CoAP Congestion Control for the Internet of Things

August Betzler*, Carles Gomez†, Ilker Demirkol†, Josep Paradells†
*i2CAT Foundation, Barcelona, Spain
†Department of Telematics Engineering
Universitat Politecnica de Catalunya, Barcelona, Spain
Email: august.betzler@i2cat.net, {carlesgo, ilker.demirkol, josep.paradells}@entel.upc.edu

***Abstract*—The Constrained Application Protocol (CoAP) is a lightweight RESTful application layer protocol devised for the Internet of Things (IoT). Operating on top of UDP, CoAP must handle congestion control by itself. The core CoAP specification defines a basic congestion control mechanism, which is however not capable of adapting to network conditions. Yet, IoT scenarios exhibit significant resource constraints which pose new challenges on the design of congestion control mechanisms. In this paper we present the CoAP Simple Congestion Control/Advanced (CoCoA), an advanced congestion control mechanism for CoAP being standardized by the Internet Engineering Task Force (IETF) CoRE working group. CoCoA introduces novel Round Trip Time (RTT) estimation techniques, together with a Variable Backoff Factor (VBF) and aging mechanisms in order to provide dynamic and controlled Retransmission Timeout (RTO) adaptation suitable for the peculiarities of IoT communications. We conduct a comparative performance analysis of CoCoA and a variety of alternative algorithms including state-of-the-art mechanisms developed for TCP. The study is based on experiments carried out in real testbeds. Results show that, in contrast with the alternative methods considered, CoCoA consistently outperforms default CoAP congestion control mechanism in all evaluated scenarios.**

I. Introduction

The quantity and diversity of devices that are interconnected in the Internet of Things (IoT) are constantly increasing, leading to a large variety of new and appealing application scenarios. More than 25 billion things are expected to be connected over the Internet by the end of 2020[1]. A significant pillar for this development is the Constrained Application Protocol (CoAP), a lightweight RESTful protocol recently standardized by the Internet Engineering Task Force (IETF). CoAP is designed as the main application-layer protocol to be used by IoT devices for IP-based, HTTP-like interactions [1]. Typically, constrained devices, such as low-power wireless sensor nodes, are used for IoT communications. These devices offer very limited processing and memory capacities. Furthermore, the communication technologies used by these devices exhibit significant limitations such as low data rates and relatively high bit error rate (BER). CoAP is tailored to these extreme resource constraints.

One of the main problems to be handled when designing a new end-to-end communication paradigm is network congestion. This phenomenon occurs when the traffic load offered to a network approaches the network capacity. In many traditional Internet applications, TCP provides end-to-end congestion control. However, CoAP operates over UDP to enable lightweight applications and must handle congestion by itself.

In IoT communications, the traffic patterns are different from the ones in conventional networks. Constrained devices often communicate periodically to notify their sensor measurements. Even when individual devices create small amounts of data, the large number of communicating devices can be a cause of network congestion. Another possible reason for congestion is traffic bursts generated as a reaction to events, for example a large number of notifications sent after a sensor network equipped with accelerometers detects a seismic event. These IoT traffic patterns, together with severe node and link constraints, pose challenges for the design of a congestion control mechanism for CoAP, which should be capable of assuring a safe network operation, while using network resources efficiently.

The core CoAP specification offers a simple congestion control mechanism, based on a Retransmission Timeout (RTO) with Binary Exponential Backoff (BEB), which is however insensitive to network conditions. Therefore, default CoAP congestion control may significantly underperform, often being too conservative or too aggressive, instead of adapting its behavior on the basis of network status information actually available to CoAP.

In this paper we present the CoAP Simple Congestion Control/Advanced (CoCoA) mechanism, defined in a draft specification[2] being standardized by the IETF CoRE working group to improve the CoAP congestion control. CoCoA combines the use of Round-trip Time (RTT) measurements, dynamic RTO backoff calculations, and RTO aging mechanisms to obtain dynamic RTO estimations for the transmission of CoAP messages. CoCoA has been designed to deliver a congestion control that is adaptive to network dynamics and suitable for IoT characteristics. CoCoA has reached maturity

Author's pre-print, accepted for publication in the IEEE Communications Magazine (2016-3-9).

[1]http://www.gartner.com/document/2625419?ref=QuickSearch&sthkw=G00259115

[2]http://tools.ietf.org/id/draft-bormann-core-cocoa-03.txt

in its current form, after initial design and tuning work [2]. We evaluate how default CoAP and CoCoA behave in two realistic IoT setups and traffic scenarios through physical experiments. The first set of experiments is performed on a testbed composed of nodes running the IPv6-based IoT protocol stack produced by the IETF over IEEE 802.15.4. The second set of experiments uses General Packet Radio Service (GPRS), which is a common Machine-to-machine (M2M) solution for IoT. Along with CoCoA, we assess the potential contribution that state-of-the-art algorithms used in TCP can provide over default CoAP congestion control. Results show that, in contrast with the alternative methods considered, CoCoA consistently outperforms default CoAP congestion control in all evaluated scenarios.

II. CoAP Congestion Control

The base CoAP specification provides congestion control by imposing conservative restrictions on the rate of outgoing messages and on the number of allowed parallel message exchanges. CoAP defines four types of messages: confirmable (CON), non-confirmable (NON), reset (RST), and acknowledgement (ACK) messages. The restrictions are applied to CON and NON messages, the first one being the limitation of outstanding interactions per destination to 1. An outstanding interaction can be a CON or a NON request for which no ACK or reply has been received yet, respectively.

In CoAP, a CON message requires an ACK from the receiver and may be retransmitted up to four times, before considering the transmission to have failed. For the first transmission of a message, a RTO value is randomly picked from the interval [2, 3] s. As in TCP, a BEB is applied to the RTO value for a retransmission, i.e., the RTO value is then doubled.

CoAP congestion control is insensitive to network conditions. In fact, it does not adapt the RTO on the basis of RTT information that is actually available to CoAP. Therefore, if the RTO chosen by CoAP congestion control is below the actual RTT, CoAP will incur spurious retransmissions. On the other hand, CoAP is likely to be used in networks with losses due to BER, which can lead to unnecessarily long idle times if the RTO timer overestimates the RTT.

Advanced congestion control mechanisms for CoAP should resolve the aforementioned issues, while assuring a safe behavior in the Internet. The proposal being standardized by the IETF CoRE working group for such advanced congestion control is made in the CoCoA draft specification.

III. CoCoA

CoCoA provides a flexible congestion control solution that relaxes the conservative message rate restrictions of the CoAP base specification, while guaranteeing a safe protocol operation. A fundamental requirement for the design of CoCoA has been to produce a mechanism that offers a performance that is better than, or at least similar to, that of default CoAP. CoCoA comprises three main components: adaptive RTO calculation, Variable Backoff Factor (VBF) and RTO aging.

A. Adaptive RTO calculation

In CoCoA, RTT measurement and adapted RTO calculation follow the principles of RFC 6298 [3]. This RFC constitutes the basis for RTO computation in most TCP implementations, where the RTO is calculated adaptively by applying an exponentially weighted moving average of RTT and RTT-variation estimates. CoCoA adapts this algorithm for IoT communications.

In TCP, a packet loss is assumed to be caused by network congestion. However, in IoT networks, a high packet loss rate is expected due to BER. The RTO estimator detailed in RFC 6298 only uses *strong RTTs*, i.e. RTT measurements from the packets for which an ACK is received before the sender runs into retransmissions. This estimator is referred to as *strong RTO estimator*. In CoCoA, also a *weak RTO estimator* is defined, which uses *weak RTTs*, i.e. RTT measurements taken from packets that have required at most two retransmissions. This increases the chances of obtaining RTT measurements in the presence of packet losses. In CoCoA, when a weak or a strong RTT is measured, the corresponding weak or strong RTO (RTO_X) is updated, respectively, following the same scheme defined in RFC 6298, as

$$\text{RTO}_X = \text{SRTT}_X + K_X \times \text{RTTVAR}_X, \tag{1}$$

where X is either *weak* or *strong*, SRTT and RTTVAR denote the well known smoothed RTT and RTT variation computed as in RFC 6298, K_{strong} is 4 also as in RFC 6298, and K_{weak} is 1. The newly calculated RTO contributes to an overall RTO value with a weighted average:

$$\text{RTO}_{overall} = \alpha \times \text{RTO}_X + (1 - \alpha) \times \text{RTO}_{overall}, \tag{2}$$

where α is 0.5 for the strong RTO estimator and 0.25 for the weak RTO estimator.

To avoid a steep RTO increase after measuring a weak RTT, and to maintain the overall RTO estimation stability, modifications were applied to the weak RTO estimator when compared to the strong RTO estimator:

- Weak RTT measurements are only allowed for up to the second retransmission in order to avoid very large weak RTT measurements (which could overestimate the RTO) and because the probability of obtaining veridical RTT information decreases with every retransmission.
- The value of K that determines the impact of RTTVAR for the weak RTO estimator is changed from 4 to 1. This reduces the impact of RTTVAR on the weak RTO estimation, since RTTVAR tends to grow large, especially if more than one retransmission is used.
- When calculating the overall RTO, the weak RTO estimator contributes less than the strong RTO estimator by using a reduced weight (0.25) for the weak RTO value. Although considering weak RTT information is necessary, strong RTTs provide more reliable input on the expected RTTs, and deliver a more accurate RTO estimation.

The parameter values chosen for RTOweak calculations were shown to reduce the fluctuations without compromising

[3]https://tools.ietf.org/html/rfc6298

the stability [2]. Like in default CoAP, CoCoA dithers the initial RTO of a transaction by choosing it from the interval [RTO$_{overall}$, $1.5 \times$ RTO$_{overall}$].

B. Variable Backoff Factor (VBF)

For small initial RTOs, a BEB may not increase the RTO fast enough to allow the network to recover from congestion, still offering high load to the network and increasing the chance for spurious retransmissions. On the contrary, for large initial RTOs, a BEB may overestimate the RTO, leading to an unnecessary delay increase.

To address these problems, CoCoA applies a VBF that adjusts the backoff factor depending on the initial RTO value of a transmission. If the initial RTO is very small (below 1 s), a larger backoff factor is applied to retransmissions (VBF = 3). If a transaction initiates with a large RTO value (above 3 s), a smaller backoff factor is chosen for retransmissions (VBF = 1.5). For transactions that initiate with an RTO between 1 and 3 s, the VBF is set to 2, corresponding to a BEB.

Several backoff factor values for the VBF have been considered and evaluated [2]. Based on the evaluation results and in consensus with the CoCoA specification authors and the IETF CoRE working group, the set of backoff factors presented above was chosen[4].

C. RTO aging

If estimated RTO values are not updated for an extended period of time, the probability that they are no longer valid becomes high. In IoT networks, network conditions, and thus the RTT, can change fast. To avoid bogus RTO values due to such changes, CoCoA applies an aging mechanism to small and large RTO estimations. If a RTO estimation is small or large (below 1 s or above 3 s, respectively), and no new RTT measurement is made for 16 or 4 times the current RTO, respectively, the RTO value is modified to approach the default initial value.

D. Congestion Control for NON messages

CoAP NON messages do not trigger ACKs from the receiver, therefore being used if end-to-end reliability is not required. Default CoAP does not limit the rate of outgoing NON messages towards a destination endpoint. CoCoA introduces congestion control for NON messages, limiting the rate of outgoing NON messages towards a destination endpoint to one message every RTO seconds. Since RTO estimation requires the presence of round trip type interaction, CoCoA mandates the use of a controlled fraction of CON messages among the NON messages to be transmitted. For the sake of brevity, in this paper we only focus on congestion control for CON messages. The interested reader is referred to [3] for evaluations of NON type message traffic.

[4]For the determination of other parameter settings of CoCoA, the same approach was followed. The investigation of the performance-complexity trade-off brought by adaptive parameter setting approaches is a future research direction for CoCoA.

IV. Alternative Congestion Control Mechanisms

For a wider understanding of congestion control for the IoT, besides comparing default CoAP with CoCoA, in this paper we also analyze other RTO calculation algorithms such as the Linux TCP RTO (Linux-RTO) estimator [4], the peak-hopper TCP RTO estimator (PH-RTO) [5] and also a CoCoA variant that only uses the strong RTO estimator (CoCoA-S).

Linux-RTO adds two mechanisms to the basic TCP RTO algorithm. First, when a new RTT measurement is smaller than the previously gathered RTT information, the RTO is not increased, avoiding peaks in the RTO value when the channel seems to improve. Second, Linux-RTO avoids the RTO estimator to converge into a RTT value after repeatedly measuring constant RTT values [4], which could lead to spurious retransmissions.

PH-RTO reacts to a sudden RTT increase with a RTO increase by using a short term RTT history, which then decays over time towards the value of a long term RTT history. PH-RTO intends to avoid spurious retransmissions by using the long term history, when the channel suffers from sudden delays. RTO dithering is not defined for the Linux and PH-RTO algorithms, which were not designed for IoT scenarios.

Above these two state-of-the-art algorithms used in TCP, we include a minimalist Basic-RTO' (B-RTO) estimator in the evaluations as a benchmark, which always sets the initial RTO for a transmission to a random value between 1 and 1.5 times the previously measured RTT. B-RTO can use weak RTT measurements. An overview of the features of all six analyzed congestion control mechanisms is given in Table I. None of the two considered TCP-oriented congestion control mechanisms takes into account the peculiarities of IoT traffic, such as high BER, sporadic transmissions and traffic bursts. On the other hand, B-RTO provides poor RTO estimation due to its simplicity. In contrast, CoCoA introduces features like the weak RTT, the VBF, and RTO aging, designed for IoT traffic. As a result, CoCoA is expected to outperform the alternative congestion control mechanisms.

V. Experimental Setup and Test Configuration

A. Testbeds

GPRS and IEEE 802.15.4 are used in this paper for evaluations in two respective experimental setups, which also use different hardware to run CoAP servers and clients.

GPRS is a common M2M technology that allows a flexible network setup for Internet connectivity. IEEE 802.15.4 targets low-power communication and is a common interface employed by many IoT standards, including ZigBee and 6LoWPAN. GPRS and IEEE 802.15.4 are interesting for this study because they have different bit rates and delay characteristics. Moreover, GPRS involves a single wireless hop, whereas IEEE 802.15.4 networks are often deployed as multihop networks.

In the first setup, a laptop running CoAP clients uses a Matrix MTX-65-ULP GPRS modem to connect to the Internet, from where packets are routed towards a PC running a CoAP server. When compared to a wired connection, much larger RTTs and a much higher RTT jitter are observed over the

TABLE I: Overview of the Features of the Different Congestion Control Mechanisms

	Strong RTTs	Weak RTTs	Dithering	Backoff method	RTO aging	Use backed-off RTO after no RTT update	RAM usage per client	Main Goal
Default CoAP	No	No	Yes	BEB	No	No	2 Bytes	IoT Traffic
CoCoA	Yes	Yes	Yes	VBF	Yes	No	29 Bytes	IoT Traffic (adaptive)
CoCoA-S	Yes	No	Yes	VBF	Yes	No	19 Bytes	IoT Traffic (adaptive)
Basic RTO	Yes	Yes	Yes	BEB	No	Yes	2 Bytes	Minimalist RTO Solution
Linux RTO	Yes	No	No	BEB	No	Yes	21 Bytes	TCP RTO Enhancement
PH-RTO	Yes	No	No	BEB	No	Yes	43 Bytes	TCP RTO Enhancement

GPRS link, as well as a higher chance for packet losses. In our testbed we observe uplink/downlink data rates of approximately 15/40 kbit/s.

In this setup, both the CoAP clients and server run the Java Californium (Cf) CoAP [6] implementation. The alternative congestion control mechanisms considered in this paper are implemented and publicly available in Cf[5]. For a fair comparative evaluation, while the PH-RTO and Linux algorithms were designed for TCP, we have implemented these algorithms for CoAP (over UDP). Other TCP features are not present in our evaluation.

In the second setup, the interaction between a cloud service and a multihop, low-power wireless network is analyzed. CoAP clients running in the cloud service are connected via Ethernet to an IEEE 802.15.4 testbed where all motes run CoAP servers. FlockLab, a publicly available IEEE 802.15.4 indoor/outdoor testbed composed of 30 TelosB motes [7], is chosen for this setup, a publicly available IEEE 802.15.4 indoor/outdoor testbed composed of 30 TelosB motes [7]. The FlockLab motes run ContikiMAC [8] with radio duty cycling (RDC) enabled, which is required to save energy in real deployments. On the client side, Cf is used as CoAP implementation, while the server motes in FlockLab run the full IPv6-based ContikiOS stack for constrained devices, including the Erbium (Er) CoAP implementation [9].

When compared to the GPRS setup, FlockLab imposes additional challenges for the congestion control mechanisms, such as a considerable packet loss rate and RTT variance due to different route lengths and RDC [9]. Packet losses in this setup mostly emerge from lossy links and from packet drops due to full buffers in the border router and relay nodes close to it. The border router is the FlockLab node that provides Internet connectivity to the Flocklab motes.

B. *Traffic Scenarios*

For both testbeds (GPRS and FlockLab), two traffic scenarios are defined to explore the effect of different congestion control mechanisms on the performance of CoAP communications:

1) *Continuous Traffic*: In this scenario, CoAP clients send CON requests to a CoAP server. When a client receives a reply from the server, the client immediately sends another CON request. Sending messages back-to-back by many clients simultaneously can create congestion. The number of clients is varied from 10 to 40 (in steps of 10) in order to achieve different degrees of congestion. In the GPRS setup, one server is running on the destination device. In FlockLab, one client is assigned to each CoAP server mote in the testbed. A continuous traffic test lasts 180 s.
2) *Burst Traffic*: This scenario starts with a low congestion level, where 10 clients (GPRS) or 5 clients (FlockLab) generate continuous traffic of back-to-back CON requests. Then, a burst of traffic is generated by a new group of clients that send 50 (GPRS) or 25 (FlockLab) back-to-back CON requests to the servers. Such traffic patterns can correspond to a local event (such as alerts about presence, temperature, etc.). The burst of messages causes a congestion peak. For the GPRS setup, we vary the number of clients that generate burst traffic. In FlockLab, for each mote that is not a destination of continuous traffic, we create a client that generates burst traffic.

Tests are repeated 15 times for each specific configuration.

VI. Congestion Control Evaluation Results

A. *Performance Metrics*

In the continuous traffic scenario, the overall throughput as successfully finished transactions per second is chosen as performance metric, merging delay and packet delivery ratio into one single value.

In the burst traffic scenario, we analyze the Settling Time of the different congestion control approaches. We define the Settling Time as the time it takes for the clients to finish at least 80% of the burst traffic transactions. This is an important metric, since traffic bursts are expected when CoAP transactions are event-based or transmissions from various senders are synchronized.

Furthermore, the congestion control mechanisms behavior is analyzed regarding their fairness in FlockLab, which is challenging given the different path lengths and the tree-like topology of the scenario. Jain's Fairness Index (FI) [6] is used as the fairness metric, which ranges between 0 and 1, and a higher FI indicates a higher fairness level.

B. *Throughput Results*

Nearly all RTT-sensitive mechanisms outperform default CoAP independently from the network setup in terms of throughput in the continuous traffic scenario (Fig. 1).

In the GPRS setup, since the packet loss rate is low, the performance mainly depends on how the RTO algorithms adapt to the RTT. In this setup, RTT increases with the amount

[5]https://github.com/eclipse/californium/tree/congestion-control

[6]http://www.rfc-base.org/rfc-5166.html

TABLE II: Comparison of the Average RTT and Initial RTO Values in Milliseconds for Different Numbers of Clients in the GPRS Setup and the FlockLab Setup

	10 GPRS clients		20 GPRS clients		30 GPRS clients		40 GPRS clients		FlockLab	
	RTT	RTO	RTT	RTO	RTT	RTO	RTT	RTO	RTT	RTO
Default CoAP	-	2497	-	2499	-	2499	-	2506	-	2505
CoCoA	661	1505	1437	3379	1936	4119	2796	5431	1507	3710
CoCoA-S	625	1428	1275	2903	1880	4122	2795	4928	656	3227
B-RTO	1025	1152	1962	2198	2983	3272	4733	4441	3172	3266
Linux RTO	682	1325	1550	2801	1863	3345	2931	5797	598	4424
PH-RTO	746	1797	1835	3703	1827	4112	3194	6213	625	4796

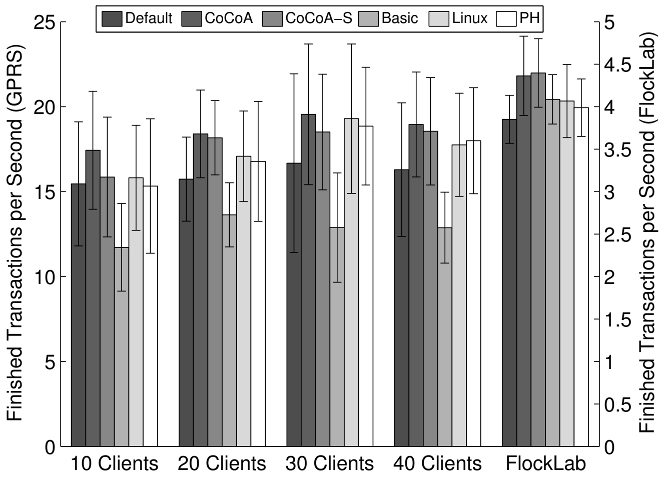

Fig. 1: Average throughput with 95% confidence intervals achieved by the evaluated congestion control mechanisms in the GPRS and FlockLab setups.

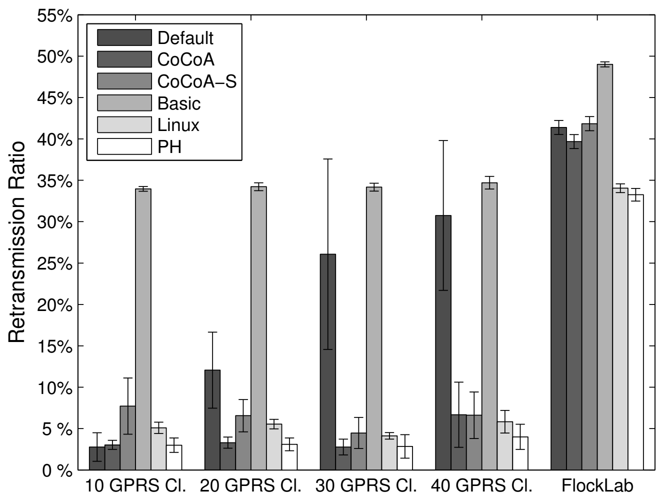

Fig. 2: Average percentage of CoAP retransmissions over all CoAP transmissions observed for the evaluated congestion control mechanisms in the GPRS and FlockLab setups.

of active clients due to the delay introduced by the queuing of packets in the GPRS modem, which occurs since the overall generated data rate exceeds the uplink capacity. With the average RTT increase, the RTT-sensitive RTO algorithms increase their initial RTO values (Table II). In FlockLab, the average RTT is small, yet the average initial RTO for RTT-sensitive algorithms is larger than in the GPRS setup. This can be ascribed to a greater amount of weak RTTs that increase the RTO value and backed-off RTO values due to packet drops as a consequence of overflowing buffers near the border router, where traffic mostly concentrates.

Default CoAP underperforms independently from the setup (except in the GPRS setup with B-RTO) since it uses a fixed range of initial RTO values and does not adapt to the current RTT. If the real RTT is noticeably below the default RTO range, CoAP reacts slowly to losses. If the RTT lies in the RTO range or even exceeds it, spurious retransmissions are likely to happen, as indicated by the increasing percentage of retransmissions with the number of clients in GPRS (Fig. 2).

CoCoA achieves the highest throughput in the GPRS setup. In the same setup, CoCoA-S does not perform as well as CoCoA since it only allows strong RTT measurements, generally resulting in slightly lower RTO values and thus increasing the probability of spurious retransmissions (see Fig. 2). In FlockLab, CoCoA and CoCoA-S perform very similarly. Their features allow to benefit especially from links with good connections and small RTTs, but they also adapt the RTO and avoid bogus values even in a lossy network: the VBF and the aging mechanisms effectively limit the growth of RTO values when retransmissions are necessary. Without these features, CoCoA would tend to calculate very large RTO values for retransmissions using the BEB, whereas the overall RTO would not be shifted towards the default value of 2 s in absence of further RTO updates.

The throughput obtained with B-RTO suffers noticeably due to its simplicity. If after measuring a small RTT the following transaction RTT is larger, which is likely given the RTT fluctuations in both network setups, the RTO timer will fire ahead of time with a high probability. In fact, B-RTO exhibits the highest retransmit ratio of all tested algorithms in all settings (Fig. 2). On the other hand, when a large RTT is measured, the next RTO used by B-RTO can grow very large due to the random multiplier, potentially leading to low throughput.

While Linux and PH-RTO perform better than default CoAP, they are not able to outperform CoCoA. Linux often calculates smaller RTO values, causing a higher amount of spurious retransmissions (Fig. 2), since contrarily to CoCoA it does not increase the RTO when the RTT decreases. The PH-RTO reacts to a sudden RTT increase with a peak in the RTO that then slowly decays in the following transactions. However, given the continuous RTT jittering that is characteristic for both network setups, a sudden RTO increase may not

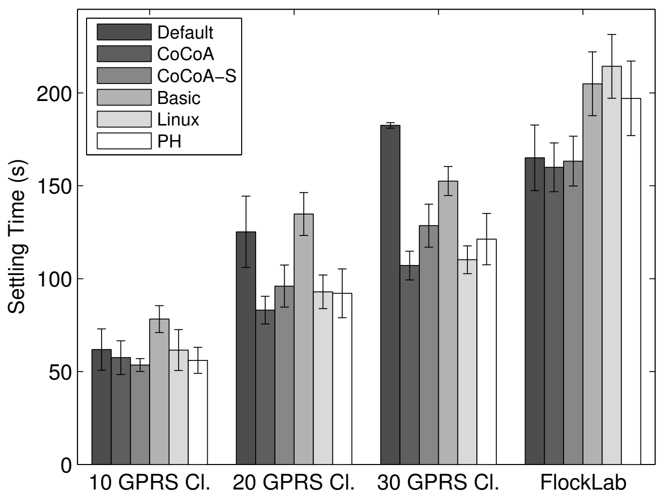

Fig. 3: Average Settling Times with 95% confidence intervals achieved by the different RTO mechanisms in the burst traffic scenario. For the GPRS experiment results, Settling Times larger than 180 s are valuated as 180 s.

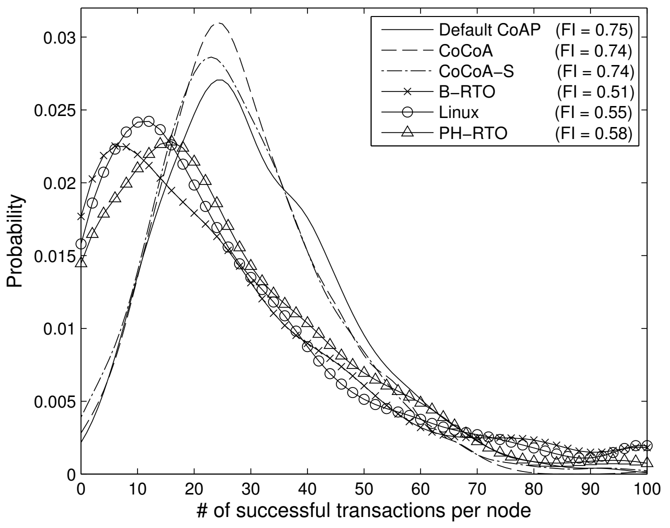

Fig. 4: Probability Mass Function for the number of finished transactions per node and the Fairness Index (FI) achieved by each of the analyzed algorithms. For illustration purposes, more than 100 finished transactions per node are valuated as 100 finished transactions per node.

be necessary and can lead to larger idle times if subsequent packets are lost. A disadvantage of both Linux and PH-RTO is their limitation to using only strong RTTs, while weak RTT measurements could provide additional RTO estimator updates. Instead, the old RTO is maintained and backed-off, i.e., large RTO values are reused for new transactions. The reuse of backed-off RTO values by these two algorithms happens frequently in FlockLab due to packet losses, leading to long idle times that reduce throughput.

C. Settling Time Results

Fig. 3 shows the average Settling Times obtained by the congestion control mechanisms in the burst traffic scenario. In the GPRS setup, results reveal that all RTT-sensitive mechanisms, except B-RTO, are able to improve the performance of default CoAP when there is congestion. For 10 burst clients (i.e., low congestion), the different congestion control approaches perform similarly, except for B-RTO since it tends to produce spurious retransmissions. When the amount of burst clients increases to 20 and 30, the RTT-sensitive mechanisms adjust their RTOs to higher RTT values. CoCoA does this most efficiently, followed by Linux-RTO, PH-RTO, and CoCoA-S, which perform slightly worse. While CoCoA-S tends to be too aggressive during the burst, Linux-RTO and PH-RTO need slightly longer time to adjust their RTO timers during congestion, since they do not exploit weak RTT information. The VBF and the aging mechanisms used in CoCoA reduce the chances for spurious retrans- missions or long idle times, effectively increasing performance. With few clients and short RTTs, B-RTO deteriorates performance, being too aggressive. However, it gets more conservative with more clients and larger RTTs, eventually increasing performance when compared to default CoAP, which does not adapt the RTO timers at all.

In FlockLab, CoCoA yields an improvement over default CoAP in terms of Settling Time and it shows the most stable behavior with the narrowest confidence intervals. Over the course of the tests, CoCoA adapts client RTOs efficiently, so the continuous and burst traffic can be processed in parallel. CoCoA-S behaves similarly, also leading to a minor Settling Time improvement. In contrast, B-RTO, Linux-RTO, and PH-RTO lead to larger Settling Times in average. For these algorithms we observe a prevalent behavior of the continuous background traffic that hampers the sudden burst from being processed quickly and abates slowly since few RTT measurements can be made because of packet losses.

D. Fairness Evaluations

In terms of fairness, important differences are observed between the different congestion control algorithms in FlockLab. Fig. 4 shows the Probability Mass Function (PMF) for the number of finished transactions per destination node for each congestion control algorithm, measured during the continuous traffic experiments along with the FI.

As seen in the figure, with Linux-RTO, PH-RTO and B-RTO, a small group of nodes are served with a very high number of transactions (e.g. more than 80), whereas a large number of nodes obtain a very low number of finished transactions (e.g. below 10). This results in much lower FI values compared to those of CoAP, CoCoA and CoCoA-S. The former methods exploit connections with small hop counts and small RTTs, setting their RTO to small values, behaving aggressively in case of message losses and increasing throughput. However, connections with multiple hops and larger RTTs do not achieve the same data rates, since larger RTO values and consecutive backoffs to these RTO values are applied, reducing throughput.

While CoCoA and CoCoA-S adapt their RTO values as well, the VBF prevents fast retransmissions for good connections with small RTTs and slow retransmissions for bad connections with large RTTs. Moreover, the use of backed-off values when initiating new transmissions is avoided and the aging mechanism prevents from maintaining very small or very large RTO values in idle periods. Thus, CoCoA(-S) does not sacrifice from fairness when compared to default CoAP Fig. 4, while achieving the performance improvements previously presented. Default CoAP behaves neutrally in terms of fairness, since its RTO computation algorithm is independent of the characteristics of a specific path.

E. Memory footprint considerations

There exists a trade-off between performance and memory footprint of the congestion control mechanisms. Improving the behavior of default CoAP congestion control requires one order of magnitude more memory consumption per CoAP client (Table I). However, and despite the limitations of many IoT devices, the additional state required is negligible compared with the state needed for other components in a CoAP implementation such as security support. In fact, Datagram Transport Layer Security (DTLS) is mandatory as per the CoAP specification, and it consumes around 2 kB of RAM [10].

VII. Conclusions

CoAP specifies a conservative and non-adaptive congestion control mechanism. CoCoA, an advanced congestion control mechanism for CoAP, provides a flexible and adaptive solution by combining an adaptive RTO calculation, the use of weak RTTs, a VBF, and an aging mechanism to optimize performance.

In comparison with default CoAP, CoCoA increases throughput and reduces the time it takes for a network to process traffic bursts, while not sacrificing fairness. CoCoA consistently delivers a performance that is better than, or at least similar to, that of default CoAP. In contrast, other approaches may be too simple (B-RTO) or do not adapt well to IoT communications (Linux-RTO, PH-RTO), underperforming default CoAP under certain conditions, and therefore not being recommendable as congestion control mechanisms for CoAP.

VIII. Acknowledgements

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, and FEDER. The authors would like to thank Dr. Carsten Bormann for his major contribution to advanced congestion control for CoAP, and the IETF CoRE WG for providing feedback on CoCoA.

References

[1] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, Mar. 2012.

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

[3] A. Betzler, C. Gomez, and I. 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, & Ubiquitous Networks, PE-WASUN 2015, Cancun, Mexico, November 2-6, 2015*, 2015, pp. 63–70.

[4] P. Sarolahti and A. Kuznetsov, "Congestion Control in Linux TCP," in *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2002, pp. 49–62.

[5] H. Ekstrom and R. Ludwig, "The peak-hopper: a new end-to-end retransmission timer for reliable unicast transport," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, March 2004, pp. 2502–2513 vol.4.

[6] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable cloud services for the internet of things with coap," in *Internet of Things (IOT), 2014 International Conference on the*, Oct 2014, pp. 1–6.

[7] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, April 2013, pp. 153–165.

[8] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, 2011.

[9] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A Low-Power CoAP for Contiki," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, Oct 2011, pp. 855–860.

[10] A. Sehgal, V. Perelman, S. Kuryla, and J. Schonwalder, "Management of resource constrained devices in the Internet of Things," *Communications Magazine, IEEE*, vol. 50, no. 12, pp. 144–149, December 2012.