Comparison of CoAP and CoCoA+ Congestion Control Mechanisms for Different IoT Application Scenarios

Emilio Ancillotti, Raffaele Bruno
IIT-CNR, Pisa, Italy
{*a.ancillotti,r.bruno*}@iit.cnr.it

***Abstract*—CoAP is a web transfer protocol that provides basic RESTful services for IoT devices. Since CoAP operates on top of UDP, CoAP must support additional congestion control and reliability mechanisms. However, CoAP is originally designed to allow implementations that do not maintain any end-to-end connection information. Thus, the default congestion control mechanisms are not capable of adapting to network conditions. A few enhancements of the basic congestion control mechanism of CoAP have been recently proposed, and CoCoA+ is commonly considered the most mature of these proposals. However, the shortcomings of CoCoA+ have not been sufficiently investigated in the literature. In this paper, we present an in-depth analysis of the congestion control mechanisms of CoCoA+ and CoAP for a variety of traffic patterns under realistic IoT setups. The results of this evaluation indicate that CoCoA+ can perform significantly worse than default CoAP, especially with bursty traffic and in networks with few clients as a result of an improper selection of the retransmission timeouts. Furthermore, large variability of RTO values in CoCoA+ makes difficult to meet end-to-end delay requirements of typical IoT monitoring applications.**

***Index Terms*—Internet of Things, CoAP, congestion control, Cooja, performance evaluation.**

I. Introduction

The future Internet of Things (IoT) foresees information systems seamlessly integrated with smart objects, i.e. physical objects empowered with computation and communication capabilities [1]. Due to the success and ubiquity of IP-based technologies, both the academic and industrial world seems to converge towards an IP-based protocol stack for IoT. Indeed, in the past years various IETF working groups have been initiated to extend the Internet protocols to efficiently operate in IoT environments [2]. The most relevant of such WGs are: i) 6LoWPAN, which is focusing on a convergence layer to enable IPv6 on constrained networks [3]; ii) ROLL, which is specifying RPL, a routing protocol for low power and lossy networks [4]; and iii) CoRE, which is designing CoAP, a specialised web transfer protocol for IoT devices [5].

From a high-level perspective, CoAP can be considered a radically simplified UDP-based analogue of the HTTP protocol with low header overhead and parsing complexity. CoAP is designed to provide basic RESTful services for IoT devices, as it enables a request/response interaction model between application endpoints, with both endpoints acting as clients and servers. Following the REST-based model, sensors, actuators and control systems can be represented as resources, which are identified via URIs (Universal Resource Identifiers) [6] and offer their services through RESTful web services. Then, CoAP offers basic methods to create, read, update and delete resources; asynchronous message exchanges; simple proxy and caching capabilities; and optional support for resource discovery [5].

Because CoAP operates over an unreliable data-transfer protocol (i.e. UDP), it needs to implement additional reliability mechanisms. To this end, CoAP base specification proposes to use a retransmission timeout (RTO) to identify lost packets that need to be retransmitted. Clearly, retransmitted packets contribute to traffic load and CoAP recommends the sender to *retransmit lost messages at exponentially increasing intervals*, until it receives an acknowledgment (or runs out of attempts). Since CoAP operates over UDP it deals with network congestion by itself using a very simple *stop-and-wait congestion control mechanism*. To reduce protocol complexity, CoAP does not require to compute and manage any end-to-end connection information to control the exchange of CoAP messages. The downside of such simplicity is that the default CoAP behaviour is basically insensitive to the network conditions. Previous studies have shown that the simple congestion control mechanism included in the CoAP base specification can lead to poor network performance, as it tends to be too conservative in small networks and too aggressive in large networks [7], [8].

To address the aforementioned shortcomings a few enhancements of the default CoAP congestion control scheme have been recently designed. The most mature of these proposals is CoCoA+ [9], [10], which has been also recently standardised by the IETF CoRE WG [11]. CoCoA+ makes the default CoAP congestion control scheme adaptive to network dynamics by defining a novel round-trip time estimation technique, together with variable back-off factor and ageing mechanisms. Preliminary results indicate that CoCoA+ often outperforms default CoAP, especially in large networks, while it performs very similar to default CoAP under certain conditions [10]. However, there are still limitations in the algorithms and mechanisms of CoCoA+. The first one is that *CoAP cannot correctly identify the RTT of retransmitted messages*, which may lead to improper selection of RTO values in CoCoA+ [12] Furthermore, in the case of short RTTs or few clients, it is likely that the calculated RTO values are not suitable

to avoid unnecessary retransmissions. Another main problem with CoCoA+ is that large fluctuations of RTO estimates, combined with the simple stop-and-wait retransmission behaviour of CoAP, increase the probability of violating end-to-end delay requirements of IoT applications. Note that typical IoT applications generate *low duty-cycle traffic patterns* with long intervals between consecutive transmission bursts. In addition, often there is the coexistence between *unsynchronised and synchronised transmissions* due to multiple devices that react to the same/similar events. As we will show in the following, the *interplay between RTO estimators in CoCoA+ and burst and periodic traffic can deteriorate network performance*.

In this paper, we carry out an in-depth analysis of how CoAP and CoCoA+ perform into two typical IoT application scenarios: continuous monitoring and event detection. Using the Cooja simulator and an open-source implementation of an IoT stack by ContikiOS we evaluate these schemes in realistic IoT setups. Our main focus is to investigate the scalability properties of CoCoA+, as well as the impact of different traffic characteristics on the RTO estimates. To this end, we dissect the causes of packet losses in the evaluated scenarios. Our results show that, in contrast to previous studies, CoCoA+ can perform significantly worse than default CoAP. In the evaluated scenarios, CoCoA+ increases the overall packet loss rate by up to 30% with respect to default CoAP when traffic is periodic and offered load is small (or there are few clients). With bursty traffic performance degradations with CoCoA+ are even more significant. Furthermore, our results also indicate that CoCoA+ is not able to select a proper RTO values when bursty traffic exists, due to poor estimates of RTTs. Finally, large variability of RTO values in CoCoA+ makes difficult to meet end-to-end delay requirements of typical IoT monitoring applications. Thus, we believe that traffic-aware methods for selecting the most suitable RTO values for the current network conditions and traffic characteristics are needed to increase the network performance.

The rest of the paper is organised as follows. In Section II, we provide an overview of the congestion control mechanisms for CoAP and CoCoA+, and we discuss main shortcomings of these schemes. In Section III, we introduce the simulation setup and we present the results of our analysis. The conclusions of this paper and future research directions are discussed. in Section IV.

II. Background and Motivation

In this section, we first describe how congestion control is implemented in default CoAP. Then, we introduce the new mechanisms added by CoCoA+. Based on this analysis, we discuss the possible shortcomings of these two schemes.

A. CoAP-CC

The interaction model of CoAP is similar to the client/server model of HTTP. However, differently from legacy HTTP, request/response interactions between clients and servers occurs asynchronously over a datagram-oriented transport such as UDP. As CoAP is bound to unreliable transports, it implements optional end-to-end reliability for *confirmable* messages, i.e., messages that require an explicit acknowledgement from the destination endpoint. Reliability is achieved by the sender detecting lost messages and retransmitting them (up to a maximum number of times). CoAP uses a basic technique to identify packet losses: the retransmission timeout (RTO). Basically, when a confirmable message is sent a timer is set to the RTO value. If the timer expires and the sender has not yet received an acknowledgement from the destination endpoint, a loss is assumed and the CoAP message is retransmitted. CoAP was specifically designed to enable implementations that do not maintain round-trip-time (RTT) measurements. Therefore, the RTO value for the initial message transmission is randomly selected from a fixed interval[1].

Packet losses can be caused by channel errors or network congestion. Basic congestion control for CoAP is provided by a binary exponential back-off (BEB) scheme, which doubles the RTO value of the retransmitted packets. Another simple technique that is applied in CoAP in order to reduce network congestion is to limit the number of simultaneous outstanding interactions towards one particular destination endpoint to NSTART. The base specification of CoAP recommends to set NSTART equal to one, which is equivalent to implement a stop-and-wait protocol, but more sophisticated congestion control algorithms can be designed to allow a value for NSTART greater than one. For the sake of brevity, in the following we denote with CoAP-CC the default congestion control scheme of CoAP.

B. CoCoA+

CoCoA+ is a simple advancement of CoAP-CC, which makes CoAP congestion control more responsive to network conditions. Basically, CoCoA+ combines different mechanisms to adapt the RTO values of transmitted messages on the basis of RTT information that is obtained by monitoring ACK packets. Specifically, CoCoA+ maintains two separate RTO estimators: $i)$ the strong RTO estimator (RTO_s), which relies on the RTT measurements taken from message exchanges that complete on initial transmissions, and $ii)$ the weak RTO estimator (RTO_w), which relies on the RTT measurements taken from message exchanges that have required at most two retransmissions[2]. To compute the current RTO value, the CoAP source node follows the principles of RFC 6298 [13] and maintains two state variables, the $SRTT$ and $RTTVAR$ values, which denote the smoothed RTT and RTT variation. Then, following the same approach as in [13], it is assumed that:

$$RTO_X = SRTT_X + K_X \times RTTVAR_X \ , \qquad (1)$$

where $X \in \{s, w\}$, K_s is 4 as in RFC 6298, and K_w is 1. The lower K_w factor is motivated by the need of lessening the

[1]The CoAP specification recommends to use the interval [2,3] seconds, but CoAP transmission parameters can be modified and CoAP implementation in ContikiOS uses the interval [3,6] seconds.

[2]Note that to reduce packet header overheads when a request is retransmitted, it has the same message identifier as the initial one. Thus, ACKs cannot be associated to specific retransmitted packets.

impact of high $RTTVAR_w$ values due to large fluctuations of RTT measurements of retransmitted messages. Finally, the overall RTO estimate is a classical exponential weighted moving average of the strong and weak RTO estimators. More formally,

$$RTO = \alpha_X RTO_X + (1 - \alpha_X) RTO \ , \qquad (2)$$

with $\alpha_s = 0.5$ and $\alpha_w = 0.25$. Using different weights allows to reduce the contribution of the weak estimator, which might be otherwise dominant in lossy networks.

In CoCoA+ the classical binary exponential back-off is substituted by a variable back-off in which the multiplicative factor used to compute the RTO value for the retransmitted packets is not fixed but it is adaptive and changes according to the initial RTO value of a transmission (RTO_{init}). More formally, it holds that

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

The rationale behind the above selection of the variable back-off factor is to avoid that with short RTO_{init} values all available retransmissions are confined in a short interval (causing network congestion), or that large initial RTOs may lead to unnecessary long transmission delays.

Another addition of CoCoA+ is an RTO ageing mechanism that is applied to update RTO estimators when there are not new RTT measurements for long periods of time and current RTO values become outdated. Specifically, if an RTO estimator has a value that is lower than 1 s, and it is not updated during more than sixteen times its current value, the RTO estimate is doubled. On the contrary, if an RTO estimator has a value that is lower than 1 s, and it is not updated during more than four times its current value, the RTO estimate is changed to RTO $= 1 + (0.5 \times \text{RTO})$.

C. Discussion on shortcomings of CoAP-CC and CoCoA+

The design of CoCoA+ is primarily motivated by the need to improve CoAP-CC performance by providing dynamic and controlled adaptation of the retransmission timeout based on RTT measurements in a way that is suitable for the peculiarities of IoT communications. Indeed, IoT communications are expected to be subject to large and variable round-trip times due to radio duty cycling and lossy links. In addition, IoT traffic is typically sporadic, which makes more difficult to maintain updated state information. Thus, CoCoA+ makes use of all available traffic, included retransmitted packets, to collect RTT measurements while mitigating the impact of large fluctuations of RTT estimates. However, collecting reliable RTT measurements from retransmitted packets is difficult in CoAP because the message ID is maintained unchanged during retransmissions and the CoAP source node cannot establish the correct RTT of request/response interactions. Furthermore, RTT values may vary significantly in an IoT network [14] even when the network conditions are relatively stable. Small RTT estimates can easily cause unnecessary retransmissions because they lead to small RTOs, which dampen the ability of the retransmission timer to efficiently cope with RTT fluctuations.

Although CoAP-CC and CoCoA+ differ in the way they manage retransmissions, they share the same stop-and-wait approach to flow control: by default, the CoAP source nodes transmits a single message at a time. This behaviour is application agnostic and it can create potential shortcomings and inefficiencies. For instance, the source node can be blocked because it must wait for ACKs that might not arrive due to packet losses. Similarly, long messages induce long transmission delays that reduce throughput. In addition, it can be very difficult to satisfy stringent end-to-end delay requirements due to the multiplicative increase of RTO. Finally, the RTO estimators may not work properly with traffic bursts (e.g., traffic that is generated when an event is simultaneously detected by multiple monitoring nodes in the network) [12]. In this case, the selected RTO value may underestimate the real RTT, causing unnecessary retransmissions.

III. Evaluation

In this section, we give the details on the simulation setup used to conduct a comparative performance analysis of CoAP-CC and CoCoA+. Then, we discuss the results for the two traffic scenarios considered in this study.

A. Simulation Setup

CoAP-CC is currently included in the default IETF communication protocol stack that is provided with ContikiOS, while the code of CoCoA+ has been shared with us by the authors of [9]. ContikiOS toolset includes Cooja, a simulation platform that allows to precisely emulate off-the-shelf wireless sensor node hardware, and to execute real code. Thus, for our performance comparison we conduct simulations in Cooja. At the physical (PHY) and MAC layers, the nodes implement IEEE 802.15.4, using a data transmission rate of 250 kbps in the 2.4 GHz radio band. We do not use radio duty cycling (RDC) at the MAC layer to mitigate RTT fluctuations. To model realistic radio propagation and interference we use the Multipath Ray-tracer Medium (MRM), which supports multipath effects [15]. We configure MRM parameters to achieve a 100% success rate at 10 meters and an interference range of 20 meters. Figure 1 shows the link loss rate (LLR) as a function of node distance for a link under our MRM model. It is important to note that MRM provides more realistic link behaviours than the simple on-off link model that was used in [9].

RPL is used as routing protocol. As in [9] we consider a $n \times n$ grid topology, with the DODAG root at the centre of the grid, while the sink (i.e. destination endpoint of the CoAP client application) is one of the corner nodes of the grid topology. The distance between nodes is 10 meters. Two types of node are used in the simulations: TMote Sky mote for the RPL boarder router (10 kB of RAM and 48 kB of ROM), and Z1mote for the clients (8 kB of RAM and 92 kB

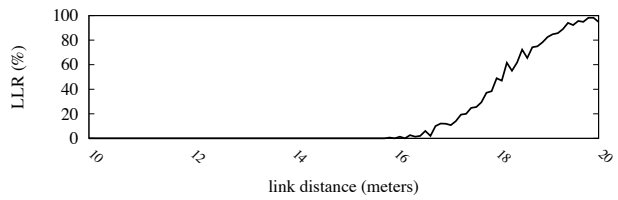


Fig. 1. Link characterisation in Cooja simulator under the MRM model.

of ROM). To evaluate different network scenarios, we vary the number of nodes from 9 to 36 (i.e. $n \in [3, 6]$). We do not consider larger networks because ContikiOS implements only the storing mode for RPL and the RAM capabilities of the TMote Sky mote do not allow to store routing information for larger networks. Finally, each node run an application that generates PUT messages that are used to update the state of a REST resource at the sink node. Different message sizes can be used by the CoAP clients. Then, CoAP messages are fragmented into small link-layer frames of size equal to 32 bytes. Congestion control mechanisms are applied to individual frames, i.e. different RTO values are computed for the first transmission of each fragment of the CoAP message. The different types of traffic patterns generated by the application are described in the following section.

All simulations last 3 hours and 95% confidence intervals are computed by replicating each simulation ten times.

B. Traffic scenarios

We analyse the efficiency of CoAP-CC and CoCoA+ CC for two common types of IoT applications. Note that the same traffic scenarios are also used in [9], [10].

1) Continuous monitoring: In this scenario each node periodically generates CoAP messages for a sink node reporting a sensor measurement. Note that the monitoring processes of different nodes are assumed to be asynchronous. Since data is generated from simple sensor readings (e.g., temperature, velocity) packet size is small. If not otherwise stated, in this scenario the CoAP message size is equal to 96 bytes, which requires three link-layer fragments. Different traffic loads are evaluated by varying the number of source nodes or the frequency of the reporting period.

2) Global event detection: In this scenario, a global event requires all nodes of the network to transmit one or several packets at the same moment to a sink node. Different traffic loads are evaluated by varying the amount of data that need to be transferred for a single detected event (from 96 bytes to 384 bytes), or the average time between two consecutive global events. We also assume that there is a low amount of periodic traffic for the same sink node (one packet every 60 seconds from each client). This traffic is needed to ensure that CoCoA+ can maintain updated RTT estimates and to conduct a fair comparison with CoAP-CC.

C. Metrics

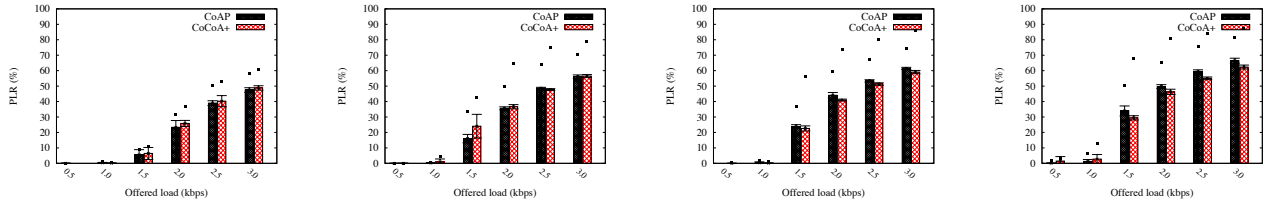
In our evaluation, we consider as application-layer performance metric the overall packet loss ratio (PLR) at the sink node, defined as the percentage of failed message transmissions over the total number of CoAP PUT messages sent by a node. To assess the efficiency of CoAP and CoCoA+ congestion control mechanisms we measure various statistics related to the RTO calculation. First, we measure the average number of retransmissions for each message fragment. Second, we measure the distribution of RTO values. Third, we compute percentage of "early" retransmissions, defined as the number of times that a fragment is retransmitted although it is correctly received at the sink node and the related ACK is received at the source node. Finally, we measure the distribution of RTT weak and strong estimates.

D. Results for continuous monitoring

The first set of experiments is conducted with the traffic pattern generated by a continuous monitoring application. The traffic generation rate at the nodes is selected in such a way to vary the network-wide offered load from 0.5 kbps to 3 kbps in steps of 0.5 kbps. We do not consider higher traffic rates because, as shown in the following, large PLRs are already observed with small traffic loads.

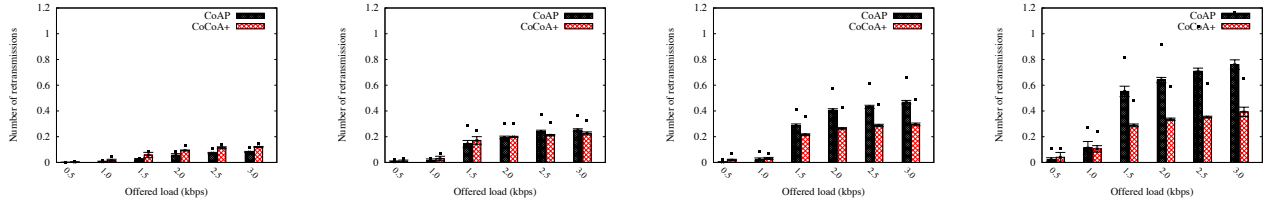
Fig. 2 shows the average PLR for each client node as a function of the offered load and the network size. Several observations can be derived from the shown results. First, a non-negligible number of CoAP messages is already lost with a small traffic load of 1.5 kbps and with 3 kbps PLR is above 60% for both CoAP-CC and CoCoA+. Second, CoAP-CC and CoCoA+ achieve very similar performance for the majority of the evaluated scenarios. Third, CoCoA+ outperforms CoAP-CC only in large networks and for high offered loads. However, PLR improvements are always smaller than 10%. On the contrary, CoAP-CC behaves better than CoCoA+ for small traffic loads or a few clients, with a PLR gain that can be higher than 30%.

To explain the above controversial results, we conduct an in-depth analysis of the inner behaviours of CoAP-CC and CoCoA+ and we dissect the different causes of packet losses. First of all, Fig. 3 shows the average number of retransmissions per message sent by a node in the same network scenarios we evaluated in Fig. 2. In a 3×3 grid topology CoCoA+ generates slightly more retransmissions than CoAP-CC, but retransmissions are sporadic overall. On the contrary, in larger networks CoCoA+ ensures significantly less retransmissions than CoAP-CC (up to a 50% reduction in a 6×6 grid topology). However, this lower number of retransmissions does not have a significant impact on the observed PLR values. To explain this behaviour, we look at the different causes of packet losses and we find out that only a small percentage of packets is dropped as a result of network congestion, i.e. buffer overflows. The large majority of lost packets occurs when the application generates a new packet to be sent and CoAP is still busy in transmitting one of the fragments



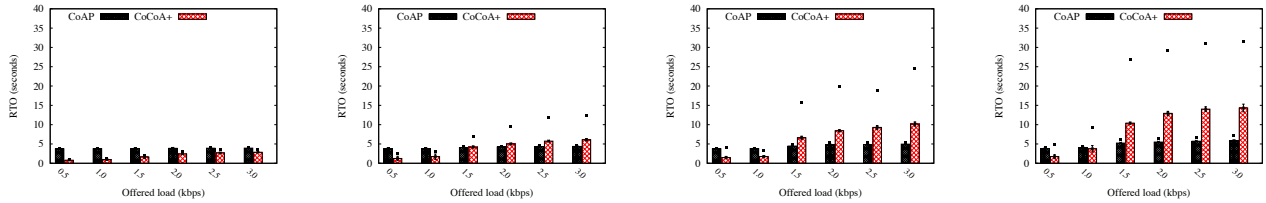
(a) $n = 9$ (b) $n = 16$ (c) $n = 25$ (d) $n = 36$

Fig. 2. Periodic traffic: average PLR of each client versus the number of network nodes and traffic loads. Square points represent the maximum observed PLR. CoAP message size is 96 bytes.



(a) $n = 9$ (b) $n = 16$ (c) $n = 25$ (d) $n = 36$

Fig. 3. Periodic traffic: average number of retransmissions per message sent by a node. Square points represent the maximum number of retransmissions.



(a) $n = 9$ (b) $n = 16$ (c) $n = 25$ (d) $n = 36$

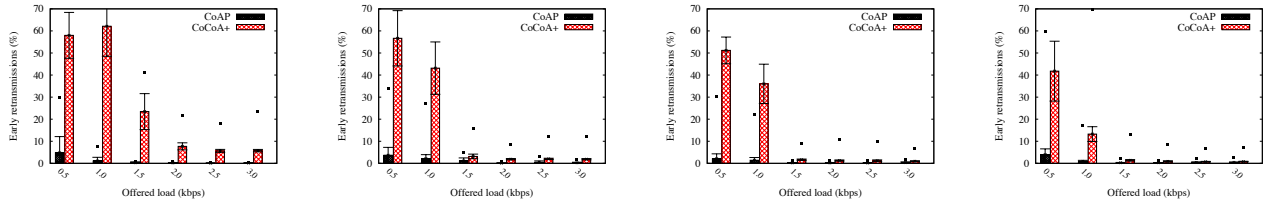
Fig. 4. Periodic traffic: average RTO values. Square points represent the maximum observed RTO values.

of the previous message[3]. The intuitive explanation of this behaviour is that CoCoA+ introduces an excessive delay when retransmitting a fragment. To confirm this claim, in Fig. 4 we show the average RTO values that are selected by CoAP-CC and CoCoA+. As expected, CoAP-CC select very similar RTO values independently of the network size and the traffic load. On the contrary, CoCoA+ behaviour is adaptive to the network conditions. Thus, the higher the network size and the traffic rate, the longer the RTO values. However, the RTO variability has also a relevant impact on the performance of the continuous monitoring application, and the maximum RTO values in CoCoA+ are significantly longer than in CoAP-CC. In Fig. 4 the maximum RTO values in the 6×6 grid with a traffic load above 2 kbps are about 30 seconds. Another limitation of CoCoA+ is that it is more prone to unnecessary retransmissions due to an aggressive selection of the RTO value of the initial transmission. Specifically, Fig. 5 shows the percentage of initial transmissions that are correctly received at the sink node, which are nevertheless retransmitted as a result of the retransmission timeout expiration. The results indicate that CoAP-CC generates a negligible number of early retransmissions, while up to 60% of the first retransmissions are unnecessary in CoCoA+ with low traffic rates and small networks. This means that CoCoA+ initialises RTOs with too short values when RTT estimates are small. Furthermore, even if the average percentage of early retransmissions computed over all the nodes is small, there are clients that suffer from up to 10% of early retransmissions.

E. Results for global event detection

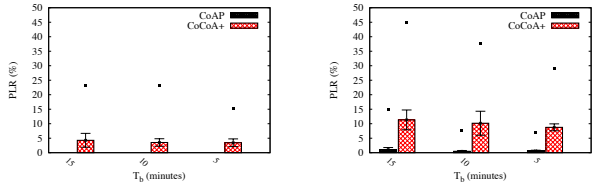
The second set of experiments is conducted with the traffic pattern generated by a global event detection. Different traffic intensities are obtained by varying the average time T_b between consecutive global events. If not otherwise stated, the CoAP message size is equal to 192 bytes in the following simulations, which is equivalent to six fragments. We omit results for the 3×3 grid and the 4×4 grid because in those

[3]We remind that the continuous monitoring application does not implement caching and data needs to forwarded to the sink immediately. Furthermore, by default NTSTART is equal to one, thus CoAP does not allow multiple transactions to the same sink node.



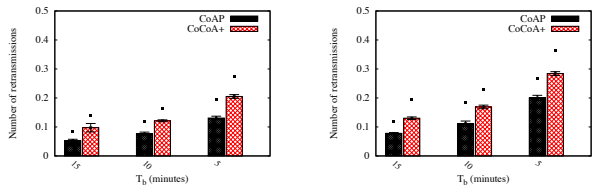
(a) $n = 9$ (b) $n = 16$ (c) $n = 25$ (d) $n = 36$

Fig. 5. Periodic traffic: percentage of early retransmissions. Square points represent the maximum observed percentage.



(a) $n = 25$ (b) $n = 36$

Fig. 6. Bursty traffic: average PLR of each client for different T_b. Square points represent the maximum observed PLR. CoAP message size is 96 bytes.

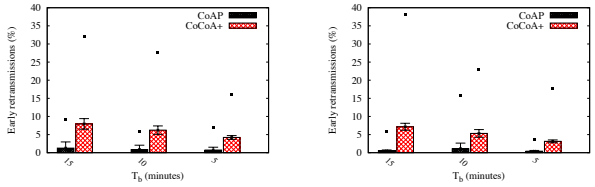


(a) $n = 25$ (b) $n = 36$

Fig. 7. Bursty traffic: average number of retransmissions per message sent by a node. Square points represent the maximum number of retransmissions.

scenarios PLR values are null and CoAP-CC and CoCOA+ behave very similarly. We remind that the main difference between the continuous monitoring scenario and the event detection scenario is that in the former all the nodes of the network transmit simultaneously.

Fig. 6 shows the average PLR of each client node as a function of the average time between two global events. The results clearly indicate the CoCoA+ induces a significant degradation of the network performance with respect to CoAP-CC. For instance, in a 6×6 grid PLR values for CoCoA+ are above 10% while CoAP achieves a PLR lower than 2%. Furthermore, there are clients in the network that can suffer from very high PLRs, up to 40%, while with default CoAP maximum PLR values are always lower than 15%. To explain these results, first of all we analyse the average number of retransmissions per message sent by a node (Fig. 7). The results show that CoCoA+ generates more retransmissions than CoAP-CC. However, the overall number of retransmission is small and it does not seem sufficient to justify the degradation



(a) $n = 25$ (b) $n = 36$

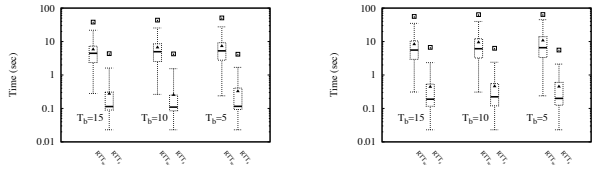
Fig. 8. Bursty traffic: percentage of early retransmissions. Square points represent the maximum observed percentage

(a) $n = 25$ (b) $n = 36$

Fig. 9. Bursty traffic: box-plots of RTO values that are selected for the first transmission of each packet.

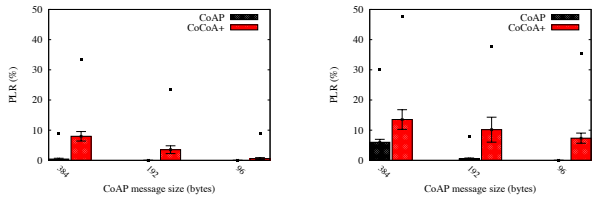
of network performance provided by CoCoA+. Then, in Fig. 8 we report the percentage of early retransmissions (average and maximum values). The results show that there are transmissions that experience a very high probability of unnecessary retransmissions. To clarify this behaviour, in Fig. 9 we report the box-plot[4] of the RTO values that are selected by CoAP-CC and CoCoA+. It is important to point out that we only consider the RTO value of the first transmission of each packet, i.e. we do not include the RTO values of retransmission in the computation of the RTO distribution to avoid the biasing effect due to the variable back-off. The graphs clearly show that there is a huge variability in the selected RTO values, which range from few tens of milliseconds to tens of seconds.

[4]We remind that a box-plot is a way of graphically depicting groups of numerical data through their five-number summaries: the bottom and top of the box are the 25th and 75th percentile, the band in the box is the median (50th percentile) and the ends of the whiskers represent the minimum and maximum of all the data. Note that box-plots are particularly useful to display differences between sets of data without making any assumption on the underlying statistical distribution.



(a) $n = 25$ (b) $n = 36$

Fig. 10. Bursty traffic: box-plots of weak and strong RTT estimates in CoCoA+.



(a) $n = 25$ (b) $n = 36$

Fig. 11. Bursty traffic: average PLR of each client for different CoAP message sizes. Square points represent the maximum observed PLR. $T_b = 10$ minutes.

This can be explained by noting that the clients maintain an updated RTO estimate using the background traffic, which is a low-rate periodic traffic. Since background traffic does not generate network congestion these RTO estimates are very small. On the contrary, when a global event is detected, there is a surge in network contention due to simultaneous transmissions of multiples nodes. CoCoA+ is slow in realising that the network conditions are changed. Thus, CoCoA+ is too aggressive when sending the initial fragments, causing many retransmissions. These retransmissions rapidly increase the RTO values of subsequent fragments. To explain why CoCoA is not be able to select a proper RTO value when bursty traffic exists, Fig. 10 shows the box-plots of the weak and strong RTT values. We can observe that the RTT estimate, in case of no retransmissions, is at least one order of magnitude lower than the RTT estimate with retransmissions. CoCoA+ tries to reduce the impact of weak RTT by using a lower weight factor for weak RTTs in the RTO calculation. However, in case of bursty traffic the difference between weak and strong RTTs may be too high to be compensated by the weight factor. To conclude this evaluation, Fig. 11 shows the average PLR of each client for different CoAP message sizes in the case $T_b = 10$ minutes. As expected, the larger the size of the CoAP message to be transmitted, the higher the number of link-layer fragments that are generated. Thus, not only there is an increase of network congestion due to a larger amount of data to be transmitted, but also the vulnerability of the CoAP transmission to fragment drops increases. Again, the degradation of network performance with CoCoA+ is more significant than with default CoAP.

IV. Conclusions

In this article, we have presented a detailed comparison of the congestion control mechanisms implemented in default CoAP and CoCoA+ for two typical IoT used cases. Differently from previous works our performance analysis reveal that CoCoA+ can perform significantly worse than basic CoAP in a variety of network conditions, such as network with bursty traffic or small RTTs. Then, we have shown that the main cause of these poor performance is that CoCoA+ might be too aggressive when selecting the RTO values of initial transmissions. In addition, RTT estimates obtained from retransmissions are quite unreliable. In future work we plan to investigate cross-layer mechanisms able to select the most suitable RTO values not only on the basis of network conditions but also of traffic characteristics, e.g. to take into account different end-to-end delay requirements of IoT applications and bursty traffic. Furthermore, flow control mechanisms should be included in CoAP to provide better support for continuous traffic.

References

[1] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1–31, 2014.

[2] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of (Important) Things," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1389–1406, Third 2013.

[3] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," IETF RFC 6282, September 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6282.txt

[4] P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF RFC 6550, March 2012. [Online]. Available: https://tools.ietf.org/pdf/rfc6550.txt

[5] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF RFC 7252 , June 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7959.txt

[6] Z. Shelby, "Constrained RESTful Environments (CoRE) Link Format," IETF RFC 6690, August 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6690.txt

[7] T. Pötsch, K. Kuladinithi, M. Becker, P. Trenkamp, and C. Goerg, "Performance Evaluation of CoAP Using RPL and LPL in TinyOS," in *Proc. of IEEE NTMS'15*, May 2012, pp. 1–5.

[8] I. Järvinen, L. Daniel, and M. Kojo, "Experimental evaluation of alternative congestion control algorithms for Constrained Application Protocol (CoAP)," in *Proc. of IEEE WF-IoT'15*, December 2015, pp. 453–458.

[9] 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.

[10] ——, "CoAP Congestion Control for the Internet of Things," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 154–160, July 2016.

[11] C. Bormann, A. Betzler, G. Gomez, and I. Demrkol, "CoAP Simple Congestion Control/Advanced," Internet Draft (expires: April, 2017), October 2016. [Online]. Available: https://www.ietf.org/id/draft-ietf-core-cocoa-00.txt

[12] J. J. Lee, K. T. Kim, and H. Y. Youn, "Enhancement of congestion control of Constrained Application Protocol/Congestion Control/Advanced for Internet of Things environment," *International Journal of Distributed Sensor Networks*, vol. 12, no. 11, November 2016.

[13] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's Retransmission Timer," IETF RFC 6298, June 2011. [Online]. Available: https://tools.ietf.org/pdf/rfc6298.txt

[14] M. Katoh, I. Sato, and N. Watanabe, "Traffic engineering for IoT," in *Proc. of ICOIN'16*, January 2016, pp. 195–200.

[15] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proc. of IEEE LCN'06*, November 2006, pp. 641–648.