Congestion Control for CoAP Cloud Services

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

Matthias Kovatsch
Institute for Pervasive Computing
Department of Computer Science
ETH Zurich, Switzerland
Email: kovatsch@inf.ethz.ch

***Abstract*—The Constrained Application Protocol (CoAP) is a new Web protocol for the Internet of Things that allows to connect IoT devices directly to services hosted in the cloud. CoAP is based on UDP to better fit the requirements of constrained environments with resource-constrained nodes and low-power communication links. Being an Internet protocol, CoAP must still adhere to congestion control, primarily to keep the backbone network stable. Thus, the base specification uses conservative parameter values for the number of open requests, the retransmission timers, and the overall message rate. More powerful CoAP nodes, however, can use metrics to optimize these parameters to achieve a better quality of service. For this, the IETF CoRE working group is designing an advanced congestion control mechanism for CoAP called CoCoA. This paper presents first evaluation results for a mechanism that improves the communication between cloud services and resource-constrained IoT devices. We implement CoCoA for the Californium (Cf) CoAP framework and evaluate its performance on a wireless sensor network testbed that runs IPv6. Our results show that CoCoA can better utilize the available network capacity and can increase throughput by 19–112%.**

I. Introduction

The Internet of Things (IoT) aims for the interconnection of numerous smart objects to create a link between the virtual world of IT systems and the real world of physical artifacts. Through the introduction of IPv6 and 6LoWPAN, this vision is now emerging. Using the Internet Protocol (IP) for seamless network convergence, it is now possible to connect smart objects directly to the Internet. To benefit from a rich ecosystem of services in the IoT, however, devices must also be interoperable at the application layer. Due to the success of the World Wide Web as the de facto application layer of the Internet, the IETF recently standardized a new Web protocol that fulfills the requirements of low-power IoT devices. The *Constrained Application Protocol (CoAP)* [12] provides a modular protocol suite that brings RESTful Web services to the IoT, and also enriches the Web with indispensable features for machine-to-machine (M2M) communication. CoAP is a compact binary protocol that uses the UDP transport. On the one hand, UDP is better suited for constrained environments with limited resources in nodes and lossy low-power multihop communication in networks [4]. On the other hand, it is a better fit for M2M applications because it allows for group communication and significantly reduces round-trip times opposed to the three-way handshake of TCP.

Internet protocols are required to provide congestion control so that the open Internet backbone remains operational and stable independent from the number of connected nodes. In most cases, this is realized by using TCP as transport. Current TCP implementations combine a plethora of mechanisms and metrics to optimize the connection between two endpoints. It can effectively react to congestion along the route through the traditional Internet. These mechanisms and metrics usually break, however, when one endpoint is part of an edge network, a low-power lossy network (LLN) that is not designed for transit traffic [4]. Yet most IoT devices are connected through such LLNs.

When using UDP, the application protocol must provide congestion control. Thus, the CoAP base specification defines a basic scheme, which uses a retransmission timeout (RTO) with exponential backoff, whereas the RTO is selected from a fixed interval of values. Furthermore, it limits the number of outstanding interactions between an endpoint and a given destination. To improve this scheme, the IETF CoRE working group is currently working on a draft specification called CoAP Congestion Control Advanced (CoCoA) [1]. It allows to dynamically determine the RTO based on RTT samples and it also allows a greater amount of outstanding interactions.

It is expected that cloud services will play a key role in the management and orchestration of the IoT [2], [8], [14]. Thus, one of the main use cases we see for CoCoA is when traditional Internet nodes communicate with IoT devices in low-power edge networks. These are usually connected through a border router (BR) that provides access to the LLN. Problems may arise if the amount of active interactions between the cloud and the network of constrained devices either surpasses the capacity of the BR (bottleneck issue) or if the bandwidth of the LLN is exceeded. In this paper we analyze if CoCoA is capable of improving the performance in such use cases in terms of throughput and request/response exchange durations when compared to default CoAP.

II. CoAP Congestion Control Advanced (CoCoA)

This section explains the congestion control mechanisms of CoAP. First, we summarize the default mechanism before we introduce CoCoA in more detail. Finally, we present optimizations that go beyond the current version of the CoCoA Internet-Draft.

A. CoAP Default Congestion Control

Besides RESTful requests and responses, CoAP includes a messaging sub-layer that provides deduplication and optional reliable transmission. For the latter, it provides two different message types to send a request:

- *Confirmable (CON)* messages must be acknowledged by the receiver and are retransmitted up to four times.
- *Non-confirmable (NON)* messages follow a best-effort delivery and are neither acknowledged nor retransmitted.

CoAP applies congestion control by imposing a limit to the number of parallel requests and by using backoffs for the retransmission of CONs. The number of outstanding interactions to a single destination is limited by the parameter NSTART, which is set to a conservative value of one by default[1]. An outstanding interaction is a CON request that has not been acknowledged yet (either by an empty ACK message or the response) or a NON request for which no reply has been received yet. When transmitting CONs, the retransmission timeout (RTO) is initialized with a random value between two and three seconds to avoid synchronization effects. If the timer expires without having received an ACK or response (which can be sent as CON as well) from the destination endpoint, the client assumes message loss and retransmits the request. To dilute network congestion, a binary exponential backoff (BEB) is applied to the next retransmission, thereby doubling its RTO value. After four retransmissions without reply, the transmission is closed and a new request can be issued to this remote endpoint (although other actions might be more appropriate for the application).

For NON requests, it is hard to define when to give up on an outstanding interaction. Thus, this timeout is usually given by the application requirements. For congestion control, however, clients must not exceed a rate of 1 Byte/second. The same applies for clients that decide to cancel an ongoing CON request to issue a new request to the same endpoint.

B. CoCoA: Confirmable Requests

There may exist network setups where the static restrictions imposed by the base specification can cause the network to underperform. Thus, CoAP Simple Congestion Control/Advanced (CoCoA) [1], attempts to make the restrictions more flexible. Its core mechanism consists of the adaptation of the RTO value for CON requests by using round-trip time (RTT) information for different destination endpoints such as IP addresses or whole IPv6 prefixes. While RTT information is also the basis for the flow control mechanisms of TCP, CoCoA specifically aims for the adaption for constrained node networks.For each destination endpoint, CoCoA maintains two RTO estimators:

- The *strong RTO estimator*, which gathers RTT information (RTT_{strong}) according to Karn's algorithm [5] and follows the guidelines of RFC 6298 [11] by measuring only when no retransmissions occurred.
- The *weak RTO estimator*, which takes RTT values from retransmitted requests (RTT_{weak}) using the time between sending the initial request and obtaining the reply.

The following formulas apply when obtaining a new RTT measurement RTT_{X_new}, where RTTVAR is the round-trip time variation as in RFC 6298 and X stands for *strong* or *weak* accordingly:

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

$$\text{RTTVAR}_X = (1-\beta)\cdot \text{RTTVAR}_X + \beta \cdot |RTT_X - RTT_{X\_new}| \tag{2}$$

with $\alpha = \frac{1}{4}$ and $\beta = \frac{1}{8}$. When RTT_X is determined, it is used to update RTO_X as

$$\text{RTO}_X = RTT_X + K_X \cdot \text{RTTVAR}_X, \tag{3}$$

with $K_{strong} = 4$ and $K_{weak} = 1$. The point of using a weak estimator in addition to the strong estimator is to increase the probability of gathering RTT information for network environments where packet losses are likely to happen. Still, the weak estimator is less reliable and likely to overestimate the RTT because of the ambiguity of which CON actually triggered the reply.

When either RTO_{strong} or RTO_{weak} is updated after obtaining a RTT measurement, the overall RTO ($\text{RTO}_{overall}$) is updated as an equally weighted average of its previous value and the newly obtained RTO estimation:

$$\text{RTO}_{overall} = 0.5 \cdot \text{RTO}_X + 0.5 \cdot \text{RTO}_{overall} \tag{4}$$

It is used as next initial RTO for CON messages to the same destination endpoint. It also adjusts the maximum transmission rate for NON messages to $1/RTO_{overall}$. Two out of 16 consecutive NON messages must be changed into CONs, however, to allow for an update of the RTO estimator. The overall RTO is initiated with two seconds, while the maximum RTO value is defined to be 60 s. The state of the RTO estimators must be kept for at least 255 s before clearing inactive destination endpoints.

CoCoA also allows to use values of NSTART greater than one, thereby permitting several outstanding interactions towards one destination endpoint in parallel. The assumption made by CoCoA is that when RTT information and adapted RTO timers are used, the outgoing traffic automatically adjusts to the capacity of the network. If NSTART is greater than one and no RTO information is available yet, the initial RTO of each new CON message is set to $2\,s \cdot 2^{\text{ACT}}$, where ACT is the number of already ongoing exchanges.

To avoid exchanges with large initial RTOs to take very long and to avoid that retransmissions are used too fast when the initial RTO is very short, the variable backoff factor (VBF) is applied. With the VBF, three different backoffs are applied to retransmissions, depending on the initial RTO of the transmission:

$$\text{VBF}(\text{RTO}_{init}) = \begin{cases} 3, & \text{RTO}_{init} < 1s \\ 2, & 1 <= \text{RTO}_{init} <= 3s \\ 1.3, & \text{RTO}_{init} > 3s \end{cases} \tag{5}$$

[1]It may be increased for closed networks without Internet connectivity.

If a small overall RTO ($\text{RTO}_{overall} < 1\ s$) is not updated during 16 times its current value, its value is increased as

$$\text{RTO}_{overall} = \text{RTO}_{overall} \cdot 16. \quad (6)$$

The values and thresholds are experimental and need to stabilize while the proposal evolves.

C. CoCoA: Optimizations

For the evaluations carried out in this paper and in line with the ongoing optimization of the CoCoA draft, we modify some of CoCoA's mechanisms and add new ones. We add an aging mechanism for RTO estimators with large RTOs. If an RTO estimator was not updated for at least one minute and its RTO value is larger than the default of two seconds, the RTT and RTTVAR of the estimators are adjusted to be reduced:

$$\text{RTT}_X = (2 + \text{RTT}_X)/2; \text{RTTVAR}_X = \text{RTTVAR}_X/2. \quad (7)$$

The reasoning for this mechanism is that RTT information may become obsolete after some time and may no longer reflect the current network state, in particular when low-power wireless links are involved.

Recent discussions in the working group indicate that using RTT measurements obtained after the first retransmission are likely to distort the values maintained by the weak RTO estimator, since they may differ essentially from the real RTT. Therefore, RTT measurements taken after the first retransmission are ignored. A further optimization is to reduce the contribution of the weak RTO estimator to the overall RTO upon a weak RTT measurement:

$$\text{RTO}_{overall} = 0.25 \cdot \text{RTO}_{weak} + 0.75 \cdot \text{RTO}_{overall} \quad (8)$$

This helps to reduce the fluctuation of the overall RTO that is caused by the ambiguity of the multiple possible CON-ACK pairs when retransmissions occur. Latter improvements are considered to be included in future versions of the draft. Another improvement to avoid strong fluctuations of the RTO is the use of a history with configurable size for implementations of CoCoA for unconstrained devices, where the last calculated values for $\text{RTO}_{overall}$ are used to calculate an averaged RTO value.

III. CoCoA Californium Implementation

We implemented CoCoA and the optimizations introduced in the last section as an optional congestion control layer for the Californium (Cf) CoAP framework [7]. It provides all necessary data structures and methods to carry out the mechanisms introduced in the previous section. A `RemoteEndpoint` object stores the information needed by the congestion control layer, such as the RTT windows and the current state of the RTO estimators. Since Californium is designed for unconstrained environments, each remote endpoint is identified at the highest granularity, that is, by its unique IP address and port number. This allows to even react to congestion at specific services, which are provided at different UDP ports of a host.

Whenever a new request-response exchange is created, it is associated to its remote endpoint. After performing this association, it is possible to access the state information of the remote endpoint from all layers of the CoAP stack, as the `Exchange` object is passed around. The congestion control layer first determines whether the transmission is a CON or a NON, as each type is processed differently.

When an exchange with a CON message reaches the congestion control layer, the NSTART limit for open interactions with the corresponding remote endpoint is checked. If less than NSTART exchanges are active to that remote endpoint, the request can be processed and it is forwarded to the reliability layer. In this context, observing does not count as active exchange. If the limit of NSTART is already reached, the new request is added to a queue that is bounded through a maximal lifetime for stored exchanges. As soon as an incoming reply is handed up from the reliability layer, the congestion control layer updates the RTO estimators of the associated endpoint. After updating the state information of the remote endpoint, the outstanding interaction is closed and the next CON exchange can be pulled from the queue.

When a NON exchange is handed over to the congestion control layer, it is stored in a queue, which implements a leaky bucket traffic shaper [13] for the associated remote endpoint. With a rate of $1/RTO_{overall}$, NONs are pulled from the queue and handed down to the lower layers in the stack. Following the guidelines of the CoCoA draft, every eighth NON is converted to a CON in order to obtain a RTT measurement to assure that the RTO value gets updated from time to time. The evaluation of this mechanism is out of scope of this paper, though, as we focus on the adaptation of the RTO value for CON requests.

IV. Evaluation

In this section we carry out a performance comparison of the default CoAP congestion control with CoCoA. We evaluate if the use of advanced congestion control mechanisms improves the quality of service in terms of higher throughput and faster processing of requests by the network. In an experimental setup, Cf-clients try to access information stored on constrained devices in an LLN over the Internet. We define three different scenarios for the evaluations that differ in the amount of generated traffic and in the interaction patterns between the Cf-clients and the LLN. In the following we introduce the general experiment setup and then go into the details and results of the different scenarios.

A. Experiment Setup

The experiment setup can be divided into two parts: the client side and the server side. On the client side, we set up a PC to run multiple instances of the Californium client to represent cloud services that access Web resources on sensor nodes. For the server side, we use the FlockLab testbed [9] with CoAP servers on 30 Tmote Sky motes [10], which provide a test resource that allows GET requests. The motes are programmed with the full ContikiOS 6LoWPAN

TABLE I: CoAP Congestion Control Schemes

Congestion Control used in clients	RTO base value	NSTART
Default CoAP	2 s	1
Default CoAP$_B$ ($\frac{\text{RTO}_{init}}{2}$)	1 s	1
CoCoA	2 s	1
CoCoA$_4$	2 s	4

communication stack. This includes the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [15] and Erbium CoAP [6]. One of the central motes in the FlockLab is set up as border router to connect the LLN with the Internet. For our evaluations, we furthermore compare two link-layer configurations: radio duty cycling (RDC) with ContikiMAC [3] and no RDC.

On the client side, four different CoAP congestion control schemes are evaluated as listed in Table I. The first one uses default CoAP, that is, our CoCoA layer for Californium is disabled. The second one explores how the performance of default CoAP is affected when it uses less conservative RTO values: we reduce the initial RTO from 2 s to 1 s (CoAP$_B$), expecting CoAP to react faster to packet losses not originating from congestion but from lossy links. The latter two configurations use our CoCoA layer, while the evaluations are carried out for the cases NSTART=1 (CoCoA) and NSTART=4 (CoCoA$_4$). With NSTART=4 we want to determine if relaxing the restriction of only one open request to an endpoint at a time increases the performance, while maintaining network stability when applying advanced congestion control mechanisms.

All test runs for a scheme are repeated ten times with a duration of 15 minutes each. In the following we define three scenarios for the interaction between the cloud services and the LLN and present the results of the measurements.

B. Baseline: 1-to-1

In the baseline scenario, we measure the performance for the clients under basic network conditions. We observe how long it takes for a single client to exchange 50 CON-ACK pairs (request and piggybacked response) with a single CoAP server in the LLN. As metrics, we give the average throughput (# of requests processed per second), the average exchange duration (time it takes to obtain an ACK to a CON request), and the average amount of retries used per original CoAP request. This experiment is repeated for every mote in the LLN separately and determines how well CoAP and CoCoA perform when there is no congestion within the LLN and at the border router.

TABLE II: Results for the Baseline Scenario (No RDC).

Config	Throughput (req./s)	Exch. duration	# of retries
CoAP	0.67	1.49 s	0.40
CoAP$_B$	0.88	1.13 s	0.52
CoCoA	1.18	0.84 s	**0.38**
CoCoA$_4$	**1.42**	**0.70** s	0.56

TABLE III: Results for the Baseline Scenario (ContikiMAC).

Config	Throughput (req./s)	Exch. duration	# of retries
CoAP	0.56	1.76 s	0.21
CoAP$_B$	0.79	1.26 s	0.35
CoCoA	0.89	1.12 s	**0.07**
CoCoA$_4$	**1.18**	**0.84 s**	0.35

Table II shows the metrics when no RDC is configured. The average throughput achieved by clients using CoCoA and CoCoA$_4$ is higher than the throughput achieved with default CoAP. The adaptive RTO mechanisms ensure that advanced congestion control mechanisms use the available bandwidth more efficiently than default CoAP. The peak performance is achieved with NSTART=4, which doubles the average throughput. The results for CoAP$_B$ reveal that reducing the fix interval of RTO$_{init}$ can help to improve the throughput as well in this scenario. Analogical to the throughput, the average exchange duration is the lowest for CoCoA$_4$ and the highest for CoAP.

With ContikiMAC (Table III), the same tendencies are observed, however, the average throughput decreases for all congestion control schemes, since the duty cycling introduces larger delays to communications within the LLN. Independent from the RDC, we draw the conclusion that in networks with low or no congestion, the use of CoCoA with NSTART$>$1 is recommended, as it leads to a significant increase in throughput.

C. Many-to-many

In the many-to-many scenario, we evaluate how the congestion control mechanisms perform in environments that suffer from heavy congestion. During the setup phase of the experiment, we assign a separate client to each of the CoAP servers in the LLN. As soon as the test starts, all clients continuously exchange CON-ACK pairs with their associated CoAP servers until the test finishes. The continuous requests cause significant congestion at the border router bottleneck and inside the LLN.

Tables IV and V show that the highest overall throughput is achieved with clients that use CoCoA. With NSTART=1, each client adapts the RTO to a saturated and highly congested network and a higher overall throughput is achieved when compared to default CoAP. Allowing more parallel exchanges with NSTART=4 does not increase the performance noticeably. In a situation where the LLN and the border router are highly congested, it is difficult to adjust the RTO timers adequately when more CoAP messages are transmitted in parallel by the clients. If the clients use CoAP$_B$, a performance loss is observed, since the more aggressive yet static RTO setting does not adapt to the heavy congestion of the network. The aggressive behavior also results in using the highest

TABLE IV: Results for the Many-to-Many Scenario (no RDC).

Config	Throughput (req./s)	Exch. duration	# of retries
CoAP	4.23	2.32 s	**0.28**
CoAP$_B$	4.05	2.15 s	1.08
CoCoA	**5.34**	1.94 s	0.43
CoCoA$_4$	4.59	**1.65** s	0.67

TABLE V: Results for the Many-to-Many Sce. (ContikiMAC).

Config	Throughput (req./s)	Exch. duration	# of retries
CoAP	1.60	5.72 s	1.81
CoAP$_B$	1.45	6.06 s	2.33
CoCoA	1.79	**4.16** s	**1.69**
CoCoA$_4$	**1.90**	5.12 s	2.16

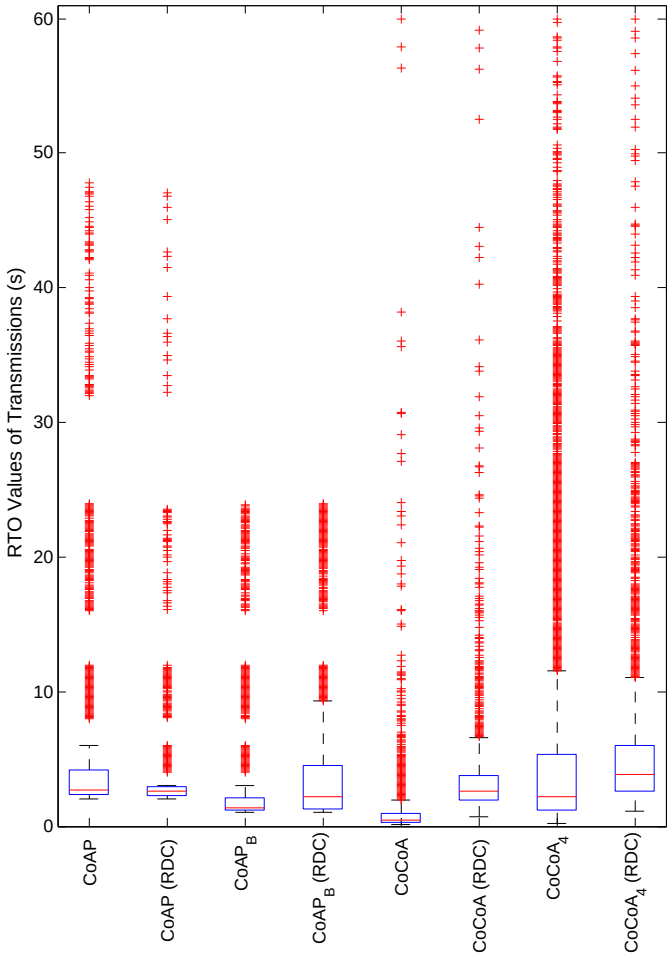


Fig. 1: Boxplots for all congestion control schemes (and RDC configurations) showing the RTO values observed during test runs in the many-to-many scenario with median (lines within the boxes), the first and third quartiles (bottom and tops of boxes), 1.5 times the interquartile range of the first and third quartiles (whiskers) and outliers (crosses).

number of retries, exceeding an average of one retry per CoAP packet sent. It is also important to notice that the clients running CoCoA have the shortest exchange durations for both configurations, with and without RDC.

The results of the many-to-many case show that the use of adaptive RTOs has a high impact on the performance. Fig. 1 shows the typical distribution of RTO values observed during the tests runs in the many-to-many scenario. The static RTO intervals of default CoAP that result from the BEB for retransmissions are visible for the CoAP and CoAP$_B$ cases. Since CoCoA dynamically adjusts the RTO, it uses a wider range of RTO values: the boxplots for CoCoA and CoCoA$_4$ show lower minimums and a wider spectrum of outliers.

D. Cross Traffic Burst

In the cross traffic burst scenario, we observe how clients that continuously exchange data with an LLN react to a state of sudden congestion in the LLN. For this experiment, we distinguish two groups of clients: The first group, consisting of four clients, continuously sends requests to four randomly picked CoAP servers in the LLN throughout the whole test duration. The second group consists of up to 25 clients, one for every other CoAP server of the LLN, and is set to initiate half-way through the test (at the instant of $t = 450\ s$). At the point the second group of Cf-clients initiates, they begin the exchange of a limited number of requests (50) between them and the CoAP servers. The sudden appearance of these CoAP messages causes a peak in the network congestion. We determine how this traffic burst affects the performance of the different congestion control schemes by measuring the overall throughput and looking at the time it takes to transmit the burst of traffic.

Figures 2a and 2b show the observed RTO values used for (re)transmissions as well as the cumulative distribution functions (CDFs) for the constant traffic and interfering burst. In both figures, the 100% mark of the constant traffic CDF refers to the amount of completed requests with CoCoA during the experiment.

Up to the burst at 450 s, the network only suffers from a low degree of congestion. During this phase, the clients are able to achieve a high throughput, since the performance is similar to the one observed in the baseline scenario, where no congestion was present. CoCoA therefore is able to transmit more messages than default CoAP in the same time interval. For CoAP$_B$, and CoCoA$_4$ (not visualized) this is the case as well. A change occurs as soon as the additional traffic is introduced in the network, causing a high degree of congestion for a limited time. Now the constant and burst traffic clients are contending. The CDF of the successfully transmitted packets of the burst traffic shows that default CoAP needs more time to process the burst traffic requests. Conversely, CoCoA processes the messages of the burst in a shorter time span and returns to the initial network state much faster, thanks to the adaptive RTO that adjusts to the sudden traffic burst.

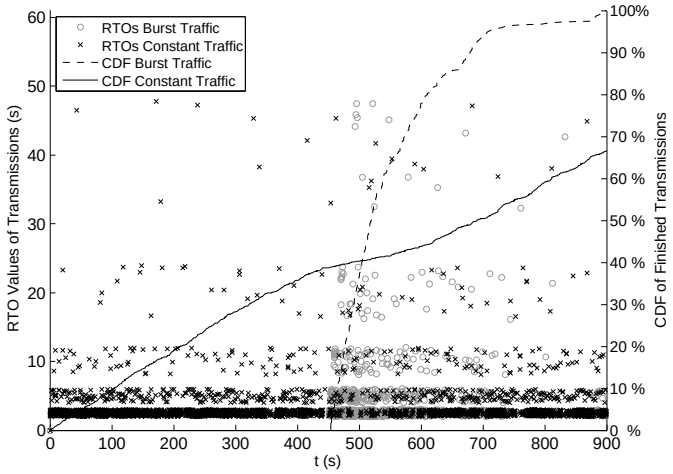
Tables VI and VII show that CoCoA performs best with and without RDC. Since with CoCoA$_4$ every client may send up to 4 requests in parallel, the network is driven into congestion in the pre-burst phase (nullRDC). Since the calculated RTOs are small, clients react with fast retransmissions to congestion losses or losses from lossy links, which increases congestion further and can lead to an important increase of the RTOs of

TABLE VI: Results for the Burst Scenario (No RDC).

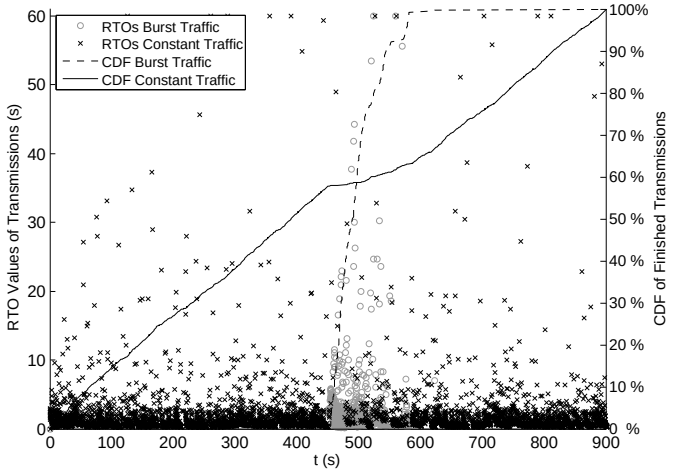
Config	Throughput (req./s)	Exch. duration	# of retries
CoAP	2.65	1.86 s	0.22
CoAP$_B$	2.98	2.21 s	0.37
CoCoA	**3.98**	**0.84 s**	**0.21**
CoCoA$_4$	2.48	1.43 s	0.52

TABLE VII: Results for the Burst Scenario (ContikiMAC).

Config	Throughput (req./s)	Exch. duration	# of retries
CoAP	1.58	2.21 s	**0.35**
CoAP$_B$	1.32	2.52 s	0.98
CoCoA	1.69	2.09 s	0.39
CoCoA$_4$	**1.95**	**1.94** s	0.79



(a) Default CoAP



(b) CoCoA

Fig. 2: RTO values for (re)transmissions and CDF of successful requests for constant and burst traffic during a testrun with clients using CoAP (a) and CoCoA (b) with no RDC.

subsequent retries with a VBF of 3. Both, fast retransmissions and large backoffs after the LLN recovers from congestion lead to a poor overall trouhgput. On the other hand, since with ContikiMAC the delays in the LLN are larger, the calculated RTOs are larger and fast retransmissions are avoided. This leads to a more stable behaviour and a better overall performance of CoCoA_4 when ContikiMAC is enabled. CoAP_B cannot adapt to the network state and the reduced initial RTO interval is prone to cause congestion. Only when RDC is disabled and during the congestion free pre-burst phase, CoAP_B is able to achieve a higher trouhput than CoAP. In all other situations (with ContikiMAC, burst phase) CoAP_B performs worst.

The generally short average exchange duration and low amount of retries can be explained by the fact that most of the measured values are obtained during the initial and end phases, where the degree of congestion is low and fewer retransmissions are needed.

V. Conclusions

In this work, we present the details of our implementation and evaluation of CoCoA for the Californium (Cf) CoAP framework as an optional congestion control layer. We carry out experiments with cloud services that communicate with CoAP servers on real sensor nodes in a testbed and compare how well the different congestion control schemes for CoAP perform. In three different traffic scenarios, we determine that the improvements achieved by CoCoA are twofold: the amount of requests that can be processed in parallel increases and the time it takes for clients to complete their tasks decreases. The advanced congestion control mechanisms achieve this by calculating optimized RTO timers and adjusting the backoff behaviour dynamically. We also show that NSTART can be increased to higher values safely, even though it may not always deliver the best performance.

In future work, CoAP observe notifications need to be evaluated in combination with the rate control of Non-confirmable CoAP messages as proposed by the CoCoA draft.

VI. Acknowledgements

This work was supported in part by the Spanish Governments Ministerio de Economía y Competitividad through project TEC2012-32531, and FEDER.

References

[1] C. Bormann. CoAP Simple Congestion Control/Advanced. I-D draft-bormann-core-cocoa-01, 2014.

[2] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota. REST Enabled Wireless Sensor Networks for Seamless Integration with Web Applications. In *Proc. MASS*, Valencia, Spain, 2011.

[3] A. Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science, 2011.

[4] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proc. SenSys*, Raleigh, NC, USA, 2008.

[5] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. *ACM SIGCOMM Computer Communication Review*, 17(5):2–7, 1987.

[6] M. Kovatsch, S. Duquennoy, and A. Dunkels. A Low-Power CoAP for Contiki. In *Proc. MASS*, Valencia, Spain, 2011.

[7] M. Kovatsch, M. Lanter, and Z. Shelby. Californium: Scalable Cloud Services for the Internet of Things. In *Proc. IoT*, Cambridge, MA, USA, 2014.

[8] M. Kovatsch, S. Mayer, and B. Ostermaier. Moving Application Logic from the Firmware to the Cloud: Towards the Thin Server Architecture for the Internet of Things. In *Proc. IMIS*, Palermo, Italy, 2012.

[9] 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 *Proc. IPSN*, Philadelphia, PA, USA, 2013.

[10] Moteiv Corporation. *TMote Sky: Ultra low power IEEE 802.15.4 compliant wireless sensor module*, 2/6/2006 edition, 2006.

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

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

[13] M. Sidi, W.-Z. Liu, I. Cidon, and I. Gopal. Congestion Control Through Input Rate Regulation. In *Proc. GLOBECOM*, Dallas, TX, USA, 1989.

[14] I. Thomas, S. Gaide, and M. Hug. Composite Business Ecosystems for the Web of Everything: Using Cloud Platforms for Web Convergence. In *Proc. IMIS*, Taichung, Taiwan, 2013.

[15] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC6550, 2012.