GÜFBED/*GUSTIJ* (2018) CMES 2018 SEMPOZYUM EK SAYISI: 53-60
DOI: 10.17714/gumusfenbil.436056 Araştırma Makalesi / *Research Article*

Comparison of CoAP and CoCoA Congestion Control Mechanisms in Grid Network Topologies

Grid Ağ Topolojilerinde CoAP ve CoCoA Tıkanıklık Kontrol Mekanizmalarının Karşılaştırılması

Alper Kamil DEMIR*[a], Fatih ABUT[b]
Adana Science and Technology University, Faculty of Engineering, Computer Engineering Department, 01250, Adana

• Geliş tarihi / *Received*: 24.06.2018 • Düzeltilerek geliş tarihi / *Received in revised form*: 20.09.2018 • Kabul tarihi / *Accepted*: 09.10.2018

Abstract
The Internet of Things (IoT) is a vision of the future Internet. Due to limited resources of IoT devices, a new generation of protocols and algorithms are being developed and standardized. The Constrained Application Protocol (CoAP) has been designed by the Internet Engineering Task Force (IETF) for application layer communication. CoAP is based on User Datagram Protocol (UDP), a simple transport layer protocol that does not handle congestion within the network. However, the phenomenon of congestion in IoT networks is also a major problem. Thus, the core CoAP specification offers a basic CoAP congestion control (CC) mechanism based on retransmission timeout (RTO) with binary exponential backoff (BEB). Default CoAP CC is insensitive to network conditions. Thus, to improve the default CoAP CC, CoAP Simple Congestion Control/Advanced (CoCoA), defined in a draft specification, is being standardized by the IETF CoRE working group. Nevertheless, comparison of default CoAP CC and CoCoA has not been sufficiently investigated in the literature. In this paper, we investigate and present comparison of default CoAP CC and CoCoA in terms of throughput (i.e. number of requests/second) by varying number of concurrent clients where each client continuously sends back-to-back traffic to servers residing in 1x6, 3x6 and 5x6 grid network topology. Our results show that CoCoA is not always better than default CoAP CC in terms of throughput in some scenarios. As a result, design and development of new CoAP CC mechanisms are open to research.

Keywords: Internet of Things, Congestion control, CoAP, CoCoA, Cooja, ContikiOS

Öz
Nesnelerin İnterneti (IoT) geleceğin İnternet'inin bir vizyonudur. IoT cihazlarının sınırlı kaynakları nedeniyle yeni nesil protokoller ve algoritmalar geliştirilmekte ve standartlaştırılmaktadır. Kısıtlı Uygulama Protokolü (CoAP), uygulama katmanı iletişimi için Internet Mühendisliği Görev Gurubu (IETF) tarafından tasarlanmıştır. CoAP, ağ içinde tıkanıklığı karşılamayan basit bir taşıma katmanı protokolü olan Kullanıcı Datagram Protokolü (UDP) üzerine kurulmuştur. Bununla birlikte, IoT ağlarında tıkanıklık olayı da büyük bir sorundur. Bu nedenle, çekirdek CoAP spesifikasyonu, ikili üstel geri çekilme (BEB) ile yeniden iletim zaman aşımına (RTO) dayalı temel bir CoAP tıkanıklık kontrolü (CC) mekanizması sunar. Mevcut CoAP CC, ağ koşullarına duyarsızdır. Bu nedenle, mevcut CoAP CC'yi geliştirmek için IETF CoRE çalışma grubu tarafından taslak bir spesifikasyonda tanımlanan CoAP Simple Congestion Control/Advanced (CoCoA) mekanizması standartlaştırılmaktadır. Ancak mevcut CoAP CC ve CoCoA'nın karşılaştırılması literatürde yeterince araştırılmamıştır. Bu çalışmada, eşzamanlı istemcilerin sayısının değiştirilmesiyle her istemcinin sürekli olarak 1x6, 3x6 ve 5x6 grid ağ topolojilerinde yer alan sunuculara arka arkaya trafik gönderilerek mevcut CoAP CC ve CoCoA'nın performansları iş/zaman oranı (yani istek sayısı/saniye) açısından karşılaştırılmış ve sunulmuştur. Elde edilen sonuçlar, CoCoA'nın bazı senaryolarda mevcut CoAP CC'den iş/zaman oranı açısından her zaman daha iyi olmadığını göstermektedir. Sonuç olarak, yeni CoAP CC mekanizmalarının tasarımı ve geliştirilmesi araştırmaya açıktır.

Anahtar kelimeler*: Nesnelerin İnterneti, Tıkanıklık kontrolü, CoAP, CoCoA, Cooja, ContikiOS*

*a Alper Kamil DEMIR; akdemir@adanabtu.edu.tr; Tel: 0 (322) 455 0000 - 2081; [a] orcid.org/0000-0002-9256-0368
[b] orcid.org/0000-0001-5876-4116

1. Introduction

The Internet of Things (IoT) is a vision of the future Internet where the network of physical devices, embedded with electronics, software, sensors, actuators, and communication unit and protocols, enable to connect and exchange data about themselves and their surroundings within the existing Internet infrastructure. Experts forecast that the IoT will include about 30 billion physical devices and the global market value of IoT will reach $7.1 trillion by 2020 (Manjarekar et al., 2018). The IoT is bringing about new generation of applications such as smart factories, cities, homes, grids, power plants, automotive, transportation, aerospace, aviation, healthcare and agriculture (Bandyopadhyay and Sen, 2011; Li et al., 2015).

Due to constrained energy, computation, memory and communication capacities of IoT devices, a new generation of protocols and algorithms are being developed and standardized. The Constrained Application Protocol (CoAP) was constructed by the Internet Engineering Task Force (IETF) for the needs of IoT application layer communication (Shelby et al., 2014). The CoAP is a specialized web transfer protocol for use with these constrained physical devices. CoAP is based on User Datagram Protocol (UDP) to better fit the requirements of constrained physical devices. UDP is a very simple and lightweight transport layer protocol that does not handle congestion within the network. However, the phenomenon of congestion in IoT networks is also a major problem.

When the queuing and storing capacities of physical devices forming the IoT network are exceeded or generated traffic within IoT network gets close to the network capacity, network congestion is inevitably observed. Typical effect of network congestion results with queuing delay and packet loss. Congestion decreases the network utilization and can lead to congestive collapse. Congestion control and avoidance mechanisms are required to avoid congestive collapse. Thus, the core CoAP specification offers a basic CoAP congestion control (CC) mechanism based on retransmission timeout (RTO) with binary exponential backoff (BEB). Default CoAP CC is insensitive to network conditions.

Because default CoAP CC mechanism is conservative, rather than adjusting its behavior to network conditions, it may significantly underperform. Thus, CoAP specification encourages further CC mechanisms that leverage information related to current network condition. As a result, to improve the CoAP CC, CoAP Simple Congestion Control/Advanced (CoCoA) is being standardized by the IETF CoRE working group (Bormann et al., 2018). CoCoA uses round-trip time (RTT) measurements, dynamic RTO backoff computations, and RTO aging method to improve the performance of CoAP.

As far as we know, there exist some rare previous studies on comparing the performance of default CoAP CC and CoCoA (Betzler et al., 2014; Betzler et al., 2015; Ancillotti and Bruno, 2017) and comparing the performance of default CoAP CC, CoCoA and alternative CCs (Jarvinen et al., 2015; Betzler et al., 2016; Lee et al., 2016). In (Betzler et al., 2014, 2015, 2016), the authors suggested CoCoA CC mechanism for CoAP and showed that CoCoA obtains better results than default CoAP CC in the larger part of considered cases. In (Jarvinen et al., 2015), two TCP-based RTO calculation methods, namely Linux RTO and Peak-Hopper RTO, are proposed for CoAP CC. The results show that all the alternatives of default CoAP CC operate more efficient particularly at higher congestion levels. In contrast, the authors in (Betzler et al., 2016) find out that Linux RTO and Peak-Hopper RTO underperform default CoAP CC under certain conditions. Hence, they do not recommend these two RTO calculation methods as CC mechanisms for CoAP. In (Lee et al., 2016), a new RTT-based adaptive CC mechanism is proposed. The results reveal that the proposed mechanism increases the throughput of default CoAP CC. In (Ancillotti and Bruno, 2017), as far as we know for the first time in literature, the authors reveal that CoCoA performs worse than default CoAP CC under burst and light traffic loads in grid network topologies.

Although the studies in (Betzler et al., 2014, 2015, 2016) show that performance of CoCoA is better than or in worst case similar to that of default CoAP CC, in (Ancillotti and Bruno, 2017) the authors showed that CoCoA does not perform the best in heavy and light traffic loads in grid topologies. In addtion to the results obtained in (Ancillotti and Bruno, 2017), in our work, we additively reveal that CoCoA also performs worse than default CoAP under moderate traffic loads besides heavy and light traffic loads in grid network topologies. We distinctly show that CoCoA does not consistently perform the best in different MAC protocol setups in grid topologies. Particularly, as being different from the rest of

previous studies, we considered three different congestion scenarios, namely lightly, moderately and heavily congested grid networks with two different MAC approaches including nullMAC and Carrier Sense Multiple Access (CSMA).

In this paper, we investigate and present comparison of default CoAP CC and CoCoA in terms of throughput (i.e. number of requests/second) by varying number of clients where each client continuously sends back-to-back traffic to servers residing in a 1x6, 3x6 and 5x6 grid network topology operated with nullMAC or CSMA. Our results show that CoCoA is not always better than default CoAP CC in our grid topologies and MAC protocol setups. As a result, design and development of new CoAP CC mechanisms are open to research.

This paper is organized as follows. Section 2 describes the default CoAP CC and CoCoA mechanisms. Sections 3 introduces the experimental setup and methodology used to compare the performance of CoCoA with default CoAP CC. Section 4 is devoted to results and discussion. Finally, Section 5 presents our conclusion.

2. CoAP Congestion Control Mechanisms

In this section, we first describe the default CoAP CC mechanism. Then, we introduce the new mechanisms leveraged by CoCoA.

2.1. Default CoAP Congestion Control

CoAP specifies four types of messages: confirmable (CON), reset (RST), non-confirmable (NON) and acknowledgement (ACK) messages (Shelby et al., 2014). When a CON message is transmitted by a client, an ACK message is needed from the receiver. A CON message may maximally be retransmitted four times before the transmission is identified as failed. The initial RTO value is set to a random value between two and three seconds. This random initialization prevents synchronization issues. If RTO runs out and no ACK or response is obtained by the receiver, the client presumes that the CON message is lost. Hence, the client retransmits the CON message again. Consequently, for the next retransmission, the RTO value is doubled. This is known as BEB algorithm. Upon four retransmissions without any reply, the transmission is recognized as failed and the client can send a new CON request to the same receiver. Moreover, CoAP limits the number of parallel pending interactions to a single destination by NSTART parameter that is set to a conventional value of one by default. A pending interaction can be a CON or NON request that has not been replied yet.

2.2. CoAP Simple Congestion Control/Advanced (CoCoA)

Default CoAP CC is insensitive to network conditions. Hence, CoCoA leverages adaptive RTO computations, a variable backoff factor (VBF), and RTO aging. An RTO for remote receiver (RTOoverall) is computed and updated in adaptive RTO computations. The RTO is computed adaptively by administering an exponentially weighted moving average (EWMA) of RTT and RTT-variation estimates. CoCoA keeps two RTO estimators for any remote receiver, referred to as the strong RTO estimator (RTT_{strong}) and the weak RTO estimator (RTT_{weak}). The RTTstrong preserves RTT information when no retransmission is obtained. On the other hand, the RTT_{weak} preserves RTT information from retransmitted requests where the time is obtained between delivering the initial request and collecting the response. Succeeding equations represent RTTVAR and RTT where RTTVAR means the round-trip time variation, RTT means round-trip time, and X represents the strong or weak correspondingly when a new RTT (RTT_{X_new}) is measured.

$$\begin{aligned} RTTVAR_X &= (1-\beta) \times RTTVAR_X + \beta \\ &\quad \times |RTT_X - RTT_{X_{new}}| \\ RTT_X &= (1-\alpha) \times RTT_X + \alpha \times RTT_{X\_new} \end{aligned} \quad (1)$$

In (1), default values for α and β are 0.25 and 0.125, respectively. Consequently, any update of RTTX results with an update of RTOX as

$$RTO_X = RTT_X + K_X \times RTTVAR_X, \quad (2)$$

where Kstrong=4 and Kweak=1. Finally, RTOoverall, which represents the total RTO value kept up for a remote receiver, is calculated as

$$RTO_{overall} = \gamma_X \times RTO_X + (1-\gamma_X) \times RTO_{overall}, \quad (3)$$

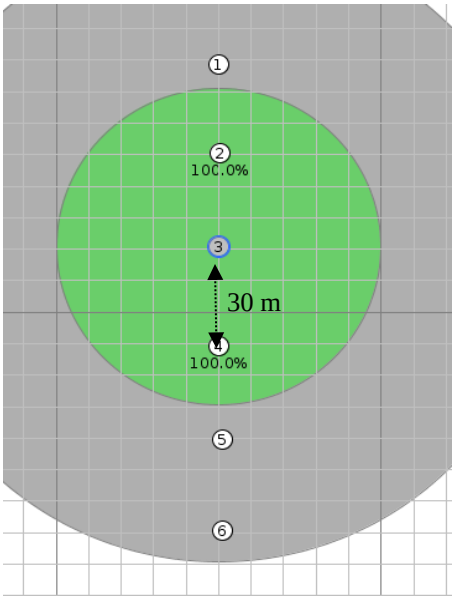
where γstrong=0.5 and γweak=0.25. RTOoveall is then utilized to decide the initial RTO (RTOinit). Contrary to the BEB used in default CoAP CC, CoCoA employs a VBF that sets the backoff value depending on the RTOinit. Additionally, to prevent the utilization of obsolete RTOoverall estimates that may have turned out to be

counterfeit after some time, CoCoA employs an aging method to the RTO estimation of remote receivers.

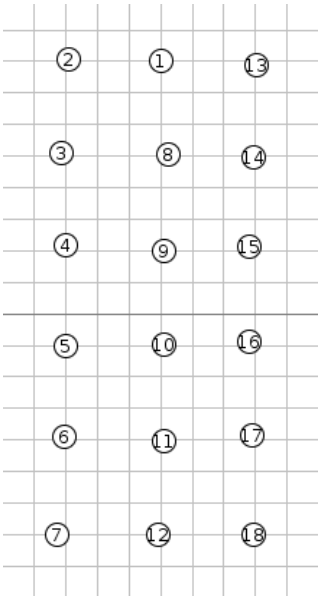
3. Experimental Setup and Methodology

For our experiments, we run varying number of clients and servers where the number of clients is always the same with the number of servers. Each client is programmed with Californium (Kovatsch et al., 2014) implementation of CoAP where default CoAP CC or CoCoA can be preferred as congestion control mechanism. On the other side, each server is programmed with Erbium implementation of CoAP in Cooja (Kovatsch et al., 2011) emulator of ContikiOS (Dunkels et al., 2004) toolset. The CoAP servers are programmed with ContikiOS that provide Erbium CoAP, UDP, uIPv6, RPL, SICSlowpan, nullMAC or CSMA based on nullRDC network stack over IEEE 802.15.4 physical layer.

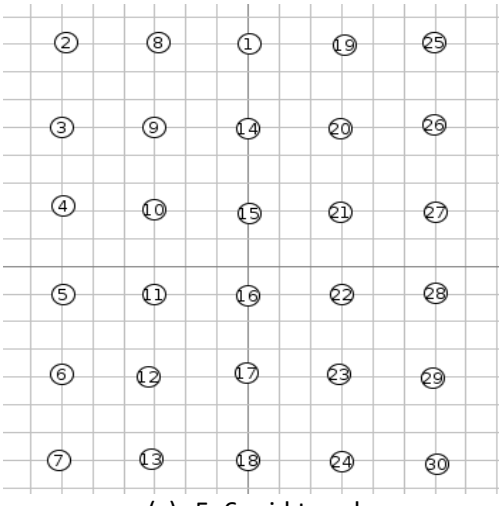
Each client continuously sends back-to-back traffic to CoAP servers residing in a 1x6, 3x6 or 5x6 grid network topology for three minutes, as illustrated in Figure 1. Moreover, each client randomly selects a server. For 1x6 grid topology, the number of concurrent clients is varied as 2, 3, 4 and 5, whereas for 3x6 grid topology the number of concurrent clients is varied as 3, 5, 13 and 17. Finally, for 5x6 grid topology the number of concurrent clients is varied as 4, 6, 21 and 29. By starting each particular client-server scenario on 24 individual PCs simultaneously, we collected and evaluated the experimental results.



(a) 1x6 grid topology



(b) 3x6 grid topology



(c) 5x6 grid topology

Figure 1. Overview of experimental setup consisting of 1x6, 3x6 or 5x6 grid topologies. The nodes are 30 m apart from each other. The nodes with ID 1 represent the border gateways.

4. Results and Discussion

Table 1 and Table 2 show the throughput results of CoCoA compared to default CoAP CC using nullMAC and CSMA for 1x6, 3x6 and 5x6 grid topologies. In 1x6 topologies, where nullMAC is used as a medium access protocol, it is observed that CoCoA consistently outperforms default CoAP CC. Contrary, in 1x6 topologies where CSMA is used instead of nullMAC, the opposite case is observed, i.e. default CoAP CC performs better than CoCoA in terms of achieved throughput. In all other cases (i.e. in 3x6 and 5x6 topologies), where nullMAC is used as a medium access protocol, default CoAP CC performs better than CoCoA. In contrast, if CSMA is used instead of nullMAC, in some cases default CoAP CC performs better than CoCoA (i.e. in experiments with case ID's 7, 8, 10, 11 and 12), and in some other cases CoCoA outperforms default CoAP CC (i.e. in experiments with case ID's 5, 6 and 9).

Table 1. Comparison of default CoAP CC and CoCoA in terms of throughput using nullMAC

Case ID	Number of clients	Topology	Default COAP CC (Requests/s)	CoCoA (Requests/s)
1	2	1x6	3.653	4.704
2	3	1x6	3.375	3.391
3	4	1x6	1.793	2.066
4	5	1x6	1.412	1.506
5	3	3x6	3.195	2.888
6	5	3x6	3.206	3.046
7	13	3x6	1.087	0.956
8	17	3x6	0.748	0.628
9	4	5x6	3.829	3.353
10	6	5x6	3.008	2.483
11	21	5x6	0.920	0.905
12	29	5x6	0.747	0.695

Table 2. Comparison of default CoAP CC and CoCoA in terms of throughput using CSMA

Case ID	Number of clients	Topology	Default COAP CC (Requests/s)	CoCoA (Requests/s)
1	2	1x6	3.166	2.612
2	3	1x6	2.046	1.682
3	4	1x6	1.613	1.374
4	5	1x6	1.379	1.052
5	3	3x6	2.639	2.670
6	5	3x6	1.635	1.766
7	13	3x6	0.850	0.828
8	17	3x6	0.623	0.565
9	4	5x6	3.250	3.403
10	6	5x6	2.733	2.493
11	21	5x6	0.896	0.780
12	29	5x6	0.626	0.600

Figure 2 through Figure 4 illustrate the percentage increase or decrease rates in throughput of CoCoA compared to default CoAP CC using nullMAC and CSMA for 1x6, 3x6 and 5x6 grid topologies, respectively.

As illustrated in Figure 2, if CoCoA and default CoAP CC run with nullMAC in 1x6 topology, CoCoA outperforms default CoAP CC relatively improving the throughput ranging between 0.47% and 22.34%. In contrast, if CoCoA and default CoAP CC run with CSMA in 1x6 topology, CoCoA underperforms default CoAP CC

relatively worsening the throughput ranging between 17.39% and 31.08%.

According to Figure 3, if CoCoA and default CoAP CC run with nullMAC in 3x6 topology, CoCoA underperforms default CoAP CC relatively worsening the throughput ranging between 5.25% and 19.11%. Differently from the previous case, when CoCoA and default CoAP CC run with CSMA in the same topology, CoCoA outperforms default CoAP CC when the number of concurrent clients is 3 or 5 improving the throughput to 1.16% and 7.42%, respectively. When the number of concurrent clients is 13 or 17, CoCoA underperforms default CoAP CC relatively worsening the throughput to 2.66% and 10.27%, respectively.

Finally, in Figure 4, it is seen that independent of whether CoCoA and default CoAP CC run with nullMAC or CSMA (except the case where CoCoA and default CoAP CC run with CSMA and the number of concurrent clients is 4) in 5x6 topology, CoCoA consistently underperforms default CoAP CC relatively worsening the throughput ranging between 1.66% and 21.14%. In case where CoCoA and default CoAP CC run with CSMA and the number of concurrent client is 4, CoCoA outperforms default CoAP CC relatively improving the throughput to 4.50%.

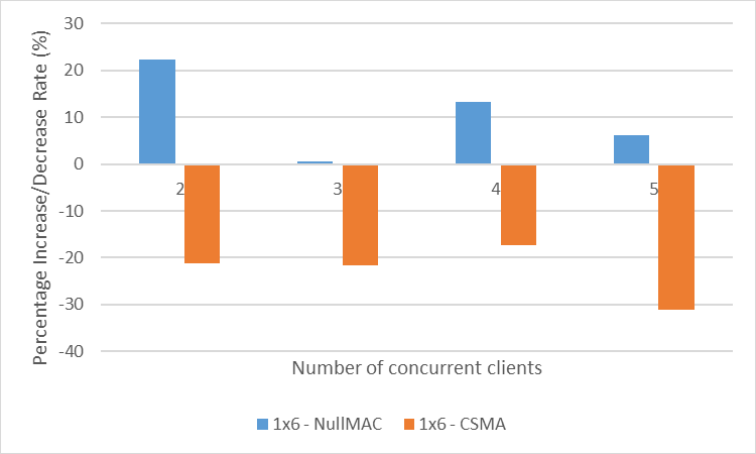


Figure 2. Percentage increase/decrease rates in throughput of CoCoA compared to default CoAP CC using nullMAC and CSMA for 1x6 topology

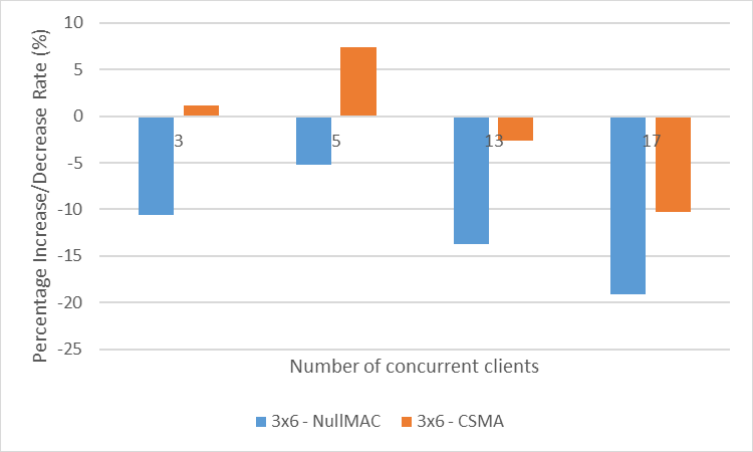


Figure 3. Percentage increase/decrease rates in throughput of CoCoA compared to default CoAP CC using nullMAC and CSMA for 3x6 topology

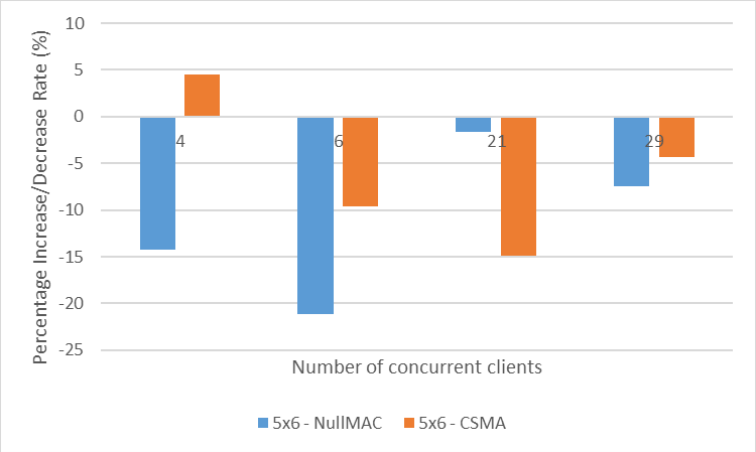


Figure 4. Percentage increase/decrease rates in throughput of CoCoA compared to default CoAP CC using nullMAC and CSMA for 5x6 topology

With these results, we demonstrate that MAC-level protocol setups influence the performance of the used CC mechanism. For example, in 1x6 topology, when nullMAC is used, CoCoA performs better than default CoAP CC in almost all traffic loads. On the other hand, in 1x6 topology, when CSMA is used, default CoAP CC performs better than CoCoA. Moreover, we investigated the performance of default CoAP CC and CoCoA in varying traffic loads, namely in lightly, moderately and heavily congested 1x6, 3x6 and 5x6 grid network topologies. Unlike other studies in the literature where CoCoA was shown to be superior to default CoAP CC under only heavy and light traffic loads, we observe and highlight that default CoAP CC outperforms CoCoA in almost all light, moderate and heavy traffic loads when CSMA, the most commonly used MAC protocol, is employed in considered grid networks.

The analysis of these results indicates that both CC mechanisms have their own shortcomings, and there is no generalizable superiority of one CC mechanism over the other that consistently performs well over all topologies, MAC protocols and varying traffic loads. Particularly, CoAP has the demerit of having a conservative retransmission timer which can cause long idle times before the lost packet is retransmitted during timeout period. On the other side, CoCoA can lead to unnecessary retransmissions, causing more congestion and resulting with reduced throughput. To overcome the shortcomings of CC mechanisms, one possible way would be to design enhanced CC mechanisms which may accurately adapt network conditions.

5. Conclusion

In this paper, we investigated and presented comparison of default CoAP CC and CoCoA in terms of throughput (i.e. number of requests/second) by varying number of concurrent clients where each client continuously sends back-to-back traffic to servers residing in 1x6, 3x6 and 5x6 grid network topologies. At the client-side, we configured 24 PCs to execute concurrent clients that use Californium implementation of default CoAP CC and CoCoA. At the server-side, we also used the same PCs to run varying number of servers that use Erbium implementation of CoAP in Cooja simulator of ContikiOS. Our results show that CoCoA is not always better than default CoAP CC in terms of throughput in some scenarios. As a result, design and development of new CoAP CC mechanisms are open to research.

References

Ancillotti, E. and Bruno, R., 2017. Comparison of CoAP and CoCoA+ congestion control mechanisms for different IoT application scenarios. IEEE Symposium on Computers and Communications, Crete, Greece, pp. 1186–1192.

Bandyopadhyay, D. and Sen, J., 2011. Internet of Things: Applications and Challenges in Technology and Standardization. Wireless Personal Communications, 58(1), 49–69.

Betzler, A., Gomez, C., Demirkol, I. and Kovatsch, M., 2014. Congestion control for CoAP cloud services. IEEE Emerging Technology and Factory Automation, Barcelona, Spain, 1–6.

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

Betzler, A., Gomez, C., Demirkol, I. and Paradells, J., 2016. CoAP congestion control for the internet of things. IEEE Communications Magazine, 54(7), 154–160.

Bormann, C., Betzler, A., Gomez, C. and Demirkol, I., 2018. CoAP Simple Congestion Control/Advanced, https://tools.ietf.org/html/draft-ietf-core-cocoa-03.

Dunkels, A., Gronvall, B. and Voigt, T., 2004. Contiki - a lightweight and flexible operating system for tiny networked sensors. 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, pp. 455–462.

Jarvinen, I., Daniel, L. and Kojo, M., 2015. Experimental evaluation of alternative congestion control algorithms for Constrained Application Protocol (CoAP). 2nd IEEE World Forum on Internet of Things, Milan, Italy, pp. 453–458.

Kovatsch, M., Duquennoy, S. and Dunkels, A., 2011. A Low-Power CoAP for Contiki. 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, Valencia, Spain, pp. 855–860.

Kovatsch, M., Lanter, M. and Shelby, Z., 2014. Californium: Scalable cloud services for the Internet of Things with CoAP. International Conference on the Internet of Things, Seoul, Korea, pp. 1–6.

Lee, J. J., Kim, K. T. and Youn, H. Y., 2016. Enhancement of congestion control of Constrained Application Protocol/Congestion Control/Advanced for Internet of Things environment. International Journal of Distributed Sensor Networks, 12(11) 1-13.

Li, S., Xu, L. Da and Zhao, S., 2015. The internet of things: a survey. Information Systems Frontiers, 17(2), 243–259.

Manjarekar, S., Rathod, S., Siddhiqi, R., Pathan, I. and Kale, M., 2018. IoT Based Home Security. International Journal of Advanced Research in Computer and Communication Engineering, 7(5), 47–50.

Shelby, Z., Hartke, K. and Bormann, C., 2014. Constrained Application Protocol. RFC 7252, https://tools.ietf.org/html/rfc7252.