Ad Hoc Networks 109 (2020) 102290

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Analysis of the interplay between RPL and the congestion control strategies for CoAP☆

Carlo Vallati [a,b], Francesca Righetti [a,b,*], Giacomo Tanganelli [a,b], Enzo Mingozzi [a,b], Giuseppe Anastasi [a,b]

[a] Department of Information Engineering, University of Pisa, Italy
[b] Largo Lucio Lazzarino, Pisa, 1, 56122, Italy

ARTICLE INFO

Keywords:
CoAP
Congestion Control
Routing
RPL
WiSHFULshful

ABSTRACT

The Constrained Application Protocol (CoAP) is gaining attention as a standardised RESTful interface for the Internet of Things (IoT). Recent studies have focused on different congestion control strategies for CoAP, in order to ensure proper operation of large-scale IoT deployments. In this paper, we carry out a performance evaluation of different congestion control policies for CoAP in a realistic environment by exploiting WiSHFUL, a platform for large-scale experimentation of network architectures. Our goal is to analyse different congestion control policies and their interplay with the routing protocol in a real environment, where unstable links and route fluctuations are frequent, due to channel variability. The results of our experiments highlight that the dynamics of the routing protocol have a noticeable impact and can impair significantly the performance of the congestion control algorithm. Specifically, the influence of the routing protocol depends on the specific congestion control policy adopted: an aggressive policy that re-transmits messages more frequently, e.g. CoCoA, is more penalised than others, in terms of throughput.

1. Introduction

The future Internet of Things (IoT) will be enabled by a pervasive fabric of sensors and actuators, which will bridge the physical environment to the information systems [1]. In these systems, a significant number of IoT devices will be connected by means of low-power and lossy wireless networks (LLNs) using multi-hop data delivery to ensure rapid and low-cost coverage of large environments.

In order to ensure interoperability, different standardisation activities have been carried out recently. In this framework, the Internet Engineering Task Force (IETF) has standardised a complete protocol suite to integrate IoT devices into the existing IPv6 network architecture. The top of this communication stack comprises the Constrained Application Protocol (CoAP)[2], an application protocol that provides a common RESTful interface for applications to interact with IoT devices. In order to implement multi-hop data delivery, the RPL routing protocol [3] is adopted at the lower layer.

The large-scale deployment of IoT systems will raise a new level of challenges for such network protocols as the limited capabilities of devices and the low available bandwidth in LLNs can lead to frequent congestion. In order to avoid congestion and ensure the proper operation of the network, research efforts have recently focused on proposing and studying congestion control policies. Since CoAP operates on top of the User Datagram Protocol (UDP), which does not implement any congestion control mechanism, specific congestion control policies to be implemented in CoAP have been proposed. Two algorithms in particular have gained attention: the *default* congestion control algorithm defined in the CoAP protocol, and the advanced congestion control algorithm, named *CoCoA*, originally proposed in [4] and now under standardisation within the CORE WG [5] as the reference congestion control algorithm for CoAP. While the default algorithm simply adopts a retransmission strategy based on a random timeout selected from a fixed interval, CoCoA relies on the continuous measurements of the round-trip-time (RTT) to adapt the transmission rate. Many works in

☆ A preliminary version of this paper has been published as Vallati, C., Righetti, F., Tanganelli, G., Mingozzi, E., Anastasi, G. "ECOAP: Experimental Assessment of Congestion Control Strategies for CoAP Using the WiSHFUL Platform." IEEE SMARTCOMP 2018.

* Corresponding author.
E-mail addresses: carlo.vallati@unipi.it (C. Vallati), francesca.righetti@ing.unipi.it (F. Righetti), giacomo.tanganelli@ing.unipi.it (G. Tanganelli), enzo.mingozzi@unipi.it (E. Mingozzi), giuseppe.anastasi@unipi.it (G. Anastasi).

https://doi.org/10.1016/j.adhoc.2020.102290
Received 16 October 2019; Received in revised form 29 July 2020; Accepted 30 July 2020
Available online 1 September 2020

literature have assessed the performance of such congestion control strategies [6–12], the majority of them, however, carries out the performance evaluation by means of simulation.

In this paper we carry out a performance evaluation of the default and CoCoA congestion control strategies by means of real-world experiments. Our goal is to analyse the interplay between the congestion control policy and the routing protocol RPL, when a real environment characterised by lossy links and route fluctuations is considered. To this end, we leveraged two different testbeds, both deployed in an indoor office environment, characterised by two different topologies. The results obtained show that the routing algorithm has a significant impact on the overall performance of the considered congestion control algorithm, decreasing significantly the overall performance, in terms of data delivered. In particular, it emerged that the influence of the routing protocol RPL depends on the specific congestion control strategy: an aggressive strategy, like CoCoA, that re-transmits more frequently is more penalised than others, in terms of throughput, as it can exacerbate the instability of the routing protocol.

Compared with the other works from the literature, this is the first work that specifically focuses on analysing the interplay between the routing protocol and the congestion control policy by means of real experiments. A preliminary version of this work appeared as a conference paper [13], where the experimental setup and a preliminary set of results were presented. Compared to this previous work, this paper has been significantly extended, with an extensive performance evaluation that presents an exhaustive analysis of the interplay between RPL and the different congestion control policies.

The remainder of the paper is organised as follows: in Section 2 we analyse the related work, in Section 3 we offer an overview of the technical background, in Section 4 we present the experimental scenario adopted for our experiments, in Section 5 we discuss the experimental results, in Section 6 we wrap up the lessons learned from our experimental analysis, finally in Section 7 we draw the conclusions.

2. Related work

Many recent works have focused on evaluating the performance of CoAP default and CoCoA congestion control algorithms [6–12,14]. The majority of these works are based on simulations [7–12,14], even though some of them also exploit real experiments [6,15]. Although the previous works have already assessed the performance of both CoCoA and the default congestion control algorithms, they have focused on evaluating the performance of the congestion control algorithm in isolation, i.e., without considering other system components. In this work, instead, we mainly focus on studying the interplay between the congestion control algorithm and the routing protocol in a real environment, where unstable links and route fluctuations are frequent. Although the issues caused by RPL instability in real experiments, especially at high loads, are known [16–18], to the best of our knowledge this is the first work that evaluates the effects of such dynamics explicitly on the performance of the congestion control algorithms.

Other works have focused on assessing the performance of CoCoA as advanced congestion control strategy. Such works show that CoCoA can improve the performance in terms of network throughput, compared to the default CoAP algorithm and other mechanisms, as it can better handle message retransmissions in LLNs. In [6], for instance, CoCoA is evaluated in a large-scale IoT scenario in which GPRS is employed to connect IoT nodes. The authors compare CoCoA against other congestion control algorithms originally proposed for TCP. The results obtained show that CoCoA performs equally, or better, than the other considered algorithms. Similar conclusions are drawn in [7], where the authors compare CoCoA to other TCP-based congestion control mechanisms on an emulated Zigbee network. In [8], the authors evaluate CoCoA by means of simulations with different traffic patterns. The authors highlight the poor performance of CoCoA with bursty traffic due to the improper selection of the retransmission timeout value.

Another group of works, instead, focuses on highlighting issues and shortcomings of CoCoA and propose modifications or enhanced versions of the algorithm. In [12] the authors carry out a performance evaluation of CoCoA by means of simulations in which a set of CoCoA shortcomings and pitfalls is highlighted. In order to overcome CoCoA limitations, a modified version, named pCoCoA, is proposed. The simulation results show that the proposed version improves the performance of the original version. In [9] the authors propose 4-states-strong, a modification to CoCoA that introduces a more complex RTT estimator to discriminate (and handle differently) losses due to transmission errors and losses due to congestion. The results show an improvement in the performance when the loss rate is particularly high. In [10], instead, the authors propose CoCoA-E, a modified version of CoCoA based on the Eifel retransmission timer. Simulations show an improvement of the retransmission timeout (RTO) estimation under two different RTT functions: *stair-step* and *saw-like*. In [14] an RTT-based congestion control is proposed and compared against CoCoA by means of simulations. The proposed approach analyses the growth of the RTT variance to detect network congestion and regulate retransmissions. Finally, in [11] the authors evaluated CoCoA-S, a version of CoCoA that uses only the strong RTO estimator. The variant is evaluated against other TCP congestion control algorithms. The results, however, confirm that CoCoA provides better performance compared to CoCoA-S and the other considered congestion control strategies.

3. Technical background

In this section, we provide an overview of the IoT protocols involved in our analysis. Specifically, we first describe the CoAP protocol along with the default and CoCoA congestion control strategies, and then we introduce the RPL routing protocol.

3.1. CoAP

CoAP is a lightweight RESTful application protocol designed for constrained devices. It inherits the same client/server paradigm adopted in HTTP, however, it runs over UDP in order to be as lightweight as possible. CoAP requests are exchanged between clients and servers and each request is sent by a client to ask for an action on a specific resource hosted on the server. In detail, actions are specified through the same basic set of methods used by HTTP (GET, POST, PUT and DELETE), whereas resources are identified by means of URIs. Clients and servers are commonly known as CoAP endpoints, as IoT applications typically results in the exchange of client and server roles.

From an architectural point of view, CoAP can be seen as a two layer protocol: a request/response sub-layer and a message sub-layer, respectively. The request/response sub-layer, handles CoAP requests by means of the execution of the required method on the requested resource. The message sub-layer, instead, deals with UDP and the asynchronous nature of the interactions. Specifically, the message layer is responsible for managing the message exchange, also called CoAP transaction, between the endpoints over UDP. It implements duplicate detection, through a message identifier field (MID for short) contained in every message, and reliable delivery if enabled. CoAP defines four different types of messages, namely *Non-Confirmable* (NON), *Confirmable* (CON), *Acknowledgement* (ACK) and *Reset* (RST). When unreliable message delivery is enabled, NON messages, which do not require confirmation of reception, are employed. When reliable delivery is enabled, requests/responses are transported within CON messages that, instead, require an acknowledgement through ACK messages. The latter can be sent in two alternative modes: *separate* and *piggyback*. In separate mode, the receiver sends an empty ACK message immediately after the reception of a request, and defers sending the actual response in a separate CON message, when it is ready. In piggyback mode, instead, the receiver waits for the response to be ready and sends it back directly, encapsulated in the body of the ACK message. The sender is responsible

for retransmitting messages that are not acknowledged, after a timeout, until a maximum number of retransmission occurs. Finally, RST messages are exploited, instead of ACKs, when the recipient is unable to process a CON message.

As example, in Fig. 1 we present two examples of CoAP transactions. The one on the left side successfully terminates after the first transmission, as both the CON and ACK messages are successfully received, while the one on the right side shows a transaction that involves a retransmission, as the initial CON transmission is corrupted.

3.1.1. Default congestion control

Considering that UDP does not provide any congestion control mechanism, CoAP specifications introduce a default congestion control algorithm that regulates the amount of data transmitted by the sender. The default congestion control included in CoAP specifications imposes a restriction on the number of parallel message exchanges and on the transmission rate of outgoing messages. Firstly, the algorithm fixes the maximum number of outstanding CoAP transactions. This value, named NSTART, specifies the maximum number of unacknowledged CoAP messages that can be transmitted, thus limiting the concurrent number of messages that can be sent by a node without receiving an acknowledgement. Secondly, the algorithm limits the transmission rate of outgoing messages by using an exponential back-off between retransmissions of lost messages. Specifically, for a new CON message, a retransmission timeout (RTO) is randomly picked from the following interval:

$$RTO = [T_{ack}, T_{ack} \cdot T_{rand}] \tag{1}$$

where T_{ack} is the acknowledgement timeout, set by default to 2000 ms, and T_{rand} is a randomisation factor that is set by default to 1.5. For each subsequent retransmission, a binary exponential backoff is applied to increment the RTO. Specifically the RTO interval for the $i-th$ retransmission is set by doubling the last RTO value:

$$RTO_i = RTO_{i-1} \cdot 2 \tag{2}$$

Finally, CoAP allows a total of four retransmissions for a CoAP message, before considering the message as a failure.

3.1.2. CoCoA congestion control

The simple strategy implemented by the default congestion control algorithm, has been demonstrated to fail in avoiding network congestion under certain circumstances [19]. For this reason, many new advanced congestion control algorithms have been proposed recently in literature. Among them, the CoCoA algorithm, originally proposed in [20] and further extended in [4], has gained significant attention and it is currently under standardisation within the CoRE Working Group [5] as the reference congestion algorithm for CoAP. In this paper we consider the last version of the CoCoA algorithm, named CoCoA+ [4]. For the sake of simplicity in the remainder of the paper we refer to it simply as CoCoA.

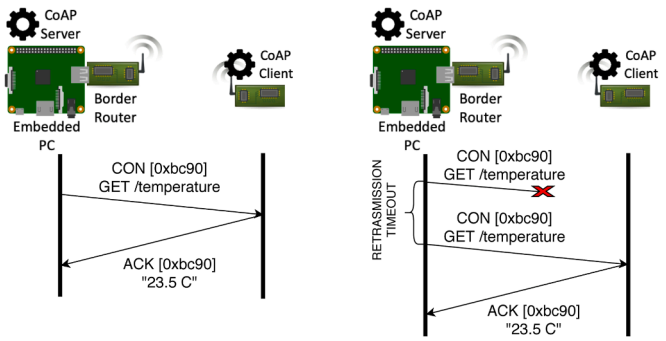

Fig. 1. CoAP transaction examples.

The CoCoA algorithm is composed of the following three main functions:

- A policy to calculate the retransmission timeout (RTO);
- A back-off policy to set the RTO for retransmissions;
- An ageing policy for the status information.

In order to calculate the RTO, CoCoA adopts the same algorithm used by TCP, which updates the RTO value based on measured Round Trip Times (RTTs), and considers only measurements on messages correctly delivered without retransmissions. In the context of lossy networks, however, many transactions are expected to experience retransmissions, thus reducing the probability of obtaining valid RTT measurements. For this reason, CoCoA considers also transactions that experienced retransmissions to obtain a more accurate RTT estimation. Specifically, two RTO values are calculated: a strong RTO, estimated using RTT samples from transactions that did not require any retransmission, and a weak RTO, estimated using RTT values of transactions that required no more than one re-transmission. It is important to highlight that the sender cannot identify the actual retransmission that generated an ACK message. For this reason, RTT samples are collected measuring the time between the first transmission and the arrival of the related ACK, even though this measurement may include multiple transmissions.

For each destination, a node maintains the following quantities:

- Two RTT estimators: RTT_{strong} and RTT_{weak};
- Two variance estimators, called $RTTVAR_{strong}$ and $RTTVAR_{weak}$;
- Two RTO estimators, RTO_{strong} and RTO_{weak}, derived from the strong and weak RTT estimators, respectively;
- A comprehensive RTO, namely, $RTO_{overall}$, which keeps track of both RTO_{strong} and RTO_{weak} changes.

Initially, the RTO estimators are initialised with a default value of 2s. The value of RTT_x and $RTTVAR_x$ ($x \in \{strong, weak\}$) are initialised when the first RTT value R is measured, as follows:

$$RTT_x \leftarrow R, \qquad RTTVAR_x \leftarrow \frac{R}{2} \tag{3}$$

Every time a new RTT sample R is measured, the corresponding strong or weak estimators (based on the number of retransmissions) are updated as follows:

$$RTT_x = (1-\alpha)RTT_x + \alpha R \tag{4}$$

$$RTTVAR_x = (1-\beta)RTTVAR_x + \beta|RTT_x - R| \tag{5}$$

$$RTO_x = RTT_x + K_x RTTVAR_x \tag{6}$$

$$RTO_{overall} = \lambda_x RTO_x + (1-\lambda_x)RTO_{overall} \tag{7}$$

where the following values are recommended: $\alpha = 0.25$, $\beta = 0.125$, $K_{strong} = 4$, $K_{weak} = 1$, $\lambda_{strong} = 0.5$, and $\lambda_{weak} = 0.25$. $RTO_{overall}$ is used to set the initial RTO (RTO_{init}) for the next CON transmission. The actual value is selected using a dithering approach, i.e., RTO_{init} is randomly chosen from the interval $[RTO_{overall}, 1.5 \cdot RTO_{overall}]$.

In case of retransmissions, CoCoA relies on a backoff mechanism to compute the retransmission timeout. Differently from the default congestion control algorithm, in which the RTO is doubled, CoCoA computes the new RTO value for retransmissions according to a variable backoff factor (VBF) that depends on the initial RTO value (RTO_{init}). Specifically, the new value of RTO for retransmissions (RTO_{new}) is evaluated as follows:

$$RTO_{new} = RTO_{previous} * VBF(RTO_{init}) \tag{8}$$

The VBF factor is set as a function of RTO_{init} to avoid frequent retransmissions in a short time, when the RTO value is low, and long delays in

retransmissions when the *RTO* value is large. Specifically, the formula adopted is the following:

$$VBF(RTO_{init}) = \begin{cases} 3 & RTO_{init} < 1s \\ 2 & 1 \leq RTO_{init} \leq 3s \\ 1.3 & RTO_{init} > 3s \end{cases} \quad (9)$$

Finally, CoCoA introduces a mechanism to age *RTO* values when *RTT* updates are not received for a certain time. The rationale is that the *RTO* estimation becomes obsolete after a certain time and should converge towards the initial value. Specifically, if $RTO_{overall}$ is larger than the minimum *RTO* value adopted in the default CoAP congestion control algorithm, which is set by default to 2s, and it is not updated for more than 30s, when a new measurement is obtained the following formula is applied:

$$RTO_{overall} = \frac{2 + RTO_{overall}}{2} \quad (10)$$

If $RTO_{overall}$ is, instead, less than 1s, and it is not updated for a time that is 16 times its actual value, $RTO_{overall}$ is reset to 1s.

Although the efficacy of CoCoA is still argued [19], and multiple alternative definitions have been proposed recently [21], it is widely recognised as the most promising congestion control strategy to handle the shared nature of the wireless medium, which results in collisions of transmissions, and to cope with the limited buffer size of IoT devices, which often produces buffer overflows.

3.2. RPL

We conclude this section on Technical Background by providing an overview of the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL).

RPL [3] is an IPv6 distance-vector routing protocol specifically designed to support resource-constrained embedded devices and handle lossy wireless networks. Specifically, the protocol takes into account the unreliable nature of wireless communication and the limited available power of devices by minimising the memory requirements and the complexity of routing functionalities and reducing the signalling overhead.

The protocol design assumes that the majority of the application traffic is upward, i.e., generated by nodes and directed towards a single collector node. Downward traffic, i.e., generated by the collector towards other nodes, is assumed to be sporadic, while node-to-node interactions are considered rare. For this reason, RPL builds and maintains a logical topology for upstream data delivery, while downward routes are established when required. Specifically, the topology is a *Destination Oriented Directed Acyclic Graph (DODAG)*, in which every node selects a set of neighbours, called parent set, as candidate neighbours for upstream data delivery. Unlike other Directed Acyclic Graph, every DODAG is rooted in a single node, the DODAG root, that is the collector of the network to which upstream data is directed. The DODAG root triggers the RPL topology formation by emitting DODAG Information Object (DIO) messages. Non-root nodes listen for DIOs and use the included information to join the DODAG. Upon joining the DODAG, a node starts emitting DIOs to advertise its presence.

Each node is assigned a rank, which is a scalar measure of its distance from the root. Rank is advertised in DIOs and it is used to select the neighbours to be included in the parent set. To avoid loops in the topology, a node can include only nodes with lower rank in its parent set. As result, the rank monotonically decreases along an upward path towards the root. Every node evaluates the rank according to an *Objective Function (OF)*. The OF is responsible for implementing routing constraints and optimisation objectives, evaluating the rank using specific metrics such as link quality, number of hops, delay, etc. The OF also specifies how the parent set is formed and how the preferred parent, the node from the parent set exploited for data forwarding, is selected. Several OFs have been defined with different objectives, e.g., to reduce route flaps or minimise energy consumption. Among them, it is worth to mention the Minimum Rank with Hysteresis OF (MRHOF for short) [22] that adopts a hysteresis mechanism to reduce routing instability in response to small metric changes.

In order to support also downward traffic, nodes generate on-demand a *Destination Advertisement Object message* (DAO for short) which propagates destination information upward in the DODAG. DAO messages are sent as unicast messages up to the root node and processed by intermediate nodes.

In order to recover from routing inconsistencies that can be caused by the loss of control messages, the RPL standard defines a procedure, called *Local Repair*, that is triggered by a node every time the current preferred parent is reset, since it becomes unreachable or unacceptable as preferred parent. When the procedure is triggered, the node resets its preferred parent and starts looking for a new preferred parent among its neighbours.

4. Experimental setup

In this section we present the methodology and the setup adopted in our experiments. The scenario we considered is shown in Fig. 2: a typical IoT system that comprises a wireless sensor network connected to the internet through a Border Router. Sensors are assumed to be equipped with a low-power wireless transceiver, e.g. an IEEE 802.15.4 radio, for low-power wireless communication. The standard protocol stack defined within IETF is adopted, i.e. the sensors leverage IPv6 for communication through the 6LoWPAN adaptation layer and the RPL routing protocol to implement multi-hop communication capabilities.

In order to report the data to a collector (i.e., the Border Router), each sensor behaves as a CoAP client, i.e., it periodically issues a request (e.g., a POST request) to a CoAP server with its data in the payload. The CoAP server is assumed to be external to the sensor network and executed on a powerful host, e.g., an embedded system or a cloud server, which executes a fully functional operating system, e.g., a Linux operating system. For instance, in our experiments, the CoAP server runs on an embedded system which connects to the wireless sensor network through a sensor node that is configured as the border router.

The experiments are run exploiting WiSHFUL [23], a platform for rapid experimentation on network testbeds. The platform, developed in the framework of the H2020 European Project WiSHFUL[1], allows to implement and run a large number of experiments on different network testbeds in a rapid and reliable manner. Specifically, the platform allows to easily configure the network devices and program their behaviour by defining a control software, running on a controller node. For the sake of brevity, an introduction on the WiSHFUL platform and the details of the implementation of our experiments on top of it are omitted here and reported into the Appendix, which the interested reader can refer to.

Experiments are executed on two different testbeds, characterised by two different network topologies: the Pisa IoT Testbed (PINT) [24] and the WiLAB testbed (WiLAB) [25]. The PINT testbed is deployed in the two-floor building of the Department of Information Engineering at the University of Pisa, within offices, laboratories and classrooms with several sources of interfering signals, including co-located WiFi networks. The core infrastructure of the testbed is a Wireless Sensor Network (WSN) composed of 22 nodes installed in different rooms, as shown in Fig. 3. The nodes are installed in spots where power and network connectivity are available, thus resulting in an irregular topology with different distances between nodes. Each node is composed of a Raspberry PI II to which one Zolertia RE-Mote sensor[2] is connected. All the nodes are connected through an Ethernet LAN to a server that is used to manage and control experiments. The controller, installed in the

[1] http://www.wishful-project.eu/ .
[2] https://github.com/Zolertia/Resources/wiki/RE-Mote .

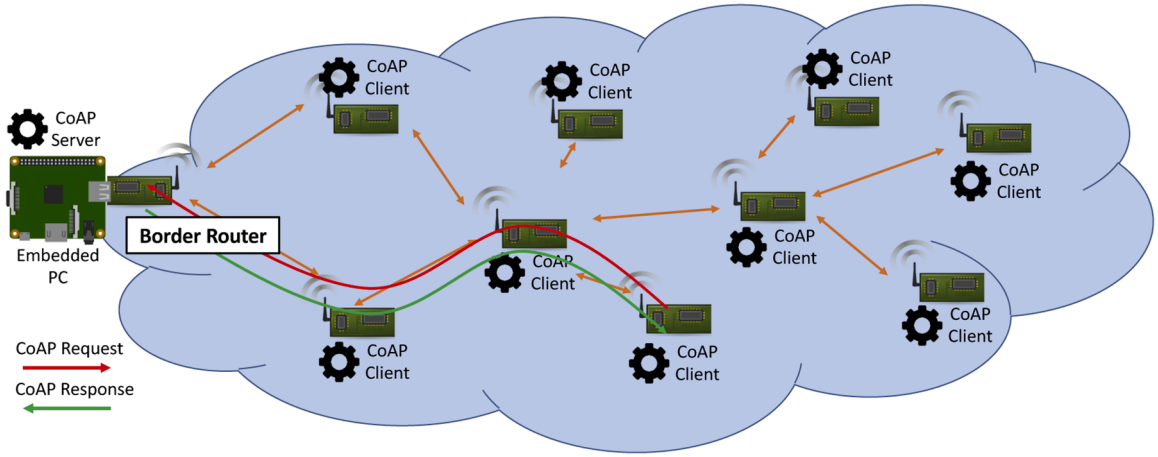

Fig. 2. Experiment scenario.

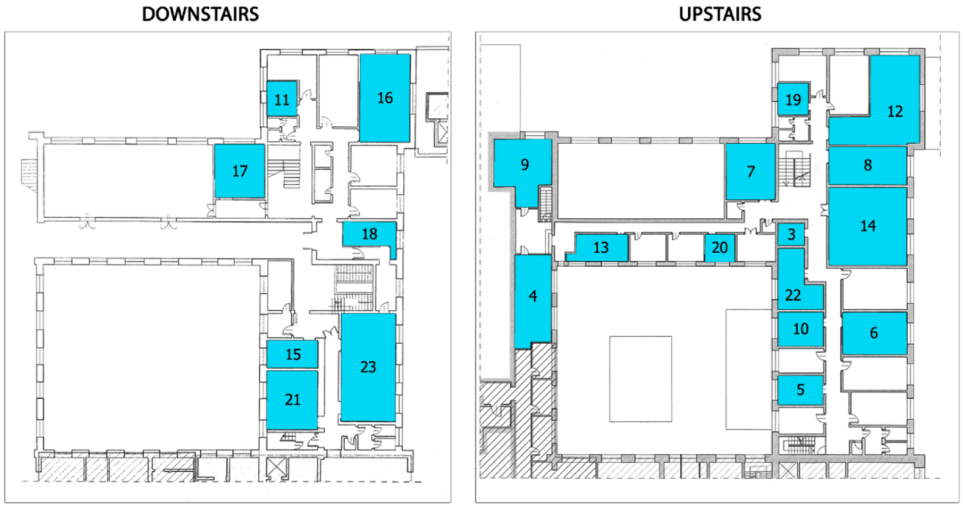

Fig. 3. PINT Testbed map.

same room of node 21, has also attached one Zolertia RE-Mote, which is programmed as Border Router to bridge the WSN with the CoAP server that is hosted directly on the controller.

The WiLAB testbed is an IoT testbed deployed in the iGent building at Gent, Belgium. A set of IoT nodes composed of an Intel NUC mini PC and a Zolertia RE-MOTE is deployed on all the floors of the building. In this testbed, nodes are installed with a more regular topology, as can be seen in Fig. 4. For our experiments we exploited the nodes deployed at Floor 11, an office environment in which 26 nodes are deployed. Among them, a node, namely node 11–25, runs the controller and hosts the CoAP server. In order to connect the WSN with the CoAP server, the RE-MOTE attached to the controller is programmed to behave as RPL root node.

Each sensor in both the considered testbeds runs the Contiki Operating System (OS), a popular operating system for constrained devices. Each device running an instance of the Contiki OS is re-programmed and configured by exploiting the capabilities of the WiSHFUL platform. The traffic considered in our experiments is a CoAP traffic generated by each node at a constant rate. Specifically, each CoAP client periodically sends a confirmable POST request towards the CoAP server, which is implemented using CoAPthon [26] and deployed on the testbed controller. For each successfully received request, the server replies with a response piggybacked onto a CoAP ACK message. Before starting to transmit CoAP requests, CoAP clients wait for a random time selected with uniform distribution between 240s and 270s. This warm-up time, of at least 4 min, is introduced in order to allow the routing protocol to stabilise,

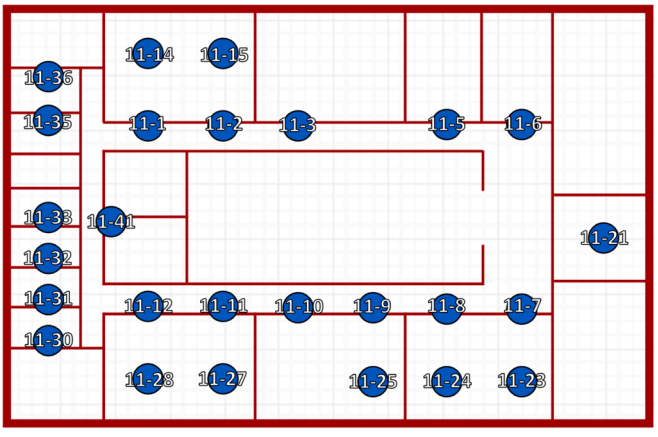

Fig. 4. WiLAB Testbed map.

and to avoid synchronisation effects among different clients. Then, nodes start generating requests with a fixed period T, which is equal for all the clients. Two different set of periods T are considered for the experiments on PINT and WiLAB, respectively, in order to take into account the different number of nodes that comprise the testbed, thus to obtain an Offered Load that is comparable.

The following performance metrics are collected and analysed using the WiSHFUL framework:

- *Carried Load (Bytes/s)*: The overall amount of data successfully delivered at the server per time unit. It includes *all* the traffic originated from *all* the clients in the network. This metric, widely adopted in literature, measures directly the efficacy of a congestion control algorithm in avoiding congestion without limiting the transmission rate.
- *Number of CoAP transmissions (# of transmissions)*: The total number of CoAP transmissions (including retransmissions) issued by *all* the clients in the network for *all* the transactions. This metric measures the aggressiveness of a congestion control strategy in issuing transmissions.
- *Dropped Packets (# of packets)*: The total number of packets dropped at the network layer due to buffer overflows, originated by *all* the clients of the network. This metric is selected as direct measure of the amount of packets dropped at the network layer due to the unavailability of a route for data transmission. It is worth to highlight that in the Contiki OS implementation considered in our experiments, at the network layer packets are dropped only due to the unavailability of a route and not due to congestion. This is due to the fact that Contiki OS exploits a shared buffer for TX/RX operations across all the layers. Consequently, when a default route is available on the routing table, packets are stored in the shared buffer and forwarded directly to the MAC layer for transmission, after the computation of the route. Instead, when a default route is not available, packets are still stored in the shared buffer, but they are handled at the network layer waiting for the routing protocol to select a preferred parent. When the routing protocol takes time to select the preferred parent, the buffer gets full and new packets may be discarded.

In addition to the above-mentioned metrics, in the following we will also analyse other indices, such as RPL Rank and CoAP RTO over time, in order to get an insight on the behaviour of the network and give an explanation on some specific trends.

The network and RPL parameters used in our experiments are summarised in Table 1. As shown in the table, different values for some of the parameters were considered, in particular for those parameters that are expected to have a major impact on the network behaviour. For the sake of brevity, however, only a subset of the scenarios is presented in this paper, as the conclusions drawn from this subset are confirmed in all the considered scenarios. In Table 1 we highlight in bold the values of the parameters considered in this paper.

As the Objective Function (OF), in our experiments we considered both MRHOF and OF0. However, we report only the results obtained with MRHOF, as it is the most used OF, since it ensures higher performance and a more stable route selection, with respect to OF0 [27]. The Expected Transmission Count (ETX) is adopted to measure the *link cost metric*. ETX, defined in [28], represents the number of transmissions required to successfully deliver a packet at the MAC layer. Its value measures directly the cost of transmitting a packet: the lower the link cost, the higher the convenience in selecting a path that includes that link. ETX is the default link cost metric adopted in MRHOF [22] to assess the cost of using a link and find the minimum cost path towards the root node. Considering that the ETX value can fluctuate significantly, e.g., due to sporadic interference or collisions caused by simultaneous transmissions, the average ETX value, measured using an Exponential Weighted Moving Average, is usually adopted as link cost metric. Finally, the routing protocol is configured to work in non-storing mode. This mode has been selected, instead of the storing mode, since (i) it is more lightweight and, hence, can be supported also by very constrained devices with scarce memory and computing capabilities, and (ii) it ensures that upward and downward routes have the same path, except for some transitory periods after a preferred parent change.

The above-mentioned configuration of the routing protocol results in two different topologies for each testbed. In Fig. 5 and Fig. 6 we report the number of hops towards the root node for each node in the PINT and WiLAB testbeds, respectively. The topology is derived running an ad-hoc experiment without traffic on both the testbeds. As can be seen, although the two testbeds includes a similar number of nodes, they result in two different topologies due to the different arrangement of the nodes. Specifically, the topology is more deep in the PINT testbed, with nodes more distant from each other, while it is more dense in the WiLAB testbed. This difference is also confirmed by the number of neighbours for each node, reported in Fig. 7 and Fig. 8: WiLAB, which has a more dense topology, results in a higher number of neighbours, as opposed to PINT that is characterised by a lower number of neighbours for each node.

Increasing values of the generation period T are considered in order to inject in the network an increasing amount of data. This allows to assess the performance of the default algorithm and CoCoA under different network loads. The specific generation periods used for the experiments in both testbeds are reported in Table 1. It may be worthwhile recalling that nodes start injecting messages in the network not earlier than minute 4, in order to ensure that the routing protocol stabilises.

Each experiment is run for 15 min. Longer experiments, up to 1 h, were also run on a subset of the scenarios in order to confirm the results. Their presentation, however, is omitted for the sake of brevity as they led to the same conclusions. In order to obtain statistically sound results, for each scenario at least 5 independent replicas are run. When required by the high variability of a specific scenario, e.g. when high traffic rates

Table 1
Operating parameters .

Parameters	Values
Routing Protocol	**RPL**
RPL Objective Function	**MRHOF**, OF0
RPL Link Cost Metric	**ETX**
RPL Max Link Cost Metric	**4**, 10
RPL DIO interval doublings	**20**
CoAP request size	**35 byte**
CoAP requests buffer size	**4 packets**
CoAP NSTART	**1**
MAC Layer	**802.15.4 CSMA/CA**
MAC Buffer Size	**8 packets**, 4 packets
MAC level max retransmissions	**8**
PINT Nominal Offered Load (B/s) (Generation period in s, T)	**7350 (0.1), 2940 (0.25) 1470 (0.5), 735 (1), 368 (2) 245 (3),184 (4), 147 (5), 74 (10)**
WiLAB Nominal Offered Load (B/s) (Generation period in s, T)	**8750 (0.1),1750 (0.5), 875 (1) 438 (2), 291 (3)**

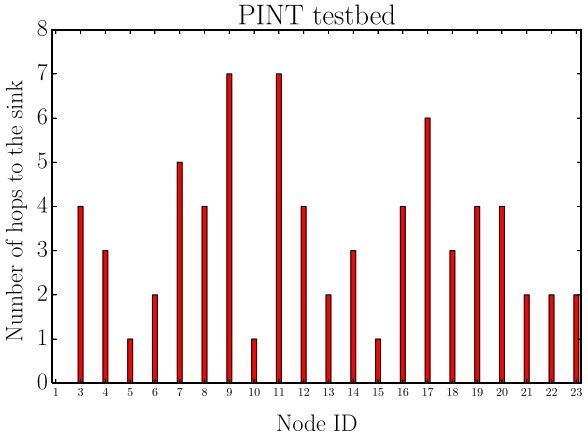

Fig. 5. Number of hops towards the sink for each node. PINT testbed.

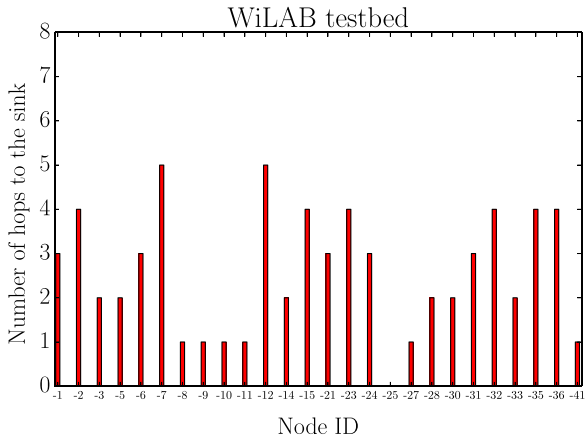

Fig. 6. Number of hops towards the sink for each node. WiLAB testbed.

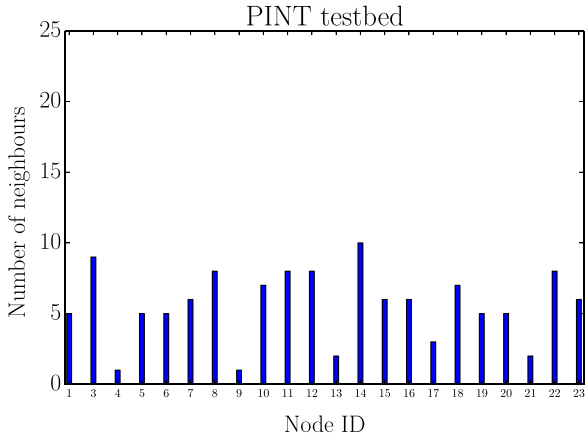

Fig. 7. Number of neighbours for each node. PINT testbed.

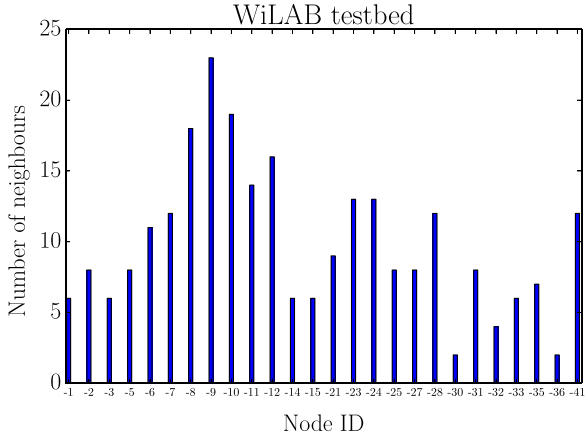

Fig. 8. Number of neighbours for each node. WiLAB testbed.

are considered, the number of replicas is increased up to 10. For each metric the average value and the 95% confidence interval are computed.

5. Results

In the following we present the results of our experimental analysis. As anticipated, we performed two different set of experiments, using two different and independent testbeds. Below, we first analyse the results from the PINT testbed (Section 5.1) and then we discuss the results from the WiLAB testbed (Section 5.2). Finally, in Section 6 we wrap up the lessons learned from our experimental study.

5.1. PINT testbed

In Fig. 9 we show the Carried Load versus the nominal Offered Load with the two considered congestion control policies. The nominal Offered Load is defined as the total amount of traffic injected by all the nodes in the network, expressed in Bytes/sec (B/s). It depends on the packet generation period T, as shown in Table 1. As expected, initially, the Carried Load increases as the nominal amount of traffic injected in the network increases. Up to an Offered Load of 368 B/s the increase in the Carried Load is proportional to the increase in the Offered Load, showing that the network is capable of handling the traffic injected, irrespective of the congestion control policy used. At higher Offered Loads, the Carried Load still increases, however, the growing rate is lower, as the network saturates and does not have sufficient resources to handle all the traffic injected. Eventually, the growth stops and, then, a significant drop in the Carried Load is observed. As shown in Fig. 9, up to 368 B/s, both the considered congestion control policies result in a similar Carried Load, while they exhibit a different behaviour at higher loads. Specifically, the default congestion control algorithm results in a higher Carried Load than CoCoA. As it will be discussed in Section 6, this result is not in agreement with previous results presented in literature [6,12] showing that CoCoA usually ensures a higher Carried Load in the majority of the considered scenarios.

The Carried Load measures the total amount of data successfully delivered to the destination. Therefore, it represents a holistic metric that indicates not only how well the congestion control algorithm performs, but also how the whole network behaves. Specifically, the resulting Carried Load is influenced not only by the congestion control algorithm but also by other network protocols, first and foremost the routing protocol, which is well known to influence significantly the performance of the network [29]. In order to evaluate how RPL directly affects the network performance and, consequently, how its dynamics affect the performance of the considered congestion control algorithm, in the following we get an insight on the results, taking a look at RPL metrics. In Fig. 10 we show the total number of times the preferred parent has been reset, by any node, during each experiment. A preferred parent is reset by a node every time a route inconsistency occurs. Among the inconsistencies that can trigger the reset, there is a steep increase in the rank above a certain threshold, named MaxRankIncrease [3]. When the amount of data traffic injected into the network increases, the number of resets increases accordingly. This is because the large amount of data transmitted in the network increases the number of collisions,

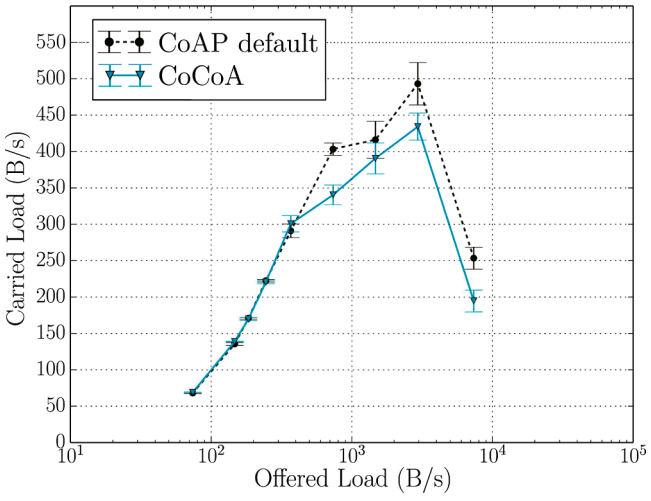

Fig. 9. Carried Load (B\s) with RPL.

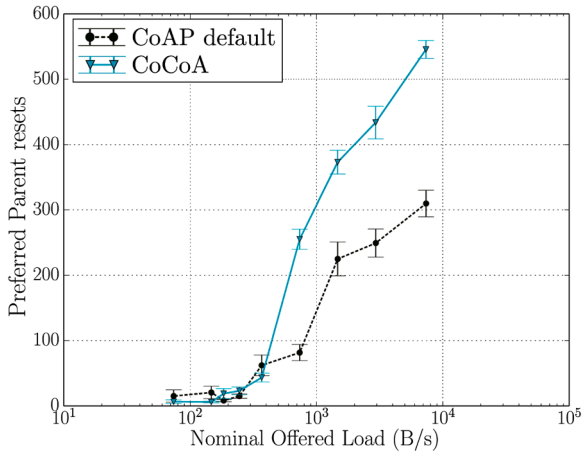

Fig. 10. Overall preferred parent resets with RPL.

thus increasing the likelihood that the overall ETX grows up. Since the rank is calculated as a function of the ETX towards the preferred parent, an increase in the ETX values towards the neighbours may increase the instability of the routing protocol, as explained below.

Every time the preferred parent is reset, the sensor triggers a *local repair* procedure, that aims at reconstructing the proper routing information and reacquiring the correct preferred parent. During the execution of this procedure, however, the node cannot forward its data packets upward. Consequently, data is buffered and may be dropped, if the buffer is full. As can be seen from our experimental results, CoCoA introduces a significantly higher number of resets than the CoAP default algorithm. Moreover, as the Offered Load increases, the difference between CoCoA and the default algorithm increases noticeably. This behaviour, as analysed in the following, can be explained by considering the different policies adopted to update the RTO.

In order to quantify the impact of the preferred parent resets in the network, we measured the overall time during which a sensor operates without a preferred parent, i.e. the time spent executing the local repair procedure, during which a node cannot forward data packets towards its destination. Fig. 11 shows the distribution of this time. As can be seen, with CoCoA there are some sensors that operate without a valid preferred parent even for more than 10 min. With the default policy the maximum time spent without a preferred parent is 7 min. Such long periods without a valid parent, can be explained by the occurrence of multiple local repair procedures in a single experiment, caused by the instability of the routing protocol. During a local repair a node may remain without any preferred parent even for some min. In some extreme situations, it might also happen that a node initiates a local repair procedure that never terminates. This particular behaviour occurs when a node resets the current preferred parent and, afterwards, it is unable to find a new acceptable parent among its neighbours. Every time a node is looking for a new preferred parent, it evaluates the status of all its neighbours, in order to select the most convenient one, according to its OF. In order to avoid the selection of neighbours characterised by poor links, a maximum link cost threshold is introduced in many OFs, (including MRHOF used in our experiments), to ensure that neighbours with very low-quality links are not selected.

In order to provide an insight on the instability of the routing protocol, as well as on the dynamics that lead to a local repair procedure with CoCoA, in the following we report some of the results obtained on a specific node, over time, in a single experiment. To this aim, we selected node number 14, a central node in the PINT network with many neighbours. In this node we observed a significant fluctuations of the routes over time and a local repair procedure that eventually lead to its isolation.

In Fig. 12 we report the RPL rank over time. As it can be seen, node 14 changes frequently its preferred parent, looking for the most convenient neighbour for data forwarding. At a certain point, i.e., right before minute 10, the node resets its preferred parent and triggers a local repair procedure, which however will never terminate, thus leaving the node without a preferred parent for the reminder of the experiment. To better understand this behaviour, in Fig. 14 we report the ETX measured by node 14 for each neighbour in its set of parent candidates. To show the same data exploited by the node for selecting its preferred parent, we consider the Exponential Weighted Moving Average (EWMA) of the ETX, as it is used by the RPL Contiki implementation to smooth small-scale fluctuations of ETX samples. In our experiments, the default RPL configuration of Contiki OS is adopted, where RPL measures the link cost using unicast packets, i.e., data packets or RPL control packets. As can be seen, before minute 9, the ETX samples towards the neighbours fluctuate, thus resulting in many parent changes over time. Right before minute 9, the ETX towards the current preferred parent (i.e., node number 20), increases steeply, triggering a local repair procedure. However, in the meantime, all the ETX values towards the other neighbours, increase above the maximum link-cost threshold, which is set to 4 in our configuration. Consequently, none of the neighbours can be selected as preferred parent and, hence, node 14 remains isolated as the local repair procedure cannot complete. This is due to the higher number of collisions occurring in the channel that worsen the links quality.

In order to explain the reason for the higher number of collisions occuring with CoCoA, we analysed the RTO. Again, for the sake of comparison, we focused on node number 14. In Fig. 13 we report the RTO as a function of time. As can be seen, the RTO fluctuates significantly over time and, in certain phases, it also assumes very small values, even below the second. If compared with the RTO value selected by the default algorithm, which is, by definition, in the range [2000–3000] ms

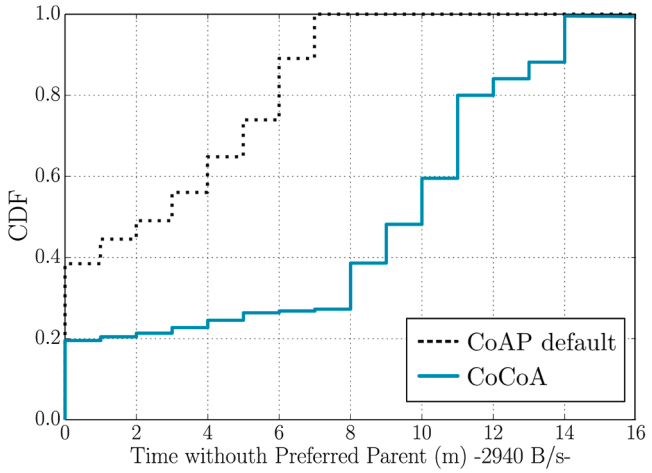

Fig. 11. Time Distribution without RPL Preferred parent.

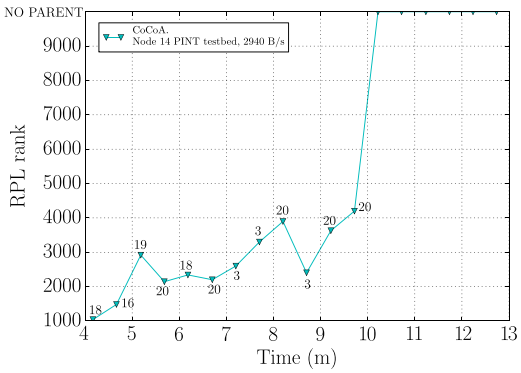

Fig. 12. Rank over time of node 14 with RPL.

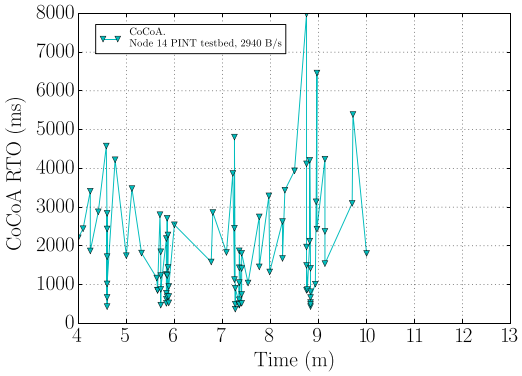

Fig. 13. RTO over time of node 14 with RPL.

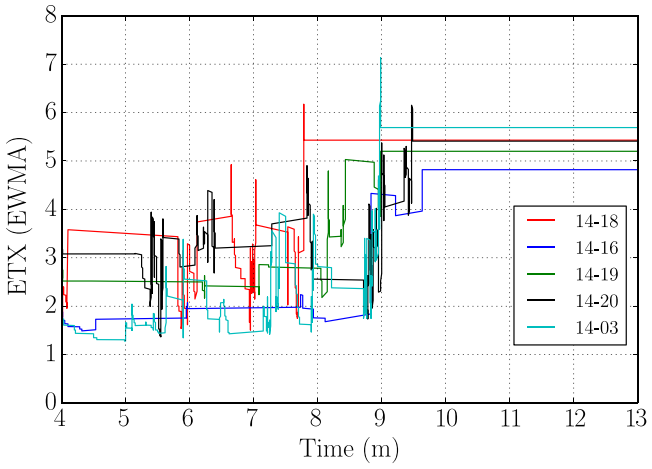

Fig. 14. EWMA of ETX estimated by node 14 during time in a single experiment.

(and is, thus, omitted in Fig. 13), we can observe that CoCoA results in a more æaggressiveg behaviour, with respect to the default algorithm, characterised by frequent retransmissions. If we compare these results with those in Fig. 12, where changes in the preferred parent, over time, are shown, we can observe that this behaviour mainly occurs after a change in the preferred parent of the node. Specifically, every time a node changes the preferred parent, a reduction in the RTO value is introduced by CoCoA. This is due to the ageing mechanism that activates because the node does not receive any acknowledgement and, hence, cannot update the RTO value for some time before selecting a new preferred parent. Right after the selection of the new preferred parent, however, the ageing mechanism is disabled, since transmissions are successful and RTT samples are collected. This may also lead to a further reduction in the RTO value. Specifically, during the time when no preferred parent is available, packets are buffered and are then transmitted in a series. These packets typically experience a low RTT, as the new path is expected to have a lower ETX and is likely to be less congested than the previous one. This series of successful transmissions reduces the RTO value and triggers an æaggressiveg behaviour by the source node, i.e., packets are re-transmitted very frequently. This eventually leads to a congestion and, ultimately, triggers a new change of the preferred parent.

This conclusion is confirmed by the analysis of the RTO distribution in the considered scenarios. To this end, Fig. 15 shows the distribution of RTO samples for both CoCoA and the default algorithm, at different traffic rates (i.e., packet generation periods). Indeed, for the default algorithm only the distribution at the highest traffic rate is reported, as the trend for the other rates is similar. For CoCoA, the RTO distribution

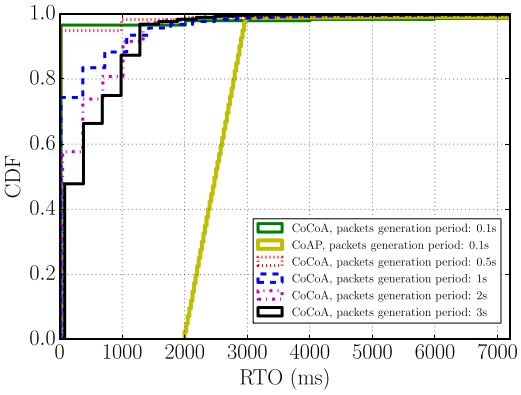

Fig. 15. RTO distribution with RPL.

shrinks as the packet generation period decreases (i.e., the Offered Load increases), thus increasing the likelihood of having small RTO values. Instead, the default algorithm generates RTO values that are always in the range [2000–3000] ms, as expected.

This behaviour, already known in literature [19], can be explained by considering the approach taken by the two policies in managing retransmissions. CoCoA sets the RTO based on the measured RTT, while the default algorithm adopts a fixed RTO. Setting the RTO based on the measured RTT values, results in more frequent retransmissions (some of which even unnecessary [19]) as the RTT value is usually significantly lower than the fixed RTO selected by the default algorithm.

The more aggressive strategy adopted by CoCoA is also confirmed by the average number of transmissions (including retransmissions) issued by a node, for each CoAP transaction, that is shown in Fig. 16. CoCoA generates a number of retransmissions much higher than the default algorithm, especially when the Offered Load is high.

In Fig. 17 we show the overall number of CoAP messages transmitted by all the nodes in the network, including both retransmissions and failed CoAP transactions, during the entire experiment. The number of transmissions reflects the behaviour of the congestion control policy: CoCoA is characterised by a more aggressive behaviour that results in a larger number of retransmissions, while the default algorithm takes a more conservative approach causing a lower number of retransmission. The higher number of retransmissions issued by CoCoA, however, does not result in a higher Carried Load, as shown in Fig. 9 from which it appears that the default algorithm outperforms CoCoA.

Moreover, if we compare the trend in the number of retransmissions (Fig. 17) and the Carried Load (Fig. 9) vs. the Offered Load, it can be

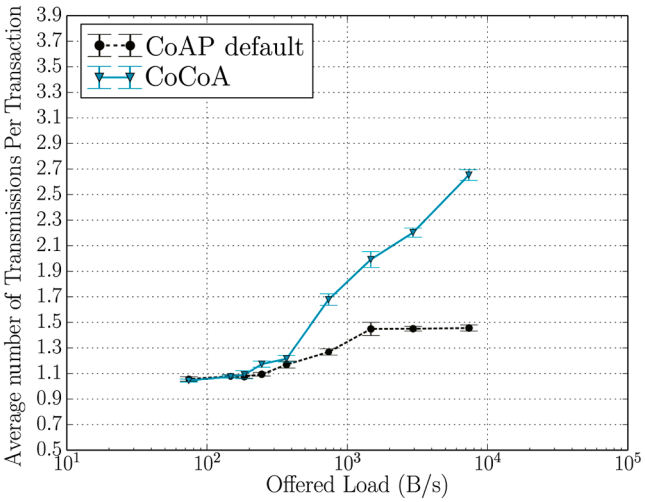

Fig. 16. Avg. number of retransmissions with RPL.

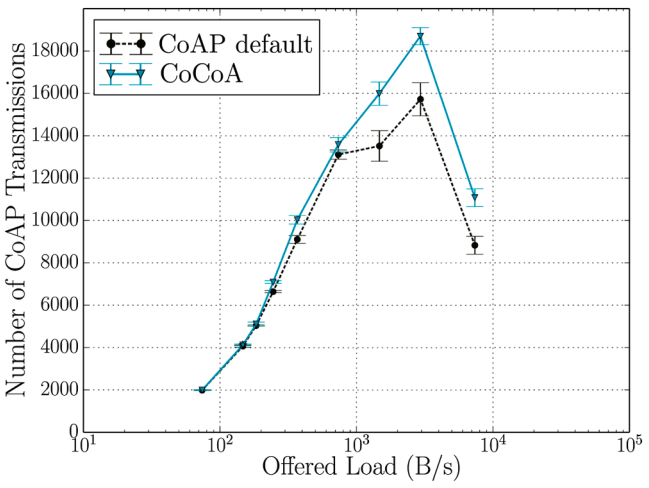

Fig. 17. CoAP transmissions issued with RPL.

noticed that, at the highest Offered Loads, the Carried Load drops significantly with both the considered congestion control algorithms. This behaviour, however, is not reflected in the number of retransmissions. This can be explained by the fact that, at the highest Offered Loads, many transactions fail because the maximum number of retransmissions allowed for a packet is reached. Considering that each source node can have, at most, a single outstanding transaction, at the highest Offered Loads many CoAP clients spend a lot of their time waiting for the ACKs for transmission that are going to fail, thus reducing also the number of CoAP transmissions issued.

An additional confirmation that the more significant drop in the Carried Load experienced by CoCoA, with respect to the default algorithm, is caused by its more aggressive behaviour, is provided by the overall number of packets dropped at the network layer, due to buffer overflows, shown in Fig. 9. This metric allows to link the instability of the routing protocol to the reduction of the Carried Load, because a packet is dropped when it cannot be forwarded due to the lack of a valid routing entry (i.e., a preferred parent), and the buffer is full. As shown in Fig. 18, at low rates the number of buffer overflows is negligible, while it increases significantly, as the Offered Load increases, highlighting that the behaviour of the routing protocol strongly influences the network performance. If we compare the results of the two congestion control algorithms, the same behaviour already observed with the previous metrics, can be found. CoCoA is characterised by a higher number of buffer overflows than the default algorithm. This confirms that the congestion control algorithm influences the instability of the routing protocol and, ultimately, the Carried Load.

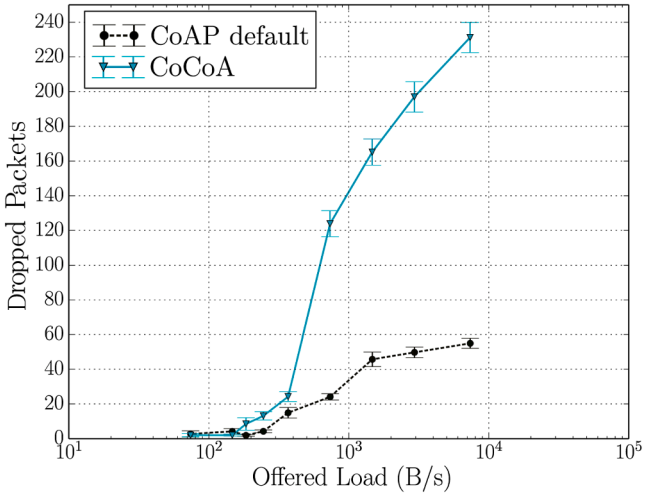

Fig. 18. Number of packets dropped at the network layer due to buffer with RPL.

5.2. WiLAB testbed

In order to confirm the conclusions drawn from the results presented in Section 5.1, the same set of experiments, presented in the previous section, was run on the WiLAB testbed.

Fig. 19 shows the Carried Load obtained with an increasing value of the Offered Load. Also in this scenario, CoCoA provides a lower Carried Load than the default congestion control algorithm. If we compare these results with the ones obtained on the PINT testbed (see Fig. 18), we can see that, now, the gap between the two algorithms is even more apparent than in the previous case. This can be explained considering that the WiLAB testbed is composed of a larger number of nodes, thus magnifying the effects of the congestion control policy on the network performance.

In order to confirm that the gap between the two congestion control policies is caused by the routing protocol, in Fig. 20 we show the overall number of preferred parent resets. As expected, the number of resets increases as the Offered Load grows up. In accordance with the results obtained on the PINT testbed, when using CoCoA, the number of resets increases significantly, thus confirming that the reduction in the overall Carried Load is caused by the RPL instability that leads to a higher number of preferred parent resets with CoCoA.

To conclude our analysis, in Fig. 21 we show the average number of transmissions per CoAP transaction to confirm that the RPL instability is influenced by the behaviour of the congestion control algorithm. The results obtained with WiLAB confirm that the CoCoA is characterised by a more aggressive behaviour that leads to issue a higher number of retransmissions for each CoAP transaction, thus exacerbating further the network congestion and RPL instability.

5.3. Static routing experiments

In order to assess the performance of the two considered congestion control algorithms without the influence of the routing protocol, we run an additional set of experiments in which the RPL protocol is disabled. In order to ensure multi-hop delivery, a static routing is implemented exploiting the functions of the WiSHFUL platform, which allows to manipulate the routing tables of the sensors. To this end, we programmed a specific module that populates the static routes of the sensors at the beginning of each experiment. Routes have been selected based on the analysis of the RPL protocol at low traffic loads, when its behaviour

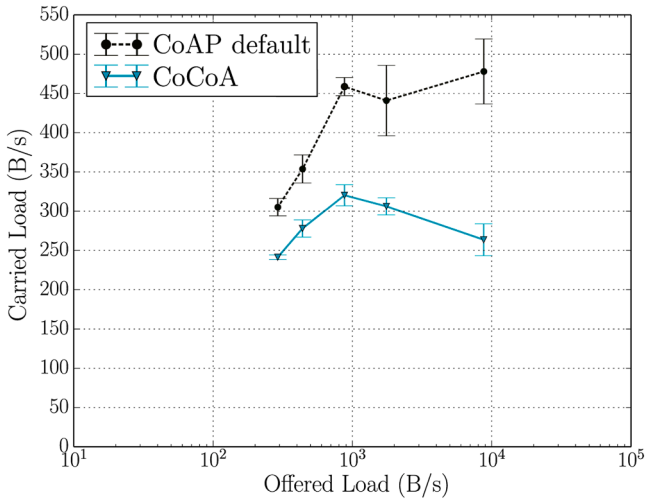

Fig. 19. WiLAB - Carried Load (*B/s*) with RPL.

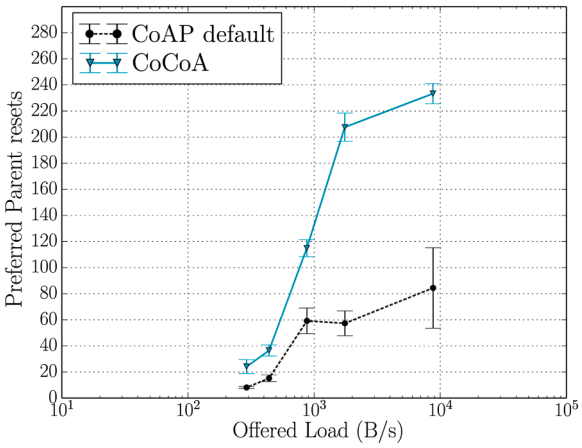

Fig. 20. WiLAB - Overall preferred parent resets.

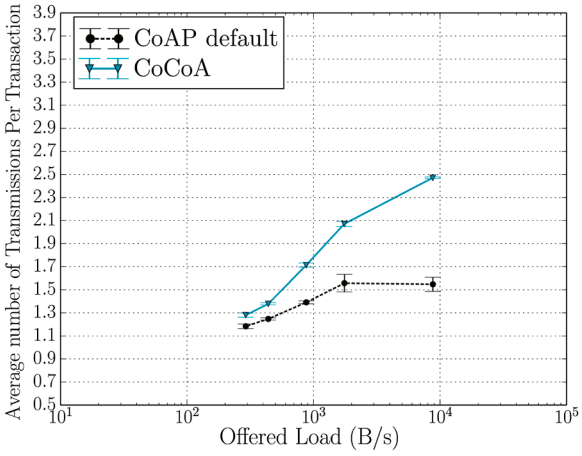

Fig. 21. WiLAB - Avg. number of transmissions with RPL.

is stable and reliable. The experiments have been run on the PINT testbed.

Fig. 22 shows the Carried Load provided by the two congestion control algorithms when using static routing. In this scenario, the results obtained are aligned with those presented in previous works evaluating the performance of congestion control algorithms for IoT [4,19]. As the network enters in a congested state, CoCoA offers a significant advantage with respect to the default algorithm. These results also confirm the conclusions of our previous study [19], showing that in the pre-congestion phase the default algorithm guarantees a performance slightly better than CoCoA. If we compare, in absolute terms, these results with the results obtained with RPL (Fig. 18), we can notice how RPL affects differently the overall Carried Load. For instance, if we consider the scenario with Offered Load equal to 2940B/s, CoCoA results in a Carried Load that is more than doubled with respect to the value obtained with RPL. The default algorithm, instead, only improves the Carried Load by around 30% with static routing.

Fig. 23 shows the overall number of transmissions issued in the scenario with static routing, by the default algorithm and CoCoA. Also in this case, the behaviour characterising the two congestion control strategies is confirmed: CoCoA issues a higher number of transmissions as it adopts a lower RTO, compared with the default algorithm.

In Fig. 25 we report the RTO samples collected, over time, on node 14 for CoCoA with static routing. If we compare these values with the ones obtained with RPL (Fig. 13) we can observe that, with static routing, very low RTO values are missing. Instead, when using RPL, we had observed that changes in the preferred parent triggers the aggressive retransmission policy of CoCoA that generates RTO values even below 1 second.

The same trend is confirmed also when comparing the overall number of transmissions issued with static routing (Fig. 23) and with RPL (Fig. 17), i.e., the overall number of transmission issued when using static routing is higher than that observed with RPL. This is because the static routing ensures a continuous transmission of data, while with RPL nodes experience long periods (during the local repair procedure) in which they do not transmit any packet. The continuous data transmission allows more transactions to be completed, thus increasing their overall number. This is also reflected by the Carried Load that increases noticeably in comparison with the values obtained with RPL (Fig. 18 vs. Fig. 23).

The significant improvement in performance with static routing can be also seen if we look at the average number of transmissions (including retransmissions) per transaction. Fig. 24 shows that the stability of the static routing helps to reduce significantly the average number of transmissions, which is always below 2. These results show that, as expected, CoCoA still results in a higher number of transmissions. However, in this case, the gap between CoCoA and the default algorithm is lower.

6. Lessons learned

Our experimental results, obtained with two different and independent testbeds, have highlighted that the RPL routing protocol has a very significant impact on the performance of the congestion control algorithm, especially at high Offered Loads. This impact, however, is stronger with CoCoA, in comparison with the CoAP default congestion-

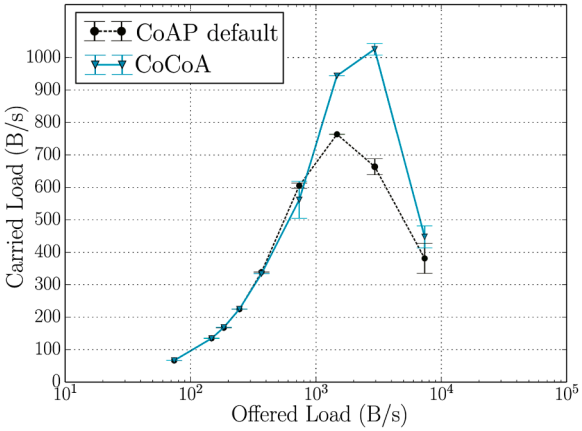

Fig. 22. Carried Load with *static* routes.

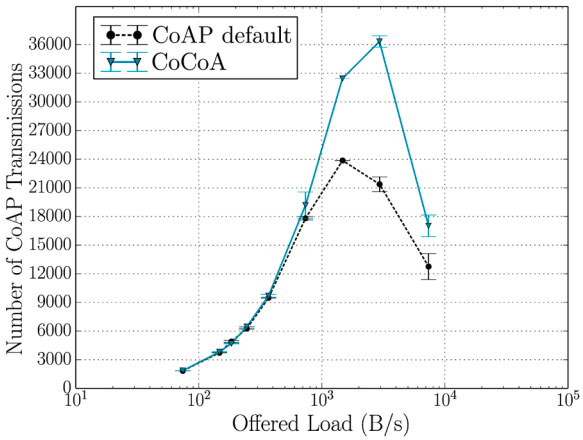

Fig. 23. CoAP transmissions issued. *Static* routing.

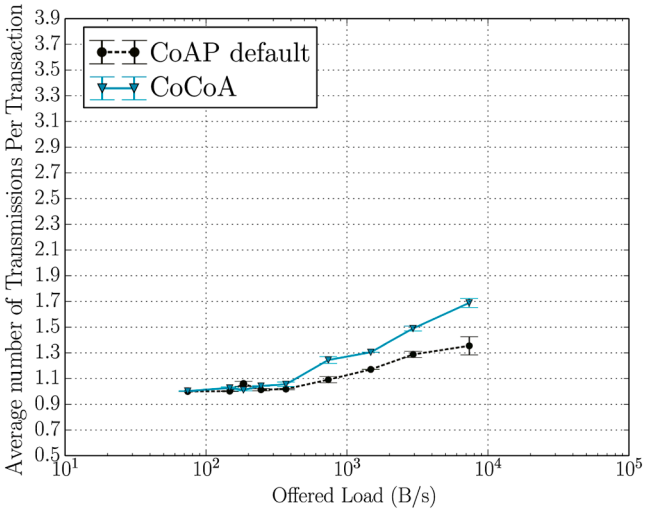

Fig. 24. Avg number of transmissions. *Static* routing.

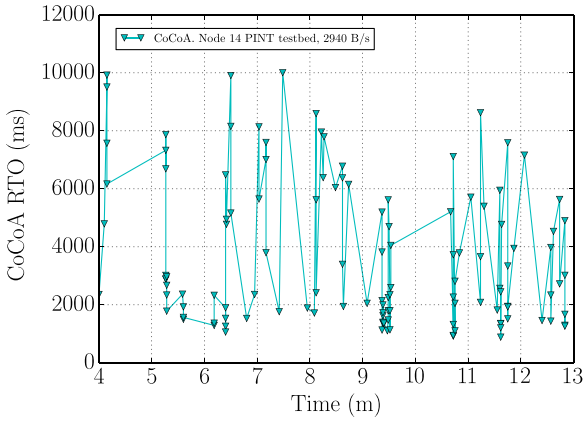

Fig. 25. RTO over time of node 14 with *static* routes.

control algorithm, as the former implements a more aggressive retransmission strategy.

Specifically, the more aggressive behaviour that characterises CoCoA increases the number of messages injected in the network, which can lead to a dramatic increase of the congestion that, in its turn, results in further exacerbation of the routing instability. This loop ultimately leads to a significant reduction of the Carried Load, with respect to the one provided by the default algorithm.

In particular, our analysis has highlighted the following vicious circle that may occur with CoCoA. Every time a node changes its preferred parent (e.g., due to channel variations), it experiences a period during which packets cannot be transmitted and are consequently buffered. When a new preferred parent is selected, a decrease in the RTO is experienced due to the combined effect of the ageing mechanism included in CoCoA and the better conditions experienced by packets with the new path. This combined effect leads to a reduction of the RTO value and, hence, to frequent retransmissions at the source node. In its turn, frequent retransmissions increase the likelihood of collisions with neighbours and, hence, the measured ETX. The increase in the ETX may result in subsequent changes in the preferred parent of the node that further exacerbates the overall congestion, thus decreasing the overall network performance. This circle eventually leads to a general increase of the ETX experienced on all the links that, ultimately, may cause the disconnection of some of the nodes from the network.

Our conclusion is not in agreement with previous results available in the literature [6,12], which show that CoCoA outperforms the default congestion control algorithm, due to its aggressive strategy. According to these studies, CoCoA would provide a Carried Load higher than that obtained with the default strategy, when the network is congested. This discrepancy can be explained if we consider that the above-mentioned studies are based on simulation. Although they rely on a realistic simulation model (e.g. they use a realistic wireless channel model and exploit a network emulator capable of emulating real hardware, such as Cooja [30]), the simulation model may not be able to capture all the aspects of a real environment. This reduces its accuracy, especially at high traffic loads, as it has been already observed in previous analysis of the RPL protocol [16–18]. Specifically, in our case, the following two main factors need to be considered to explain the discrepancy between our experimental measurements and the previous simulation results.

- *Wireless channel model*. Although Wireless Sensor Network (WSN) simulators adopt realistic channel models, they can hardly capture the full channel dynamics of a real environment, which is characterised by a time-varying level of interference [24]. External interference triggers the routing instability observed in our experiments. Of course, the negative effect of routing instability is more apparent at high traffic loads.
- *Limited buffer size*. In simulations, sensor nodes such as Cooja Motes usually have no special limitations on the available memory. Hence, large buffer sizes are typically set. Instead, real sensor nodes are usually constrained in terms of memory and, consequently, in experimental measurement small transmission buffers are typically used that can accommodate only a limited number of packets (e.g., 8 packets). The limited buffer size increases the likelihood of dropping control and data packets, thus increasing the instability of the network. As above, the negative impact of dropping messages is more apparent at high traffic loads.

7. Conclusions

In this paper, we have presented an experimental performance evaluation of two different congestion control algorithms for the Internet of Things. In our analysis, we mainly focused on analysing the interplay between the congestion control strategy and the RPL routing protocol. To make the experimental analysis more general, we used two different and independent testbeds, both using the WiSHFUL platform.

Our experimental results have shown that the performance of the considered congestion control policies is strongly influenced by the instability of the routing protocol, which can degrade noticeably the performance. However, the RPL dynamics influence the different congestion control policies in a different manner, penalising the more aggressive ones. Specifically, we found that, in a real environment, at high traffic loads the simple CoAP default congestion-control algorithm outperforms the more complex CoCoA algorithm.

For future work, we plan to extend our evaluation by considering additional network topologies and traffic patterns in order to investigate the performance of different congestion control strategies in a large number of scenarios for different use cases. In addition, we plan to evaluate the performance of congestion control protocols also in 6TiSCH networks, in order to assess their suitability and performance in networks with time-slotted access protocol. In such an environment it will be important to investigate the interplay, not only with RPL, but also with the scheduling algorithm used in 6TiSCH networks to allocate cells to nodes.

Declaration of Competing Interest

The authors Carlo Vallati, Francesca Righetti, Giacomo Tanganelli, Enzo Mingozzi and Giuseppe Anastasi declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been carried out within the activities of the project "ECOAP: Experimental Assessment of Congestion Control Strategies for CoAP using the WiSHFUL platform" funded by the WiSHFUL project H2020 (GA 645274) under the open call WiSHFUL-OC5-EXP-EXC. This work was also partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

References

[1] E. Borgia, The internet of things vision: key features, applications and open issues, Comput. Commun. 54 (2014) 1–31, https://doi.org/10.1016/j.comcom.2014.09.008.
[2] C. Bormann, A.P. Castellani, Z. Shelby, Coap: an application protocol for billions of tiny internet nodes, IEEE Internet Comput. 16 (2) (2012) 62–67, https://doi.org/10.1109/MIC.2012.29.
[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, 2012, https://doi.org/10.17487/RFC6550.(RFC 6550)
[4] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, Cocoa+: an advanced congestion control mechanism for CoAP, Ad Hoc Netw. 33 (2015) 126–139, https://doi.org/10.1016/j.adhoc.2015.04.007.
[5] C. Bormann, A. Betzler, C. Gomez, I. Demirkol, CoAP Simple Congestion Control/Advanced. Internet-Draft, IETF, 2018.
[6] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, Coap congestion control for the internet of things, IEEE Commun. Mag. 54 (7) (2016) 154–160, https://doi.org/10.1109/MCOM.2016.7509394.
[7] I. Jarvinen, L. Daniel, M. Kojo, Experimental evaluation of alternative congestion control algorithms for constrained application protocol (CoAP). 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 453–458, https://doi.org/10.1109/WF-IoT.2015.7389097.
[8] E. Ancillotti, R. Bruno, Comparison of CoAP and COCOA+ congestion control mechanisms for different IoT application scenarios. 2017 IEEE Symposium on Computers and Communications (ISCC), 2017, pp. 1186–1192, https://doi.org/10.1109/ISCC.2017.8024686.
[9] R. Bhalerao, S.S. Subramanian, J. Pasquale, An analysis and improvement of congestion control in the CoAP internet-of-things protocol. 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC), 2016, pp. 889–894, https://doi.org/10.1109/CCNC.2016.7444906.
[10] E. Balandina, Y. Koucheryavy, A. Gurtov, Computing the retransmission timeout in CoAP. Internet of Things, Smart Spaces, and Next Generation Networking, Springer Berlin Heidelberg, 2013, pp. 352–362, https://doi.org/10.1007/978-3-642-40316-3_31.
[11] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, Congestion control in reliable CoAP communication. Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems, ACM, 2013, pp. 365–372, https://doi.org/10.1145/2507924.2507954.
[12] S. Bolettieri, G. Tanganelli, C. Vallati, E. Mingozzi, pCoCoA: a precise congestion control algorithm for CoAP, Ad Hoc Netw 80 (2018) 116–129, https://doi.org/10.1016/j.adhoc.2018.06.015.
[13] C. Vallati, F. Righetti, G. Tanganelli, E. Mingozzi, G. Anastasi, ECoAP: Experimental assessment of congestion control strategies for CoAP using the wishful platform. 2018 IEEE International Conference on Smart Computing (SMARTCOMP), 2018, pp. 423–428, https://doi.org/10.1109/SMARTCOMP.2018.00040.
[14] E. Ancillotti, S. Bolettieri, R. Bruno, Rtt-based congestion control for the internet of things. Wired/Wireless Internet Communications, Springer International Publishing, 2018, pp. 3–15, https://doi.org/10.1007/978-3-030-02931-9_1.
[15] V. Rathod, N. Jeppu, S. Sastry, S. Singala, M.P. Tahiliani, Cocoa++: delay gradient based congestion control for internet of things, Future Generation Computer Systems 100 (2019) 1053–1072, https://doi.org/10.1016/j.future.2019.04.054.
[16] H. Kim, J. Paek, S. Bahk, Qu-rpl: Queue utilization based rpl for load balancing in large scale industrial applications. 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2015, pp. 265–273, https://doi.org/10.1109/SAHCN.2015.7338325.
[17] H. Kim, H. Kim, J. Paek, S. Bahk, Load balancing under heavy traffic in rpl routing protocol for low power and lossy networks, IEEE Trans. Mob. Comput. 16 (4) (2017) 964–979, https://doi.org/10.1109/TMC.2016.2585107.
[18] H. Kim, J. Paek, D.E. Culler, S. Bahk, Do not lose bandwidth: Adaptive transmission power and multihop topology control. 2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2017, pp. 99–108, https://doi.org/10.1109/DCOSS.2017.23.
[19] S. Bolettieri, C. Vallati, G. Tanganelli, E. Mingozzi, Highlighting some shortcomings of the CoCoA+ congestion control algorithm. Ad-hoc, Mobile, and Wireless Networks, Springer International Publishing, 2017, pp. 213–220, https://doi.org/10.1007/978-3-319-67910-5_17.
[20] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, Congestion control in reliable CoAP communication. Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems, ACM, 2013, pp. 365–372, https://doi.org/10.1145/2507924.2507954.
[21] A. Pramanik, A.K. Luhach, I. Batra, U. Singh, A systematic survey on congestion mechanisms of CoAPbased internet of things. Advanced Informatics for Computing Research, Springer Singapore, 2017, pp. 306–317, https://doi.org/10.1007/978-981-10-5780-9_28.
[22] O. Gnawali, P. Levis, The Minimum Rank with Hysteresis Objective Function, 2012, (RFC 6719). 10.17487/RFC6719.
[23] P. Ruckebusch, S. Giannoulis, D. Garlisi, P. Gallo, P. Gawlowicz, A. Zubow, M. Chwalisz, E. De Poorter, I. Moerman, I. Tinnirello, L. DaSilva, Wishful: enabling coordination solutions for managing heterogeneous wireless networks, IEEE Commun. Mag. 55 (9) (2017) 118–125, https://doi.org/10.1109/MCOM.2017.1700073.
[24] C. Vallati, E. Ancillotti, R. Bruno, E. Mingozzi, G. Anastasi, Interplay of link quality estimation and rpl performance: An experimental study. Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, ACM, 2016, pp. 83–90, https://doi.org/10.1145/2989293.2989299.
[25] L. Tytgat, B. Jooris, P. De Mil, B. Latré, I. Moerman, P. Demeester, Demo abstract: Wilab, a real-life wireless sensor testbed with environment emulation. EWSN 2009 adjunct poster proceedings, 2009.
[26] G. Tanganelli, C. Vallati, E. Mingozzi, Coapthon: Easy development of CoAP-based iot applications with python. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 63–68, https://doi.org/10.1109/WF-IoT.2015.7389028.
[27] T. Potsch, K. Kuladinithi, M. Becker, P. Trenkamp, C. Goerg, Performance evaluation of CoAP using rpl and lpl in tinyos. 2012 5th International Conference on New Technologies, Mobility and Security (NTMS), 2012, pp. 1–5, https://doi.org/10.1109/NTMS.2012.6208761.
[28] D.S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, Wirel. Netw. 11 (4) (2005) 419–434, https://doi.org/10.1007/s11276-005-1766-z.
[29] E. Ancillotti, R. Bruno, M. Conti, The role of the rpl routing protocol for smart grid communications, IEEE Commun. Mag. 51 (1) (2013) 75–83, https://doi.org/10.1109/MCOM.2013.6400442.
[30] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with cooja. Proceedings. 2006 31st IEEE Conference on Local Computer Networks, 2006, pp. 641–648, https://doi.org/10.1109/LCN.2006.322172.

Carlo Vallati is an assistant professor (tenured) of Computer Systems Engineering at the University of Pisa, Italy. His research activities include different areas such as wireless and sensor networks, protocols and platforms for the Internet of Things, algorithms and architectures for Edge/Fog computing. He has been involved in different national and international projects, including in particular the FP7-ICT project "BETaaS: Building the Environment for the Things as a Service". He has been the principal investigator of the project "ECOAP: Experimental assessment of congestion control strategies for the Constrained Application Protocol", funded with a grant of 45000 euro by the European Project WiSHFUL under the Fifth Open Call for experiments (WiSHFUL-OC5). He is the Coordinator of the Cloud Computing, Big Data and Cybersecurity Crosslab, founded in the framework of the Departments of Excellence funded by the Italian Ministry of Education, University and Research. He is co-author of more than 50 international publications.

Francesca Righetti is currently pursuing her PhD degree in Information Engineering at the University of Pisa. She received her Master's Degree in Computer Engineering from the University of Pisa, Italy, in May 2017. Her research interests include Wireless Sensor Networks, and their applications, Internet of Things (IoT) and Industrial Internet of Things (IIoT). She took part in the project "ECOAP: Experimental assessment of congestion control strategies for the Constrained Application Protocol". She has served as a member of the organization committee for the international conference IEEE SMARTCOMP 2018 and 2019.

Giacomo Tanganelli is a Postdoctoral Researcher at the Department of Information Engineering of the University of Pisa. In 2015, he was a PhD visiting student at the Computer Science Department of the ETH Zurich. His main research area is the Internet of Things including also wireless sensor networks, M2M communications, low power WANs and Edge/Fog computing. He is co-author of +20 peer-reviewed papers in international journals and conference proceedings. He was also guest editor of special issues on topics related to Internet of things. He has been involved in the project BETaaS, Building the Environment for the Things as a Service, funded by the European Union under the 7th Framework Program and in research projects founded by private industries. He has served as co-Chair for the international workshop IoT-SoS in 2018 and 2019.

Enzo Mingozzi is Full Professor at the Department of Information Engineering of the University of Pisa, Italy. His research activities span several areas, including resource optimization in wireless and wired networks, Mobile Edge/Fog Computing and the Internet of Things. He is a co-author of 120 peer-reviewed papers in international journal and conference proceedings, 5 book chapters and 10 patents. He currently serves on the Editorial Board of Computer Networks (Elsevier), and Computer Communications (Elsevier). He has been also guest editor of a number of journal special issues on topics related to wireless and mobile networks. He served in the organizing and technical program committee of several international conferences and was a co-founder of the IEEE IoT-SoS and IEEE MeshTech workshops.

Giuseppe Anastasi is a Full professor and the Head of the Department of Information Engineering at the University of Pisa, Italy. He is also the past (and founding) Director of the "Smart Cities" National Lab supported by CINI (Italian National University Consortium for Informatics). His research interests include Internet of Things, sensor networks, smart environments, and sustainable computing. He is currently leading the "CrossLab" project, which aims at structuring six interdisciplinary and integrated laboratories (CrossLabs) for Industry 4.0 at the University of Pisa. In addition, he has contributed to many research programs funded by both national and international institutions. He has co-edited two books and published about 150 papers in international journal and conferences. He is currently serving as Steering Committee member of the IEEE SMARTCOMP conference. Previously, he served as: Area Editor of Pervasive and Mobile Computing; Associate Editor of Sustainable Computing; Area Editor of Computer Communications; General Chair of IEEE SMARTCOMP 2018, IEEE WoWMoM 2005; Program Chair of IEEE SMARTCOMP 2016, IEEE PerCom 2010, and IEEE WoWMoM 2008.