RTT-Based Congestion Control for the Internet of Things

Emilio Ancillotti, Simone Bolettieri, Raffaele Bruno

RTT-based Congestion Control for the Internet of Things*

Emilio Ancillotti, Simone Bolettieri, and Raffaele Bruno

IIT-CNR, Via G. Moruzzi 1, 56124 Pisa, ITALY,
{e.ancillotti,s.bolettieri,r.bruno}@iit.cnr.it,

Abstract. The design of scalable and reliable transport protocols for IoT environments is still an unsolved issue. A simple stop-and-wait congestion control method and a lightweight reliability mechanism are only implemented in CoAP, an application protocol that provides standardised RESTful services for IoT devices. Inspired by delay-based congestion control algorithms that have been proposed for the TCP, in this work we propose a rate control technique that leverages measurements of round-trip times (RTTs) to infer network state and to determine the flow rate that would prevent network congestion. Our key idea is that the growth of RTT variance, coupled with thresholds on CoAP message losses, is an effective way to detect the onset of network congestion. To validate our approach, we conduct a comparative performance analysis with the two loss-based congestion control methods of standard CoAP under different application scenarios. Results show that our solution outperforms the alternative methods, with a significant improvement of fairness and robustness against unacknowledged traffic.

Keywords: Internet of things, CoAP, delay-based congestion control, flow pacing, Contiki OS.

1 Introduction

The TCP/IP architecture has significantly evolved over the years to provide new functionalities and efficient support for mobility, content delivery, multimedia applications, etc. Nowadays, there is a large consensus on assuming that one of the most important drivers for the future Internet evolution is the design of solutions to enable access to services and information provided by billions of *smart things*, i.e. resource-constrained devices with sensing and actuation capabilities [1]. This network of physical objects embedded with IP-based networking capabilities is commonly known as the Internet of Things (IoT).

IoT networks differ from traditional wired computer networks in several ways. IoT devices are typically battery-powered and have limited computing power. Furthermore, IoT systems often employ low-energy communication technologies,(e.g., IEEE 802.15.4, Bluetooth LE, etc), which operate with smaller MTUs

* This work was supported by the SIGS project, funded by the region of Tuscany under the FAR-FAS 2014 framework.

and lower transmission rates compared to traditional wired links. Finally, IoT networks have to rely on mesh communications to cope with limited communication ranges. For these reasons, many IETF working groups (e.g., 6LoWPAN [2], ROLL [3], CoRE [4]) have been established to modify the standard TCP/IP stack to fullfill IoT requirements, as well as to develop new protocols [5]. While important achievements have been obtained, the design of a scalable and reliable transport protocol for IoT deployments is still an unsolved issue.

TCP is the dominant transport layer on the Internet, which provides congestion control and reliable delivery. TCP has been primarily designed to efficiently deliver a large bulk of data over a long-lived connections without strict latency requirements. On the contrary, IoT applications are characterised by different communication patterns as IoT devices typically send small amounts of data either periodically or when sporadic events occur. In addition, IoT applications that implement actuation tasks may request low network delays. Lastly, most TCP congestion control algorithms perform poorly with lossy links, and when MAC transmission delays are similar to or longer than the retransmission timeouts. For these reasons, the Internet architecture for IoT networks only envision the use of an unreliable transport protocol (i.e. UDP). A stop-and-wait congestion control method and a lightweight reliability mechanism are only implemented in CoAP, an HTTP-like web transfer protocol for constrained environments.

CoAP uses the received ACKs to decide when transmitting a new packet. In addition, packet loss is implicitly assumed as a network congestion indicator and an exponential back-off is introduced between retransmissions to reduce sending rates. Recently, CoAP extensions are proposed to provide dynamic retransmission timeout adaptation [6]. However, recent studies have investigated the pitfalls of these techniques, such as unfairness and spurious retransmissions when offered loads are close to congestion [7, 8]. In this work, we investigate an alternative approach for congestion control that employs measurements of round trip times to infer network state and to determine the flow rate that would prevent network congestion. We are inspired by some of the delay-based congestion control algorithms that have been proposed for the TCP under specialised conditions (e.g. data centres [9] or high-speed networks [10]). One of our aims is to show that reliable RTT measurements can be obtained in constrained environments with limited additional overheads. Differently from other similar schemes, we do not use fixed RTT thresholds to predict the onset of congestion. Instead, we leverage RTT variability to define a rate control scheme that aims at maintaining a low probability of data transaction losses. Our insight is that in 6LoWPAN networks RTT measurements are not a deterministic function of queue delays and propagation times, while the growth of RTT variance is a more efficient indication of network load increases. To prove our claims we implement the proposed rate control algorithm in CoAP, and we conduct a comparative performance analysis with the two loss-based congestion control methods of standard CoAP under different application scenarios. Results show that our solution outperforms the alternative methods, with a significant improvement of fairness and an increase of packet delivery ratios in presence of unacknowledged traffic.

The rest of this paper is organised as follows. Related works are discussed in Section 2. The proposed RTT-based congestion control method for CoAP is presented in Section 3. In Section 4 we introduce the simulation setup and we present the performance evaluation results. Finally, concluding remarks are discussed in Section 5.

2 Background and Related Works

The design of congestion control algorithms for reliable transport protocols in wired and wireless networks is a deeply investigated topic. In the following, we discuss the key design principles of congestion control solutions for TCP [11]. Then, we present a brief overview of CoAP and its congestion control mechanisms.

2.1 TCP congestion control

The majority of existing end-to-end congestion control proposals for TCP are based on the *congestion window* concept. Specifically, the congestion window is an estimate of the maximum number of unacknowledged data packets that a TCP flow can transmit without congesting the network. Thus, the TCP transmission rate can be indirectly controlled by adjusting the congestion window size. Typically, TCP senders update their congestion windows based on feedback signals they receive from the network. The most popular TCP congestion algorithms, such as TCP NewReno and its many variants, use packet losses as a binary congestion control signal[1]. However, loss-based congestion control algorithms are *reactive* schemes because they reduce sending rates only when network congestion causes a packet loss. The seminal work of TCP Vegas [12] introduces a *proactive* congestion avoidance method that tries to quantify the congestion state of the network before a congestion event occurs using round-trip delays. Specifically, the minimal RTT value is considered a baseline indicating a congestion-free network state. Then, the congestion window is decreased if the delay increases. Other modern TCP variants, such as FAST TCP and Compound TCP, use delay-based congestion control algorithms trying to maintain buffer occupancy at the bottleneck queue around some predefined thresholds. The advantage of such schemes is that they can stabilise the network around the full utilisation. The inherent weakness of delay-based algorithms is that their performance degrades if the delay measurements are noisy (e.g., due to delayed ACKs and route changes). Furthermore, minimum RTT can be a bias estimate of queuing delays in a non congested network. For these reasons, delay is typically used in hybrid congestion control schemes as a secondary congestion signal in addition to packet loss information [10].

[1] Typically, TCP packet losses are detected through the receipt of duplicate and/or selective ACKs, or the timeout of a retransmission timer.

2.2 CoAP

The Constrained Application Protocol (CoAP) is a web transfer protocol based on a REST architecture with a Request/Response interaction model. Requests and responses can be carried in *non-confirmable* (NON) or *confirmable* (CON) messages. The latter are required to be acknowledged once received by the destination endpoint. Reliability is then implemented by detecting message losses and retransmitting the packet (until a maximum number of attempts). Packet losses are detected by using retransmission timeouts (RTOs): RTO is set by the sender when a CON transaction starts; If the RTO expires before the ACK reception, the packet is retransmitted.

In CoAP, the congestion control regulates CON transmission: packet losses are interpreted as a congestion signal; therefore, retransmissions are spaced out using a binary exponential back-off (BEB) policy (RTO doubled at each retransmission). By default, the initial value for the RTO is randomly chosen from a fixed interval. Furthermore, the number of parallel outstanding transmissions towards the same destination is limited to NSTART, which is set to one in standard CoAP (i.e., no more that one notification per RTT to a client on average).

A more advanced congestion control scheme is proposed in CoCoA+ [6], which introduces an algorithm to dynamically adjust the RTO using measured RTTs. The key idea is to overcome the drawbacks of having a fixed RTO that could underestimate node's RTTs leading to unnecessary retransmissions or that could overestimate them leading to increased transaction delays. Specifically, CoCoA+ maintains two RTO estimators for each destination: a strong RTO estimator (RTO_s) uses RTT samples that come from transactions which did not require a retransmission, and weak RTO estimator ($RTOw$) that uses RTT samples of transactions that required less than three retransmissions. Individual $RTOx$, where $x \in \{s, w\}$, are computed using the following formula:

$$RTO_x = SRTT_x + K_x * RTTVAR_x \ , \tag{1}$$

where $SRTT_x$ and $RTTVAR_x$ denote the smoothed RTT and RTT variation respectively, as defined in RFC 6298 [13]; K_s is set to 4, while K_w is set to 1. Upon ACK reception, the computed RTO sample (weak or strong depending on the nature of the last sampled RTT) is used as input for a classical exponential moving average filter to calculate the RTO for the next CON transaction (RTO_{init}). A smaller K_w and a smaller weight for weak RTO estimates are used to avoid large increments in RTO_{init} due to large $RTTVAR_w$, which is affected by the high variability of RTT measurements of retransmitted messages.

Furthermore, CoCoA+ introduces a new back-off policy where the multiplicative factor, used to update the RTO, is variable and depends on the RTO_{init} value. Ideally, this should avoid that with short RTOs all retransmissions are occurring in a short interval, and that, on the contrary, with large RTOs unnecessary delays are introduced. Finally, CoCoA+ proposes an ageing scheme for the RTO estimate. Specifically, if no RTT samples are collected for a sufficiently long period of time, the RTO values should converge toward their initial value.

3 RTT-CoAP Design

RTT-CoAP is an extended version of the standard CoAP web transfer protocol that implements our RTT-based congestion control scheme for 6LoWPAN networks. Unlike classical TCP that employs a window-based congestion avoidance method, RTT-CoAP uses a rate-based algorithm that directly controls the pacing rate of packet transmissions to reduce traffic burstiness and data losses. As discussed in Section 1, we are inspired by recent delay-based congestion control protocols that rely on delay measurements for inferring network congestion [14, 9]. The typical pitfalls of most delay-based congestion control schemes is that it is difficult to obtain a reliable estimate of RTT trends, especially in wireless networks. Furthermore, in constrained environments with small buffers and low transmission rates, RTT can not be considered a deterministic function of the queuing and propagation delays. Thus, it is not easy to define a RTT baseline that is indicative of network congestion. Most solutions simply assume that the minimum of all measured RTTs is the RTT value of a non-congested connection. However, the random channel access method employed in the 802.15.4 MAC standard, which cause huge variability of channel access delays, and bursty packet arrival processes leads to sudden fluctuations in RTT measurements. Furthermore, changing network conditions or network paths generate varying RTT measurements. Thus, single RTT values or the difference between RTT values may not be indicative of the network state. In the following, we explain our RTT measurement framework and how we leverage RTT variations to classify the different network states. Depending on the current network state coupled with thresholds on data transaction losses, RTT-CoAP updates the sending rate of each CoAP source to stabilise network delays and prevent packet losses.

3.1 Monitoring RTT variations

We assume that a CoAP sender can use ACKs to obtain RTT samples. Specifically, an RTT measurement is obtained as the difference between the transmission time of a CON CoAP message and the time in which the ACK is received. According to Karn's algorithm [13], RTT samples should not be taken from packets that were retransmitted, as it is ambiguous to assign the ACK message to the first transmission of that packet or to later retransmissions. To remove this ambiguity we introduce the CoAP timestamp option: the CoAP sender places a timestamp in each CON message, and the CoAP receiver echos these timestamps back in the ACKs. Timing each CoAP message leads to a better RTT estimator without generating excessive protocol overheads, as discussed in Section 3.3.

Inspired by the TCP rules for computing retransmission timers [13], we exploit RTT samples to compute three state variables $SRTT_L$, $SRTT_S$, and $RTTVAR_L$. The first two state variables are obtained by applying exponential smoothing to RTT samples. The third state variable is the well-known smoothed estimate of RTT variations as computed in TCP. Unlike CoAP, we do not distinguish between weak and strong estimators but we use all CON messages to update the RTT state variables. More formally, let r be the latest RTT sample

that is obtained by a CoAP sender. It holds that

$$SRTT_S = \alpha_S r + (1 - \alpha_S) \times SRTT_S \tag{2}$$

$$SRTT_L = \alpha_L r + (1 - \alpha_L) \times SRTT_S \tag{3}$$

$$RTTVAR_L = \beta_L |SRTT_L - r| + (1 - \beta_L) \times RTTVAR_L \ , \tag{4}$$

with $\alpha_S >> \alpha_L$. Using a small smoothing factor α_L is useful to remove the impact of noise on RTT measurements (e.g., due to random backoffs, or bursty packet arrival processes), and to obtain a more accurate estimate of the *long-term* RTT trends. On the other hand, using a large smoothing factor α_S is needed to monitor *short-term* RTT differences that occur due to sudden changes of network conditions. As better explained in the following, this improves the responsiveness of the rate control technique, especially when RTT samples are infrequent due to long network delays or low traffic intensity. It is important to point out that constrained devices have very limited memory, which makes impractical to implement more sophisticated averaging filters that require a long measurement history.

Our key idea is that the growth of RTT variance is a good indication of network state as a contention increase leads to longer backoff intervals, and higher variability of channel access delays. Thus, we leverage $RTTVAR_L$ variable to define *characteristic regions* of the network state, which will be used to determine the direction of sending rate changes (i.e., whether sending rates of CoAP sources should be increased or decreased). Let **S** denote the network state at the reception of an ACK. Furthermore, let us define the decision boundary $T(\gamma)$ as follows:

$$T(\gamma) = SRTT_L + \gamma \times RTTVAR_L \ . \tag{5}$$

To some extent, $T(\gamma)$ is analogous to a confidence level of the mean RTT, as it is a measure of RTT variability and it can be used to assess the distance of RTT samples from the mean. Now, we can identify four characteristic regions using the following decision boundaries:

- (**LC**): $SRTT_S < T(-1)$ (low congestion). In this state we can assume that there is no congestion in the network, as the short-term RTT estimate is well below the long-term averages. Thus, the sending rate can be aggressively increased.
- (**NO**): $T(-1) \leq SRTT_S < T(+1)$ (normal operating point). In this state, the short-term RTT estimate is comparable to typical RTT measurements. This corresponds to a normal operating point and the sending rate should left unchanged. Depending on the experienced packet losses a moderate increase of sending rates is also possible to check if more bandwidth is available.
- (**MV**): $T(+1) \leq SRTT_S < T(2)$ (medium variability). In this state, short-term RTT variability grows and this can be a signal of ramping network congestion. Thus, a controlled decrease of sending rate can be necessary to avoid congestion.
- (**HV**): $SRTT_S \geq T(+2)$ (high variability). In this state, the short-term RTT estimate is significantly higher that the long-term RTT estimate. Thus, the

CoAP source needs to aggressively reduce the sending rate so as to return to the normal operating state as soon as possible.

3.2 Rate control algorithm

Algorithm 1 shows the pseudo-code for our rate adaptation scheme. RTT-CoAP maintains a single sending rate R for each connection and updates it on every ACK reception using the above-described state variables, coupled with a measure of the probability of data transaction loss. Specifically, each CoAP sender maintains also an exponential moving average, say μ, of the probability that a confirmable CoAP message is retransmitted. Since RTT-CoAP aims at keeping this probability very low, it employs two thresholds to respond to extreme cases of high packet losses or possible underutilisation of available bandwidth. We are inspired by the idea of control dead zones in TCP Vegas, which defines a range of network delays that correspond to a steady state during which no changes are applied to the sending rate. This approach is known to mitigate the impact of network condition fluctuations and to facilitate the convergence to a stable behaviour. More formally, we define L_{high} as an upper bound of the tolerable CoAP message loss ratio. It provides a way to infere that there is no more available bandwidth for the CoAP sender. Moreover, we define a low threshold L_{low} to filter out RTT spikes: RTT fluctuations above the *normal operating region* are not taken into consideration when the loss ratio is lower than L_{low}. The rate adjustments in the various states are described in detail in the following. At the reception of a ACK the CoAP sender updates the RTT state variables and the CoAP message loss ratio, and it determines the new sending rate. Basically, there are four possible conditions:

1. $\mathbf{S} \in \mathbf{LC}$ and there is a *fast increase* of sending rate (see line 3).
2. $\mathbf{S} \in \mathbf{NO}$ and there is a *slow increase* of sending rate (see line 5).
3. $\mathbf{S} \in \mathbf{MV}$ and there is a *slow decrease* of sending rate (see line 7).
4. $\mathbf{S} \in \mathbf{HV}$ and there is a *fast increase* of sending rate (see line 9).

However, if $\mu < L_{low}$ we can safely assume that no congestion is experienced. As a consequence, the decreases of sending rates are disabled (see lines 6 and 8). On the other hand, if $\mu > L_{high}$ the network is very likely operating under congestion state, and the increase of sending rates in the *normal operating* region should be disabled (see line 4). Concerning the rate adaptation, RTT-CoAP probes for more bandwidth by performing an additive increment step. However, the bandwidth additive increment is not constant. Specifically, RTT-CoAP differentiates between two regimes: a sending rate that is smaller than one CoAP message per RTT (called *base rate*, see line 18), and a sending rate that is higher than this value. In the first case, the rate increase will be proportional to the difference between the base rate and the current rate. In the latter case, the rate increment is a fraction of the base rate. The rationale of this design choice is to avoid that CoAP sources that already transmit faster than the base rate get a disproportionate advantage in comparison with slower flows. As in TCP Vegas, we follow an additive increase/additive decrease (AIAD) policy. Thus, when

Algorithm 1 RTT-CoAP congestion control

```
1: function updateRate(S, SRTT_L, SRTT_S, RTTVAR_L, μ)

2:     if (S ∈ LC) then
3:         R = R + compRate(R, SRTT_L, fast);
4:     else if (S ∈ NO)  AND  (μ < L_high) then
5:         R = R + compRate(R, SRTT_L, slow);
6:     else if (S ∈ MV)  AND  (μ ≥ L_low) then
7:         R = R − compRate(R, SRTT_L, slow);
8:     else if (S ∈ HV)  AND  (μ ≥ L_low) then
9:         R = R − compRate(R, SRTT_L, fast);
10:    end if
11: end function

12: function compRate(R, SRTT_L, speed)

13:    if (speed = fast) then
14:        ω = 0.2
15:    else if (speed = slow) then
16:        ω = 0.05
17:    end if

18:    R_b = 1 / SRTT_L
19:    if (R < R_b) then
20:        δ = ω × (R_b − R)
21:    else
22:        δ = ω × R_b
23:    end if
       return δ
24: end function
```

network congestion is detected and the sending rate has to be decremented, a linear decrement factor δ is selected following the aforementioned rules.

3.3 RTT-CoAP Implementation

In this section we describe the key points of our implementation of RTT-CoAP leveraging Contiki, an open source operating system for the IoT. In particular, we use CoAP Erbium, a C implementation of CoAP for Contiki, that is compliant with RFC 7252 [4] and supports blockwise transfers and observing.

CoAP specification defines a number of options that can be included in a message. Specifically, CoAP uses a short fixed-length binary header (4 bytes) that may be followed by compact binary options and a payload. Each option instance in a message specifies the option number of the defined CoAP option, the length of the option value, and the option value itself. Since a length of a timestamp is typically four bytes, we follow an alternative approach for timing the CoAP messages. Specifically, the CoAP senders stores within a local table the timestamp of each transmissions of the same CoAP message, and use the retransmission counter as a one-byte timestamp option. The transmission counter is echoed in the corresponding ACK and it is used in the CoAP sender to retrieve the correct transmission time, and to estimate the RTT without ambiguity. Finally, we mark the CoAP timestamp option as: *i*) a critical option, i.e., it must be understood by the receiving endpoint in order to properly process the message; and *ii*) a safe-to-forward option, i.e., it can be safely forwarded by a proxy even if it can't process the option. The default value of this option is 0.

4 Performance Evaluation

In this section, we give the details on the simulation setup used to compare standard CoAP-CC, CoCoA+ and RTT-CoAP. Then, we presents results for a variety of different scenarios.

4.1 Simulation setup

For our performance comparison we conduct simulations in Cooja, a simulation platform included in Contiki toolset, that allows to emulate off-the-shelf wireless sensor node harwdare. At the physical (PHY) and MAC layers, the nodes implement IEEE 802.15.4, using a 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), configuring MRM parameters to achieve a 100% success rate at 10 meters and an interference range of 20 meters.

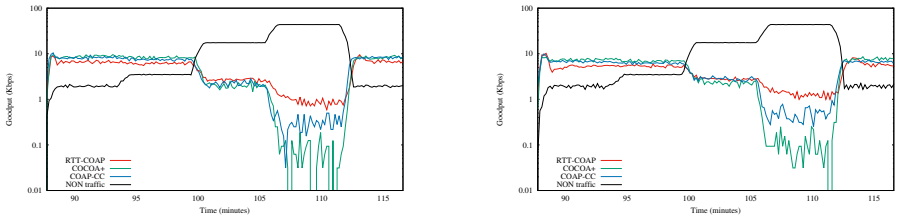
RPL is used as routing protocol [3]. We consider two large network scenarios with 57 and 112 sensor devices, respectively. The nodes are deployed along concentric circles, and the distance between consecutive circles is 10 meters. Nodes are CoAP servers that send observing notifications toward a common data collector node placed at the center of the network. We will assume that each server can generate two kinds of data flows: *i*) a data flow made of NON-messages (*background traffic*); and *ii*) a data flow made of CON-messages (*reliable traffic*). The main performance metric is the *aggregated goodput* measured as the number of successfully received acknowledgments (i.e. successfully completed transactions), and expressed in Kbit per seconds (Kbps). We also computed the flow fairness with the well-known Jain's Index applied to goodput measurements.

4.2 Mixed traffic

The first set of experiments assumes that both background traffic and reliable traffic are active on the same CoAP source. The goal of this scenario is to assess the impact of an increasing congestion-unaware traffic on the reliable flow, which is instead regulated by the congestion control mechanism. More in detail, the reliable traffic has a packet generation rate of approximately 1Kbps per CoAP source and it is maintained constant for the whole simulation. Such rate ensures that there is always a packet ready to be sent, thus, reliable traffic follows an asymptotic behaviour. Instead, the sending rate of the background traffic evolves as follows; it starts with an aggregate value of approximately 2Kbps, and it is incremented every 400 seconds until reaching a maximum of about 44 Kbps, and finally it is reverted to 2Kbps.

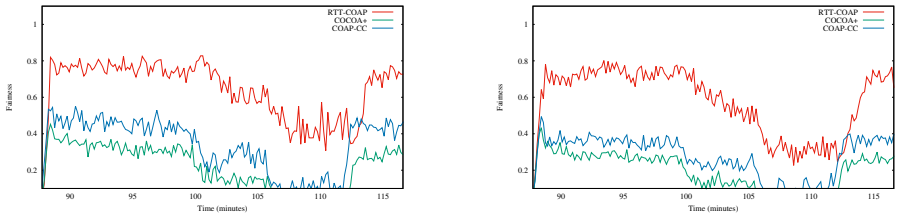
In Figure 1a and Figure 1b we show the instantaneous goodput carried by the different protocols for $n = 57$ and $n = 112$, respectively. For convenience, we report the rate of the background traffic with a solid black line.

The results reported in Figure 1a show that when the network is not congested, RTT-CoAP achieve a slightly lower aggregate goodput compared to both



(a) $n = 57$ (b) $n = 112$

Fig. 1. Goodput under mixed traffic scenario for two network deployments.

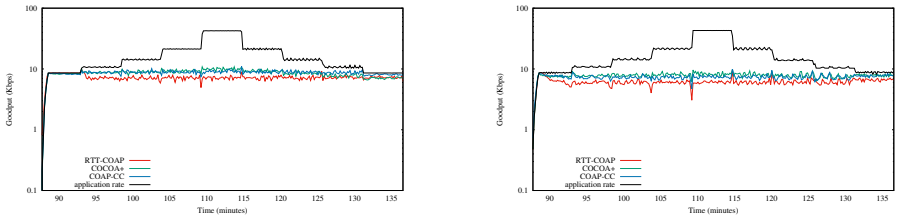


(a) $n = 57$ (b) $n = 112$

Fig. 2. Fairness under mixed traffic scenario for two network deployments.

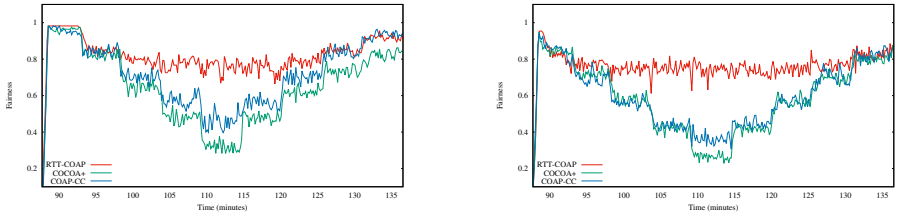
CoAP and CoCoA+ (the difference in goodput is compensated by a sensible increase of the fairness index as showed in Figure 2a). However, as the background traffic increase RTT-CoAP tends to outperform both CoAP and CoCoA+, achieving about the same aggregate goodput in the presence of medium congestion (simulation time between 100min and 105min), and obtaining a sensible goodput improvement in the presence of high congestion. We can observe that when the background traffic reaches its maximum value the CoCoA+ flows almost completely stall. The main reason is that CoCoA+ (as CoAP) indirectly regulates the sending rate using RTO, which under consecutive losses can grow up to very large values [7, 8], thus leading to nodes not sending data at all. Interestingly, CoAP is more resilient against non-confirmable traffic than CoCoA+ as RTO values are chosen in a fixed interval. On the other hand, RTT-CoAP can detect the onset of congestion earlier than CoCoA+, and it adjusts the pacing rate before overwhelming the network. Similar conclusions can be drawn from Figure 1b. However, we can see that the gap between the algorithm is reduced when the number of sources increases. This can be explained by observing that CON traffic is asymptotic. Thus, it is enough that a few CoAP sources are able to get access to the channel to be able to saturate the available bandwidth. As better explained in the following, the downside of this asymptotic behaviour is an unfair share of the network bandwidth among the existing flows.

In Figure 2a and Figure 2b we show the fairness that is measured in the same settings of the above figures. The results clearly show that our proposed congestion control solution is able to significantly outperform both standard CoAP and CoCoA+ in terms of fairness, even when the network is deeply congested.



(a) $n = 57$ (b) $n = 112$

Fig. 3. Goodput under mixed traffic scenario for two network deployments.



(a) $n = 57$ (b) $n = 112$

Fig. 4. Fairness under homogenous traffic scenario for two network deployments.

4.3 Homogenous traffic

This second set of experiments assumes that each CoAP source generates only reliable traffic. The rate at which the application send the data to the CoAP layer is incremented every 300 seconds. The goal of this scenario is to assess the efficiency of our rate-based congestion control mechanism to fully utilise the available bandwidth while maintaining a fair share of network resources between the CoAP sources. To this end, Figure 3a and Figure 3b show the instantaneous goodput carried by the different protocols for $n = 57$ and $n = 112$, respectively, while Figure 4a and Figure 4b show the fairness index in the same scenarios. The results confirm that although the aggregate goodput achieved by RTT-CoAP is slightly lower than that achieved by CoAP and CoCoA+, RTT-CoAP guarantees a significantly higher fairness among different traffic sources. In particular, we observe an increment of the fairness index of about 250% when the offered load is set to the maximum value (simulation time between 110min and 115min Figure 4b), while the goodput decrease is below 10%.

5 Conclusions

In this paper we have proposed RTT-CoAP, a novel RTT-based congestion control scheme for CoAP. Our key idea was to monitor long-term and short-term RTT trends to classify network state and to proactively anticipate network congestion. Differently from other delay-based congestion control scheme our solutions doe not require to define fixed RTT thresholds that avoid congestion, but we leverage the growth of RTT variance to directly adjust the sending rate of each CoAP source. In comparison to basic CoAP and CoCoA+, RTT-CoAP

ensures a significant improvement of fairness, and an increase of packet delivery ratios in presence of congestion-unaware 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. 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
3. T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF RFC 6550, March 2012.
4. 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
5. 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.
6. 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.
7. S. Bolettieri, C. Vallati, G. Tanganelli, and E. Mingozzi, "Highlighting Some Shortcomings of the CoCoA+ Congestion Control Algorithm," in *Proc. of IEEE ADHOC-NOW'17*. Springer, 2017, pp. 213–220.
8. E. Ancillotti and R. Bruno, "Comparison of CoAP and CoCoA+ congestion control mechanisms for different IoT application scenarios," in *Proc. of IEEE ISCC'17*. IEEE, 2017, pp. 1186–1192.
9. R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: RTT-based Congestion Control for the Datacenter," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 537–550, August 2015.
10. S. Liu, T. Başar, and R. Srikant, "TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks," *Performance Evaluation*, vol. 65, no. 6, pp. 417–440, 2008.
11. A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-Host Congestion Control for TCP," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 304–342, Third 2010.
12. L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Select. Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, October 1995.
13. V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's Retransmission Timer," IETF RFC 6298, June 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6298.txt
14. D. A. Hayes and G. Armitage, "Revisiting TCP Congestion Control Using Delay Gradients," in *Proc. of IFIP NETWORKING'11*, 2011, pp. 328–341.