Experimental Evaluation of Congestion Control for CoAP Communications without End-to-End Reliability

August Betzler[a], Javier Isern[b], Carles Gomez[c], Ilker Demirkol[c], Josep Paradells[c]

[a]*I2CAT Foundation, Barcelona, Spain*
[b]*Urbiotica, Barcelona, Spain*
[c]*Department of Network Engineering, Universitat Politecnica de Barcelona, Barcelona, Spain*

Abstract

The Constrained Application Protocol (CoAP) has been designed by the Internet Engineering Task Force (IETF) for Internet of Things (IoT) communications. CoAP is a lightweight, request/response-based RESTful protocol that has been tailored to fulfill the requisites of IoT environments, such as severely limited device hardware and link capacities. In IoT networks, congestion is a major issue that causes performance losses or may even render the network useless. Thus, the use of a congestion control mechanism is essential for the performance of such networks. CoAP defines a very basic congestion control mechanism for the reliable exchange of messages between endpoints, however it does not specify congestion control for communications without end-to-end reliability, even though the latter represent a relevant share of CoAP communications. Two extensions to CoAP, Observe and Simple CoAP Congestion Control/Advanced (CoCoA), introduce rate control mechanisms for such communications yet these extensions have not yet been compared or evaluated. In this paper, we empirically evaluate these rate control mechanisms for unreliable CoAP communications between devices over emulated GPRS/UMTS links and in a real IEEE 802.15.4 multihop testbed of constrained devices. The results show that in contrast to Observe, CoCoA performs better than, or at least similarly to, default CoAP in terms of both packet delivery ratio and delay in all analyzed scenarios.

Keywords: Internet of Things, Congestion Control, 6LoWPAN, RPL, CoAP, Contiki

Email addresses: `august.betzler@i2cat.net` (August Betzler), `javier.isern@urbiotica.com` (Javier Isern), `carlesgo@entel.upc.edu` (Carles Gomez), `ilker.demirkol@entel.upc.edu` (Ilker Demirkol), `josep.paradells@entel.upc.edu` (Josep Paradells)

1. Introduction

The Constrained Application Protocol (CoAP) has been designed by the Internet Engineering Task Force (IETF) to be the de-facto protocol for Internet Protocol (IP)-based communications between devices in the Internet of Things (IoT). The IoT aims to interconnect billions of Internet-capable smart objects, building a bridge between the physical and the virtual world. Constrained devices represent a key element of IoT networks that use low-power wireless technologies to communicate over the Internet. As a consequence of the limited radio capacities and the tight hardware limitations of constrained devices, the risk of network congestion in IoT networks is high, representing a recurring phenomenon. Moreover, IoT traffic patterns, such as the bursty transmission of messages as a reaction to an event detected in the network or periodical transmission of status information from large groups of devices, tend to result in congestion. Network congestion can deteriorate the performance of a network significantly, manifesting in increased packet latencies or packet losses, while a network may even render useless if congestion collapse occurs.

The CoAP base specification [1] addresses this issue by defining a basic congestion control mechanism for the reliable exchange of messages. However, for unreliable transmissions (i.e., without end-to-end reliability), the CoAP base specification does not determine any congestion control mechanism. Due to the simplicity of congestion control for reliable communications and the missing control mechanisms for unreliable message exchanges, the CoAP base specification envisages the definition of alternative advanced congestion control mechanisms in form of extensions. Accordingly, the IETF released RFC 7641 [2] that extends CoAP with *Observe*, a publish-subscribe mechanism. Observe defines a basic congestion control mechanism for unreliable CoAP communications, by introducing an upper limit on the allowed rate of outgoing messages. Another IETF document that extends CoAP with an adaptive, advanced congestion control mechanism for reliable and unreliable CoAP communications is the Simple CoAP Congestion Control/Advanced (CoCoA) Internet Draft specification [3].

Since CoAP is a novel protocol (the CoAP specification was completed in 2014), investigation in the area of congestion control for CoAP has been limited as of the writing. While published studies and proposals have focused on congestion control for reliable CoAP communications [4, 5, 6, 7, 8, 9], congestion control for unreliable CoAP exchanges have been considered to a very limited extent.

To our knowledge, the only study on congestion control for unreliable CoAP communications to date is the preliminary work of this paper [10][1]. In [10], GPRS/UMTS-based communications were investigated, which are typical for machine-to-machine (M2M) communications in the IoT. The GPRS/UMTS technologies used in these initial investigations are typical for machine-to-machine (M2M) communications in the IoT. However they only represent a subset of typi-

[1]This paper is an extended version of [10].

cal IoT communication technologies, leaving aside other important technologies.

Thus, in this paper we extend the evaluations to another realistic IoT scenario to provide a more general comparative study of congestion control for unreliable CoAP communications. We carry out evaluations in a network of constrained devices that use IEEE 802.15.4. This scenario is significant since IEEE 802.15.4 is the default radio type used for constrained IoT devices, and also because it is based on a multihop topology, which complements the single-hop GPRS/UMTS scenario and involves different phenomena which cannot be found in the latter. To implement a typical use case with different patterns and degrees of traffic, we use one of the IoT-Lab [11] testbeds, which offer publicly available networks of constrained devices. The evaluation results confirm that, in contrast with the other analyzed approaches, CoCoA is able to reduce congestion and to maintain high performance in almost all the considered scenarios, thanks to its flexibility. Default CoAP and Observe fail to achieve good performance due to their lack of sensitivity to network conditions. We also conclude that the benefits of CoCoA are greater in the GPRS/UMTS scenario, which is more bursty and challenging in terms of congestion than the IEEE 802.15.4 one. CoCoA is capable of increasing the packet delivery ratio up to 44% in the GPRS/UMTS scenario and up to 14% in the IEEE 802.15.4 scenario.

This paper is structured as follows: In Section 2 we explain the details of congestion control for NON messages as implemented in Observe and CoCoA. Section 3 specifies the scope of the evaluations that are carried out in this paper and details the two network setups used for the experimental evaluations. The evaluation results are presented in Section 4. Section 5 concludes this paper and proposes future lines of work.

2. Congestion Control for CoAP

IoT applications are diverse regarding their requirements on reliable message delivery. In order to support a wide range of applications, CoAP was designed to operate over User Datagram Protocol (UDP), which offers a lightweight unreliable transport, and CoAP was provided with an optional mechanism for reliable delivery. Whereas default CoAP defines a basic congestion control mechanism for reliable transmissions, it does not define congestion control for unreliable ones. Proposals for congestion control mechanisms for unreliable CoAP communications are made in other IETF documents that extend the CoAP base specification. In this paper we consider the two IETF documents that introduce congestion control for unreliable end-to-end communications in CoAP: 1) the Observe extension [2] that by default relies on the exchange of NON-messages and introduces a static rate control. 2) The CoCoA [3] extension, which introduces advanced congestion control mechanisms for both CONs and NONs. In the following subsections, we explain the default CoAP congestion control mechanism, as well as the extensions provided by the IETF specifications Observe and CoCoA.

2.1. Default CoAP Congestion Control

CoAP message transmissions can be initiated with or without end-to-end reliability, by either using confirmable (CON) or non-confirmable (NON) messages, respectively. Since CoAP operates over UDP, which does not provide end-to-end reliability, CoAP introduces reliability in form of CON messages. In a CoAP transaction, a CON requires to be confirmed with an acknowledgment (ACK) by the destination endpoint of the data exchange within a retransmission timeout (RTO) interval. If no ACK is received before the RTO expires, the CoAP message is retransmitted. This procedure may be repeated up to 4 times, before the exchange is considered to have failed. CoAP applies congestion control by doubling the RTO of each retransmission, i.e., by applying a binary exponential backoff (BEB), to keep the rate of outgoing messages at bay. Further, it determines an upper limit of 1 allowed uncompleted transaction per endpoint. The fact that default CoAP does not set any limitations to the transmissions of NON messages supposes a high risk of congestion when using default CoAP in unreliable communications with typical IoT traffic patterns, such as bursts of packets.

The use of NON messages is common for many use cases, for example the transmission of notification messages as defined in Observe [2]. Independently from whether CONs or NONs are used, the traffic generated in large networks of constrained devices can easily exceed the network capacity, saturating the wireless medium and resulting in congestion. Further, the small buffers and limited packet queue sizes inherent to constrained devices may not be sufficiently large to handle generated and forwarded data packets. Nevertheless, the CoAP base specification does not define any congestion control mechanism for NON-based communications, leaving CoAP susceptible to suffer from the consequences of congestion.

2.2. Observe

Observe is an extension for the CoAP base specification that defines publish-subscribe interactions between CoAP endpoints. This mechanism is used when clients are interested in being kept up to date about resources located on a CoAP server. Once a client subscribes to the desired resource, the CoAP server holding the resource adds the subscriber to a list of endpoints that are interested in the specific resource.

Typically, a resource generates periodic or event-based notification messages to inform the subscribers about its state. When either the timer of a periodically checked resource expires or the status of the resource is updated as a result of an event, the server sends a notification message with the current status of the resource to each of its subscribers. In Observe, the messages sent during a notification process are normally transmitted as NONs.

In IoT use cases, for example a sensor network that collects environmental data, CoAP clients, e.g. Internet services, typically subscribe to several resources offered by the sensor nodes, such as temperature, humidity, or air pollution. In networks with many devices and numerous subscribers, the Observe notification process can lead to the transmission of a high number of messages: one notification message is sent per subscriber and observed resource.

In IoT networks that follow the paradigm of gathering and processing information, large numbers of sensor nodes may maintain multiple resource-states that are reported to a central device for further processing. This device is often referred to as sink node, which can be a border router (BR) or a proxy server that is capable of storing the gathered data and providing connectivity with the Internet. The many-to-one communication patterns we observe in networks composed of sensor nodes and sink, can easily lead to congestion as a result of the persistent transmission of large quantities of notification messages. To reduce congestion, Observe introduces a mechanism that dilutes the transmission of notifications over time: It limits the transmission frequency of notification messages, applying a static rate control that only allows 1 message to be sent every 3 s to each subscriber. This rate control is applied independently from the network state or any other parameter, such as the number of subscribers or the amount of observed resources. The static rate control behaves very conservatively, which can help to dilute the traffic in highly congested networks. Like as with the default initial RTO timeout used for CONs in the CoAP base specification, large packet travel times or processing delays are assumed in such networks. For such cases, the conservative approach followed by Observe is expected to work well. Yet, in scenarios with even larger or with smaller delays and RTTs, as well as in scenarios with high degrees of connection dynamicity (varying delays and varying degrees of congestion) Observe's rate control may not deliver a good performance. A dynamic approach that represents an alternative to this rigid rate control is proposed in CoCoA, which is introduced in the following subsection.

2.3. CoCoA

CoCoA extends the CoAP base specification with an alternative, advanced congestion control mechanism for CONs and NONs. It is based on using round-trip time (RTT) information that is measured from the exchange of CONs and corresponding ACKs to maintain a RTO estimation for each Internet device (destination endpoint) it communicates with. In general lines, a RTO for an endpoint ($RTO_{overall}$) is maintained and updated following principles similar to those of the RTO estimator defined in RFC 6298 [12] for the Transmission Control Protocol (TCP). However, in order to achieve a better performance in IoT networks, which incur specific types of communication patterns and conditions, CoCoA introduces new mechanisms to those defined in RFC 6298. For unreliable communications, CoCoA specifies a transmission rate of $\frac{1}{RTO_{overall}}$ NONs per second. Since NON messages do not require ACKs, but the ACKs are needed to perform RTT measurements and to update the RTO, in CoCoA, 2 out of every 16 NONs are converted to CONs. In order to be able to understand how CoCoA applies congestion control for NONs, it is also necessary to understand the RTO estimation mechanisms used for CONs. The basics of these mechanisms are introduced in the following subsections.

The strong and weak RTO estimators

Like in TCP, CoCoA assumes an initial RTO for a destination endpoint (2 s) and then uses RTT measurements, calculated as the time between the initial transmission attempt for a CON and receiving an ACK, to update the RTO estimation for a destination. We refer to an estimator that is updated after receiving an ACK without any retransmission of the packet as *strong* estimator.

However, communications in networks of constrained devices often suffer from packet losses due to high Bit Error Rates (BER), and also packet drops resulting often from the low memory capacities that limit the routing and data packet queueing capabilities of the devices. The RTO estimator used in TCP, which is not designed for constrained networks, does not perform any updates of the RTO when running into retransmissions, since it is not capable of correlating an ACK to a specific (re)transmission of the corresponding packet. Given the idiosyncrasy of high packet loss rates in constrained networks, the use of solely a strong RTO estimator as it is done in TCP is not recommendable, since the frequency with which RTT measurements are obtained would be low.

Therefore, CoCoA introduces a so called *weak* RTO estimator (RTO_{weak}), which is maintained alongside with a strong RTO estimator (RTO_{strong}) for each endpoint. The weak RTO estimator uses RTT information obtained from up to the second retransmission of a CON, where the RTT is the time between the initial transmission and the reception of any of the ACKs. This behavior allows CoCoA to obtain RTT measurements in spite of packet losses, even if it faces the ambiguity about the correlation between the transmission intent and the reception of an ACK.

CoCoA maintains averages of strong and weak RTT measurements, RTT_{strong} and RTT_{weak}, respectively, that are calculated as follows, when a new RTT measurement RTT_{X_new} is performed:

$$\mathrm{RTTVAR}_X = (1-\beta) \times \mathrm{RTTVAR}_X + \beta \times |RTT_X - RTT_{X\_new}| \tag{1}$$

$$RTT_X = (1-\alpha) \times RTT_X + \alpha \times RTT_{X\_new}, \tag{2}$$

where X stands for *strong* or *weak* accordingly and using $\alpha = \frac{1}{4}$ and $\beta = \frac{1}{8}$.

Subsequently, each update of a RTT_X leads to an update of RTO_X as

$$\mathrm{RTO}_X = RTT_X + K_X \times \mathrm{RTTVAR}_X, \tag{3}$$

where $K_{strong} = 4$ and $K_{weak} = 1$.

The update of either the weak or strong RTO estimators in both cases leads to an update of $RTO_{overall}$ which is the overall RTO value maintained for a destination.

$$\mathrm{RTO}_{overall} = \lambda \times \mathrm{RTO}_X + (1-\lambda) \times \mathrm{RTO}_{overall}, \tag{4}$$

where λ is 0.5 for a strong estimator update and 0.25 for a weak estimator update.

$RTO_{overall}$ is then used to determine the initial RTO (RTO_{init}) of a CON transmission and to set the outgoing data rate of NONs to $\frac{1}{RTO_{Overall}}$ notifications per second, choosing randomly from the interval [$\mathrm{RTO}_{overall}$, $\mathrm{RTO}_{overall} \times$

1.5] to avoid synchronization effects among a set of transmitters. With the combination of *strong* and *weak* estimators, CoCoA is expected to dynamically adapt the RTO for different endpoints in IoT networks, being capable of adjusting to large or small RTTs, but also being capable of reacting to sudden changes of network conditions.

Variable Backoff Factor (VBF)

Contrarily to the BEB used in TCP, which doubles the RTO for each retransmission, CoCoA uses a so called VBF that adjusts the backoff value depending on the initial RTO of a transmission, RTO_{init}. When RTO_{init} is small (<1 s), the backoff is set to 3. This is done to avoid quick successive retransmissions of CoAP messages within a short interval of time. The main goal of this modification is to reduce the chance for spurious retransmissions that may occur when there is network congestion. On the other hand, the VBF applies a backoff factor of only 1.5 to CONs that are initiated with a RTO_{init} of more than 3 s, since applying the BEB to a transmission that starts with a very large RTO_{init} can lead to long idle times. Contrarily to the use of a BEB, as a result of applying larger or smaller backoffs determined by the VBF, CoCoA is expected to achieve better performance with low or high per destination RTOs, respectively.

RTO aging

In CoCoA, the RTO estimator may adopt values that are below or above the default RTO value, depending on whether small or large RTTs are measured, respectively. Since the network conditions and thus the RTT change over time, an RTO estimation may become obsolete after some time without receiving any updates. To avoid the use of outdated $RTO_{overall}$ values that may have become bogus over time, CoCoA applies an aging mechanism to the RTO estimation of endpoints. This is an important aspect in IoT networks, where CoAP messages may be transmitted sporadically or the dynamicity of channel conditions may cause RTO information to become obsolete quickly.

If $RTO_{overall}$ is very small (less than 1 s) and during more than 16 times its value no new RTO update is performed, $RTO_{overall}$ is doubled. On the other hand, if the RTO is above the default initial value of 2 s and it is not updated during 4 times its current value, the RTO is shifted towards the default initial value of 2 s using

$$\mathrm{RTO}_{overall} = (2 + \mathrm{RTO}_{overall})/2\, s. \tag{5}$$

3. Experimental Setup and Test Scenarios

The investigations in this paper analyze how different approaches to congestion control for NONs affect the overall network performance. The first set of experiments focuses on communications over a single, emulated GPRS/UMTS link between a CoAP server and an Internet cloud service. This scenario highlights how congestion can deteriorate the end-to-end performance in a typical M2M use case, and how the congestion control mechanisms proposed in Observe

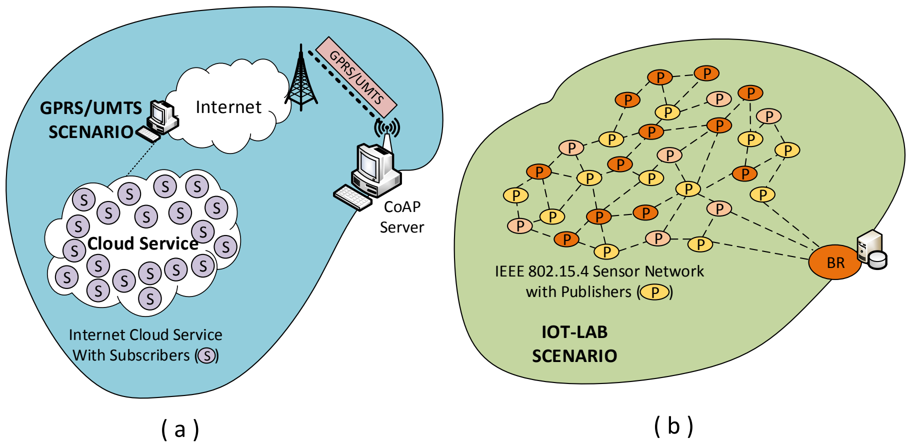
GPRS/UMTS SCENARIO
Internet
GPRS/UMTS
CoAP Server
Cloud Service
Internet Cloud Service With Subscribers (S)
(a)
IEEE 802.15.4 Sensor Network with Publishers (P)
BR
IOT-LAB SCENARIO
(b)

Figure 1: The two scenarios evaluated in this paper: a) the GPRS/UMTS scenario evaluates how the information stored by a CoAP server is transmitted via NONs to an Internet Cloud service over GPRS/UMTS links, b) the IoT-Lab scenario evaluates the CoAP communications without end-to-end reliability between an IEEE 802.15.4 sensor network that generates resource state information and a CoAP border router (BR).

and CoCoA act in order to improve the performance. The evaluations then are extended to IEEE 802.15.4-based, multihop networks of constrained devices. A plentitude of constrained devices build a mesh network, each of the nodes of the network acting as a publisher of a resource state information for a central CoAP server. In this scenario, the congestion control mechanisms have to face heterogenous links with different link qualities, varying number of hops between the source and destination of a packet, and the limited hardware capacities of the constrained devices.

In this section we introduce the two network setups used for the evaluations of the different congestion control mechanisms for CoAP communications without end-to-end reliability.

3.1. GPRS/UMTS Scenario

In the first network setup, a multitude of clients running in an Internet cloud service subscribe to status updates for a set of resources maintained by a CoAP server that is connected to the Internet via GPRS/UMTS (Fig. 1 (a)). This scenario represents a typical use case, where plenty of clients are interested in the information gathered by a network of sensor nodes. The sensors send the information they measure towards a border router (BR) or proxy server, where the data is processed further. In our setup, the BR/proxy server corresponds to the CoAP server, which is responsible for providing interested clients with the information gathered from the sensor network. In the evaluations of this use case, the CoAP server maintains several virtual resources, to which the clients located in the Internet cloud service can subscribe to. Each of the subscribers is then kept up to date about the resource state via notification messages.

The connection between the two physical machines that run the cloud service and the CoAP server, respectively, is established over the Internet. While the machine running the cloud service is connected to Internet over a broadband connection, the CoAP server connects over a GPRS/UMTS link that is emulated on top of an Ethernet connection via the Network Conditioner Software [13][2]. The software emulates several link parameters: the data rate, packet loss rate, and the delay. The GPRS link has an Uplink/Downlink datarate of 20/80 kbps, respectively, a loss ratio of 0.1% and a delay of 605 ms. On the other hand, the UMTS link has an Uplink/Downlink datarate of 128/348 kbps, respectively, a loss ratio of 0% and a delay of 71 ms.

In the analyzed scenario three different types of subscription models are offered to the clients of the Internet cloud service. Each of the subscription models determines with which frequency the CoAP server sends notification messages. Two so called *delay-tolerant* types of subscription are analyzed, where the CoAP server publishes notifications in intervals of 60 s or 30 s. In the so called *real-time* subscription, clients are notified every second about the state of the resources located on the server. Apart from choosing different notification intervals, the clients of the Internet cloud service also have the choice of observing either 1, 3, or 5 of the resources located on the CoAP server, denoted as *1n*, *3n*, *5n*, respectively. Further, for the evaluations of this use case, we vary the numbers of subscribers to be 25, 50, or 100, denoted as *25S*, *50S*, and *100S*, respectively.

Considering all available options, the number of messages sent during each notification interval depends on the number of subscribers (CoAP clients) and the number of resources that are observed by each client. The combination of the options results in a total of 27 different configurations that are evaluated for each considered congestion control mechanism.

To run the CoAP clients and the server on the physical machines[3] we use the Java-based Californium (Cf) implementation [14]. The core version of Cf implements default CoAP. CoCoA is implemented as an additional module as part of an optional congestion control layer.

3.2. IoT-Lab Scenario

The first scenario evaluated in this paper considers the communications between a CoAP server acting as BR of a wireless sensor network and a multitude of clients of an Internet cloud service. The second network setup (Fig. 1 (b)) complements this setup by considering the communication taking place between the nodes of a wireless sensor network and the CoAP BR. For the second set of evaluations carried out in this paper, we use one of the IoT-Lab [11] testbeds, that allows to experiment with constrained devices equipped with IEEE 802.15.4-based radio transmitters. We pick a subset of 60 *M3* nodes [15] distributed across the Grenoble test site to perform the experiments.

[2]The two machines running the cloud Service and the CoAP server are separated geographically from each other in Europe (they are located in Spain and Greece, respectively)

[3]CoAP Server: Runs on a MacBook Pro with 2.4 GHz Intel Core i5 CPU and 8GB of RAM; Clients: Run on a PC with a 3.4 GHz Intel Core i7 CPU and 8GB of RAM

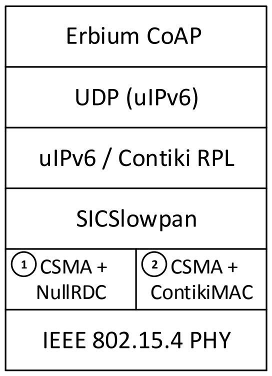

Figure 2: Overview of the different layers of the Contiki communication protocol stack.

For the evaluations of congestion control mechanisms, we program the network nodes with Contiki [16] to form a large mesh network of routing-capable devices. Contiki is an operating system designed for devices with limited hardware capacities. It implements a fullly IPv6-capable communication protocol stack, including an implementation of CoAP called Erbium [17]. In the following we explain the different layers of the Contiki stack, as well as the settings used for our experiments.

3.2.1. The Contiki Stack

Figure 2 depicts the Contiki protocol stack as used in our evaluations. The physical layer implements the IEEE 802.15.4 specification [18], which defines a frame size of up to 127 bytes. The radio operates in the unlicensed 2.4 GHz spectrum and offers data rates of up to 250 kbit/s. By default, Contiki is set to use the highest of the available channels (channel 26). Since we observe interference by other testbed users in this channel, we configure Contiki to use channel 18, in order to reduce the chances of suffering from such interference.

On top of the IEEE 802.15.4 physical layer, Contiki implements a MAC layer that can be divided into 2 components: the carrier sense multiple access (CSMA) layer and the radio duty cycling (RDC) layer. CSMA is the default mechanism used in Contiki to access the radio medium. It requires a node to detect an idle radio channel before proceeding to a packet transmission. This is done to avoid possible packet collisions with already ongoing transmissions or other sources of interference. Further, to reduce the number of packet losses caused by collisions (or interference), the CSMA layer also implements MAC layer reliability. It requires unicast transmissions from one device to another to be acknowledged with a MAC layer ACK. If a node does not receive a MAC layer ACK after the transmission of a packet, the message is retransmitted up to 3 times. After that, the transmission of a packet is considered to have failed and the packet is dropped.

Further, Contiki allows to choose from a set of RDC mechanisms that feature different duty cycling methods. The two mechanisms that are commonly used are either *ContikiMAC* or *NullRDC*. ContikiMAC introduces duty cycling by basically turning the radio on and off periodically to reduce energy consumption. This is important for battery driven devices, since the energy consumption can be reduced considerably by turning the radio off. When transmitting a packet, ContikiMAC strobes the data frame many times[4] to ensure that a destination node also using ContikiMAC receives one of the packets during its periodic duty cycle.

NullRDC, however, can be used whenever no duty cycling is necessary: the radio is maintained permanently in reception mode while no packets are transmitted. Further, using NullRDC, no strobing is required, only one packet is sent per transmission intent. When sending a packet, it is handed over to the CSMA mechanism responsible for accessing the radio medium and for performing packet retransmissions in case that no MAC layer ACK is received. Given a problematic behavior we observe with ContikiMAC's strobing mechanisms continuously occupying the radio channel when used by large numbers of nodes, causing network malfunctioning, we decide to use the NullRDC mechanism in combination with CSMA for the experimental evaluations.

Since the IPv6 Maximum Transmission Unit (MTU) exceeds the maximum capacity of an IEEE 802.15.4 frame, Contiki implements SICSlowpan, an IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) adaptation layer. The functionality of this layer includes compression of IPv6 headers and IPv6 packet fragmentation.

Contiki implements the *uIPv6* stack, which includes the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), responsible for the routing of packets between IP-based end devices. It is designed to meet the requirements of IoT networks. RPL maintains one or several Destination Oriented Directed Acyclic Graphs (DODAGs) in order to determine how packets in the network are routed. Upon network initialization, RPL starts building a routing tree-like structure, with the RPL-root at the top, which in our experiments is always the border router (BR). Each node maintains a set of possible parent nodes, from which one is the preferred parent. When sending data to the BR (being the RPL root), packets are forwarded via each node's preferred parent until reaching the root.

To control the dissemination of RPL control messages, RPL uses the Trickle algorithm [19]. Trickle runs in each node and determines when a control message is sent. It sets a timer within an interval of time that is adjusted by the algorithm. After the expiration of the timer, a control message is sent if not already K other control messages were overheard since the timer was set. The initial interval in Contiki lies between 0 and 4 s and it is doubled after each interval, up to a certain limit. However, every time a node changes its preferred parent, Trickle resets the interval size to the (short) initial one, in order to distribute routing information in a shorter time.

CoAP is located at the top of the stack. Contiki comes with the Erbium

[4]At least for the duration of the duty cycle.

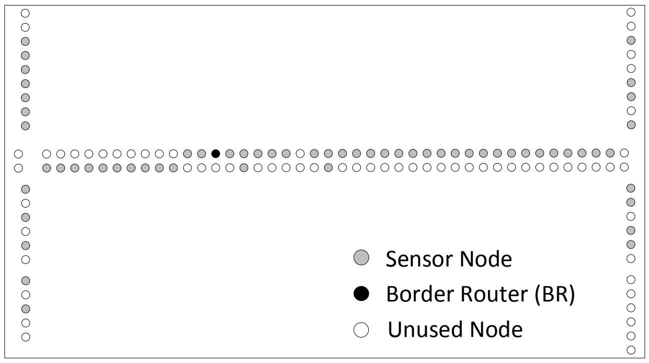

Figure 3: Detail of the testbed used for the experimental evaluations.

implementation of CoAP [17], which we extended in previous work to include CoCoA for the exchange of messages with end-to-end reliability [5]. For the evaluations performed in this paper, we extend CoCoA to also implement the rate control for NON messages as proposed in the CoCoA Internet Draft. The draft states that 2 out of every 16 NONs need to be converted to CONs in order to obtain RTT measurements. In our implementation, we convert every 8th NON into a CON. Finally, we also implement the static rate control defined in Observe. The configuration presented in [20] has been used to set crucial parameters of the Contiki stack, like queue sizes or RPL parameters.

3.2.2. IoT-Lab Network Setup and Traffic Scenarios

In this subsection we define the basic network setup upon which the experimental evaluations are carried out. The topology used for the experiments is shown in Fig. 3.

We configure node 231 (as per the identity system used in the Grenoble IoT-Lab testbed) to be the RPL root and also the sink node of the network. The rest of the nodes in the network are CoAP servers (from here on referred to as sensor nodes) that maintain state information about 5 virtual resources that represent measurements, such as temperature or humidity. As defined in Observe, the sensor nodes periodically publish the state of their resources to the sink node that is responsible for gathering and storing the state information, to be processed further, e.g., to be published to an Internet cloud service.

To adjust the traffic generation (and with it the degree of congestion), we define experiments where the sensor nodes in the network publish information for different numbers of resources. Also, we vary the periodicity of the notification timers in the sensor nodes that determine when the sink node is notified about the state of each of the resources. In total, we define a set of 9 experiments, where each sensor node reports the status of 1, 3, or 5 resources (*1n*, *3n*, *5n*), with notification intervals of *5 s*, *10 s*, and *20 s*, respectively. Further, we evaluate three different approaches to congestion control, namely the ones followed by default CoAP, Observe and CoCoA. The combinations of the analyzed mechanisms, number of observed resources, and notification interval periodicity result in a total of 27 experiments.

In each experiment, after an initial setup phase that is slightly different for each node and during which the RPL DODAG is built, the nodes begin with the generation of traffic. The duration of each experiment is 10 minutes and each experiment is repeated 6 times, leading to a total experimental duration of 1 hour for each configuration.

3.3. Performance Metrics

To measure the performance of the different congestion control approaches we use a set of performance metrics: The overall packet delivery ratio (PDR), the end-to-end delay, and the number of packet drops at the medium access (MAC) layer. The PDR is measured in both network scenarios, whereas the end-to-end delay and the MAC layer drops are only used to elaborate in depth evaluations of the IoT-Lab scenario. In the following subsections we give a short definition of these metrics and detail their relevance for the analyzed use cases.

Overall PDR

The overall PDR is an indicator of the reliability of the network and therefore is an important Quality of Service (QoS) metric used in IoT systems. The PDR is calculated as the ratio of total number of CoAP notification messages received by subscribers over the total number of notifications generated during an experiment. The use of NON messages (i.e. without reliability) does not imply that reliability of the network is irrelevant for the operation of the network. In fact, a low PDR can render networks useless, especially if the main purpose of such networks is to gather information from the network devices, such as environmental sensor readings. Further, notification messages may be of critical nature, like for example alarm notifications sent when sensor thresholds are exceeded. The loss of such messages may cause a grave deterioration or may lead to malfunctioning of the services offered by the network. Therefore, in main terms, a high PDR is always desirable and an efficient congestion control mechanism should be able to maintain the PDR of a network as high as possible.

End-to-end Delay

We define the end-to-end delay of a notification message to be the time between the generation of a notification message and its reception at the subscriber. The end-to-end delay is relevant for use cases where the ‘freshness’ of the information carried in notification messages is important. In the evaluations performed in this paper, we consider the PDR to be more relevant for the correct operation of the network and usually, there is a trade-off between end-to-end delay and PDR. Yet, the end-to-end delay observed during the experiments is relevant to understand the behavior of the different congestion control mechanisms. End-to-end delay is another common QoS metric considered in IoT systems. While not being a critical parameter for unreliable CoAP communications, low delays are indicators of a better system responsiveness.

MAC Layer Packet Drops

When a high degree of traffic drives a network into saturation or congestion, the chance for packet collisions increases since nodes are competing for access to the radio channel. The CSMA applied by the network devices reduces the number of collisions but it does not completely eliminate them. Also, even when applying MAC layer reliability, the transmission of a packet may fail after using all retransmissions. However, another main reason of packet losses at the MAC layer needs to be considered: if buffers are full and no newly generated or received packets can be stored in the node for further processing, packets are dropped. Buffers may quickly fill if large numbers of packets are generated in a short amount of time or when nodes act as relays for other messages.

MAC layer packet losses due to network congestion can decrease the network performance considerably, not only affecting the data plane (e.g. CoAP), but also the control plane (i.e., the routing protocol). Thus, an efficient congestion control mechanism should be able to prevent network congestion, which could lead to packet losses at the MAC layer. We analyze the MAC layer packet drops for the IoT-Lab network scenario.

4. Evaluation Results

In this section we present the results of the experimental evaluations carried out in the GPRS/UMTS and IoT-Lab testbed scenarios. First, we show the results for the GPRS/UMTS scenario and provide a comparison of the performance of default CoAP, Observe, and CoCoA. We then extend the evaluations to the 60-node IEEE 802.15.4 IoT-Lab scenario.

4.1. Results for the GPRS/UMTS Scenario

In the GPRS/UMTS scenario, we focus on the evaluation of the PDR results achieved by the different congestion control mechanisms with the delay-tolerant and real-time subscription models, over the emulated GPRS and UMTS links, respectively..

4.1.1. Delay-tolerant Subscriptions (60s/30s) over GPRS

Figure 4 shows the improvement of the PDR obtained during the experiments with different numbers of subscribers (25/50/100) and observed resources (1/3/5) with 60 s notification intervals, taking default CoAP as reference. If the generated traffic is small (*25S1n*, *50S1n*, *100S1n*), the performance of all three congestion control mechanisms is very similar. However, when the number of observed resources and therefore the number of notifications sent with every notification interval increases, Observe and CoCoA clearly perform better than default CoAP.

Default CoAP tries to transmit the notification messages back-to-back during every notification round. This creates a peak of traffic that exceeds the limitations of the GPRS link and results in packet losses. On the other hand, the rate control implemented by Observe and CoCoA dilutes the transmission

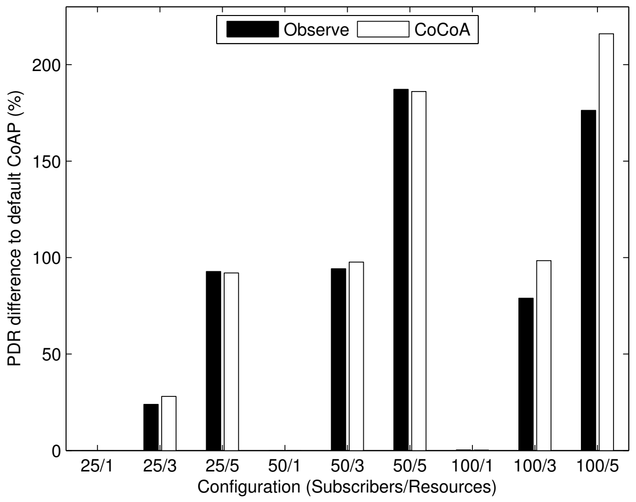

Figure 4: Relative improvement of the PDR of Observe and CoCoA when compared to default CoAP in the 60s delay-tolerant GPRS scenario.

of the bursts of 3 or 5 notifications over time. While the transmission rate of Observe is fixed (1 message every 3 s), CoCoA uses the dynamic RTO, which adapts depending on the measured RTTs. The difference between the static and dynamic RTO affects the performance in the 100-node case with 3 and 5 observed resources, where Observe does not yield the same improvement as CoCoA. The fact that the fixed rate control used by Observe dilutes the transmissions over time does not prevent a certain degree of synchronicity among the large number of subscribers, since the transmission intervals (every 3 s) are the same for each subscriber. The sum of notification messages sent to every subscriber exceeds the GPRS link capacity. CoCoA on the other hand maintains a different RTO estimation for each subscriber that is used to calculate the allowed notification rate. This reduces the chances for synchronicity, avoiding peaks of traffic and leading to an improvement of the PDR.

Similar results are obtained for the delay-tolerant scenario with *30 s* notification intervals, as can be seen in Fig. 5. Again, default CoAP's lack of congestion control mechanisms leads to heavy congestion when 3 or 5 resources are observed by the subscribers in the Internet cloud service. The approach followed by Observe again delivers improvements in this scenario that, however, are surpassed by CoCoA, which delivers the best performance in the majority of scenarios, most remarkably in the most message intensive ones (*100S3n*, *100S5n*). Note that the slightly smaller PDR observed for CoCoA (in comparison with Observe) in the *50S5n* case can be explained by the fact that CoCoA needs some time to adjust the RTO timers in the experiments, before it adapts them to the observed network conditions. During the adjustment period, the rate control applied by CoCoA may not yield an optimal performance, which

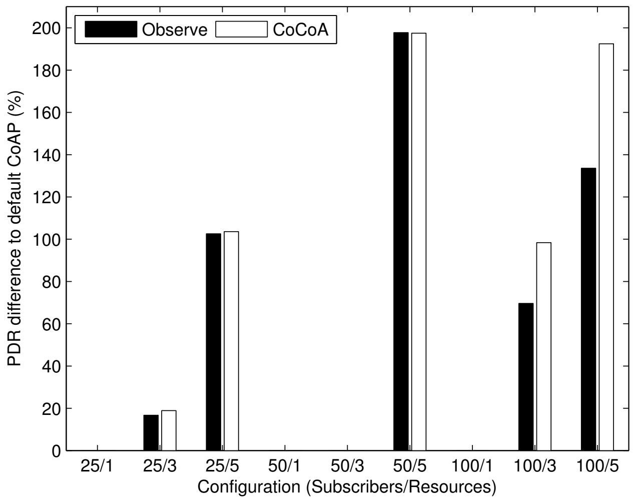

Figure 5: Relative improvement of the PDR of Observe and CoCoA when compared to default CoAP in the 30s delay-tolerant GPRS scenario.

reflects in a slightly lower PDR. Further discussion will be given about this phenomenon in section 4.1.2..

4.1.2. Real-time Subscriptions (1s) over UMTS

For the real-time subscriptions, a link with a higher supported data rate and a lower latency is required, therefore, the GPRS link from the previous evaluations is replaced with a UMTS link that yields a higher performance. In the real-time scenario, the CoAP server continuously sends notification messages towards the Internet cloud service. The experiments reveal the PDR results shown in Fig. 6. Observe yields the worst performance in every scenario, whereas default CoAP and CoCoA show a very similar performance.

The fix rate limitation of Observe does not allow all the generated packets to be transmitted over the UMTS link, since the notification generation rate is higher than the outgoing message rate. Observe starts losing packets as soon as the internal buffers of the CoAP server are full and no newly generated packets can be buffered.

On the other hand, there is a small difference in the performance between default CoAP and CoCoA, which favours default CoAP. This difference is caused by the RTO adaptation phase, during which CoCoA, starting with a default RTO of 2 s, slowly adapts to the small RTT of the UMTS link. Since only 2 out of 16 NON messages are used to update the RTO information (see Fig. 7), the adjustment takes CoCoA several notification rounds, during which the internal buffers of the CoAP server reach their limit and packets are dropped (the same behavior was detected in Observe). Once the RTO estimation reaches a steady state, CoCoA optimizes the rate of outgoing NONs and achieves a good

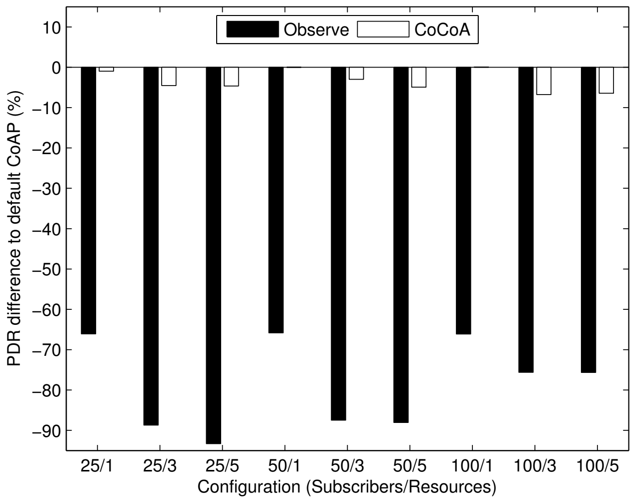

Figure 6: Relative deterioration of the PDR of Observe and CoCoA when compared to default CoAP in the real-time UMTS scenario.

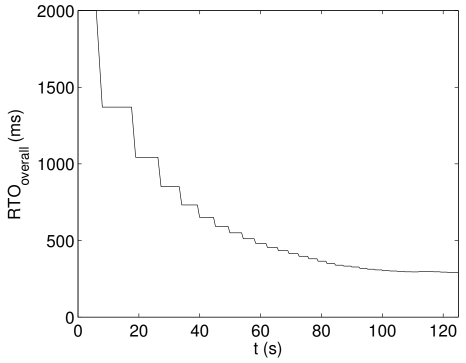

Figure 7: RTO values used by CoCoA to control the rate of outgoing notification towards an endpoint during the first 120 s of a real-time subscription experiment (*25S3n*).

and steady performance. Note that the relatively short duration of each single experiment of 10 minutes penalizes CoCoA. If a use case involves a greater duration, CoCoA performance will increase.

Overall, CoCoA delivers a very dynamic congestion control mechanism that, in contrast with Observe, provides a performance similar to or better than that of default CoAP A minimal deterioration is measured in the real-time scenario, where CoCoA suffers from a minor amount of packet losses until the RTO estimator is adjusted. In the next subsection, we perform in-depth evaluations of CoAP end-to-end communications without reliability in an IEEE 802.15.4-based multihop network of constrained devices.

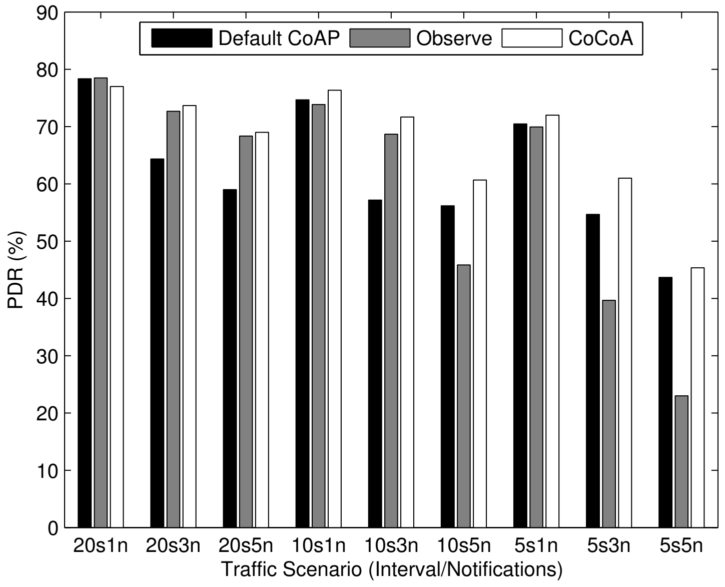

Figure 8: PDR measured for different notification intervals and notification message numbers in the 60-node topology.

4.2. Results for the 60-Node IoT-Lab Scenario

Contrarily to the GPRS/UMTS scenario, where two devices are communicating over a single link, the IoT-Lab scenario evaluates CoAP communications in a setup with 60 constrained devices. These form a mesh network, where all nodes generate traffic towards the sink node, which in this case is the BR.

4.2.1. PDR results

Figure 8 shows the overall PDR achieved with the different congestion control mechanisms (*Default, Observe, CoCoA*) in experiments with varying periodic intervals (*5s, 10s, 20s*) and number of notifications sent per interval (*1n, 3n, 5n*). In the scenarios with the lowest amount of generated traffic (*20s1n*), the PDR of the three mechanisms is similar, reaching overall PDRs of around 80 %. Since no end-to-end reliability is used, the rest of messages are lost due to packet drops caused due to lossy links, packet collisions, or buffer overflows by the aggregate traffic in the network (RPL control traffic + CoAP traffic). As soon as the number of notifications per interval is increased (*20s3n, 20s5n*), we start seeing clearer differences in the performance of congestion control mechanisms. Since in the scenarios with multiple notifications per interval several packets are sent in a burst, the chances for the network to suffer from congestion increase. Observe and CoCoA both apply their rate control mechanisms, sending notifications only every 3 s (Observe) or $\frac{1}{RTO}$ seconds (CoCoA), respectively. Default CoAP, on the other hand, sends the notifications back to back, as soon as they are generated every 20 s. These small bursts of notifications lead to packet losses, mainly caused by a higher amount of MAC layer packet drops.

In the *10s* scenarios, we observe slightly different results as the traffic is doubled, compared to the *20s* scenarios. First of all, the overall PDR decreases in general when compared to the *20s* notification interval results, which is a result of the higher degree of traffic increasing the chances for congestion, which in return can lead to packet losses. Second, while the difference in the performance between the different congestion control mechanisms for 1 and 3 notifications is similar to the one observed for the 20 s interval scenario, the 5 notification scenario yields different results. Here Observe clearly delivers a lower PDR, only allowing one NON to be transmitted every 3 s. Thus, with an interval of *10s* and 5 notification messages generated per interval, more notifications are generated than can actually be transmitted. In this situation, notification messages are buffered over longer periods of time. However, each node has a limited amount of buffer space dedicated to CoAP messages, meaning that at some point the buffers fill and some of the messages need to be dropped.

In the scenario with the highest degree of traffic (5 s notification interval), in general, the overall PDR drops even further, since the network suffers from even a higher degree of congestion than in the scenarios analyzed so far. While the network seems to be able to cope with the traffic when only transmitting a single notification (*5s1n*), the results indicate that with bursts of 3 and 5 notification messages the network clearly has to deal with a high degree of congestion. CoCoA clearly performs better than default CoAP when the notification bursts consist of 3 messages (*5s3n*), whereas Observe clearly underperforms default CoAP. In the *5s5n* scenario, CoCoA no longer performs clearly better than default CoAP, only yielding a minor improvement. Under these heavy traffic conditions, Observe only delivers a PDR of 23%. This, again, can be attributed to the fact that the notification generation rate is higher than the rate of messages that are actually allowed to be transmitted. Overall, the adaptive rate control applied by CoCoA reduces the risk of packet losses due to network congestion. Some increase of transmission reliability can be achieved by Observe as well, yet there are traffic scenarios where Observe's static rate control decreases performance noticeably.

4.2.2. Delay results

The end-to-end delays in the scenarios with 1 notification are nearly negligible for all congestion control mechanisms, as can be seen in Fig. 9. Assuming that the radio channel is almost idle when packets are transmitted, the notifications generated by the sensor nodes only need tens of milliseconds to reach the sink node. Slightly larger delays are measured for CoCoA, since some of the messages are CONs that may require a retransmission after a packet loss, contributing to a higher average delay.

Very small delays are also observed for default CoAP in the 3 and 5 notification scenarios, where notifications are sent back to back and either the packets arrive at the BR shortly after their transmission or they are lost on the route due to packet drops. Compared to the *1n* scenarios, the delay may double due

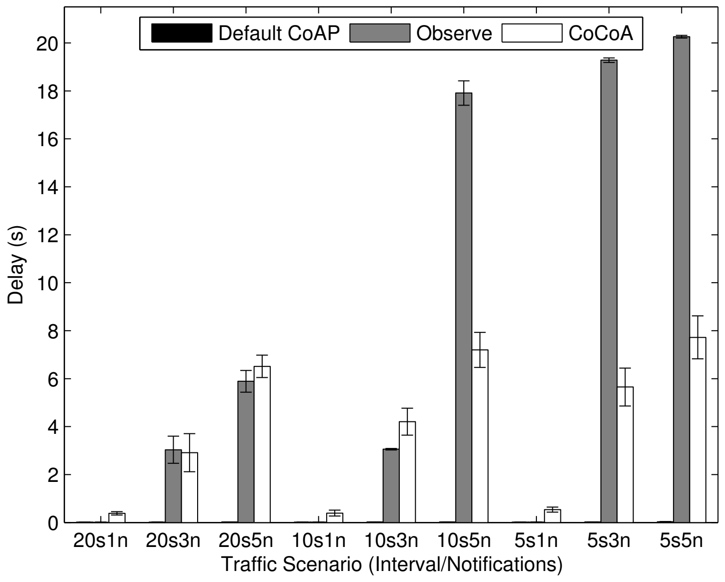

Figure 9: End-to-end delay with 95% confidence intervals between sensor nodes and sink node for different notification intervals and notification message numbers in the 60-node topology.

to packets being buffered at relay nodes, but the values remain low (i.e., less than 50 ms). Observe and CoCoA on the other hand show clearly different results, which is a consequence of introducing rate control, consisting mainly in holding back packets after they have been generated. With the alternative congestion control mechanisms, only the initial message of a burst of notification messages is sent immediately. The second and subsequent messages are each retained until the rate control mechanism allows their transmission. In Observe, each retained message after the initial one is delayed by additional 3 s. In the *20s3n* scenario this leads to an average end-to-end delay of 3 s and in the *20s5n* scenario to an average of 6 s. In the *10s3n* scenario Observe again yields an average delay of 3 s, however, in the *10s5n* scenario, the delay does not increase linearly to 6 s like in the *20s5n* scenario. It grows up to nearly 18 s, since the allowed NON transmission rate is now lower than the packet generation rate. We observe the same effect in the *5s3n* and *5s5n* scenarios, where the delay grows to values between 19 s and 21 s respectively.

On the other hand, CoCoA adapts the rate control based on the RTO estimator that is updated by RTT measurements thanks to the use of CONs for a subset of messages, as described in subsection 2.3. The measured RTT varies depending on the network state in which the measurement is performed and it also depends on where the node is located in the network topology, directly impacting the RTO. In spite of varying packet retention durations as a result of varying RTO estimations, a relatively small variation of end-to-end delays is observed in CoCoA, as indicated by the 95% confidence intervals (Figure 9).

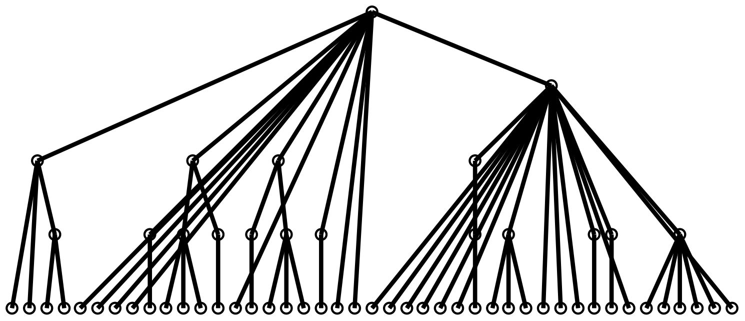

Figure 10: Snapshot of the routing tree of the 60 nodes, taken during an experiment. The circles represent the nodes and their correspondent preferred parent, connected via a line. The BR is located at the root.

Figure 10 depicts a snapshot captured during one of the experiments of the network tree showing the relationship between preferred parents and their children nodes. Each hop between the BR and a sensor node increases the chance for packet losses, since for each hop a transmission over a potentially lossy link is required. Moreover, each relay node may drop a packet if its message queues are already full, thus not being able to buffer incoming packets. Therefore, the chances of congestion are higher for nodes that are closer to the BR.

Figure 11 shows the average RTO maintained by the nodes when using CoCoA in all analyzed scenarios, laying between 1.4 s and 2.2 s. While a majority of the sensor nodes maintain rather small RTO values, several nodes have large RTO values, which is a consequence of obtaining weak RTO estimates due to packet losses. The average RTO increases slightly as the notification intervals get smaller (Fig. 11). Since packet losses are more frequent during the exchange of a CON and the correspondent ACK with increasing traffic, it becomes more likely for CoCoA to measure weak RTTs. Weak RTTs can lead to an increase of the RTO for an endpoint, which overall reflects in a larger average of the RTO values across the network. This has a controversial effect on the performance of the nodes: Nodes that have large RTOs introduce less traffic into the network, which reduces congestion. On the other hand they are likely to suffer from buffer overflows losses, since the notification generation rate may exceed the outgoing NON rate, forcing them to queue NONs until the maximum buffer limit is reached.

The impact on the end-to-end delay of the variable NON transmission rate in CoCoA is therefore different for every node. The average delays measured for CoCoA in the *20s3n*, *20s5n*, *10s3n* scenarios resemble the values measured for Observe. However, in the rest of scenarios with higher network loads, the rate control applied by CoCoA clearly leads to lower delays. There is a clear trade-off in between the delay introduced by CoCoA's rate control to prevent CoCoA from sending packets immediately after their generation like default CoAP, and risking network congestion.

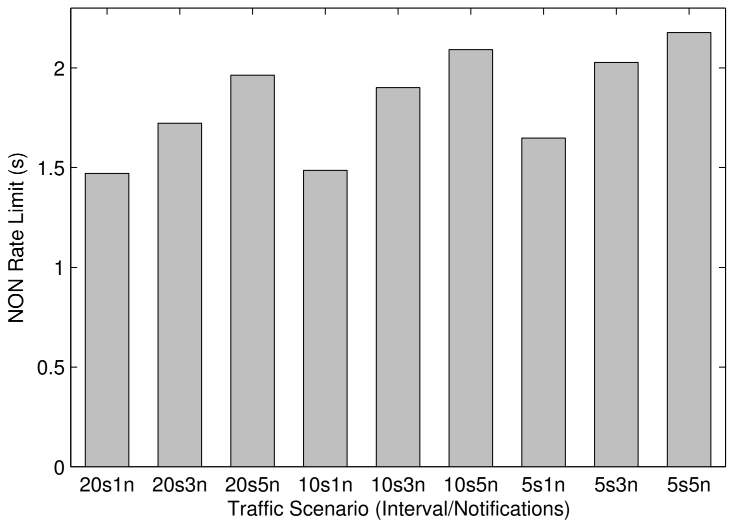

Figure 11: Average RTO values applied for CoCoA rate control by the nodes in the 60-node topology.

4.2.3. MAC Layer Drops

The effects of the congestion control applied by Observe and/or CoCoA also reflect in the MAC layer drops, which are considerably less than the ones measured if no congestion control is applied as in default CoAP (see Fig. 12). Diluting the transmissions of notifications over time decreases the chance for packet collisions in the radio and also reduces the traffic load the relay nodes of the network have to handle. The reduction of MAC layer packet drops achieved by Observe and CoCoA applies to all the evaluated traffic scenarios. In the *20s* interval scenarios, CoCoA can clearly reduce the number of MAC layer drops applying its dynamic rate control. While in general still performing better than default CoAP in the *10s* and *5s* scenarios, the drops measured with CoCoA no longer are lower than the ones measured with Observe. This can be explained by the fact that Observe behaves much more conservatively, which makes it cause less congestion. However, the main drawback of this overly conservative behavior are the losses introduced at the CoAP layer due to overflowing message buffers which cannot be compensated by the (slightly) lower number of MAC layer drops.

5. Conclusions and Future Work

This paper has evaluated the performance of default CoAP, Observe and CoCoA as the main approaches to congestion control for unreliable CoAP communications, for various load caracteristics and in two setups: a single-hop, GPRS/UMTS emulated link, and a real 60-node IEEE 802.15.4 multihop testbed.

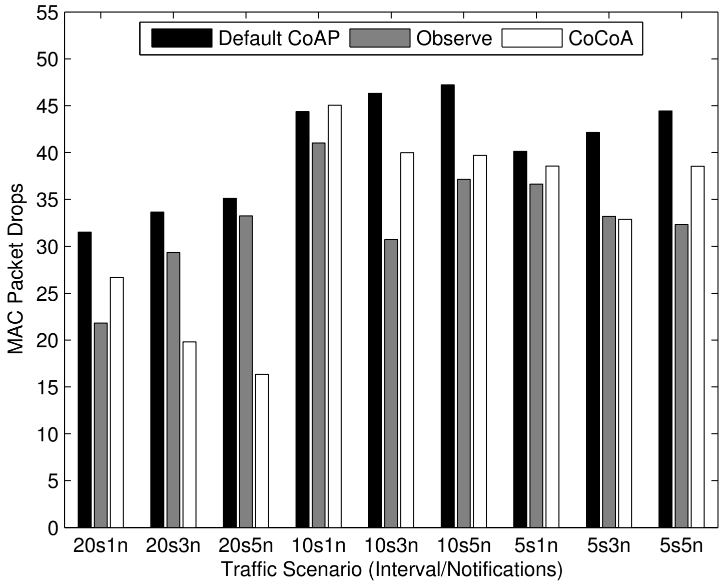

Figure 12: Average number of MAC layer packet drops in the 60-node topology.

The risk of suffering network performance degradation when not using any congestion control mechanisms is high. Default CoAP faces serious issues when dealing with bursts of packets that cause peaks of congestion which result in packet losses independent from the communication technologies used. Using a fixed rate control as it is done in Observe is a two-edged sword: in scenarios with moderate traffic or seldom bursts of packets it can increase the performance of a network by diluting notification transmissions over time. Yet, it also can potentially lead to issues when the notification generation rate exceeds the permitted transmission rate, and the rate limitations cause a bottleneck effect, leading to a degradation of the performance when compared to default CoAP. Performance losses are also observed when there is heavy congestion and the constant outgoing message rate is too high to effectively decrease congestion. On the other hand, due to its adaptive nature, CoCoA maintains relatively high QoS or shows improvements over default CoAP and Observe in nearly all traffic scenarios. The results show that a dynamic rate control as applied by CoCoA adapts much better than a static rate control (Observe) or no rate control at all (default CoAP) in all analyzed scenarios.

In order to improve the performance of CoCoA further, we also propose a future line of work: If a multitude of destinations are notified with NON messages, even when applying rate control on a per destination basis, the sum of traffic sent to the entirety of destination endpoints may cause congestion. In such cases, congestion control mechanisms need to go beyond a per-destination basis, applying an aggregate congestion control is required to solve this issue.

6. Acknowledgments

This work was supported in part by the Spanish Government's Ministerio de Economía y Competitividad under grant number RYC-2013-13029, through project TEC2012-32531, FEDER, the Spanish Government's Ministerio de Industría, Energía y Turismo, under the program AEESD 2013, through the project Smart Gateway Evolution and by the Generalitat de Catalunya, Departament d'Economia i Coneixement through the project of Doctorat Industrial. We would like to thank the operators of the IoT-Lab for granting us access to their testbed facilities and for their technical support.

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

[2] K. Hartke. Observing Resources in the Constrained Application Protocol (CoAP). https://datatracker.ietf.org/doc/rfc7641/, September 2015.

[3] C. Bormann, A. Betzler, C. Gomez, and I. Demirkol. CoAP Simple Congestion Control/Advanced (work in progress), October 2015.

[4] August Betzler, Carles Gomez, Ilker Demirkol, and Matthias Kovatsch. Congestion control for CoAP cloud services. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation, ETFA 2014, Barcelona, Spain, September 16-19, 2014*, pages 1–6, 2014.

[5] August Betzler, Carles Gomez, Ilker Demirkol, and Josep Paradells. CoCoA+: An advanced congestion control mechanism for CoAP . *Ad Hoc Networks*, 33:126 – 139, 2015.

[6] August Betzler, Carles Gomez, Ilker Demirkol, and Josep. CoAP Congestion Control for the Internet of Things (Accepted for publication). *IEEE Communication Magazine*, June 2016.

[7] Rahul Bhalerao, Sridhar Srinivasa Subramanian, and Joseph Pasquale. An Analysis and Improvement of Congestion Control in the CoAP Internet-of-Things Protocol. In *12th IEEE Consumer Communications and Networking Conference (CCNC 2015), Las Vegas, Nevada, USA, January 9-12, 2016.*, 2015.

[8] Ilpo Jarvinen, Laila Daniel, and Markku Kojo. Experimental evaluation of alternative congestion control algorithms for Constrained Application Protocol (CoAP). In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 453–458, Dec 2015.

[9] L. Eggert. Congestion Control for the Constrained Application Protocol, January 2011.

[10] August Betzler, Carles Gomez, and Ilker Demirkol. Evaluation of Advanced Congestion Control Mechanisms for Unreliable CoAP Communications. In

Proceedings of the 12th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, 38; Ubiquitous Networks, PE-WASUN '15, pages 63–70, New York, NY, USA, 2015. ACM.

[11] Eric Fleury, Nathalie Mitton, Thomas Noel, and Cédric Adjih. FIT IoT-LAB: The Largest IoT Open Experimental Testbed. *ERCIM News*, (101):14, April 2015.

[12] C. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's Retransmission Timer (RFC 6298), June 2011.

[13] Apple Inc. Network Link Conditioner, July 2015.

[14] Matthias Kovatsch, Martin Lanter, and Zach Shelby. Californium: Scalable Cloud Services for the Internet of Things with CoAP. In *Proceedings of the 4th International Conference on the Internet of Things (IoT 2014)*, 2014.

[15] Fit/IoT-LAB. M3 Open Node. https://github.com/iot-lab/iot-lab/wiki/Hardware_M3-node, June 2014.

[16] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, Nov 2004.

[17] Matthias Kovatsch, Simon Duquennoy, and Adam Dunkels. A Low-Power CoAP for Contiki. In *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011)*, Valencia, Spain, October 2011.

[18] IEEE. 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), September 2006.

[19] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The Trickle Algorithm (RFC 6206), March 2011.

[20] Javier Isern, August Betzler, Carles Gomez, Ilker Demirkol, and Josep Paradells. Large-Scale Performance Evaluation of the IETF Internet of Things Protocol Suite for Smart City Solutions. In *Proceedings of the 12th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, 38; Ubiquitous Networks*, PE-WASUN '15, pages 77–84, New York, NY, USA, 2015. ACM.