. The results regarding ASSET's robustness—summarized in Table 11—indicate that the proposed system is the most robust among its related works since it can handle 13 attacks. Certain attacks were excluded from the analysis, such as *Sinkhole*, *Neighbor*, and *Sybil* attacks due to their high similarities with *Decreased Rank*, *Replay*, and *Clone-ID* attacks, respectively. Moreover, *Decreased Rank* and *DODAG Inconsistency* attacks appear twice in the Table to highlight how alternative mechanisms can handle them.

More specifically, each row of Table 11 represents a three-hour simulation, divided in 5 min time-slots, that regards the same 25-nodes' network. The first two rows refer to Chebyshev's and Dixon's operations in case of non-attack. In contrast, each of the rest rows represents a type of attack (1st column), occurring

at the 80^{th} min, along with the detection mechanism (2nd column) in place. The basic implementation details and configuration for each attack follow:

- In *Blackhole* and *Grayhole* attacks, the malicious node suspends forwarding of UDP data packets traveling towards the sink. For Blackhole, all received packets are silently dropped, while for Grayhole, the attacker decides to forward or not the received data packet based on a fair coin toss;
- *Decreased Rank* attack is carried out by a malicious node advertising a fake rank calculated after subtracting four times the RPL's parent switching threshold (*MinHopRankIncrease*) from attacker's actual rank (i.e., *fake_rank = actual_rank - 4*MinHopRankIncrease*);
- For *DODAG Version* attack, an attacker keeps sending DIO messages with increasing version numbers, triggering continuous Trickle Timer resets, in addition to Global and Local Repairs;
- *DODAG Inconsistency* attack is applying erroneous headers in RPL messages [72]) triggering also Trickle Timer resets, Global and Local Repairs;
- Since *Global* or *Local Repair* attacks can be caused by various reasons, they were replicated with a DODAG inconsistency attack;
- *Flooding* attack was implemented with the attacker continuously dispatching forged RPL & data packets, limited by Cooja processing capabilities since a high communication load crashes the (emulated) serial port. Hence, some zeros in the results towards the end of the experiment are indicative;
- *Replay* attack was implemented in a similar way to *Flooding* attack, by having an attacking node continuously resending the RPL messages it receives;
- The *Clone-ID* attacker duplicates existing RIME, MAC, and/or IPv6 addresses, i.e., leading to double node IDs.

The specific attack detection mechanism employed for each attack is also indicated in Table 11. Chebyshev's inequality's and Dixon's settings are *window-size=8*, $p_1 = 0.95$ and *window-size=5*, $confidence = q99$, respectively. The configuration of threshold F was set to 10 (half of the one proposed by RPL, assuming a hostile environment), and adaptable λ is described in Equation 1. These mechanisms operate both at node- and *controller*-side, depending on the attack type. K-Means converges once its objective function improvement between two consecutive iterations is less than a minimum amount of improvement specified. In the current case, it is 0.1. Rank Validation (RV) is described in 4.5.1.

The main cells in Table 11 indicate the number of nodes signaling an attack at the given time-slot, based on the mechanism referenced in the particular row. Indicated with bold, the time-slot that attacks start, e.g., slot 16 was selected on 80^{th} min for all different cases, and cells are colored differently when the attacks are detected (gray), and mitigated (dark gray-white fonts), as well as those reflecting false positives (light gray). For example, there are a few false positives caused by single nodes. As previously discussed, an event is considered an attack when at least three nodes declare its detection, except for Clone-ID and Global Repair attacks. The corresponding mechanisms do not cause false positives, e.g., the Global Repair detection mechanism operates at the sink only. Moreover, regarding Decreased Rank detection, although four rank inconsistencies are reported in time-slot 18, the dedicated RV mechanism needs to mandate the nodes to enable *full-function mode* to send all neighbor's data (i.e., [SN] message) and compare all declared ranks for discrepancies before identifying the attacker.

An attack is considered mitigated when the proper mitigation action is enforced, independently of the time it takes. An indication of the latter appears in the Table through the declining number of nodes signaling the attack immediately after the mitigation time-slots. Once the notation is described, observations come next, based on the results in each Table's row.

The first two rows consider simulations without attacks to highlight the overhead of *ASSET* during regular system operation. On the one hand, Chebyshev's inequality did not produce any false positives. However, there were some rare false positives with more relaxed confidence levels (e.g., $p_1 = 0.90$), but without triggering attack detection. On the other hand, the Dixon-Q test faces 5 cases of single-node detecting outliers, e.g., node 22^{nd} on time-slots 23, 24, and 25. It is also noted that when an attack has taken place, Dixon-Q detects some infrequent outliers once the attack is mitigated since the network settles down progressively. This causes a minor communication overhead increase in the particular nodes, i.e., enabling the transmission of ICMP statistics to the *controller* who then, based on a configurable timeout, disables this feature by mandating the nodes to go back to essential mode ([EI] message).

Table 11: ASSET's Robustness Evaluation.

Time (180 min)		*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*	*55*	*60*	*5*	*10*	*15*	***20***	*25*	*30*	*35*	*40*	*45*	*50*	*55*	*60*	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*	*55*	*60*
Time-slot		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**16**	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
No Attack	**Mech**																																				
Chebyshev's Inequality	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dixon-Q Test	Di	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0
Attack																																					
Blackhole	K	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	**0**	0	0	1	2	4	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Grayhole	K	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	**0**	0	0	0	0	1	1	2	4	5	0	0	0	0	0	0	0	0	0	1	0
Decreased Rank	RV	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	**0**	0	4	5	5	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Decreased Rank	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	2	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DODAG Version	$\lambda(C,n)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	15	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DODAG Inconsistency	λ(C,n)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DODAG Inconsistency	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Global Repair	$\lambda(C)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Local Repair	$\lambda(C)$,F(n)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	11	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flooding	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	2	5	9	10	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Replay	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	7	11	12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Clone-ID	Δ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Attack initiation	Ch: Chebyshev's Inequality, Di: Dixon-Q Test, K: K-Means
False Positives	λ: Adaptable Threshold, F: Fixed Threshold, RV: Rank Validation, Δ: Node ID Validation
Attack Detection	*C*: Controller, *n*: node
Attack Mitigation	

However, those aspects and possible implications (e.g., the reappearance of an attacker) deserve a further analysis that is out of this work's scope.

Blackhole and *Greyhole* attacks impact data rather than control packets. The K-Means algorithm is employed, which continuously clusters the nodes into two groups based on their UDP packets arrived at the sink-node. An attack is considered present whenever a small cluster appears with a limited number of nodes that present a low number of UDP packets, i.e., assuming that the attack does not impact most nodes. Consequently, the sporadic false positives do not cause any issue. It was noticed that topology size and severity of attack impact false positives and attack mitigation time. For example, it takes three more time-slots for *ASSET* to mitigate the less severe *Grayhole* attack, compared to *Blackhole*. Such issues deserve a dedicated investigation.

Regarding the *Decreased Rank* attack, results are provided for both Rank Validation (RV) and Chebyshev mechanisms. The former needs four time-slots until its mitigation time, while the latter can detect the attack in just two time-slots. However, Chebyshev is not equipped to mitigate this particular attack. In this particular experiment, RV is characterized by two false positives, before and after the attack, without impacting the attack detection process. These results highlight the need for dedicated specification-based mechanisms.

DODAG Version attack is mitigated within two time-slots, because of frequent DIO packets with increasing DODAG versions. In the first and second time-slots, the adaptable λ thresholds are being crossed at the node and controller-levels, respectively, i.e., the latter confirming the attack detection. An equivalent result exists for *DODAG Inconsistency* attack since their outcome is similar given the same spatial position of the attacker. For such attacks, identifying the attacker is challenging and beyond this work's scope since it requires additional software or equipment [79]. Consequently, the outcome of the attack is mitigated, i.e., suspend resetting Trickle Timer, Global, and Local Repairs. We also provide the outcome of the Chebyshev's mechanism in the case of *DODAG Inconsistency* attack, highlighting its inability to detect the latter and the advantages of *ASSET's* specification-based mechanisms. It is noted that Chebyshev with a lower sensitivity (e.g., $p_1 = 0.90$ and the same window-size) can detect the attack at time-slot 20 and mitigate it at 21, i.e., later than adaptable λ. Although these aspects call for further investigation, they highlight that anomaly detection and specification-based mechanisms can be operating in a parallel manner, complementing each other.

In the case of *Global Repair* attack (affecting the sink only), *ASSET* needs three time-slots to mitigate it (i.e., the sink ignores further Global Repair actions). This process involves the communication of nodes with the sink and the follow-up involvement of the *controller*. The mitigation time is shorter by one time-slot for *Local Repair* attacks. In this case, nodes signal an attack as soon as their fixed threshold F is reached, which is confirmed by the *controller* with its adaptable threshold λ.

It takes four time-slots for *ASSET* to detect both *Flooding* and *Replay* attacks. One node detects an outlier for the Replay attack, at 28^{th} time-slot, which is ignored by the *controller*. Mitigation for both attacks involves disabling Global and Local repairs, as well as Trickle Timer resets. Since Cooja faces stability issues with these two attacks, conducting these experiments in a test-bed environment and studying the network's behavior under actual network conditions is another open

issue.

Clone-ID attackers are rapidly identified by the *controller* with 100 percent accuracy, due to the centralized nature of *ASSET*, i.e., nodes with duplicated IDs are immediately detected and black-listed. *Sybil* attacks will also be equivalently mitigated.

The above results demonstrate that *ASSET*, under the given scenario, configuration settings and network conditions: (i) can detect 13 attacks (i.e., including those three identified as having a very similar behavior) without false positives in attack detection, i.e., only some rare false alarms are noticed from nodes to the *controller*; (ii) handles effectively the infrequent false alarms due to the requirement that at least three nodes should signal an attack before a mitigation action being triggered; (iii) employs multiple attack detection mechanisms, including three anomaly detection and four specification-based, contributing to both *width* and *depth* of attack detection; (iv) requires up-to four time-slots and one for mitigation in the particular experiments, i.e., mitigation time depends on the attack type, severity and behavior; and (v) manages to identify and exclude the attackers for *Blackhole*, *Greyhole*, *Decreased Rank*, and *Clone-ID* attacks, while for the rest of them it mitigates the outcome of the attack, i.e, the attack may still be present.

Due to the high complexity of the experiments, a more thorough investigation of *ASSET's* performance, including its statistical evaluation, is future work. However, the results are representative and suffice to highlight *ASSET's* novelties.

4.11 Chapter Summary

In this chapter, the design, implementation, and operational features of *ASSET* a state-of-the-art Intrusion Detection Softwarized controller for RPL was introduced, capable of confronting more RPL-related attacks than any other similar technology in the literature today. Moreover, *ASSET* is modular and easily expandable. As such, it can be enriched to confront some attacks with higher precision, but also to tackle newly emerged attacks. Moreover, the source code is freely distributed (Table 1), so it can be utilized as a platform for future research by the scientific community.

In the following chapter, the final remarks and conclusions of this dissertation are presented.

5 Conclusion & Future Work

In this dissertation, the following basic notions were covered:

In Chapter 2, there are several innovative solutions proposed for two RPL/IoT issues, namely peer-to-peer communication and mobility. Among the proposed solutions, there is an innovative algorithm for peer-to-peer communications, adaptable versions of RPL for the same, while also focusing on the mobility of the nodes, and several experiments and extended testing regarding those provided innovations.

In Chapter 3 there is a systematic bibliography mapping of the security problems and issues that the RPL protocol is facing, along with the main limitations the current state-of-the-art IDS solutions have. Moreover, the chapter reveals some open research questions towards centralized, intelligent, SDN-like IDSs for IoT, and for RPL in particular, which were exploited in chapter 4.

In Chapter 4, *ASSET*, a novel Intrusion Detection System was introduced, employing: (i) a holistic workflow handling 13 well-known attacks; (ii) 3 anomaly and 4 specification-based attack detection mechanisms, operating both at node and controller-level and exhibiting a low number of false positives; (iii) a set of alternative mitigation actions and an original attacker identification process; and (iv) an adaptable control and monitoring protocol, trading communication overhead for attacker detection accuracy. *ASSET* is inspired by the softwarization paradigm, providing centralized intelligence and extendability while keeping in acceptable levels the control overhead. There are also several experimental results validating the above novel characteristics.

As an aftermath, the RPL routing protocol is a relatively mature technology that allows IPv6 routing in LLNs. As distributed and robust, it can also benefit by applying specific abstractions and offloading some functionalities from the constrained nodes to the abundant in resources central infrastructure, following the SDN-paradigm.

The investigation of RPL attacks revealed that there is room for research regarding holistic solutions with specific tailored-made characteristics, such as monitoring and exploiting several features in conjunction, e.g., network conditions and protocols' mechanisms, handling mobility, respecting resource constraints, while at the same time providing a high level of security reflected in robustness and low false positives.

5.1 Future Work Discussion

Regarding the point-to-point communication of nodes, a possible pathway would be to determine the peer-to-peer path and which one is the best available according to ad-hoc or preset criteria. Such works [140], are describing an extra ICMP packet to be sent from the originating node, or a temporary DODAG [53], looking for the best path toward the target node.

Moreover, the link coloring additions to RPL in Chapter 2.5.3, have a lot of potential and capabilities to be further explored. The link-coloring can be utilized for various new, advanced Objective Functions by setting priorities and pathways quality-estimation. Such Objective Functions could assign colors to paths back to the sink, depending on a variety of specific conditions. As an example, different

pathways, colored accordingly, could be utilized during the day when node batteries are charging from solar panels, while during the night, alternative fail-safe pathways will be enabled, avoiding nodes with limited power (e.g., this area was cloudy during the day, and butteries did not fully charge).

The energy sector, and battery preservation, is a significant domain that needs special attention and craves for innovative solutions and more experiments. In this direction, *ASSET* could be utilized in experimenting even in real testbeds, and accurately measure the different behaviors and battery preservation of the nodes under specific conditions. For example, how does a particular attack affect the nodes' battery exhaustion? Moreover, *ASSET* would help measure and directly compare the various power exhaustion simulation algorithms against real measurements of actual network implementations.

In the mobility of nodes detection front, CORAL [19, 141] has successfully cooperated with solutions where the back-pressure algorithm was utilized for such a purpose [57]. The mechanism of finding quickly and accurately according to certain criteria the "optimal" path between the sender and the receiver can be implemented in the following way: In RPL each node that does not have any parents "poisons" its rank so it can choose no legitimate parent. In such cases, the node has to start its own (floating) DODAG [3]. After this DODAG is created, the node communicates it with the supervising SDN mechanism and then stops "poisoning" its rank and triggers a local repair to re-join the original "legitimate" rank. Now the SDN above "knows" the "optimal" path from the sender to the receiver and can set the nodes along accordingly.

Regarding *ASSET*, the next steps include the following aspects:

- To further improve the attack detection and mitigation, as well as the attacker identification mechanisms, including employing change-point analysis for anomaly detection [135], [142].
- To conduct extensive experimentation with multiple attacks, multiple (also co-existing), attackers, topology structures and sizes, experiment configurations, including based on real IoT testbeds.
- To incorporate a separate control channel with a long-range interface, inspired by [143], [144], which can significantly improve *ASSET's* operation, in terms of communication overhead and attack mitigation capability.
- To assess the impact of nodes' mobility and how it can affect attack detection since it can also increase control overhead.

In another view, *ASSET* can be utilized and expanded as an IoT centralized console and monitoring infrastructure. A pathway would be to utilize other languages (e.g., RUST) or other communication ways (e.g., experiment with Remote Procedure Call (RPC) frameworks like gRPC). Towards this path, researchers can implement *ASSET* in the cloud and then connect it to an IoT testbed somewhere on the planet. Such experiments will not only produce more valuable data but will also reveal new potential, new possible attacks, and also reveal the limitations and possible attack fronts against *ASSET* itself. For example, it is worth exploring the different ways that *ASSET* could be cut-off from the network it is monitoring if a certain amount of nodes are compromised, or when "smart" attackers, with knowledge of *ASSET's* detection mechanisms, manage to forge the statistics they are dispatching, to pass undetected.

In general, attacks against the controller and attacks originated outside RPL

(e.g., Denial of Service or all eavesdropping attacks) were considered outside the scope of this dissertation and were not covered. It would be worth exploring how *ASSET* could be utilized in such areas.

Moroever, some of *ASSET's* security vulnerabilities that are outside the scope of this paper and deserve further investigation are provided right below.

For simplicity, we currently assume that *ASSET Controller* and corresponding communication (e.g., packets carrying measurements from nodes to the *Controller*) is safe and not tampered. For example, attacks oriented to Software-Defined IoT solutions could be relevant to *ASSET*, e.g., targeting a centralized *Controller*. Consequently, there is a need for hardening the related security. Several techniques could be potentially applied, including Byzantine Fault Tolerance [145], n-versioning, or secure tokens and enclaves. Moreover, a sophisticated attack could potentially tamper with the measurements traveling to the sink to "hide" an ongoing attack or to work around an *ASSET* mechanism. This may be challenging for *ASSET* since it operates many attack detection mechanisms in parallel, i.e., another one may detect the attack. We consider such aspects complementary with our solution but complicated enough to deserve an independent study.

Furthermore, our proposal may be vulnerable to more sophisticated attacks than the considered ones. For example, neighboring nodes may collude to exclude nodes from the graph or apply a Clone-ID attack after collapsing the node to be duplicated. In the latter case, reputation-based mechanisms can be implemented as a scheme with multi-path duplication of messages, i.e., to verify node's compliance. Although this is always the case with IDSs, we consider *ASSET* as a descent solution to many different attacks, in contrast to the related works.

As a bottom line, the solutions proposed in this dissertation should be considered as enabling platforms for research on SDN-like capabilities and expansions for IoT devices, and most eminent, they are enabling the security aspect during all the phases of IoT technologies and communications. A future goal is to enhance *ASSET* with new such features and capabilities, transforming it into a mature, robust application utilized by other researchers, industrial applications, or even applied projects.

—***End of Dissertation***—

References

[1] Cormen, Thomas H and Leiserson, CE and Rivest, RL and Stein, Clifford, "Introduction to algorithms, book," 2009.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE industrial electronics magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "Rfc 6550: Rpl: Ipv6 routing protocol for low-power and lossy networks," *IETF Request For Comments*, 2012.

[4] O. Gaddour and A. Koubâa, "Rpl in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163–3178, 2012.

[5] 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.

[6] G. Violettas, S. Petridou, and L. Mamatas, "Routing under heterogeneity and mobility for the Internet of Things: a centralized control approach," in *Global Communications Conference (GLOBECOM), 2018 IEEE Conf. on*. IEEE, dec 9 13 2018, pp. 1–7.

[7] T. Tsvetkov and A. Klein, "RPL: IPv6 routing protocol for low power and lossy networks," *Sensor Nodes-Operation, Network and Application (SN)*, vol. 59, p. 2, 2011.

[8] A. Mayzaud, R. Badonnel, and I. Chrisment, "A Taxonomy of Attacks in RPL-based Internet of Things," *International Journal of Network Security*, 2016.

[9] A. Verma and V. Ranga, "Security of RPL based 6LoWPAN Networks in the Internet of Things: A Review," *IEEE Sensors Journal*, vol. 20, no. 11, pp. 5666–5690, 2020.

[10] A. Arena, P. Perazzo, C. Vallati, G. Dini, and G. Anastasi, "Evaluating and improving the scalability of RPL security in the Internet of Things," *Computer Communications*, 2020.

[11] P. Perazzo, C. Vallati, A. Arena, G. Anastasi, and G. Dini, "An implementation and evaluation of the security features of rpl," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2017, pp. 63–76.

[12] P. O. Kamgueu, E. Nataf, and T. D. Ndie, "Survey on RPL enhancements: a focus on topology, security and mobility," *Computer Communications*, vol. 120, pp. 10–21, 2018.

[13] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the internet of things: a survey of existing protocols and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[14] A. Raoof, A. Matrawy, and C.-H. Lung, "Routing attacks and mitigation methods for RPL-based internet of things," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1582–1606, 2018.

[15] G. Simoglou, G. Violettas, S. Petridou, and L. Mamatas, "Intrusion Detection Systems for RPL Security: A Comparative Analysis," *Computers & Security*, vol. 104, p. 102219, 2021.

[16] P. Pongle and G. Chavan, "A survey: Attacks on RPL and 6LoWPAN in IoT," in *2015 International conference on pervasive computing (ICPC)*. IEEE, 2015, pp. 1–6.

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

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

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

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

[21] 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.

[22] G. Violettas, S. Petridou, and L. Mamatas, "ASSET: A Softwarized intruSion dEtection SysTem for RPL," *Computers & Security*, 2021.

[23] J. M. Talavera, L. E. Tobón, J. A. Gómez *et al.*, "Review of IoT applications in agro-industrial and environmental fields," *Computers and Electronics in Agriculture*, vol. 142, pp. 283–297, 2017.

[24] J. Qi, P. Yang, G. Min *et al.*, "Advanced internet of things for personalised healthcare systems: A survey," *Pervasive and Mobile Computing*, vol. 41, pp. 132–149, 2017.

[25] S. R. Islam, D. Kwak, M. H. Kabir *et al.*, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[26] Y. B. Zikria, M. K. Afzal, F. Ishmanov *et al.*, "A survey on routing protocols supported by the Contiki Internet of things operating system," *Future Generation Computer Systems*, vol. 82, pp. 200–219, 2018.

[27] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL)," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*. IEEE, 2011, pp. 365–372.

[28] E. T. Phinney, P. Thubert, and RA. Assimiti Nivis, "RPL applicability in industrial networks draft-ietf-roll-rpl-industrial-applicability-02," *IETF draft*, Oct. 2013. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-roll-rpl-industrial-applicability-02.txt

[29] M. Goyal, E. Baccelli, M. Philipp, A. Brandt, and J. Martocci, "RFC 6997: Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks," 2013.

[30] R. Silva, J. S. Silva, and F. Boavida, "A proposal for proxy-based mobility in WSNs," *Computer Communications*, vol. 35, no. 10, pp. 1200–1216, 2012.

[31] P. Ruckebusch, S. Giannoulis, E. De Poorter *et al.*, "A unified radio control architecture for prototyping adaptive wireless protocols," in *European Conf. on Networks and Commun. (EuCNC)*. IEEE, 2016, pp. 58–63.

[32] K. C. Lee, R. Sudhaakar, J. Ning, L. Dai, S. Addepalli, J. Vasseur, and M. Gerla, "A comprehensive evaluation of RPL under mobility," *International Journal of vehicular technology*, vol. 2012, 2012.

[33] 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 Network of the Future (NOF)*. IEEE, 2012, pp. 1–8.

[34] 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.

[35] P. Thubert, "Objective function zero for the routing protocol for low-power and lossy networks (rpl)," March, Tech. Rep., 2012.

[36] O. Gnawali and P. Levis, "The minimum rank with hysteresis objective function," *RFC 6719*, 2012.

[37] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th annual IEEE international conference on local computer networks*. IEEE, 2004, pp. 455–462.

[38] J.-P. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "RFC 6551, routing metrics used for path calculation in low-power and lossy networks," RFC 6551, IETF, Tech. Rep., 2012.

[39] O. Alay, A. Lutu, D. Ros, R. Garcia, V. Mancuso *et al.*, "MONROE: Measuring mobile broadband networks in Europe," in *Proc. of the IRTF & ISOC Workshop on Research and Applications of Internet Measurements (RAIM)*, 2015.

[40] D. Johnson, C. Perkins, J. Arkko *et al.*, "RFC 3775: Mobility support in IPv6," *IETF, June*, pp. 1–165, 2004.

[41] C. Cobârzan, J. Montavont, and T. Noel, "Integrating mobility in RPL," in *European Conf. on wireless sensor networks*. Springer, 2015, pp. 135–150.

[42] F. Gara, L. B. Saad, R. B. Ayed, and B. Tourancheau, "RPL protocol adapted for healthcare and medical applications," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2015 International*. IEEE, 2015, pp. 690–695.

[43] L. B. Saad and B. Tourancheau, "Sinks mobility strategy in IPv6-based WSNs for network lifetime improvement," in *4th IFIP Intl. Conf. on New Technol., Mobility and Security (NTMS)*. IEEE, 2011, pp. 1–5.

[44] S. Sanshi and C. Jaidhar, "Enhanced mobility aware routing protocol for low power and lossy networks," *Wireless Networks*, pp. 1–15, 2017.

[45] S. Sanshi and C. Jaidhar, "Enhanced mobility routing protocol for wireless sensor network," *Wireless Networks*, pp. 1–15, 2018.

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

[47] F. Gara, L. B. Saad, E. B. Hamida, B. Tourancheau, and R. B. Ayed, "An adaptive timer for RPL to handle mobility in wireless sensor networks," in *Int. Wireless Commun. and Mobile Computing Conf. (IWCMC)*. IEEE, 2016, pp. 678–683.

[48] J. Ko and M. Chang, "MoMoRo: Providing mobility support for low-power wireless applications," *IEEE Systems Journal*, vol. 9, no. 2, pp. 585–594, 2015.

[49] F. Somaa, I. El Korbi, C. Adjih, and L. A. Saidane, "A modified RPL for wireless sensor networks with bayesian inference mobility prediction," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International*. IEEE, 2016, pp. 690–695.

[50] J. Ko, J. Jeong, J. Park, J. A. Jun, O. Gnawali, and J. Paek, "Dualmop-rpl: Supporting multiple modes of downward routing in a single rpl network," *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 2, p. 39, 2015.

[51] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi, "A performance analysis of point-to-point routing along a directed acyclic graph in low power and lossy networks," in *2010 13th International Conference on Network-Based Information Systems*. IEEE, 2010, pp. 111–116.

[52] M. Goyal, E. Baccelli, A. Brandt, and J. Martocci, "A mechanism to measure the routing metrics along a point-to-point route in a low-power and lossy network," *Internet Eng. Task Force, Fremont, CA, USA, RFC*, vol. 6998, 2013.

[53] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," RFC 3561, No., Tech. Rep., 2003.

[54] J. Tripathi, J. C. De Oliveira, and J.-P. Vasseur, "Proactive versus reactive routing in low power and lossy networks: Performance analysis and scalability improvements," *Ad Hoc Networks*, vol. 23, pp. 121–144, 2014.

[55] P. Thubert, T. Watteyne, R. Struik, and M. Richardson, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4," *Working Draft, IETF Secretariat, Internet-Draft draft-ietf-6tisch-architecture-14*, 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6tisch-architecture-15

[56] M. Zhang, A. Sangi, C. Perkins, and S. Anand, "Asymmetric aodv-p2p-rpl in low-power and lossy networks (llns) - draft-satish-roll-aodv-rpl-03," 2016. [Online]. Available: https://www.ietf.org/archive/id/draft-satish-roll-aodv-rpl-03.txt

[57] Y. Tahir, S. Yang, and J. McCann, "BRPL: Backpressure RPL for High-throughput and Mobile IoTs," *IEEE Transactions on Mobile Computing*, 2017.

[58] 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.

[59] T. Theodorou and L. Mamatas, "Software defined topology control strategies for the internet of things," in *2017 IEEE Conf. on Network Function Virtualization and Software Defined Networks (NFVSDN)*, 2017.

[60] N. Abdolmaleki, M. Ahmadi, H. T. Malazi, and S. Milardo, "Fuzzy topology discovery protocol for SDN-based wireless sensor networks," *Simulation Modelling Practice and Theory*, vol. 79, pp. 54–68, 2017.

[61] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-defined WSN management system for IoT applications," *IEEE Systems J.*, 2016.

[62] C. Jacquenet and M. Boucadair, "A software-defined approach to IoT networking," *ZTE Communications*, vol. 1, pp. 1–12, 2016.

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

[64] E. Municio, J. Marquez-Barja, S. Latré, and S. Vissicchio, "Whisper: Programmable and flexible control on industrial iot networks," *Sensors*, vol. 18, no. 11, p. 4048, 2018.

[65] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for low-power IoT networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, June 2018, pp. 71–79.

[66] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the rpl-based internet of things," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, 2013.

[67] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A specification-based IDS for detecting attacks on RPL-based network topology," *Information*, vol. 7, no. 2, p. 25, 2016.

[68] A. Mayzaud, R. Badonnel, and I. Chrisment, "A distributed monitoring strategy for detecting version number attacks in RPL-based networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 472–486, 2017.

[69] A. Kamble, V. S. Malemath, and D. Patil, "Security attacks and secure routing protocols in rpl-based internet of things: Survey," in *2017 International Conference on Emerging Trends Innovation in ICT (ICEI)*, 2017, pp. 33–39.

[70] A. Le, J. Loo, Y. Luo, and A. Lasebae, "The impacts of internal threats towards routing protocol for low power and lossy network performance," in *2013 IEEE*

Symposium on Computers and Communications (ISCC), 2013, pp. 000 789–000 794.

[71] P. Pongle and G. Chavan, "A survey: Attacks on RPL and 6LoWPAN in IoT," in *2015 International conference on pervasive computing (ICPC)*. IEEE, 2015, pp. 1–6.

[72] A. Sehgal, A. Mayzaud, R. Badonnel, I. Chrisment, and J. Schönwälder, "Addressing DODAG inconsistency attacks in RPL networks," in *Proceedings of Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2014, pp. 1–8.

[73] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai, "The Impact of Rank Attack on Network Topology of Routing Protocol for Low-Power and Lossy Networks," in *IEEE Sensors Journal*. IEEE, 2013, vol. 13, pp. 3685–3692.

[74] A. Aris, S. F. Oktug, and S. Berna Ors Yalcin, "Rpl version number attacks: In-depth study," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 776–779.

[75] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "A study of rpl dodag version attacks," in *Monitoring and Securing Virtualized Networks and Services, LNCS, volume 8508*. Springer, 2014, pp. 92–104.

[76] P. Pongle and G. Chavan, "Real Time Intrusion and Wormhole Attack Detection in Internet of Things," *International Journal of Computer Applications*, vol. 975, p. 8887, July, 2015.

[77] D. Airehrour, J. Gutierrez, and S. K. Ray, "Secure routing for Internet of Things: A survey," *Journal of Network and Computer Applications*, vol. 66, pp. 198–213, 2016.

[78] D. Sharma, I. Mishra, and S. Jain, "A detailed classification of routing attacks against RPL in Internet of Things," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 3, pp. 692–703, 2017.

[79] P. Perazzo, C. Vallati, G. Anastasi, and G. Dini, "DIO suppression attack against routing in the Internet of Things," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2524–2527, 2017.

[80] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi, "Routing Loops in DAG-Based Low Power and Lossy Networks," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 888–895.

[81] K. Chugh, A. Lasebae, and J. Loo, "Case Study of a Black Hole Attack on 6LoWPAN-RPL," in *Proc. of the Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE), Rome, Italy (August 2012)*, 07 2012, pp. 157–162.

[82] A. Kumar, R. Matam, and S. Shukla, "Impact of packet dropping attacks on rpl," in *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 2016, pp. 694–698.

[83] K. Zhang, X. Liang, R. Lu, and X. Shen, "Sybil attacks and their defenses in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 372–383, 2014.

[84] F. Medjek, D. Tandjaoui, M. R. Abdmeziem, and N. Djedjig, "Analytical evaluation of the impacts of sybil attacks against rpl under mobility," in *2015 12th International Symposium on Programming and Systems (ISPS)*, 2015, pp. 1–9.

[85] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano, and M. Richardson, "A security threat analysis for the routing protocol for low-power and lossy networks (RPLs)," *RFC 7416*, p. 131, 2015. [Online]. Available: https://tools.ietf.org/html/rfc7416

[86] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[87] B. Lokesak, "A Comparison Between Signature Based and Anomaly Based Intrusion Detection Systems," *PPT). www. iup. edu*, 2008. [Online]. Available: http://www.iup.edu/WorkArea/DownloadAsset.aspx?id=81109

[88] R. A. Sadek, M. S. Soliman, and H. S. Elsayed, "Effective anomaly intrusion detection system based on neural network with indicator variable and rough set reduction," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 6, p. 227, 2013.

[89] H. Sedjelmaci, S. M. Senouci, and T. Taleb, "An accurate security game for low-resource IoT devices," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 9381–9393, 2017.

[90] M. N. Napiah, M. Y. I. B. Idris, R. Ramli, and I. Ahmedy, "Compression Header Analyzer Intrusion Detection System (CHA - IDS) for 6LoWPAN Communication Protocol," *IEEE Access*, vol. 6, pp. 16 623–16 638, 2018.

[91] J. Kaur, "An Ultimate Approach of Mitigating Attacks in RPL Based Low Power Lossy Networks," *Proceedings of 17th International Conference on Security and Management (SAM'19)*, 2019.

[92] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[93] H. Bostani and M. Sheikhan, "Hybrid of Anomaly-Based and Specification-Based IDS for Internet of Things Using Unsupervised OPF Based on MapReduce Approach," *Computer Communications*, pp. 52–71, 2016.

[94] S. Dharmapurikar and J. W. Lockwood, "Fast and Scalable Pattern Matching for Network Intrusion Detection Systems," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 10, pp. 1781–1792, 2006.

[95] E. Aydogan, S. Yilmaz, S. Sen, I. Butun, S. Forsström, and M. Gidlund, "A Central Intrusion Detection System for RPL-Based Industrial Internet of Things," in *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2019, pp. 1–5.

[96] S. M. Othman, N. T. Alsohybe, F. M. Ba-Alwi, and A. T. Zahary, "Survey on Intrusion Detection System Types," *International Journal of Cyber-Security and Digital Forensics*, vol. 7, no. 4, pp. 444–463, 2018.

[97] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito, "An IDS framework for internet of things empowered by 6LoWPAN," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1337–1340.

[98] A. Verma and V. Ranga, "ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things," in *2019 4th International conference on Internet of Things: Smart innovation and usages (IoT-SIU)*. IEEE, 2019, pp. 1–6.

[99] P. Ioulianou and V. Vasilakis, "Denial-of-Service Attacks and Countermeasures in the RPL-Based Internet of Things," *Katsikas S. et al. (eds) Computer Security. CyberICPS 2019, SECPRE 2019, SPOSE 2019, ADIoT 2019*, vol. 11980, pp. 374–390, 2020.

[100] S. Deshmukh-Bhosale and S. S. Sonavane, "A Real-Time Intrusion Detection System for Wormhole Attack in the RPL based Internet of Things," *Procedia Manufacturing*, vol. 32, pp. 840–847, 2019, 12th International Conference Interdisciplinarity in Engineering, INTER-ENG 2018, 4{5 October 2018, Tirgu Mures, Romania.

[101] P. Ioulianou, V. Vasilakis, I. Moscholios, and M. Logothetis, "A signature-based intrusion detection system for the Internet of Things," *Information and Communication Technology Form*, 2018.

[102] A. Verma, V. Ranga *et al.*, "Cosec-rpl: detection of copycat attacks in rpl based 6lowpans using outlier analysis," *Telecommunication Systems: Modelling, Analysis, Design and Management*, pp. 1–19, 2020.

[103] M. Surendar and A. Umamakeswari, "InDReS: An Intrusion Detection and response system for Internet of Things with 6LoWPAN," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2016, pp. 1903–1908.

[104] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 606–611.

[105] F. Gara, L. B. Saad, and R. B. Ayed, "An intrusion detection system for selective forwarding attack in IPv6-based mobile WSNs," in *13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 276–281.

[106] K. Sentz, S. Ferson, and K. Sentz, "Combination of evidence in dempster-shafer theory," US Department of Energy (US), Tech. Rep., 2002.

[107] Barnett, V. and Lewis, T., *Outliers in statistical data. 3rd edition*. J. Wiley & Sons, 1994, vol. 37, no. 2.

[108] E. Kfoury, J. Saab, P. Younes, and R. Achkar, "A Self Organizing Map Intrusion Detection System for RPL Protocol Attacks," *International Journal of Interdisciplinary Telecommunications and Networking (IJITN)*, vol. 11, no. 1, pp. 30–43, 2019.

[109] A. Nikam and D. Ambawade, "Opinion Metric Based Intrusion Detection Mechanism for RPL Protocol in IoT," in *3rd International Conference for Convergence in Technology (I2CT)*. IEEE, 2018, pp. 1–6.

[110] F. Nygaard, "Intrusion detection system in IoT," Master's thesis, NTNU, 2017.

[111] L. Zhang, G. Feng, and S. Qin, "Intrusion detection system for RPL from routing choice intrusion," in *2015 IEEE International Conference on Communication Workshop (ICCW)*. IEEE, 2015, pp. 2652–2658.

[112] U. Shafique, A. Khan, A. Rehman, F. Bashir, and M. Alam, "Detection of rank attack in routing protocol for Low Power and Lossy Networks," *Annals of Telecommunications*, 2018.

[113] F. Ahmed and Y.-B. Ko, "A Distributed and Cooperative Verification Mechanism to Defend against DODAG Version Number Attack in RPL," in *PECCS*, 2016, pp. 55–62.

[114] H. Svensson and A. Jøsang, "Correlation of intrusion alarms with subjective logic," in *Proceedings of the sixth Nordic Workshop on Secure IT systems (NordSec2001), Copenhagen, Denmark*. Citeseer, 2001.

[115] T. Matsunaga, K. Toyoda, and I. Sasase, "Low false alarm attackers detection in RPL by considering timing inconstancy between the rank measurements," *IEICE Communications Express*, vol. 4, no. 2, pp. 44–49, 2015.

[116] D. Shreenivas, S. Raza, and T. Voigt, "Intrusion detection in the RPL-connected 6LoWPAN networks," in *Proceedings of the 3rd ACM international workshop on IoT privacy, trust, and security*, 2017, pp. 31–38.

[117] L. Rocha, F. Cappabianco, and A. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, pp. 50 – 68, 06 2009.

[118] S. Sahu and B. M. Mehtre, "Network intrusion detection system using j48 decision tree," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2015, pp. 2023–2026.

[119] T. Wauters *et al.*, "Federation of internet experimentation facilities: architecture and implementation Federation of internet experimentation facilities: architecture and implementation," in *European Conf. on Networks and Communications (EuCNC)*. IEEE, 2014, pp. 1–5.

[120] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[121] P. Valsamas, P. Papadimitriou, I. Sakellariou, S. Petridou, L. Mamatas, S. Clayman, F. Tusa, and A. Galis, "Multi-PoP Network Slice Deployment: A Feasibility Study," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*. IEEE, 2019, pp. 1–6.

[122] P. D. Maciel, F. L. Verdi, P. Valsamas, I. Sakellariou, L. Mamatas, S. Petridou, P. Papadimitriou, D. Moura, A. I. Swapna, B. Pinheiro *et al.*, "A marketplace-based approach to cloud network slice composition across multiple domains," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 480–488.

[123] Flux Research Group, "The University of Utah," https://www.flux.utah.edu/index, 2020.

[124] M. F. Elrawy, A. I. Awad, and H. F. Hamed, "Intrusion detection systems for iot-based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, p. 21, 2018.

[125] E. Frank, M. A. Hall, and I. H. Witten, *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, 4th ed. Morgan Kaufmann, 2016.

[126] S. Schaller and D. Hood, "Software defined networking architecture standardization," *Computer Standards & Interfaces*, vol. 54, pp. 197 – 202, 2017, sI: Standardization SDN&NFV.

[127] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, "Research on the architecture of internet of things," in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol. 5. IEEE, 2010, pp. V5–484.

[128] M. R. Abdmeziem, D. Tandjaoui, and I. Romdhani, "Architecting the internet of things: state of the art," in *Robots and Sensor Clouds*. Springer, 2016, pp. 55–75.

[129] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné, "GraphStream: A Tool for bridging the gap between Complex Systems and Dynamic Graphs," *EPNACS'2007*, p. 63, 2007.

[130] GraphStream, https://github.com/graphstream, 2018.

[131] S. S. Kalamkar, A. Banerjee, and A. Roychowdhury, "Malicious user suppression for cooperative spectrum sensing in cognitive radio networks using dixon's outlier detection method," in *2012 National Conference on Communications (NCC)*. IEEE, 2012, pp. 1–5.

[132] C. E. Efstathiou, "Estimation of type I error probability from experimental Dixon's \Q" parameter on testing for outliers within small size data sets," *Talanta*, vol. 69, no. 5, pp. 1068–1071, 2006.

[133] D. B. Rorabacher, "Statistical treatment for rejection of deviant values: critical values of Dixon's" Q" parameter and related subrange ratios at the 95% confidence level," *Analytical Chemistry*, vol. 63, no. 2, pp. 139–146, 1991.

[134] B. G. Amidan, T. A. Ferryman, and S. K. Cooley, "Data outlier detection using the chebyshev theorem," in *2005 IEEE Aerospace Conference*. IEEE, 2005, pp. 3814–3819.

[135] S. Skaperas, L. Mamatas, and A. Chorti, "Real-time video content popularity detection based on mean change point analysis," *IEEE Access*, vol. 7, pp. 142 246–142 260, 2019.

[136] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[137] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE transactions on neural networks*, vol. 5, no. 1, pp. 3–14, 1994.

[138] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," *Computers & Mathematics with Applications*, vol. 7, no. 1, pp. 67–72, 1981.

[139] M. Zaminkar and R. Fotohi, "SoS-RPL: Securing Internet of Things Against Sinkhole Attack Using RPL Protocol-Based Node Rating and Ranking Mechanism," *WIRELESS PERSONAL COMMUNICATIONS*, 2020.

[140] M. Goyal, A. Brandt, and E. Baccelli, "A Mechanism to Measure the Routing Metrics along a Point-to-Point Route in a Low-Power and Lossy Network," 2012. [Online]. Available: https://tools.ietf.org/html/draft-ietf-roll-p2p-measurement-03#section-3

[141] "2nd Open Call Scheme of the Wireless Software and Hardware platforms for Flexible and Unified radio and network controL project (WiSHFUL)," Accessed: Jul. 20, 2020. [Online]. Available: http://www.wishful-project.eu/OpenCall2.html

[142] S. Skaperas, L. Mamatas, and A. Chorti, "Real-time algorithms for the detection of changes in the variance of video content popularity," *IEEE Access*, vol. 8, pp. 30 445–30 457, 2020.

[143] 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.

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

[145] S. Marano, V. Matta, and L. Tong, "Distributed detection in the presence of Byzantine attacks," *IEEE Trans. on Signal Process.*, vol. 57, no. 1, pp. 16–29, 2008.

A Appendix

Below, there are some basic instructions, technical details, and screenshots of *ASSET* in action. More information, and the full code of *ASSET* can be found in the relevant GitHub repository.

A.1 ASSET Controller Usage instructions

The *ASSET* follows the SDN paradigm. Hence it only communicates with the sink. A basic setup of the network is depicted in A.38.

The controller can identify an intruder and the attacked nodes. The controller will automatically discover the underlying network, monitor in real-time, and depict changes. A video example here, where the network starts, and then node no 7 changes position. After a while (remember, RPL takes time to adjust), the node's new position and parent are automatically depicted in the GUI. An actual attack is identified in this example, by two attacker (purple color).

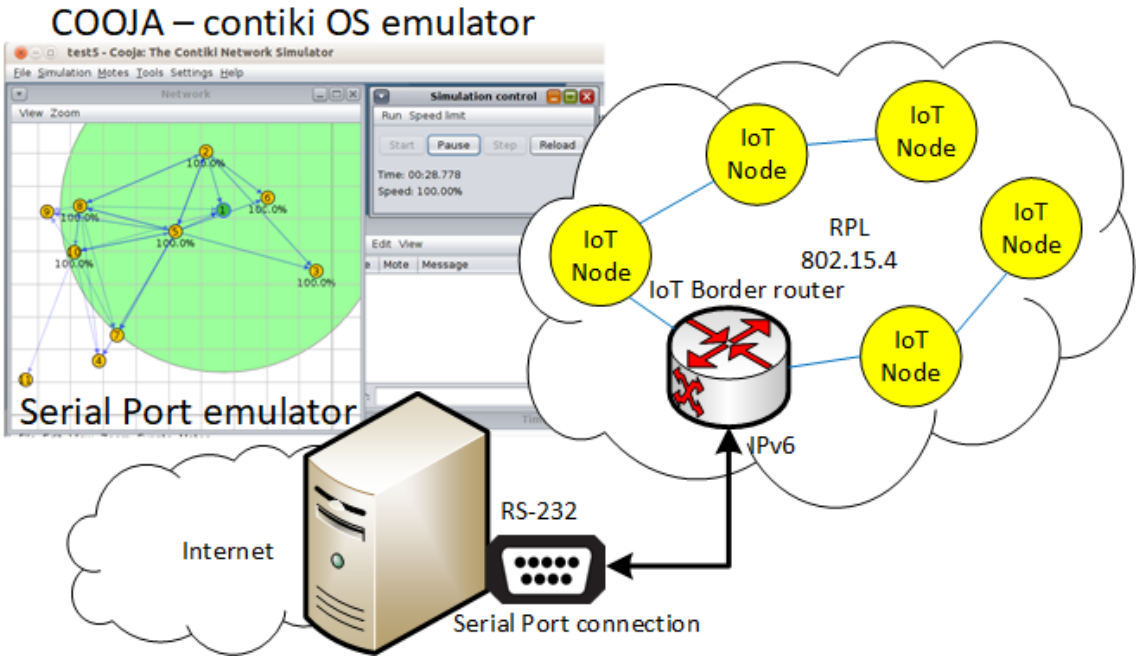


Figure A.38: Basic Network Setup for experimentation. *ASSET* controller usually relies in the computer and connects via the serial port with the sink.

The network nodes are connected via the attackers who are implementing two attacks: rank attack and grayhole attack. After minute 5, the attack is identified.

The controller runs the k-Means algorithm, finds all suspicious nodes, runs a Kosaraju algorithm to discover how many strongly connected graphs there are, and at the end finds the mother of each such sub-graph. The "mother(s)" identified are the attackers.

As a meta-step, the network can exclude the attackers from being selected as parents by using "coloring" from a previous work, (a video here), and papers here, and here.

The *ASSET* controller can be freely run, modified, and adapted, or used for any research project. It also needs a Contiki OS with specifically adapted nodes **(sink-client-intruder)** to run experiments. The "professional" way which gives you access to all, is to download Contiki OS TWICE:

1. One basic Contiki (slightly altered with a lot of custom messages, etc.) from here. Go to contiki/examples/ipv6/rpl-udp-fresh/ and run one of the many *.csc files there. they all use the same two nodes: udp-server.c and udp-client.c. The *dixon*.csc emulations are using the respective *dixon*.c sink/client code.
2. In order to include one or more intruder node(s), you have to download another Contiki version (completely separated), from contiki-malicious, or contiki-malicious-controller-aware, or contiki-malicious-controller-aware-version-attack.
   Again, in all cases, the intruder code is in contiki/examples/ipv6/rpl-udp-fresh/*.c.

A.2 Basic Algorithms Implementations

Listing 1: Basic Continuous Main Method

```
1
2 /* TODO: This method needs restructuring.
3  * First, JSON messaging or similar should be implemented,
4  * for automatically handling messages.
5  * Right now, it is simple string handling/concatenation,
6  * which is difficult to handle, and prone to errors.
7  * Second, threading or similar mechanism(s) should be
```

```
 * implemented, along with synchronization and time delays.
 * This way, those if/then/else, can be dramatically
 * reduced, making the code robust, and easy to follow.
 * Most importantly, implementing the above will make
 * the code easy to be expanded, add new features, or
 * implement it for different implementations
 * (e.g., outside RPL).
 */
public class Client implements Runnable{

  private volatile boolean exit = false; /* to start/restart/stop
      the thread */
  private static String ipServer; /* hardwire the sink's IP if
     not found */
  int roundsCounter=0;
  SerialPortProbe serialportprobe = new SerialPortProbe();
  ClientHelper clienthelper = new ClientHelper();
  SerialPort motePort = null;
  ClusterMonitor clustermonitor;

    @Override
    public void run(){
      while(!Thread.interrupted()) {
        debug("Client searching for Serial port...");
        /* it will continue, ONLY when port is found */
        while(motePort == null) {
          motePort = serialportprobe.getSerialPort();
        }
      clienthelper.setMotePort(motePort);
          /* read serial port output line by line */
          Scanner lineReader = new Scanner(motePort.
             getInputStream());

      while(lineReader.hasNextLine() ){
```

```
39          String inComingLine = lineReader.nextLine();
40          if(inComingLine!=null){
41                  if(inComingLine.startsWith("Tentative")){ /* only
                        the sink prints at the serial port */
42                  String[] parts = inComingLine.split("Tentative
                       link-local IPv6 address ",2);
43                  ipServer = "["+parts[1]+"]";
44                  debug("found ipServer: "+ipServer);
45                  clienthelper.setSink(ipServer); /* ONCE ONLY, at
                       the beginning */
46
47                  //TODO: Is this redundant? ipServer already
                       exists by now?
48                  //clienthelper.checkNode(ipServer);
49
50                }/* end if InPut.startsWith("Tentative") */
51                  else
52                  if (inComingLine.startsWith("Route")){
53                  String[] parts = inComingLine.split(" ",4);
54                  String ip1 = parts[1];
55                  String ip2 = parts[2];
56                  String[] ltime = parts[3].split(":",2);
57                  String lt = ltime[1];
58                  int intlt = Integer.parseInt(lt); //TODO: use the
                        intlt in the graph
59
60                  if(ip1.equals(ip2)) {/* node is a direct child of
                        sink's */
61                    debug("found a direct child of sink: "+ip1);
62                    if(ipServer == null) {
63                      debug("ATTENTION: Found a sink's child, SINK'
                           s IP IS NOT SET YET!");
64                      ipServer = "[fe80:0000:0000:0000:c30c
                           :0000:0000:0001]";
```

```
65                  //clienthelper.setSink(ipServer); //TODO:
                        Redundant? Sink is always found
66                }
67                clienthelper.checkEdge(ipServer,ip2);
68              }else { /* ip1 != ip2 */
69                clienthelper.onlyAddNodeifNotExist(ip1);
70                  clienthelper.checkNode(ip2);
71              }
72            }/* end if InPut.startsWith("Route") */

73              else
74              if(inComingLine.startsWith("NP")){
75              try{
76                String[] parts = inComingLine.split("NP:",2);
77                parts = parts[1].split(" ",3);
78                String ipParent = parts[0];
79                String ipChild = parts[2];
80                debug("R:"+roundsCounter+" edge "
81                      + clienthelper.IPlastHex(ipParent)
82                      + "-->"+clienthelper.IPlastHex(ipChild));
83
84                clienthelper.onlyAddNodeifNotExist(ipParent);
85                clienthelper.checkNode(ipChild);
86                clienthelper.checkEdge(ipParent, ipChild);
87
88              }catch (ArrayIndexOutOfBoundsException e) {
89                debug("NP line problem: "+ e.toString());
90              }
91            } /* end if InPut startsWith "NP" */
92
93            /* Nodes were forced to send neighbors. They don't
                 necessary
94             * have an edge with those neighbors, so we only
                 add the node
```

```
95                  * if it does not exist, BUT NOT the edge.
96                  * After that, we need to ask the node to print its
                        father.
97                  */
98                   else
99                   if(inComingLine.startsWith("N1")){
100                  try{
101                    String[] parts = inComingLine.split("N1:",2);
102                    parts = parts[1].split(" ",3);
103                    String neighbor = parts[0];
104                    String nodeProbed = parts[2];
105
106                    /* sometimes an IP "0000" comes along */
107                    if(clienthelper.legitIncomIP(neighbor) &&
108                       clienthelper.onlyAddNodeifNotExist(neighbor)
                              ) {
109                      /* The node exists(?) but there was no
                           contact with it yet */
110                       debug("found a NEW neighbor! Lets probe its
                              parents");
111                       String message = "SP "+neighbor+"\n";
112                       clienthelper.sendMsg2Serial(message);
113                    }
114                    if(clienthelper.legitIncomIP(nodeProbed) &&
115                       clienthelper.checkNode(nodeProbed)) {
116                       debug("found a NEW node! Lets probe its
                              parents");
117                       String message = "SP "+nodeProbed+"\n";
118                       clienthelper.sendMsg2Serial(message);
119                    }
120
121                  }catch (ArrayIndexOutOfBoundsException e) {
122                    //debug("could not break apart: "+inComingLine)
                          ;
```

```
123                     debug("N1 line problem: "+e.toString());
124                   }
125                 }
126                   else
127                 if(inComingLine.startsWith("Custom ")){ /* Custom
                       Data coming from node */
128                   try{
129                     String[] parts = inComingLine.split("from ",2);
130                     String nodeAlive = parts[1];
131
132                     if(clienthelper.legitIncomIP(nodeAlive)) {
133                       clienthelper.checkNode(nodeAlive); /* it will
                             also reset the keepAliveTimer */
134                       clienthelper.addRecvdPacket(nodeAlive); /*
                             keep the num of received data packets */
135                     }
136
137                   }catch (ArrayIndexOutOfBoundsException e) {
138                     //debug("could not break apart: "+inComingLine)
                          ;
139                     debug("Custom line problem: "+e.toString());
140                   }
141                 }
142                   else /* Version number attack */
143                 if(inComingLine.startsWith("[VA")){ /* Version
                       attack(s) number */
144                   try{
145                     String[] parts = inComingLine.split(" from ",2)
                          ;
146                     String nodeUnderVerAttack = parts[1];
                                          if(clienthelper.legitIncomIP
                          (nodeUnderVerAttack)) {
147                       parts = parts[0].split(":",2);
```

```
148                     String verNumAttacks =  parts[1].substring(0,
                            parts[1].length() - 1); /* remove ']' */
149                     clienthelper.checkNode(nodeUnderVerAttack);
                           /* it will also reset the keepAliveTimer
                           */
150                     clienthelper.addVerNumAttacks(
                           nodeUnderVerAttack, verNumAttacks); /*
                           keep the num of version number attacks
                           suffered */
151                   }
152                 }catch (ArrayIndexOutOfBoundsException e) {
153                   //debug("could not break apart: "+inComingLine)
                          ;
154                   debug("Version Attack line problem: "+e.
                          toString());
155                 }
156               }
157               else
158               /* Info from Attacker(s) when they Start/Stop. Only
                     for logging */
159               if(inComingLine.startsWith("DATA Intercept")){
160                 Main.debugEssential(inComingLine);
161               }
162               else
163               if(inComingLine.startsWith("[SI:")){ /* ICMP
                     statistics coming from node */
164                 try{
165                   String[] parts = inComingLine.split(" from ",2)
                          ;
166                   String nodeAlive = parts[1];            if(
                          clienthelper.legitIncomIP(nodeAlive)) {
167
168                     // this creates a DOUBLE NODE !!!!!
```

```
169                     // The above statement SEEMS WRONG. CHECK
                           AGAIN if in doubt
170                     clienthelper.checkNode(nodeAlive); /* it will
                            also reset the keepAliveTimer */
171
172                     parts = parts[0].split(":",2);
173                     parts = parts[1].split(" ",2);
174                     String ICMPRecv = parts[0];
175                     String ICMPSent =  parts[1].substring(0,
                           parts[1].length() - 1); /* remove ']' */
176
177                     /* keep the num of Send/Recv ICMP packets */
178                     //clienthelper.addICMPStats(nodeAlive,
                           ICMPRecv, ICMPSent);
179
180                     // using an array now, not one old value
                           above
181                     clienthelper.addICMPArrays(nodeAlive,
                           ICMPRecv, ICMPSent);
182                   }
183                 }catch (ArrayIndexOutOfBoundsException e) {
184                   debug("SI line problem: "+e.toString());
185                 }
186               }
187             clienthelper.probeForHiddenEdges(roundsCounter);
188
189               if(roundsCounter%100==0 && roundsCounter > 100) {/*
                      Every hundred rounds */
190                 clienthelper.getInDegrees(roundsCounter); /* Just
                      in case there is someone hiding? */
191
192                 //TODO: This is not needed anymore since we have
                      spanning tree?
193               }
```

```
194
195             // kMeans on UDPRecv. Clustering =2
196                 /* Initial delay & GUI button pressed to start */
197                 if(Main.appTimeStarted > 2*Main.keepAliveNodeBound
198                     && roundsCounter > 100
199                     && Main.kMeansStart /* GUI toggle button */
200                   ){
201                   clienthelper.runKMeans(2); /* BE CAREFUL: Nothing
                          else than two for now */
202                 }
203           }/* end if InPut!=null */
204
205             roundsCounter++;
206
207       }/* end while nextline() */
208       }/* end while(!exit) */
209
210     }/* end run() */
211
212 }
```

Listing 2: Kosaraju's Alrgorithm Indicative Java Implementation

```
213 /* Java implementation of Kosaraju's algorithm to print all SCCs
       */
214 import java.util.*;
215 import java.util.LinkedList;
216
217 /* This class represents a directed graph using adjacency list
       representation */
218 class Graph {
219  private int V;   // No. of vertices
220  private LinkedList<Integer> adj[]; //Adjacency List
221
222  //Constructor
```

```
223  Graph(int v){
224      V = v;
225      adj = new LinkedList[v];
226      for (int i=0; i<v; ++i)
227          adj[i] = new LinkedList();
228  }
229
230  // Function to add an edge into the graph
231  void addEdge(int v, int w)  { adj[v].add(w); }
232
233  // A recursive function to print DFS starting from v
234  void DFSUtil(int v,boolean visited[]){
235      // Mark the current node as visited and print it
236      visited[v] = true;
237      System.out.print(v + " ");
238
239      int n;
240
241      // Recur for all the vertices adjacent to this vertex
242      Iterator<Integer> i =adj[v].iterator();
243      while (i.hasNext())
244      {
245          n = i.next();
246          if (!visited[n])
247              DFSUtil(n,visited);
248      }
249  }
250
251  // Function that returns reverse (or transpose) of this graph
252  Graph getTranspose(){
253      Graph g = new Graph(V);
254      for (int v = 0; v < V; v++)
255      {
256          // Recur for all the vertices adjacent to this vertex
```

```
257             Iterator<Integer> i =adj[v].listIterator();
258             while(i.hasNext())
259                 g.adj[i.next()].add(v);
260         }
261         return g;
262     }
263
264     void fillOrder(int v, boolean visited[], Stack stack){
265         // Mark the current node as visited and print it
266         visited[v] = true;
267
268         // Recur for all the vertices adjacent to this vertex
269         Iterator<Integer> i = adj[v].iterator();
270         while (i.hasNext())
271         {
272             int n = i.next();
273             if(!visited[n])
274                 fillOrder(n, visited, stack);
275         }
276
277         // All vertices reachable from v are processed by now,
278         // push v to Stack
279         stack.push(new Integer(v));
280     }
281
282     /* The main function that finds and prints all strongly
        connected components */
283     void printSCCs(){
284         Stack stack = new Stack();
285
286         // Mark all the vertices as not visited (For first DFS)
287         boolean visited[] = new boolean[V];
288         for(int i = 0; i < V; i++)
289             visited[i] = false;
```

```
290
291         /* Fill vertices in stack according to their finishing
               times */
292         for (int i = 0; i < V; i++)
293             if (visited[i] == false)
294                 fillOrder(i, visited, stack);
295
296         // Create a reversed graph
297         Graph gr = getTranspose();
298
299         // Mark all the vertices as not visited (For second DFS)
300         for (int i = 0; i < V; i++)
301             visited[i] = false;
302
303         // Now process all vertices in order defined by Stack
304         while (stack.empty() == false)
305         {
306             // Pop a vertex from stack
307             int v = (int)stack.pop();
308
309             // Print Strongly connected component of the popped
                   vertex
310             if (visited[v] == false)
311             {
312                 gr.DFSUtil(v, visited);
313                 System.out.println();
314             }
315         }
316     }
317
318     // Driver method
319     public static void main(String args[]){
320         // Create a graph given in the above diagram
321         Graph g = new Graph(5);
```

```
322       g.addEdge(1, 0);
323       g.addEdge(0, 2);
324       g.addEdge(2, 1);
325       g.addEdge(0, 3);
326       g.addEdge(3, 4);
327
328       System.out.println("Following are strongly connected
              components "+ "in given graph ");
329       g.printSCCs();
330    }
331 }
332 //This code is contributed by Aakash Hasija
```

Listing 3: Chebyshev's Inequality Detection Algoritm

```
333
334 /*
335  * Provides Chebyshev's Inequality method to find
336  * outliers for non-normally distributed data sets
337  */
338 public class ChebyshevInequality {/* implements IOutlierDetector
        */
339
340   DecimalFormat df = new DecimalFormat();
341
342     /* threshold to determine the minimum of values within  * a
          data set must lie in. 90% = 0.9
343      */
344     private double probabilityThreshold = 0.90;
345
346     public List<Double> getOutlierScoreIterator(double
          valueToCheck, Iterator iter) {
347       List<Double> myList = (List<Double>) iter;
348       int count = 0;
349     while(iter.hasNext()) {
```

```
350         int  k = (int)iter.next();
351         double d = k; //straight casting from int to double
352         count++;
353       }
354       return myList;
355       }
356       /*
357        * Check if given value is an outlier in the data set
358        * using Chebyshev's Inequality.
359        * @param valueToCheck   the given values to check
360        * @param timeSeriesData the time series data
361        * @return true if value is an outlier. Otherwise false.
362        */
363       public boolean isOutlier(double valueToCheck, double[]
             timeSeriesData, int nodeId) {
364          double outlierScore = getOutlierScore(valueToCheck,
                timeSeriesData, nodeId);
365          return isOutlierByChebyshevsInequality(valueToCheck,
                outlierScore);
366       }
367
368       /**
369        * Calculates the outlier score using value to be checked and
              time series data
370        *
371        * @param valueToCheck   the given values to check
372        * @param timeSeriesData the time series data
373        * @return an outlier score
374        * @throws Exception
375        */
376       public double getOutlierScore(double valueToCheck, double[]
             timeSeriesData, int nodeId) {
377         df.setMaximumFractionDigits(3); //max double number digits
378
```

```
379         // get mean and standard deviation
380           DescriptiveStatistics statistics = new
                 DescriptiveStatistics(timeSeriesData);
381           double sampleMean = statistics.getMean();
382           double sampleStdDev = statistics.getStandardDeviation();
383
384           // get k (number of standard deviations away from the
                 mean)
385           double k = getK(probabilityThreshold);
386
387           validateChebyshev(valueToCheck, sampleMean, sampleStdDev,
                  k);
388
389           double acceptableDeviation = k * sampleStdDev;
390           double min = sampleMean - acceptableDeviation;
391           double max = sampleMean + acceptableDeviation;
392           double currentDeviation = Math.abs(valueToCheck -
                 sampleMean);
393
394           double outlierScore = currentDeviation /
                 acceptableDeviation;
395           if(outlierScore > 1.) {
396             debugBoth("--------Node:"+nodeId+" isOutlier outside
                    limits--------");
397             debugBoth("outlierScore = " + df.format((outlierScore))
                    );
398             debugBoth(df.format(Math.floor(acceptableDeviation)) +
                    " acceptableDeviation");
399             debugBoth(valueToCheck + " valueToCheck");
400             debugBoth(df.format(Math.floor(min)) + " min");
401             debugBoth(df.format(Math.floor(max)) + " max");
402             debugBoth(df.format(currentDeviation) + "
                    currentDeviation");
403             debugBoth(df.format(sampleStdDev) + " sampleStdDev");
```

```
404             debugBoth(df.format(sampleMean) + " sampleMean");
405             String valuesString = "values: ";
406             for (int i=0; i<timeSeriesData.length;i++){
407               valuesString+=timeSeriesData[i]+", ";
408             }
409             debugBoth(valuesString);
410             debugBoth("--------------------------");
411         }
412         return outlierScore;
413     }
414
415     /**
416      * Calculates the k (number of standard deviations away from
           the mean)
417      * for the specified probability
418      *
419      * @param probability the probability that a minimum of just
           'probability' percent
420      *                    of values within a data set must lie
           within k standard deviations of the mean.
421      * @return the number of standard deviations away from the
           mean for the given probability.
422      */
423     public double getK(double probability) {
424
425         if (Math.abs(probability) > 1) {
426             return 0;
427         }
428
429         double k = Math.sqrt(1. / (1. - probability));
430         return k;
431     }
432
433     /*
```

```
434         * Check if given value is an outlier.
435         * @param valueToCheck the value to check
436         * @param outlierScore the score of the outlier
437         * @return true if value is an outlier. Otherwise false.
438         */
439        private boolean isOutlierByChebyshevsInequality(double
               valueToCheck, double outlierScore) {
440            // value in range of min and max
441            //boolean isOutlier = !((valueToCheck > min) && (
                   valueToCheck < max));
442            //boolean isOutlier = outlierScore > 1.;
443            //return !((valueToCheck > min) && (valueToCheck < max));
444            return outlierScore > 1.;
445        }
446
447        /*
448         * Check the condition to be able to use Chebyshev's
                Inequality Theorem.
449         * @param valueToCheck the given values to check
450         * @param sampleMean   the mean of the data set
451         * @param sampleStdDev the standard deviation of the data set
452         * @param k the number of standard deviations away from the
                mean
453         */
454        private void validateChebyshev(double valueToCheck, double
               sampleMean, double sampleStdDev, double k) {
455            // standard deviations more than 1;
456            if (sampleStdDev <= 1) {
457                //debug("Chebyshev's Inequality does not work,
                       because stdDev < 1");
458            }
459
460            //TODO: Check this
461            //double zScore = new ZScore().getZScore(valueToCheck,
```

```
            sampleMean, sampleStdDev);
462         // as long as the z score/'s absolute value is less than
               or equal to k
463         //if (k >= Math.abs(zScore)) {
464             //throw new Exception(``Chebyshev/'s Inequality wont
                   work, because k < z-score'');
465         //}
466     }
467 }
```

Listing 4: Method to identify malicious nodes. Follows the Kosaraju's Algorithm.

```
468 /*
469  * This is step No 2 in clustering suspicious nodes.
470  * Step No 1 already identified "suspicious' nodes and
471  * created a (sub)graph. Hence, the incoming graph must
472  * be a (sub)graph with only 'suspicious' nodes.
473  * This class will do two things:
474  * 1. identify and separate the incoming graph in
475  * strongly connected components, i.e. identify how
476  * many different neighborhoods of attacked nodes exist.
477  * 2. For each neighborhood, find the 'mother' node,
478  * i.e., node with no incoming edge. This exact node
479  * is the intruder/attacker.
480  *
481  * Not implemented step: the intruders are returned as
482  * such to the controller, which in return communicates
483  * them with all nodes to avoid using them as "parents".
484  */
485
486 public class FindConnectedComponents {
487
488   ConnectedComponents cc = new ConnectedComponents(); /*
         graphstream algo using Kosaraju style */
489
```

```
490   List<Node> motherNodes = new ArrayList<>();
491
492   FindConnectedComponents(){
493   }
494
495   public List<Node> findCC(Graph graph) {
496
497     //printInGraphEdges(graph); /* use for debugging */
498
499     cc.init(graph); /* subgraph of attacked / attacker only */
500     cc.compute();
501     int ccGraphEnum = cc.getConnectedComponentsCount();
502     debug("There are "+ccGraphEnum+" connected (sub)graphs");
503
504     Stream<Node> nodes =graph.nodes();
505     Iterator<Node> n = nodes.iterator();
506     while (n.hasNext()){
507       Node node = n.next();
508       String comp = cc.getConnectedComponentOf(node).toString();
509       comp = comp.substring(comp.lastIndexOf("#")+1); //
             components numbering from zero
510       debug("Node: "+IPlastHex(node.toString())+" belongs to
             component No "+comp);
511
512       if(node.enteringEdges().count() == 0) { /* orphan node,
             hence ATTACKER */
513         debug("Node "+IPlastHex(node.getId().toString())+" has no
               EdgeToward ==> IT IS AN ATTACKER");
514         motherNodes.add(node);
515       }
516     }
517     //TODO: Do we need the edges of the node under attack?
518     return motherNodes;
519   }
```

```
}
```

Listing 5: Method to probe the serial port for cooja connection.

```
/*
 * This class needs total restructuring. It needs
 * more innovative ways (threads?) to probe continuously
 * all possible serial ports for cooja, inputs.
 * It also needs to depict the
 * actual port number chosen.
 * Also, it needs to be reset, and re-probe properly,
 * when cooja is restarted. Right now, the whole
 * controller needs to be restarted, if cooja is
 * restarted.
 * In certain cases (not clear when...) the port is not
 * identified, although correctly probed.
 */

import com.fazecast.jSerialComm.SerialPort;

public class SerialPortProbe {

  SerialPort motePort ;

  public SerialPortProbe() {
    //motePort = getSerialPort();
  }

  public SerialPort returnSerialPort() {
    return motePort;
  }

  //TODO: Rewrite with a while loop
  public SerialPort getSerialPort() {
```

```
552
553   /********* Set & open the serial port
          **************************/
554     if (motePort == null)
555       motePort=findPort("dev/pts/1");
556     if (motePort == null)
557       motePort=findPort("dev/pts/2");
558     if (motePort == null)
559       motePort=findPort("dev/pts/3");
560     if (motePort == null)
561       motePort=findPort("dev/pts/4");
562     if (motePort == null)
563       motePort=findPort("dev/pts/6");
564     if (motePort == null)
565       motePort=findPort("dev/pts/7");
566     if (motePort == null)
567       motePort=findPort("dev/pts/14");
568     if (motePort == null)
569       motePort=findPort("dev/pts/15");
570     if (motePort == null)
571       motePort=findPort("dev/pts/17");
572     if (motePort == null)
573       motePort=findPort("dev/pts/18");
574     if (motePort == null)
575       motePort = findPort("dev/pts/19");
576     if (motePort == null)
577       motePort=findPort("dev/pts/20");
578     if (motePort == null)
579       motePort=findPort("dev/pts/21");
580     return motePort;
581   }
582 /***********METHODS*******************************************/
583   protected SerialPort findPort(String portName ) {
584     try{
```

```
585         /********* Set & open the serial port
                ************************/
586         //debug("Opening port:"+ portName);
587         motePort = SerialPort.getCommPort(portName);
588         motePort.closePort();
589         motePort.setBaudRate(115200);
590         //motePort.setParity(1);
591         motePort.setComPortTimeouts(SerialPort.TIMEOUT_SCANNER, 0,
                0);
592         if(motePort.openPort()==true){
593           debug("Serial Port found: "+motePort.
                 getDescriptivePortName());
594           debug("Baud Rate:"+ motePort.getBaudRate());
595           debug(" Parity:"+ motePort.getParity());
596           debug(" Write-Timeout:"+ motePort.getWriteTimeout());
597           return motePort;
598         } else {
599           debug("Serial Port not found. Check if port number exists
                  in SerialPortProbe.java");
600           debug("Going to sleep for 300ms");
601           Thread.sleep(300);
602           return null;
603         }
604       } catch (Exception e) {
605         debug(e.toString());
606       }
607       return null;
608     }
609 }
```

A.3 ASSET Controller in Action screenshots

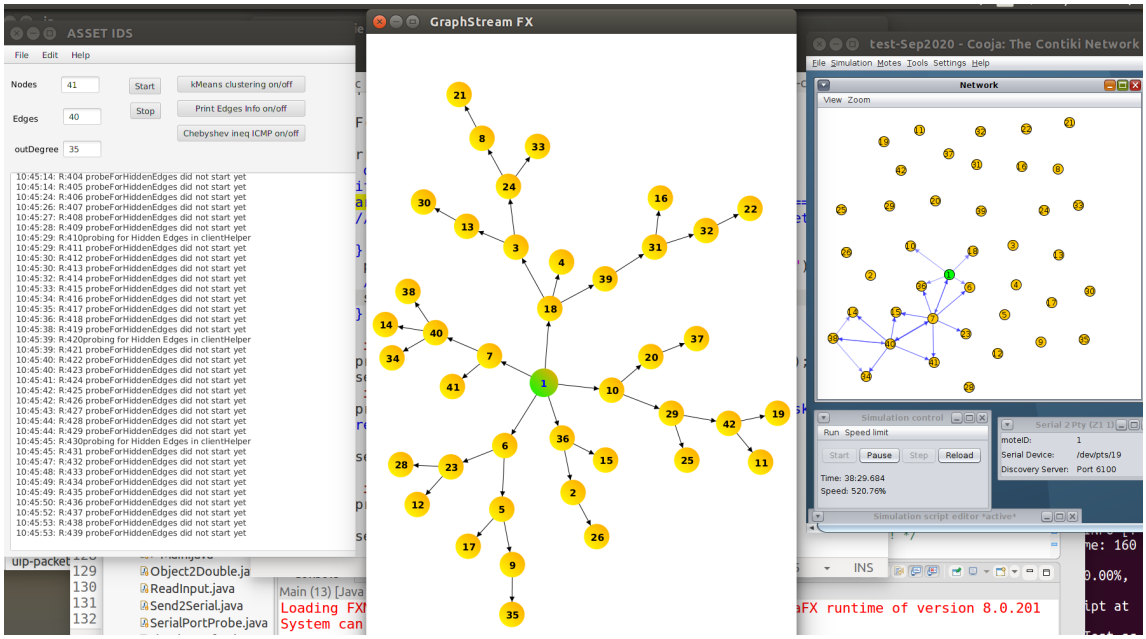


Figure A.39: *ASSET* in action, monitoring a network of 40 nodes.

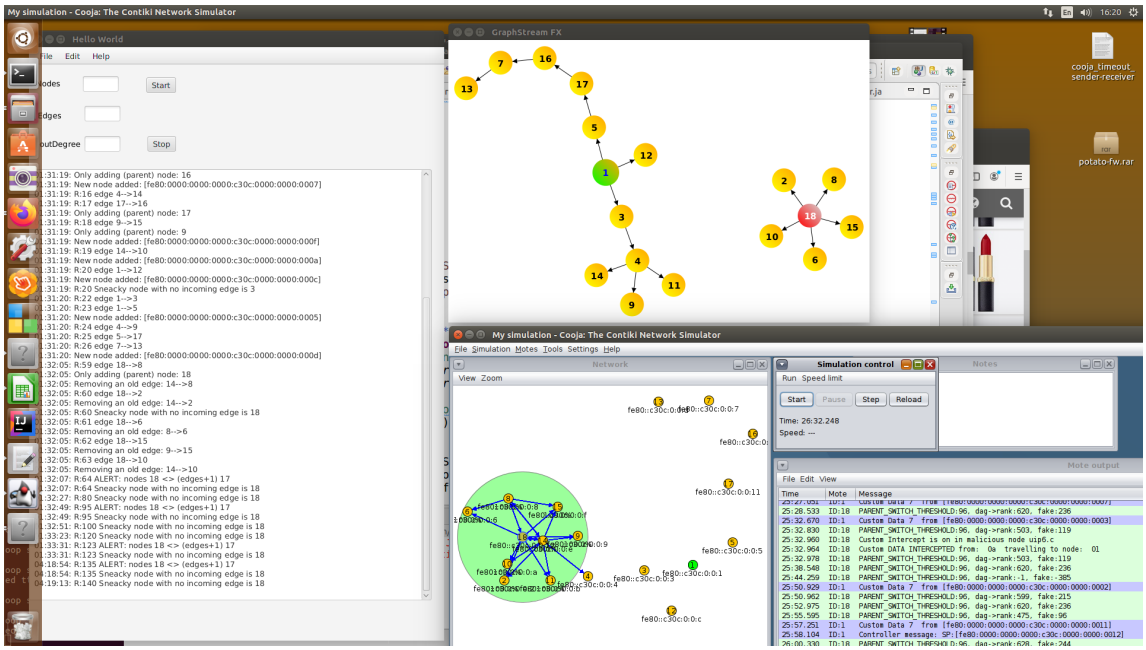


Figure A.40: *ASSET* identifies an outsider. The attacker is not responding to *ASSET*'s commands.

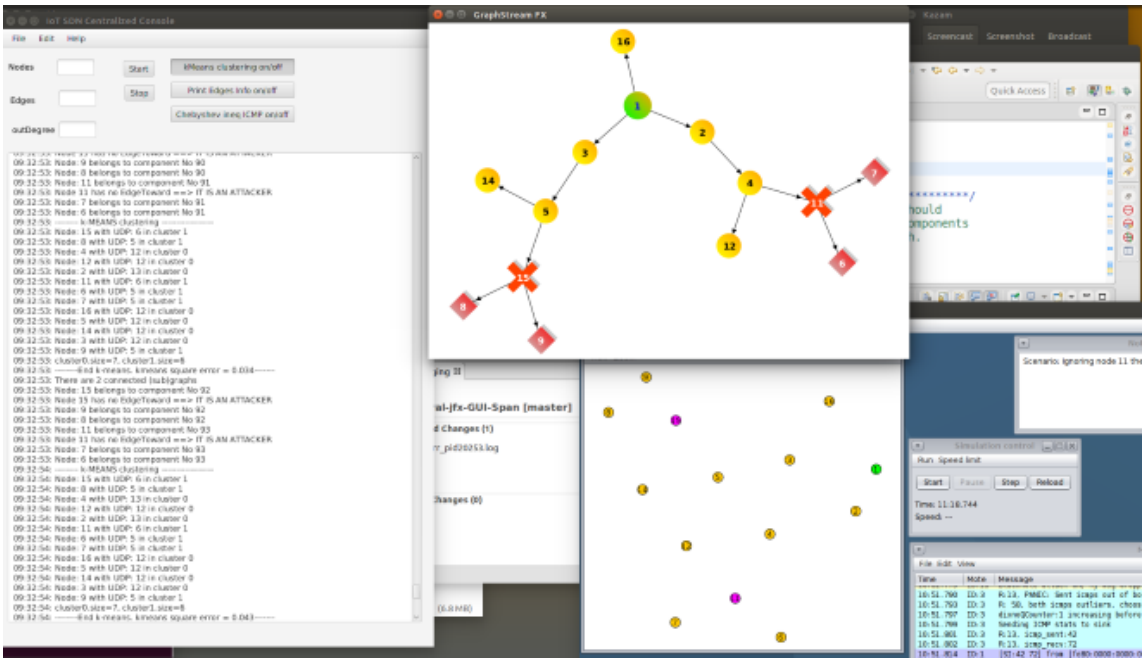


Figure A.41: *ASSET* identifies two concurrent attackers.

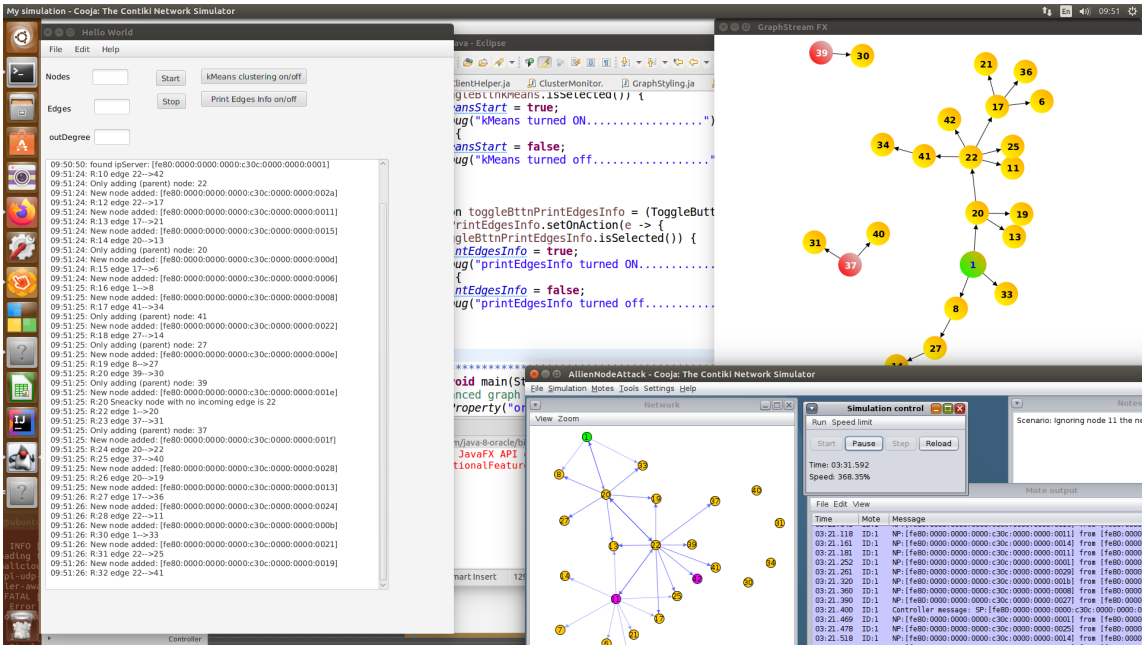


Figure A.42: *ASSET* identifies two attackers, not responding to commands. Sucha attacks, where the intruder does not possess the network's specific firmware, are trivially detected by *ASSET*.