IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 January 2023, accepted 27 January 2023, date of publication 1 February 2023, date of current version 6 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3241327

RESEARCH ARTICLE

Investigating the CoAP Congestion Control Strategies for 6TiSCH-Based IoT Networks

FRANCESCA RIGHETTI, CARLO VALLATI, (Member, IEEE), DAVIDE RASLA, AND GIUSEPPE ANASTASI, (Member, IEEE)
Department of Information Engineering, University of Pisa, 56126 Pisa, Italy
Corresponding author: Francesca Righetti (francesca.righetti@unipi.it)

This work was supported by the Italian Ministry of Education and Research (MIUR) in the Framework of the CrossLab and FoReLab Projects (Departments of Excellence).

ABSTRACT The *Constrained Application Protocol (CoAP)* is a RESTful protocol standardized by the IETF and widely used for IoT applications. CoAP includes a default congestion control algorithm to ensure efficient operation under high traffic conditions. Other congestion control algorithms for CoAP have been proposed and evaluated in the literature, including the very popular CoCoA algorithm. All these algorithms assume that the underlying wireless communication is regulated through the 802.15.4 CSMA-CA protocol. Today, many IoT systems are based on the 6TiSCH architecture that, instead, leverages the TSCH (*Time Slotted Channel Hopping*) mode of IEEE 802.15.4, i.e., a synchronous and time-slotted access protocol. In this paper we investigate, by simulation, the suitability of existing CoAP congestion-control algorithms to the 6TiSCH architecture. Our results show that the performance of the considered algorithms are strongly influenced by the Scheduling Function used to allocate communication resources to nodes. In addition, our analysis emphasizes that CoCoA does not provide a significant advantage over the default algorithm in 6TiSCH networks. We investigate the motivations for such a behavior and propose an optimized version of CoCoA, namely 6CoCoA, specifically tailored for 6TiSCH networks. 6CoCoA is able to provide up to a 15% improvement of the Transaction Delivery Ratio and up to a 25% reduction of the end-to-end Transaction Delay, when the network is congested.

INDEX TERMS CoAP, congestion control, CoCoA, 6TiSCH scheduling function, 6CoCoA.

I. INTRODUCTION

The *Constrained Application Protocol (CoAP)* [1] is a RESTful protocol standardized by the Internet Engineering Task Force (IETF) and widely used for supporting applications in the Internet of Things (IoT). It includes a default congestion control algorithm to ensure efficient operation under high traffic conditions. Other congestion control algorithms for CoAP have been proposed and evaluated in the literature. Among them, *CoCoA (CoAP Simple Congestion Control/Advanced)* [2] is certainly the most popular one. However, all these algorithms assume that the underlying wireless communication is regulated by the CSMA-CA (*Carrier Sense Multiple Access with Collision Avoidance*) protocol defined in the original IEEE 802.15.4 standard [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

Recently, the IETF has defined the 6TiSCH architecture that, instead, relies on the TSCH (*Time Slotted Channel Hopping*) mode of IEEE 802.15.4 [4]. TSCH leverages *time-slotted* access, *multi-channel* communication, and *frequency hopping* to provide improved reliability, bounded latency, increased network capacity, and high energy efficiency.

The concept of congestion in 6TiSCH-based IoT networks is very different from that of traditional IoT networks based on the CSMA-CA protocol. In the latter case, congestion is originated by many nodes trying to transmit simultaneously. This causes long waiting times and packet dropping, due to exceeded number of retransmissions and/or exceeded number of back-off stages [5]. Instead, in 6TiSCH networks the access method is slotted and timeslots are pre-assigned to nodes, according to a certain schedule. Hence, nodes can transmit only during their assigned timeslots, without contention. Congestion may still occur. However, it is due to an

insufficient number of resources allocated to a node, resulting in queuing delay and possible packet dropping [6]. In both cases, the final result is similar, but the causes behind are quite different.

More specifically, in 6TiSCH networks, a congestion may occur even when there is a single transmitting node, if the amount of allocated communication resources (i.e., timeslots) is too low for the traffic that the node needs to manage. In principle, congestion may be avoided by adjusting dynamically the amount of communication resources through the scheduling algorithm. However, if the overall traffic injected increases more and more, it surely happens that some node in the network has to manage more packets than those allowed by the current allocation and no more resources are available for a new allocation. Hence, an end-to-end congestion control mechanism is still required to reduce the injected traffic and recover from congestion.

As anticipated, CoAP specifications [1] define a default congestion control algorithm that includes some basic functionalities to regulate the end-to-end traffic transmission. However, this basic algorithm suffers many limitations under high traffic conditions [7]. For this reason, an advanced congestion control has been proposed in literature, the CoCoA algorithm [2]. CoCoA includes a set of mechanisms to adapt the transmission of data to the current network conditions, and provides significant advantages over the default basic approach. CoCoA [2] is widely considered the most popular congestion control strategy for low-power and lossy networks.

So far, all the proposed congestion control algorithms, including the default algorithm and CoCoA, have been evaluated assuming that the underlying Medium Access Control (MAC) protocol is CSMA-CA [2], [8], [9], [10], [11]. In this paper, we focus on 6TiSCH-based IoT networks (based on TSCH), and carry out a simulation analysis to compare the performance of the two most popular CoAP congestion control algorithms, namely *CoAP default* and *CoCoA*. To the best of our knowledge, this is the first work investigating the suitability of CoAP congestion control algorithms for IoT networks based on the 6TiSCH architecture.

Our simulation results show that the performance of the considered congestion control algorithms strongly depends on the scheduling algorithm used to allocate TSCH communication resources. In our analysis, we considered two different 6TiSCH Scheduling Functions that take a different approach in allocating communication resources to nodes, namely the *Minimal Scheduling Function* (MSF) [12], standardized by IETF as the default scheduling algorithm for 6TiSCH networks, and the *Autonomous Link-based Cell Scheduling* (ALICE) [13].

Moreover, we found that existing CoAP congestion control algorithms, conceived for asynchronous and random-access communication environments, do not work efficiently in the 6TiSCH architecture that relies on synchronous and scheduled communication. In particular, our results show that the CoCoA algorithm, which is conceived as an advanced congestion control and is proved to be very efficient in CSMA-CA environments, is unable to guarantee a performance that is inline with the results obtained in literature when adopted in 6TiSCH environments, especially if we compare its performance with that of the default congestion control algorithm. In order to improve the performance of CoCoA in 6TiSCH networks, we investigated the motivations for such a behavior and came up with a new configuration that is specifically tailored for the 6TiSCH architecture. The proposed version, throughout referred to as *6CoCoA*, is a fine-tuned version of CoCoA whose parameters are set in order to perform better in TSCH networks, and it allows a 15% improvement of the Transaction Delivery Ratio and a 25% reduction of the end-to-end Transaction Delay, in congestion situations.

In summary, the contribution provided by this paper is twofold, and can be summarized as follows:

- A simulation analysis to assess the suitability and performance of the two most popular CoAP congestion-control algorithms in 6TiSCH networks;
- A fine-tuned version of CoCoA whose parameters are tailored to 6TiSCH networks.

The reminder of this paper is organized as follows. In Section II, we describe the 6TiSCH architecture and we introduce the CoAP protocol and its congestion control algorithms. In Section III, we present our simulation setup. In Section IV, we compare the default and the CoCoA congestion control algorithms in 6TiSCH networks and investigate the interplay between congestion control and 6TiSCH scheduling. In Section V, we analyze the motivation for the low performance of CoCoA in 6TiSCH networks. Based on these outcomes, in Section VI, we present 6CoCoA, a revised version of CoCoA targeted to 6TiSCH networks. In Section VII we assess the performance of 6CoCoA, while in Section VIII we draw our conclusions.

II. 6TiSCH ARCHITECTURE

6TiSCH is the architecture defined by IETF to support industrial IoT applications [14]. It aims at integrating wireless networks based on the TSCH access mode of the IEEE 802.15.4 standard [4] into existing IPv6 infrastructures. The reference network model and the complete protocol stack are shown in Figure 1.

The main difference, with respect to the traditional IoT architecture, is the MAC protocol. As shown in Figure 1, 6TiSCH relies on the TSCH access mode defined in the IEEE 812.15.4 standard [4], while the former architecture was based on the CSMA-CA protocol defined in a previous version of the same standard [3]). CSMA-CA implements a random-access protocol using a single communication channel and provides a best-effort service model. Instead, TSCH relies on *time-slotted channel access*, *multi-channel communication*, and *frequency hopping* and provides increased network capacity, guaranteed bandwidth, bounded delay and energy efficiency. Hence, the 6TiSCH architecture can meet the requirements of a larger number of IoT application

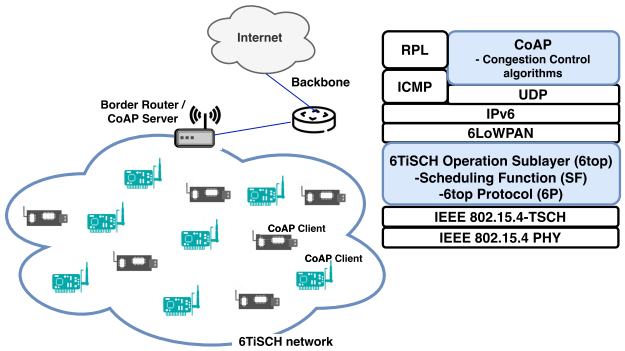


FIGURE 1. 6TiSCH architecture: reference network model (left) and protocol stack (right).

domains, including industrial, medical, and critical applications with stringent requirements, in terms of timeliness and reliability. The TSCH protocol is described in Section II-A.

While TSCH provides the mechanism to allocate and deallocate communication resources (TSCH cells), it does not specify how cells are allocated to nodes. To this purpose, the 6TiSCH architecture includes the 6TiSCH Operation (6top) sublayer [15] that manages the allocation of cells to nodes in such a way to meet the application requirements. Specifically, the 6top layer implements a *Scheduling Function* (SF), to determine the amount of cells to allocate, and the *6top protocol* (6P), to negotiate cells among neighbors.

In principal, many different SFs can be used, depending on the application domain. The 6TiSCH has defined the *Minimal Scheduling Function* (MSF) [12] to be used as the default SF. However, many other SFs have been proposed in the literature [16]. For the purposes of this study, we will focus on two popular SFs, namely MSF [12] and ALICE [13], taking a different approach in allocating TSCH cells to nodes. This will allow us to investigate the interplay between 6TiSCH scheduling and CoAP congestion control. The two considered SFs are described in Section II-B.

Above the 6top layer, the 6LoWPAN adaptation protocol [17] is responsible for encapsulating IPv6 datagrams into TSCH frames. The *IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)* [18] ensures multi-hop delivery of IPv6 datagrams. RPL organizes the network nodes in a *Destination Oriented Directed Acyclic Graph (DODAG)*, where every node selects a neighbor, called *preferred parent*, as the next hop for upstream data delivery. The DODAG is rooted at a single node, the RPL root node, that is typically the collector of the network to which upstream data is directed.

Finally, the delivery of messages originated by the application is managed by the *Constrained Application Protocol (CoAP)* running on top of UDP [1], as shown in Figure 1. CoAP is a lightweight RESTful application-layer protocol specifically designed for constrained devices. It inherits the same client/server paradigm used in HTTP, however leverages UDP (instead of TCP) to be as lightweight as possible. CoAP requests are exchanged between clients and servers to perform some actions, and are specified through the same basic set of methods used by HTTP (GET, POST, PUT and DELETE). Since CoAP runs on top of UDP, it also includes a congestion control mechanism to avoid congestion and ensure efficient management of network resources under high traffic conditions. A detailed description of the CoAP protocol is presented in Section II-C.

A. TIME SLOTTED CHANNEL HOPPING (TSCH)

The *Time Slotted Channel Hopping* (TSCH) mode, defined in the IEEE 802.15.4 standard [4], divides the time into time intervals of fixed duration (*timeslots*), each of which allows to transmit a data packet and receive the corresponding acknowledgment. A number of consecutive timeslots is grouped to form a *slotframe*, which repeats periodically over time. TSCH allows for multi-channel communication, increasing the network capacity, by allowing different nodes to transmit simultaneously on the same timeslot, using a different channel. Specifically, 16 different channels are available, identified by a channel offset (an integer value in the range 0-15) and, hence, each cell in this two-dimensional slotframe is identified through a pair, namely *timeslot*, and *channel offset*. Finally, to mitigate the negative effects of multipath fading and interferences, TSCH leverages frequency hopping. A predefined frequency-hopping sequence is shared among all the nodes in the network, so that they can select a different operating frequency at each timeslot.

B. SCHEDULING FUNCTIONS

TSCH cells are allocated to nodes for communication according to a SF implemented at the 6top layer. In this section, we describe the two SFs considered in our analysis, namely, MSF and ALICE.

MSF [12] takes a distributed approach to resource allocation, where each node negotiates the allocation and deallocation of cells with its neighbors, depending on its traffic requirements and network conditions. Cell negotiation is carried out through the 6P protocol [15]. Basically, MSF calculates dynamically the required number of cells and, consequently, triggers the allocation or deallocation of cells, through 6P.

MSF is designed to operate in a wide range of application domains. However, it is optimized for applications with regular upstream traffic, from nodes to the root of the DODAG, while downstream traffic is assumed to be sporadic [12]. Consequently, cells are allocated only for upstream traffic, while downstream traffic is typically managed through shared cells.

Indeed, many IoT applications (e.g., monitoring applications) follows this convergecast model, where data are sent by sensors to the root node and, hence, downstream traffic is very limited or absent. However, applications using the Confirmable CoAP service, like the ones considered in this paper, generate both upstream and downstream data flows. For this class of applications, the standard MSF in unsuitable, as it provides very low performance to the downstream traffic.

Algorithm 1 B-MSF Algorithm

Input:
NEC = Number of elapsed cells
MAX_NEC = Max number of elapsed cells
NCU = Number of cells used for transmission
LIM_NCU_HIGH = Threshold to cell
LIM_NCU_LOW = Threshold to delete cell
NEC_C[c] = Number of elapsed **child** cells
MAX_NEC_C = Max number of elapsed **child** cells
NCU_C[c] = Number of **child** cells used for transmission
LIM_NCU_HIGH_C = Threshold to add **child** cell
LIM_NCU_LOW_C = Threshold to delete **child** cell

Output:
ADD/DEL one cell
ADD/DEL one **child** cell

```
1  if NEC > MAX_NEC then
2      if NCU > LIM_NCU_HIGH then
3          trigger 6P to ADD one cell
4      if NCU < LIM_NCU_LOW then
5          trigger 6P to DEL one cell
6  for each child c do
7      if NEC_C[c] > MAX_NEC_C then
8          if NCU_C[c] > LIM_NCU_HIGH_C then
9              trigger 6P to ADD one child cell
10         if NCU_C[c] > LIM_NEC_LOW_C then
11             if number of child cells > 1 then
12                 trigger 6P to DEL one child cell
```

Hence, for the purposes of our analysis, we modified the standard MSF by allocating resources in both upstream and downstream directions. Throughout, we will refer to this modified version of MSF as Bidirectional MSF (or B-MSF, for short).

B-MSF is shown in Algorithm 1, where we emphasized in blue the differences with respect to the standard (i.e., unidirectional) MSF. It leverages an utilization-based approach. Initially, each node negotiates a single cell with its preferred parent and every child in the DODAG. Then, B-MSF periodically (every MAX_NEC) checks the cell utilization of the node, i.e., the percentage of used cells with respect to scheduled cells. Different thresholds are used for the preferred parent and each of the children to decide when to add, or remove, cells. In Algorithm 1, we distinguish the part related to the preferred parent (lines 1-5) from the one related to the children (lines 6-12). However, in both cases, the actions to perform are similar. Specifically, if the cell utilization is higher than the upper threshold, one more cell is negotiated with the parent/child node. Instead, if the cell utilization falls below the lower threshold, one cell is deleted.

Unlike B-MSF, ALICE [13] is an autonomous SF that allocates cells autonomously (i.e., without negotiation), using a hash function applied to the address of nodes. ALICE adopts a link-based approach that allows each node to allocate one dedicated cell for each unidirectional link with every neighbor node (i.e., preferred parent or child node). The timeslot and channel offset of each cell are calculated by the hash function based on (i) addresses of the node itself and its corresponding; (ii) direction of the communication link (i.e., upstream or downstream); and (iii) slotframe length. The result is a bidirectional schedule (i.e., for both upstream and downstream traffic), in which each node has many transmission and reception cells as the number of its neighbors. The main advantage of ALICE, with respect to MSF, is that the allocation is generated autonomously, without any exchange of control messages between neighboring nodes. On the other side, the number of cells allocated for communication link is fixed, and equal to 1.

C. CoAP PROTOCOL

From an architectural point of view, CoAP consists of two different sub-layers, namely, the *request/response* sub-layer and the *message* sub-layer. The request/response sublayer manages CoAP requests by executing the required method on the requested resource. Instead, the message sub-layer is responsible for managing the message exchange, referred to as *CoAP transaction*, between the two endpoints, over the UDP protocol. Specifically, the message sub-layer provides: (i) duplicate detection, and (ii) reliable message delivery, if enabled.

CoAP defines four different messages types, namely *Non-Confirmable* (NON), *Confirmable* (CON), *Acknowledgment* (ACK) and *Reset* (RST). When unreliable delivery is required, NON messages are used; they do not require a confirmation from the receiver. Instead, when reliable end-to-end delivery is enabled, requests/responses are transported within CON messages. Upon receiving such a message, the destination must reply with an ACK message, using either the *separate* or *piggyback* mode. In separate mode, the receiver sends an empty ACK message after the reception of a request, and defers the transmission of the actual response in a separate CON message, when it is ready. In piggyback mode, the receiver waits for the response to be ready and sends it back directly, in the body of the ACK message. The sender is responsible for re-transmitting messages that are not acknowledged, after a timeout, until a maximum number of re-transmissions is reached. Finally, RST messages are exploited, instead of ACKs, when the recipient is unable to process a CON message.

Figure 2 shows two examples of CoAP transactions. In the first case (left side), both the CON and ACK messages are received correctly and the transaction terminates successfully after the first transmission. In the second example (right side), the CON message is not received correctly by the destination and it is re-transmitted by the sender after the timeout. The

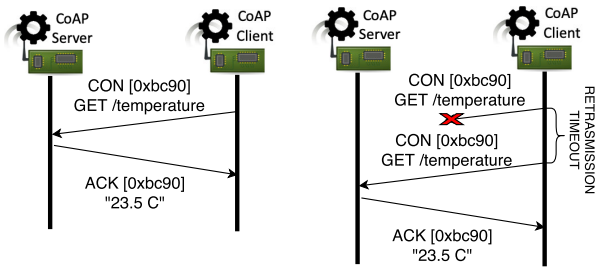


FIGURE 2. CoAP transaction examples.

transaction terminates successfully but requires more time to be completed.

Since CoAP relies on the UDP protocol, that does not implement any congestion control, it must include a congestion control mechanism to regulate the amount of traffic transmitted by the sender in order to prevent or manage possible network congestion. CoAP specifications define a default congestion control algorithm. Then, many alternative congestion control algorithms for CoAP have been proposed in the literature. Among them, the most popular one is the CoCoA algorithm, originally proposed in [19] and further extended in [2].

D. CoAP DEFAULT CONGESTION CONTROL

The default congestion control included in CoAP specifications [1] imposes a restriction on the number of parallel message exchanges, as well as 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 acknowledgment. Secondly, the algorithm limits the transmission rate of outgoing messages by using an exponential backoff between the consecutive re-transmissions 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 acknowledgment timeout, set by default to 2s, and T_{rand} is a randomization 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 re-transmissions for a CoAP message, before considering the message as a failure.

E. CoCoA CONGESTION CONTROL

The CoCoA congestion control, originally proposed in [19] and further extended in [2], is the most popular congestion control algorithm. It is composed by the following three main functions:

- a policy to calculate RTO;
- a backoff policy to set the RTO for re-transmissions;
- an aging policy for the status information.

In order to compute the RTO, CoCoA uses the measured Round Trip Times (RTTs), and considers only measurements on messages delivered without retransmissions. In the context of lossy networks, however, many transactions are expected to experience retransmissions, thus reducing the probability of obtaining significant RTT measurements. For this reason, CoCoA considers also transactions that experienced retransmissions to obtain a more accurate RTT estimation. Specifically, two RTO values are computed, namely, a *strong RTO*, using RTT samples from transactions that were successful at the first attempt, and a *weak RTO*, 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 (re)transmission that generated the 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 retransmissions.

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

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

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:

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

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

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

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

The following set of values are recommended: $\alpha = 0.25$, $\beta = 0.125$, $K_{strong} = 4$, $K_{weak} = 1$, $\lambda_{strong} = 0.5$, and $\lambda_{weak} = 0.25$. Such configuration was selected after a set of experiments in CSMA-CA networks, where the values resulting in the highest performance was selected [19].

$RTO_{overall}$ 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:

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

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 outdated 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:

$$RTO_{overall} = \frac{2 + RTO_{overall}}{2} \tag{10}$$

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 [20], and multiple alternative definitions have been proposed recently [21], it is widely recognized as the most promising congestion control strategy for CoAP.

III. SIMULATION SETUP

In this Section, we present the simulation setup and methodology used in our analysis to compare the considered congestion control algorithms in 6TiSCH networks. We used simulation as it allows to evaluate a large number of scenarios and to investigate the impact of many different factors, such as traffic period (i.e., Offered Load), network size, and Scheduling Function.

The reference scenario is shown in Figure 3. We considered a typical IoT system that comprises a Low-power and Lossy Network (LLN) connected to the Internet through a border router, which also behaves as the RPL root node. We assume that IoT devices are equipped with a radio, compliant to the IEEE 802.15.4 standard, for low-power wireless communication, and implement the 6TiSCH architecture and protocols described in Section II.

In our simulation experiments, IoT nodes periodically send their data to a collector node. To this end, each IoT device behaves as a CoAP client, i.e., it (periodically) issues a request (e.g., a POST request), with its data in the payload, to the collector node acting as the CoAP server. The period *P* between two subsequent requests determines the Offered Load, i.e., the sum of the cumulative traffic injected by all the nodes in the network. The CoAP server is typically external to the LLN and executed on a powerful host, e.g., an embedded system or a cloud server. For simplicity, in our experiments we assumed that the CoAP server runs on the border router.

To carry out our analysis, we exploited the Contiki-NG Operating System (OS),[1] a popular OS for IoT devices that runs on a wide range of hardware platforms. Contiki-NG includes the basic components of the 6TiSCH architecture, where we implemented the two considered SFs (B-MSF and ALICE). Specifically, for our simulations we exploited Cooja [22], a network emulator that is part of the Contiki-NG suite. In all the experiments, we used the RPL routing protocol with default parameters to allow multi-hop communication. The routing protocol is tuned according to its Contiki-NG default configuration.

In our analysis, we considered the following performance metrics.

- **Transaction Delivery Ratio (TDR)**, defined as the ratio between the number of successful CoAP transactions and the total number of CoAP transactions issued during each experiment.

[1] https://github.com/contiki-ng/contiki-ng

TABLE 1. Simulation settings.

Simulation Settings	
Simulation Duration	60 min
Warmup Duration	12 min
Application level buffer	32
Network level buffer	16
L_0	101 slots
L_1	31 slots
L_2	29 slots
RPL Mode	Storing
RPL Objective Function	MRHOF
MSF MAX_NEC	48
MSF LIM_NCU_PARENT_LOW	12 (25% of MAX_NEC)
MSF LIM_NCU_PARENT_HIGH	36 (75% of MAX_NEC)
MSF LIM_NCU_CHILD_LOW	12 (25% of MAX_NEC)
MSF LIM_NCU_CHILD_HIGH	36 (75% of MAX_NEC)

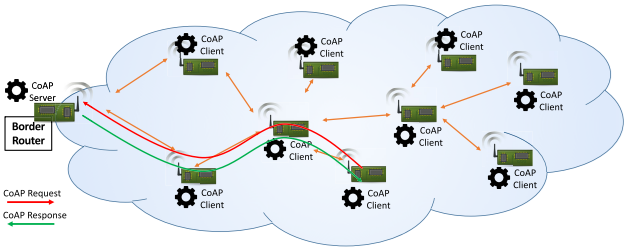


FIGURE 3. Experiment scenario.

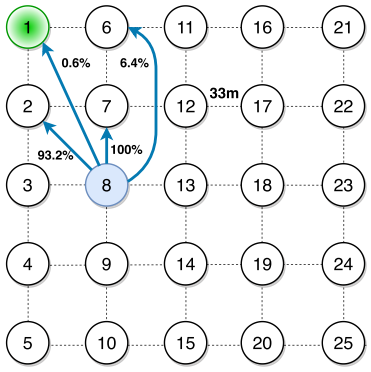


FIGURE 4. Example of 5 × 5 Grid topology and nominal Packet Delivery Probability (PDP).

- **End-to-end Transaction Delay**, defined as the time interval between the transmission of the CoAP request and the reception of the related ACK message at the CoAP client in a successful transaction. This metric will be reported in the form of 95^{th} percentile of its distribution, in order to provide an estimate of the delay experienced in the worst case.

The goal of our experiments is to assess the performance of the considered congestion control algorithms in 6TiSCH networks and investigate their interplay with the SF in different scenarios, characterized by increasing Offered Loads. Specifically, we want to test their ability to react to congestion in 6TiSCH networks.

For our analysis, we considered a grid network topology with a varying number of nodes (N), deployed equally distant, i.e., 33m from each other. In Figure 4 we display an example of a network topology with $N = 25$. To make the communication channel realistic, each link was modeled through the *Multi-path Ray-tracer Medium* (MRM) model [22]. MRM implements ray-tracing techniques with various propagation effects (e.g., multi-path, refraction, diffraction, etc.), and associates a *Packet Delivery Probability* (PDP) to each link, which changes over time due to propagation effects and concurrent transmissions. The nominal PDP for each specific kind of link is shown in Figure 4.

All the parameter settings used in simulation experiments are summarized in Table 1. We considered different slotframe lengths, namely 29, 47 and 101 slots. However, for the sake of brevity, below we will show only the results obtained with slotframe length of 29, as the results obtained with the other values lead to the same conclusions.

Each simulation experiment was run for 1 hour. Each CoAP client was programmed to start generating CoAP requests after 6 minutes from the beginning of the experiment. This allows to complete the network formation phase (i.e., TSCH join and RPL formation) that may take up to some minutes [6]. In order to obtain statistically sound results, we performed 10 independent replicas for each experiment. For each metric, we derived the confidence interval with a 95% confidence level.

IV. ANALYSIS OF CoAP CONGESTION CONTROL ALGORITHMS IN 6TiSCH

In this and next Sections, we discuss the results obtained in our simulation analysis. In all the experiments, we considered two different network configurations with a different number of IoT nodes N, namely 25 and 64, in order to analyze the performance on small and large networks. We also assumed that each CoAP client generates CoAP requests towards the CoAP server periodically, with a period P in the range [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20]s. This corresponds to different Offered Loads.

We started our analysis with a set of experiments aimed at investigating the impact of the Scheduling Function on the performance of the considered congestion control algorithms.

Figure 5 shows the Transaction Delivery Ratio provided by the default and CoCoA algorithms, for a network with 25 (left side) nodes and 64 nodes (right side). The trend is quite similar in both cases. We can observe that the two considered congestion control algorithms exhibit roughly the same performance, when B-MSF is used for scheduling communication resources. Instead, when using ALICE, CoCoA provides a significant advantage, with respect to the default algorithm. This is because B-MSF is adaptive and, hence, it allocates communication resources depending on the operating conditions. Instead, ALICE is a static SF that allocates a fixed amount of resources on all the links in the network, irrespective of the traffic conditions. Hence, when the total Offered Load increases over a certain threshold (i.e., for low P values and/or high number of nodes), the allocated resources may not be enough to manage all the traffic passing through a certain link. Under such overload conditions, the CoCoA algorithm performs better than the default algorithm, as it better handles congestion.

This conclusion is also supported by Figure 6 showing the 95^{th} percentile of the end-to-end Transaction Delay. When TSCH cells are allocated through B-MSF, the default and CoCoA algorithms exhibit similar performance, with the default algorithm introducing a slightly lower delay than CoCoA. Instead, when using ALICE, CoCoA always provides a lower end-to-end Transaction Delay.

In conclusion, the previous results highlight that CoCoA can offer some advantage, over the default congestion control algorithm, in 6TiSCH networks. The advantage, however, is limited to some cases and is significantly lower to what observed in other scenarios, where CSMA-CA based networks have been considered [19]. Specifically, when communication resources are adjusted dynamically (through B-MSF), CoCoA does not provide any significant advantage over the default algorithm, both in terms of Transaction Delivery Ratio and Transaction Delay.

In the following section we analyze this behavior in detail to better understand the reason why CoCoA does not perform as expected in 6TiSCH networks. This will allow us to propose some modifications in order to improve its performance in this kind of IoT networks.

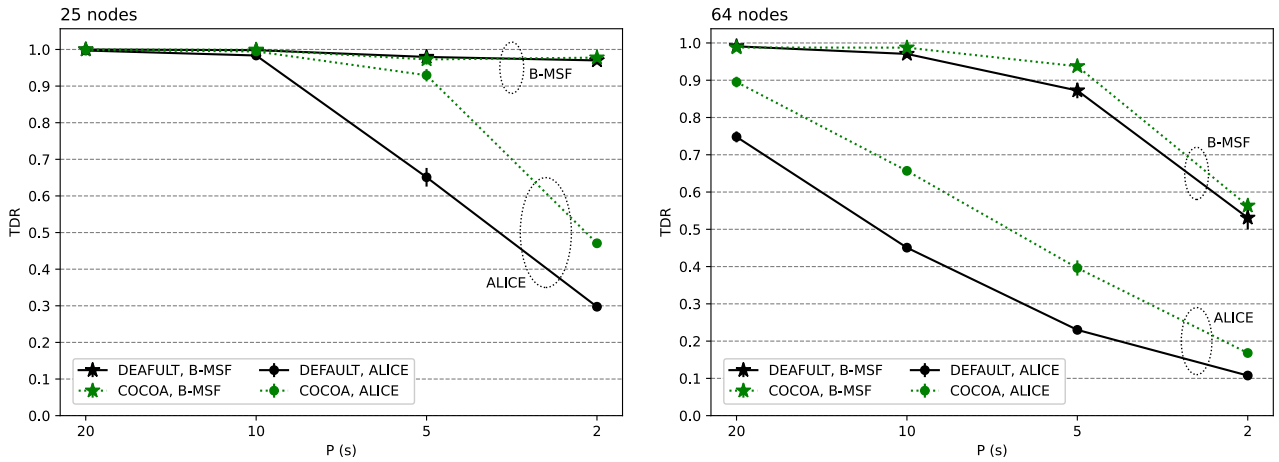


FIGURE 5. Transaction Delivery Ratio (TDR) for CoAP default and CoCoA with 25 nodes (left) and 64 nodes (right).

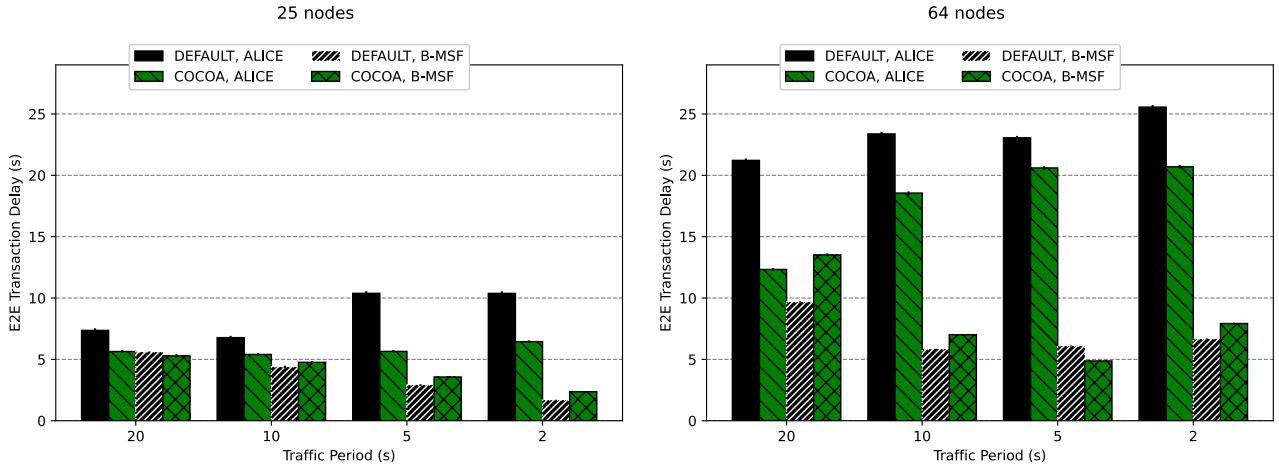


FIGURE 6. End-to-end (E2E) Transaction Delay for CoAP default and CoCoA with 25 nodes (left) and 64 nodes (right).

V. LIMITS OF CoCoA IN 6TiSCH NETWORKS

Previous evaluations of CoCoA [2] have been carried out referring to the traditional IoT architecture based on the CSMA-CA MAC protocol. The algorithm itself was conceived for this kind of networks, and its parameters were set based on the results obtained in such an environment. At the best of authors' knowledge, the results presented in Section IV are the first ones assessing the performance of CoCoA in 6TiSCH networks. From these results, it clearly emerges that CoCoA does not outperform the default algorithm in a significant way, especially when communication resources are adjusted dynamically, according to traffic conditions. This can be explained with the different approach of TSCH access that guarantees the two following advantages over the CSMA-CA random-based access.

- *Collisions on data transmissions are minimal*. In TSCH networks almost all data packets are transmitted in dedicated timeslots and, hence, collisions never occur. Only sporadically (e.g., at network bootstrap), some data packets are transmitted in shared slots where collisions may occur. For this reason, packet loss is mainly caused by buffer overflow, rather than collisions due to concurrent transmissions.
- *End-to-end delays are more stable over time*. Collision free transmissions result in a more stable environment, with less variability. As a result, end-to-end delays are more stable over time, as they are only caused by queuing delays experienced at intermediate nodes.

As mentioned above, CoCoA was conceived to work in a CSMA-CA environment where, instead, collisions are very frequent, especially at high Offered Loads. For this reason, the algorithm has been designed and configured to react to bursty losses and variable end-to-end delays. While this is very beneficial in a CSMA-CA environment, it has no effects in a 6TiSCH network, and may be even harmful.

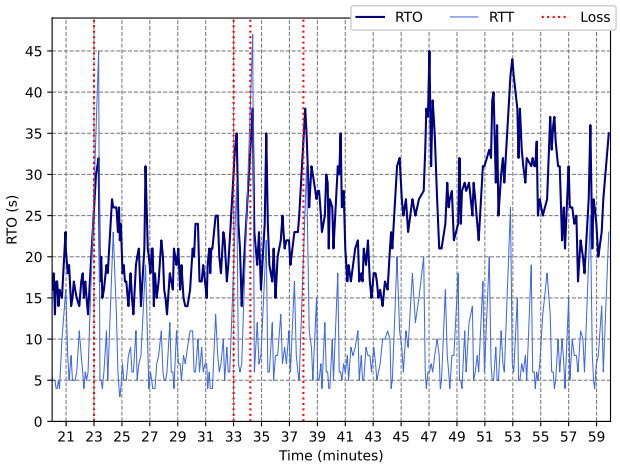


FIGURE 7. RTT and RTO, over time, CoCoA.

The main drawback of CoCoA, when operating in a 6TiSCH network, is the weight assigned to RTT samples from retransmissions through the weak estimators. In a CSMA-CA environment, where losses are likely to be caused by collisions, it is appropriate for the sender to backoff immediately for a long period of time (e.g., several seconds), in order to avoid subsequent concurrent transmissions. In CoCoA, this is achieved by handling RTT samples from transactions that experienced at least one retransmission differently via the weak estimator and by tuning the parameters of the algorithm in order to give more weight to the weak estimator in the RTO calculation. This results in long periods, where nodes select large RTO values every time a retransmission occurs.

In 6TiSCH networks, however, congestion is typically due to an insufficient allocation of TSCH cells, compared to the amount of traffic that each node has to manage. When a dynamic SF is used, like B-MSF, such situations are usually solved by the SF in a short amount of time (e.g., tens of seconds). Congestion still causes increased queuing delays and, in extreme cases, packet dropping at nodes. However, it has a shorter lifespan, as it is managed and solved locally by the SF that allocates more resources for data transmission on the interested links. Hence, selecting a large RTO value for long periods every time a retransmission occurs is not appropriate in 6TiSCH networks, as there is no collision-caused congestion that requires a long backoff to be resolved.

In order to get more insight on this, we analyzed the behavior of a single node in a simulation experiment. Specifically, we considered a leaf node that is located 7 hops away from the root node. We selected this node because it is quite distant from the root node. This exacerbates the CoCoA behavior in reacting to sporadic losses, since the long path towards the root node leads to fluctuations in the resource allocation at intermediate nodes.

Figure 7 shows the RTT samples measured at the considered node, and the corresponding RTO values calculated by CoCoA. The results are extracted from one simulation run with B-MSF in the scenario with 64 nodes and $P = 2$s. We can observe that, as soon as a loss occurs (we can notice a first one at minute 23) – e.g., due to buffer overflow caused by an insufficient number of cells allocated – CoCoA reacts by increasing the RTO. As other losses occur (three losses are experienced at minute 33, 34 and 38, respectively) – e.g., because the SF (B-MSF in this case) takes some time to react thus causing some buffer overflows – the RTO is further increased. The gap between the RTO and the RTT is kept higher even if the RTT decreases quickly after such losses, thanks to the allocation of new cells by the SF. This can be noticed especially in the last part of the experiment, where RTO and RTT values are apparently uncoupled: RTO is set to a value that is quite higher than the current RTT value. Clearly, this reduces the number of requests that the source node can send, thus limiting the overall performance.

TABLE 2. 6CoCoA parameters.

Parameter	6CoCoA	CoCoA
λ_{strong}	0.75	0.5
λ_{weak}	0.1	0.25
Z_{strong}	2	4
Z_{weak}	0.5	1
α	0.9	0.125
β	0.5	0.25
Lower VBF Threshold	20s	1s
Upper VBF Threshold	50s	3s
Large VBF	2.2s	2.5s
Medium VBF	1.5s	2s
Small VBF	1.1s	1.5s

VI. 6CoCoA: CoCoA FOR 6TiSCH NETWORKS

In order to improve the performance of CoCoA in 6TiSCH networks, we finely tune the CoCoA algorithm by changing the default settings adopted in CSMA-CA networks and originally proposed in [19]. It is important to highlight that no modification to the algorithm is introduced here. Instead, we only perform a reconfiguration of its parameter values in order to find the appropriate configuration for a 6TiSCH environment. Throughout, this configuration will be referred to as 6CoCoA.

To find out the most appropriate parameter values for CoCoA in 6TiSCH networks, we followed a number of guidelines that comes from the lessons learned through the previous analysis. These guidelines, and the related motivations, are described below.

- *The weight of the RTT samples corresponding to retransmissions should be reduced.* A large weight for RTT samples collected from retransmissions was set in CoCoA through a λ_{weak} close to 0 and a λ_{strong} close to 1. These values result in selecting an RTO that is close to the last RTT value when a retransmission occurred. In 6TiSCH networks, retransmissions are less frequent and not caused by collisions. Hence, they do

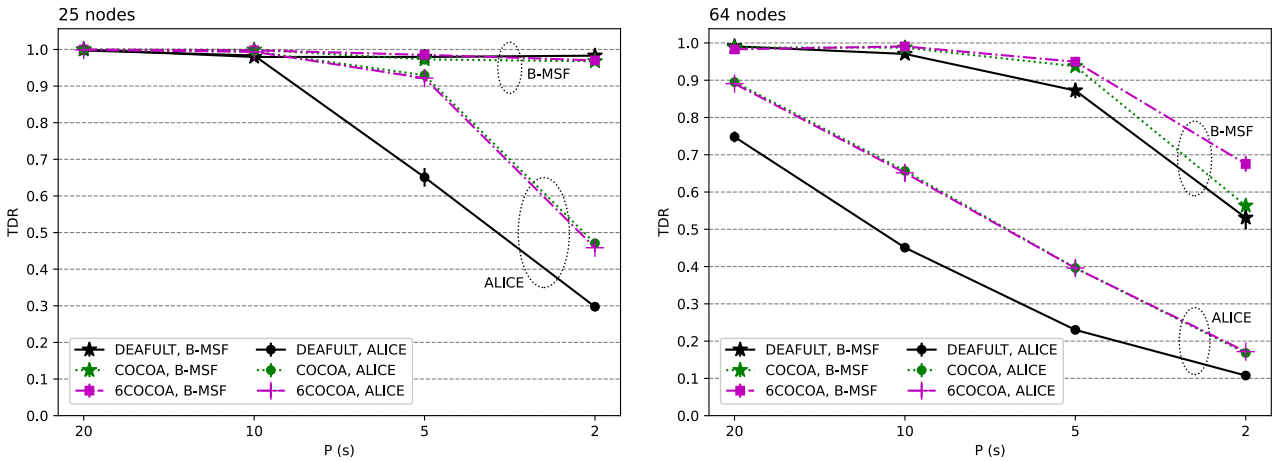


FIGURE 8. Transaction Delivery Ratio (TDR) for CoAP default CoCoA and 6CoCoA with 25 nodes (left) and 64 nodes (right).

not require to immediately back-off to reduce the effects of collisions. Guideline: increase the value of λ_{weak} and decrease the value of λ_{strong}.

- *RTT history and its variability should have a reduced weight in calculating RTO.* The reason for sporadic peaks in RTT are sporadic losses. For this reason, they should not cause an increase of the overall RTO for a long period, but only for a short time interval, just to allow the SF to adjust the amount of resources allocated to the node. Guideline: reduce the α and β factors in order to increase the weight of the last value w.r.t. history.
- *RTT variability should have less impact in the RTO calculation.* CoCoA was set to provide a significant weight to the variability of RTT through high values of the K_{strong} and K_{weak} parameters. The rationale was to make the algorithm to react to the first increase of the RTT by rapidly increasing the RTO caused by the first collisions. 6TiSCH networks, however, are characterized by a lower RTT variability as collisions are avoided. Guideline: minimize the impact of RTT variability in RTO calculation by reducing K_{strong} and K_{weak}.
- *Variable Back off range should be reduced.* 6TiSCH networks do not require long back-off periods to recover from congestion as transmissions do not experience collisions. Shorter variable back-off are desirable in order to increase the reactivity after short term congestion. Guideline: reduce the threshold for VBF and the back-off values.

Based on the above guidelines, we performed a large set of simulation experiments, where we considered and evaluated a wide range of parameter values. In each experiment, we measured the Transaction Delivery Ratio and end-to-end Transaction Delay. For these experiments, we considered the same scenarios adopted in our previous experiments. Following the same approach used in [19], we finally selected the set of parameter values, as those resulting in the highest performance. The selected parameter set is shown in Table 2. For comparison, we also report the default parameters values used in the original CoCoA.

VII. EVALUATION OF 6CoCoA

In this Section, we compare the performance achieved when using 6CoCoA with that obtained with the original CoCoA configuration. As above, we considered two different SFs, namely B-MSF and ALICE, and different network sizes.

Figure 8 shows the TDR provided by the default, CoCoA and 6CoCoA congestion control algorithms, in networks of different size, i.e., $N = 25$ (left figure) and $N = 64$ (right figure). We can observe that, when ALICE is used as SF, there is no real benefit in using 6CoCoA, with respect to the original CoCoA. This is because packet losses are originated by an insufficient allocation of resources at nodes that will persist forever, since ALICE is not adaptive. When using B-MSF, instead of ALICE, the TDR is significantly higher, thanks to the ability of the SF to adapt the resource allocation to the traffic conditions (as already highlighted). In addition, using 6CoCoA may provide a further benefit, especially when the network size is large ($N = 64$) and the Offered Load is high ($P = 2s$). In the latter case, we observed an increase of roughly 15% when using 6CoCoA, instead of CoCoA.

In terms of end-to-end Transaction Delay, Figure 9 shows that 6CoCoA introduces a delay that is lower than, or equal to, that introduced by the original CoCoA in all the considered scenarios. Also in this case, the major gain is obtained, with B-MSF, in the most challenging scenario (i.e., $N = 64$ and $P = 2s$). In this specific scenario, we observed a reduction in the Transaction Delay of approximately 25%, when using 6CoCoA.

The previous results confirm that the fine tuning of CoCoA parameters provides a certain advantage in 6TiSCH networks, especially in challenging scenarios with large number of nodes and/or high Offered Load, as they allow to fit better the characteristics of scheduled 6TiSCH networks.

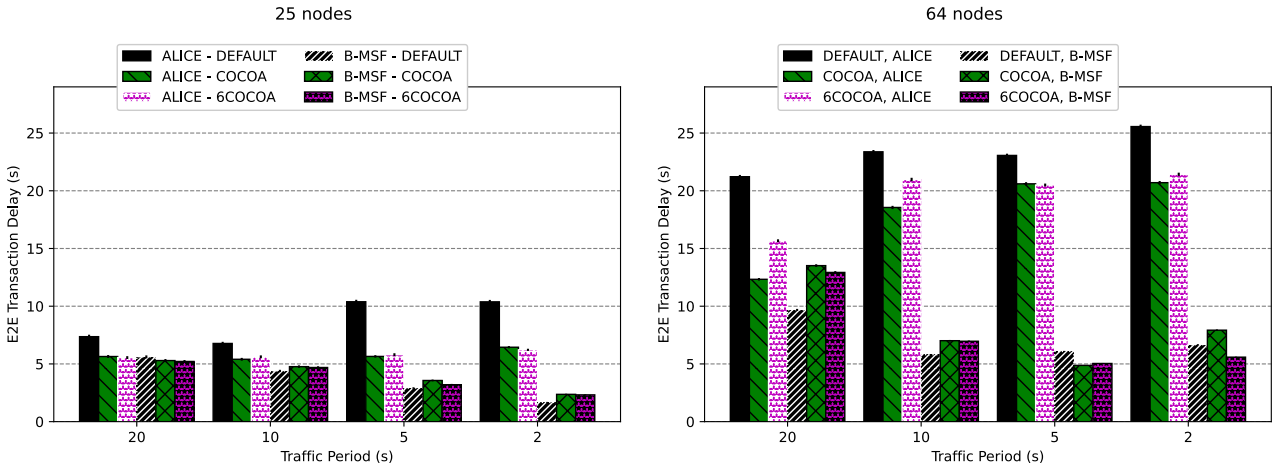


FIGURE 9. End-to-end (E2E) transaction delay for CoAP default, CoCoA and 6CoCoA with 25 nodes (left) and 64 nodes (right).

VIII. CONCLUSION

In this paper we have compared, through simulation, two congestion control algorithms for the Constrained Application Protocol (CoAP), namely CoAP default and CoCoA, in 6TiSCH-based IoT networks. In our analysis we have considered two different scheduling algorithms for 6TiSCH (i.e., B-MSF and ALICE), taking a different approach in allocating communication resources, in order to investigate also the interplay between congestion control and scheduling. Our analysis has emphasized that, despite its higher complexity, CoCoA may not provide a significant benefit, in comparison with the default algorithm, in 6TiSCH networks. We have shown that this is due to the to the different nature of congestion in scheduled 6TiSCH networks, with respect to traditional IoT networks, based on CSMA-CA, for which it was originally conceived.

To improve the performance of CoCoA in 6TiSCH networks, we have proposed 6CoCoA, a version of CoCoA whose parameter values are specifically tailored to the 6TiSCH environment. We have observed, through simulation, that this version actually provides better performance, with respect to the original version, in terms of both Transaction Delivery Ratio (up to 15%) and end-to-end Transaction Delay (up to 25%).

As future work, we plan to investigate the possibility to design from scratch a new congestion control algorithm specifically conceived for 6TiSCH networks.

ACKNOWLEDGMENT

This work was supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab and FoReLab projects (Departments of Excellence).

REFERENCES

[1] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, document RFC 7252, Jun. 2014.

[2] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP," *Ad Hoc Netw.*, vol. 33, pp. 126–139, Oct. 2015.

[3] *IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANS) 1: MAC Sublayer*, IEEE Standard 802.15.4e-2012 (Amendment to IEEE Standard 802.15.4-2011), 2012, pp. 1–225.

[4] *IEEE Standard for Low-Rate Wireless Networks*, Standard IEEE 802.15.4-2020, Jul. 2020.

[5] G. Anastasi, M. Conti, and M. Di Francesco, "A comprehensive analysis of the MAC unreliability problem in IEEE 802.15.4 wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 7, no. 1, pp. 52–65, Feb. 2011.

[6] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "An evaluation of the 6TiSCH distributed resource management mode," *ACM Trans. Internet Things*, vol. 1, no. 4, pp. 1–31, Jul. 2020.

[7] C. Vallati, F. Righetti, G. Tanganelli, E. Mingozzi, and G. Anastasi, "Analysis of the interplay between RPL and the congestion control strategies for CoAP," *Ad Hoc Netw.*, vol. 109, Dec. 2020, Art. no. 102290.

[8] S. Bolettieri, G. Tanganelli, C. Vallati, and E. Mingozzi, "PCoCoA: A precise congestion control algorithm for CoAP," *Ad Hoc Netw.*, vol. 80, pp. 116–129, Nov. 2018.

[9] G. A. Akpakwu, G. P. Hancke, and A. M. Abu-Mahfouz, "CACC: Context-aware congestion control approach for lightweight CoAP/UDP-based Internet of Things traffic," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, Feb. 2020, Art. no. e3822.

[10] I. Jarvinen, I. Raitahila, Z. Cao, and M. Kojo, "FASOR retransmission timeout and congestion control mechanism for CoAP," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.

[11] S. Deshmukh and V. T. Raisinghani, "AdCoCoA-adaptive congestion control algorithm for CoAP," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–7.

[12] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. R. Dujovne, *6TiSCH Minimal Scheduling Function (MSF)*, document RFC 9033, May 2021.

[13] S. Kim, H.-S. Kim, and C. Kim, "Alice: Autonomous link-based cell scheduling for TSCH," in *Proc. 18th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2019, pp. 121–132.

[14] P. Thubert, *An Architecture for IPv6 Over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)*, document RFC 9030, May 2021.

[15] Q. Wang, X. Vilajosana, and T. Watteyne, *6TiSCH Operation Sublayer (6top) Protocol (6P)*, document RFC 8480, Nov. 2018.

[16] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "Analysis of distributed and autonomous scheduling functions for 6TiSCH networks," *IEEE Access*, vol. 8, pp. 158243–158262, 2020.

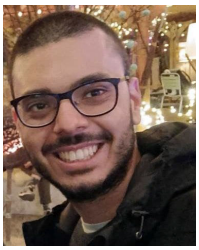
[17] G. Montenegro, J. Hui, D. Culler, and N. Kushalnagar, *Transmission of IPv6 Packets Over IEEE 802.15.4 Networks*, document RFC 4944, Sep. 2007.

[18] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, document RFC 6550, Mar. 2012.

[19] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoAP congestion control for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 154–160, Jul. 2016.
[20] S. Bolettieri, C. Vallati, G. Tanganelli, and E. Mingozzi, "Highlighting some shortcomings of the CoCoA+ congestion control algorithm," in *Proc. AdHoc-Now*. Messina, Italy: Springer, Sep. 2017, pp. 213–220.
[21] A. Pramanik, A. K. Luhach, I. Batra, and U. Singh, "A systematic survey on congestion mechanisms of CoAP based Internet of Things," in *Advanced Informatics for Computing Research*. Singapore: Springer, 2017, pp. 306–317.
[22] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 641–648.

FRANCESCA RIGHETTI received the master's degree in computer engineering and the Ph.D. degree in information technology from the University of Pisa, Italy, in 2017 and 2021, respectively. She is currently an Assistant Professor with the Department of Information Engineering, University of Pisa. Her research interests include the Internet of Things (IoT), the Industrial Internet of Things (IIoT), and security in IoT. She has been involved in several national and international research projects, including EdgeFlooding (funded by NGI Atlantic), "ECOAP: Experimental assessment of congestion control strategies for the Constrained Application Protocol" (funded by the EC within the Horizon 2020 Programme), SmartEP, SMARIERS, and STINGRAY. She has served as the TPC Co-Chair of the IEEE International Workshop in Smart Service Systems (SmartSyS 2022), Co-Located with IEEE SMARTCOMP 2022. In addition, she has been involved in the TPC of many international conferences and workshops, including IEEE SMARTCOMP, IEEE CCNC, IEEE MSN, and IEEE MELECON.

CARLO VALLATI (Member, IEEE) received the master's (magna cum laude) and Ph.D. degrees in computer systems engineering from the University of Pisa, in 2008 and 2012, respectively. In 2010, he visited the Computer Science Department, University of California at Davis. He is currently an Associate Professor with the Department of Information Engineering, University of Pisa. He is the coauthor of more than 70 peer-reviewed papers in international journals and conference proceedings. He has been involved in multiple national and international research projects, and several research projects supported by private industries. He has served as a program committee member for more than 30 international conferences and workshops. He has served as the Workshop Chair for the IEEE IoT-SoS and IEEE SmartSys workshops. He has served as the TPC Co-Chair for IEEE SMARTCOMP 2020 and the General Vice Chair for PERCOM 2022. He is on the editorial board of three international journals, the *Ad Hoc Networks*(Elsevier), *Journal of Reliable Intelligent Environments* (Springer), and *Applied Sciences* (MDPI). He is the Coordinator of the Cloud Computing, Big Data and Cybersecurity Crosslab founded in the framework of the Departments of Excellence (Dipartimenti di Eccellenza) funded by the Italian Ministry of Education, University and Research (Ministero dell'Istruzione dell'Università e della Ricerca).



DAVIDE RASLA received the master's degree in embedded computing systems from the University of Pisa, Italy, in 2021. His master's thesis Analysis of the Interplay between 6TiSCH Scheduling and CoAP Congestion Control strategies contributed for this work of research. He is currently working as a Software Engineer. His research interests include applications of the Internet of Things (IoT), the Industrial Internet of Things (IIoT), and wireless communications.

GIUSEPPE ANASTASI (Member, IEEE) was the Head of the Department of Information Engineering (DII), University of Pisa, Italy, from 2016 to 2020. From 2015 to 2018, he also served as the (Founding) Director of the CINI Smart Cities National Laboratory, a nation-wide competence center on smart cities and communities, consisting of 29 nodes (local labs) located at different Italian universities. He is currently a Professor of computer engineering with DII, University of Pisa. He is also the Director of the Industry 4.0 CrossLab, funded by the Italian Ministry of Education and Research (MIUR) in the framework of the Departments of Excellence Program, which consists of six interdisciplinary and integrated research laboratories (CrossLabs) covering all the key areas of Industry 4.0. He is a Co-Editor of two books: *Advanced Lectures in Networking* (LNCS 2497, Springer, 2002) and *Methodologies and Technologies for Networked Enterprises* (LNCS 7200, Springer, 2012). He has published more than 160 research papers in the area of computer networking and distributed systems. His publications have received more than 10,000 citations, according to Google Scholar (H-index=44). His current scientific interests include the Internet of Things, fog/edge computing, cyber-physical systems, cybersecurity, and smart environments.

He is currently serving as a Steering Committee Member of the IEEE SMARTCOMP conference. He was the General Chair of IEEE SMARTCOMP 2018 and IEEE WoWMoM 2005; the Program Chair of IEEE SMARTCOMP 2016, IEEE MSN 2015, IFIP/IEEE SustainIT 2012, IEEE PerCom 2010, and IEEE WoWMoM 2008. He has co-founded many successful international workshops and conferences. Previously, he served as an Area Editor of *Pervasive and Mobile Computing* (PMC, 2007–2016); an Associate Editor of *Sustainable Computing* (SUSCOM, 2010–2015); and an Area Editor of *Computer Communications* (ComCom, 2008–2010).

Open Access funding provided by 'Università di Pisa' within the CRUI CARE Agreement