Received: 21 October 2023 | Revised: 7 November 2023 | Accepted: 9 November 2023

IET Wireless Sensor Systems

DOI: 10.1049/wss2.12072

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Congestion control in constrained Internet of Things networks

Lotfi Mhamdi | **Hussam Abdul Khalek**

School of Electronic and Electrical Engineering, University of Leeds, Leeds, West Yorkshire, UK

Correspondence
Lotfi Mhamdi.
Email: l.mhamdi@leeds.ac.uk

Abstract

The Internet of Things (IoT) is a growing technology that remotely connects multiple devices (ranging across many fields and applications) over the Internet. The scalability of an IoT network mandates a reliable transport infrastructure. Traditional transport control protocol (TCP) control protocol is unsuitable for such domain, mainly due to energy and power consumption reasons. A lighter version of TCP, light weight IP (lwIP) provides a promising solution for current and projected future scalable IoT infrastructures. However, the original lwIP is just a simple mapping of the protocol, without insight into the IoT specific requirements. This paper examines the lwIP congestion control mechanism and addresses its shortcomings. In particular, a detailed examination is devoted to the various metrics such as retransmission time-outs and its back-off epochs, the congestion window behaviour and progress in the absence (and presence) of congestion. In particular, we propose a set of novel algorithms to address both the IoT constraints nature (light-weight) as well as keeping up with scalability in IoT network size and performance. A detailed simulation study has been conducted to endorse the viability of our proposed set of algorithms for next-generation IoT networks.

KEYWORDS

data communication, Internet of Things, performance evaluation, protocols

1 | INTRODUCTION

The Internet of Things (IoT) describes the wireless network of embedded devices and systems that connect and exchange data with other devices and systems over the Internet [1]. IoT has become an integral part of the current technological advancement and is expected to substantially grow in the coming future. IoT technologies are included in multiple applications such as smart houses, smart cars, smart factories, smart security systems, smart and remote healthcare applications, cloud technologies and many other platforms spanning over multiple industries [1]. Currently, there are at least 10 billion active IoT devices, and it is estimated that by 2030 the number of active IoT devices will rise to around 25.4 billion devices [2].

The Internet Engineering Task Force (IETF) classified IoT devices into different classes, mainly depending on their random access memory and Flash memory capacity [3]. The class resources defined by the IETF have limited capacity, making them inadequate for handling moderate to high levels of traffic. As a result, transmitting data from Internet-constrained IoT devices effectively over the Internet using legacy network architecture, protocols, and communication technologies is challenging for such resource-limited devices. The research as well as industrial communities have identified various issues in IoT networks, including heterogeneity, security, congestion, energy efficiency, mobility, reliability, and quality of service, among others [4]. Congestion control has been identified as one of challenges that needs to be addressed in order to keep pace with the steep growth in IoT networks [5, 6]. Figure 1 illustrates a typical example of an IoT network. A growing IoT device count would cause congestion. This problem can even escalate in the presence of concurrent connections [8], resulting in Transport Control Protocol (TCP) incast issues [9].

Like all traditional wired and wireless communication networks, IoT-based networks also suffer from network congestion which occurs when the network has more data

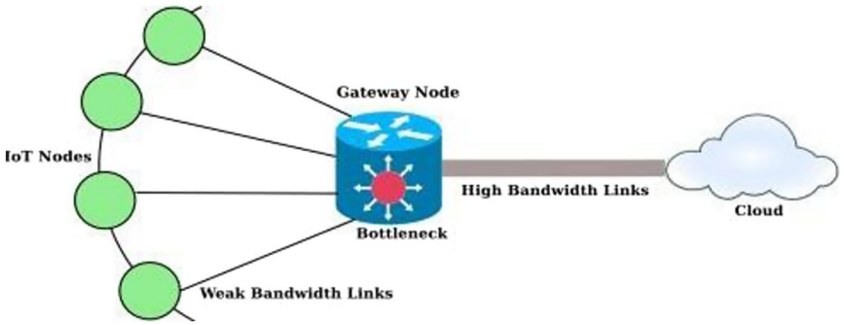

FIGURE 1 IoT network bottleneck [7]. IoT, Internet of Things.

traffic than it can handle due to the excessive number of devices exchanging data on the network [10, 11]. Unlike User Datagram Protocol (UDP), where middle-boxes such as firewalls and Network Address Translation devices might effectively block UDP packets [12], the TCP protocol in IoT yields a seamless integration with existing networks. Furthermore, the latest industry standardisation with regards to the Constrained Application Protocol (CoAP) [13] and its corresponding advanced congestion control algorithm (CoCoA) [14] suggests that TCP will be gaining considerable support in IoT scenarios in the future.

The main obstacle with TCP in IoT-based networks lies in the header over-head, stack size and memory usage [15]. Moreover, the conventional CoCoA poses the issue of multiple end-to-end retransmissions of lost segments over lossy networks with high bit-error rate causing elevated power usage in such cases [15]. To overcome such obstacles, multiple solutions were considered, such as utilising the 6LoWPAN (IPv6 low-power wireless personal area network) [16] adaptation layer over IPv6 to achieve packet compression suitable for IoT applications and embedded systems [17–19]. Furthermore, light-weight TCP implementations such as micro-IP TCP (μIP) [20] and light-weight IP TCP (lwIP), provide a much smaller stack size with minimal memory usage making them suitable for IoT scenarios. The light-weight implementations can be made even more suitable for IoT scenarios by the use of radio duty cycle (RDC) mechanisms [21] at the Medium Access Control layer to further minimise the power usage [15, 22]. To date, although multiple aspects of light-weight TCP implementations have been investigated for IoT scenarios, congestion control remains a challenge that is rarely investigated for constrained IoT networks.

The main objective of this paper is to explore TCP-based implementations for constrained IoT networks. The focus will be on enhancing the performance of the CoCoA for light-weight TCP, particularly the open source lwIP implementation [20, 23]. More specifically, the aim is to increase the total network goodput as well as reducing the network congestion and consequently minimising the number of end-to-end retransmissions since they are a primary factor in energy consumption. This can be achieved by tuning congestion control parameters and setting the right algorithmic adjustments. For this, a set of novel techniques has been proposed. In particular, we propose a novel algorithm named *lwIP Back off* where proper adjustments to the round-trip time (RTT) and retransmission time-out (RTO) estimation mechanism and RTO back-off were carefully chosen. A further improvement, named *lwIP cwnd* was to adopt a more accurate congestion window behaviour than it is originally set in lwIP. Finally, we combined both techniques into a one scheme that we named *lwIP Back off & cwnd* that has been proven through extensive simulation to exhibit good performance and outperform existing algorithms.

The remainder of the paper is structured as follows. Section 2 discusses some relevant existing related work on light-weight congestion control algorithms. In Section 3, analyses existing lwIP implementations. Section 4 gives details of the newly proposed set of algorithms that adaptively tune relevant congestion metrics to improve the IoT network utilisation. We describe three different algorithms in Section 4.1, Section 4.2 and Section 4.3 respectively. Section 5 presents the simulation settings and discuss the experimental results. Finally, Section 6 concludes the paper.

2 | RELATED WORK

The congestion issue is expected to escalate in the coming years with the continuous steep increase in the number of IoT devices and the continuous growth in the size of IoT networks, unless such obstacle is mitigated. Currently, the debate stands at the choice of protocols for the network stack layers of IoT devices [7]. In particular, the protocols for the transport layer and the application layer require careful consideration as they would highly affect the behaviour and performance of the IoT network being implemented [10, 24]. Various combinations of protocols can be used depending on the type of IoT device and its corresponding application. Considering that, the UDP stands as the most popular transport protocol and as an underlying communication protocol alongside a significant amount of the application protocols being used in various IoT scenarios [15]. For instance, the Message Queueing Telemetry Transport messaging protocol [25] is a popular light-weight messaging application used in IoT monitoring applications, and it utilises UDP as a main underlying communication protocol. This comes down to the fact that UDP is connectionless, light-weight, fast, efficient and simple to utilise for IoT scenarios. However, it lacks connection reliability, data

sequencing and acknowledgement. Meanwhile, the TCP is the de facto transport layer protocol for traditional networks. Unlike UDP, it is connection oriented, it provides maximum reliability, data sequencing and acknowledgement, retransmission of lost packets, extensive error checking and guaranteed delivery. It is also significantly dominant over the current network infrastructure [15].

A study by Lim [15] investigated a way of improving congestion control of the light-weight μIP TCP stack for constrained IoT networks [20]. The study proposes a scheme involving parameter tuning as well as algorithmic and system-level adjustments. Lim's approach involved investigating the performance of μIP TCP in a grid topology network with RDC via the Cooja network simulator in Contiki OS [26].

It was established that RDC is a significant tool for saving battery power in wireless sensor networks. However, RDC causes a lot of retransmissions when using light-weight μIP TCP due to the fixed RTO and the large RTT variations caused by the hidden node problem. The aim was to investigate the effect of RTT and RTO estimation on the performance of μIP TCP in constrained IoT networks, to examine the possibility of implementing variable RTO back-off and weak/strong RTT estimation inspired by CoCoA in CoAP and to propose additional mechanisms to further improve μIP TCP performance in constrained IoT networks with RDC enabled.

To evaluate the performance, Lim [15] proposed RPL-based 4 × 4 and 5 × 5 grid topology networks with a Linux RPL (Routing protocol for Low-Power and Lossy Networks) [27] border router at the edge of the grid, maintaining a slip connection to a Linux TCP server. Each client maintains a custom μIP TCP implementation with an RDC option. The data exchange included 48 bytes of TCP payload at 64 bits/s over a period of 10 min. As for the proposed scheme, weak RTT estimation was utilised alongside exponential back-offs with variable limits as well as dithering. The latter is implemented by setting the actual retransmission timer and adding a random duration to it. The results depicted a relatively improved performance when considering their proposed solutions especially on RDC-enabled scenarios. Their proposed scheme provided an increased number of segments across both 4 × 4 and 5 × 5 grid networks. However, the original implementation with the addition of dithering only provided a better performance with regards to the network goodput and fairness index.

Although μIP TCP can provide a good experimental starting point, it does not include the relevant TCP features to properly implement congestion control for IoT-based networks. For example, μIP TCP does not support a sliding window and maintains only a single MSS (maximum segment size) in its window size. It also does not support neither slow start, nor fast recovery/retransmit [22, 28]. Furthermore, utilising a grid topology network does not isolate the performance measurement to congestion control only, due to the possibility of the hidden node problem as well as multi-hops and consequently packet loss.

In this paper, we propose a more suitable approach and testing scheme to evaluate the congestion control performance of a light-weight TCP implementation in IoT native scenarios.

3 | OVERVIEW OF LIGHT-WEIGHT TCP IMPLEMENTATIONS

Exploring opportunities to make the congestion control mechanism more suitable for IoT scenarios, is central for devising an optimal approach to utilise light-weight TCP adaptations for constrained IoT networks and applications. Most TCP implementations follow the three main congestion control algorithms. Slow start, congestion avoidance as well as fast retransmit/recovery are utilised in that specific order to manage the number of outstanding data being sent over the network. For the original full TCP implementation, if the congestion window, *cwnd*, is less than *ssthresh*, slow start is used, while if *cwnd* is greater than *ssthresh*, congestion avoidance is used. If *cwnd* is equal to *ssthresh*, either of the algorithms can be used.

Initially, slow start algorithm is used to slowly probe the network; however, it is also used after mitigating loss detected by the retransmission timer. The initial *cwnd* size during slow start should adhere to the following guidelines [20]:

$$If\ MSS > 2190\ bytes,\ cwnd = 2 \times MSS$$

$$If\ MSS > 1095\ bytes,\ cwnd = 3 \times MSS$$

$$If\ MSS \leq 1095\ bytes,\ cwnd = 4 \times MSS$$

During the slow start phase, the *cwnd* is incremented by a maximum of one MSS for every valid ACK received, such that: $cwnd += \min(N, MSS)$ where, N is the number of previously unacknowledged bytes that are acknowledged in the incoming ACK. As for congestion avoidance, the *cwnd* is incremented by 1 full-sized segment per RTT until congestion is detected [20].

Light-weight TCP implementations, such as lwIP, follow a similar but more aggressive approach to adhere to IoT scenarios. Primarily, the initial *ssthresh* starts at a lower point at a maximum of $10 \times MSS$. Initially, and upon congestion or upon reaching the *ssthresh*, the *cwnd* is set to always start or restart from $1 \times MSS$. This forces a more conservative approach to try to minimise the points of congestion and ultimately retransmissions since the *cwnd* will take longer to reach the *ssthresh* limit. Meanwhile, important aspects of successful TCP congestion control are the RTT and RTO estimation mechanisms, which ultimately affect the algorithm's reactive response to retransmissions as well as the proactive response to prevent future congestion and retransmissions [20].

The TCP CoCoA constantly checks whether a packet is received and whether the periodic timer expires. The timer determines when a retransmission should happen whenever it reaches zero and prompts an interrupt. It is also utilised in

measuring RTT samples by sampling its value whenever an acknowledgement (ACK) is received. Since the Karn algorithm is used for implementations like lwIP and μIP, RTT is not sampled for retransmitted segments. Meanwhile, the RTT estimation is utilised in the Van Jacobson fast algorithm in order to estimate an RTO value.

The RTO estimation algorithm goes as follows [15]: where *SRTT* represents the estimated RTT, *Err* represents the error between the measured RTT and the estimated RTT and *MDEV* represents the mean deviation [20]. The RTO tends to follow an exponential back-off procedure depending on the number of retransmissions [20].

Algorithm 1. RTO estimation algorithm.

$$Err \leftarrow RTT - SRTT$$
$$SRTT \leftarrow SRTT + 0.125 \times Err$$
$$MDEV \leftarrow MDEV + 0.25 \times (|Err| - MDEV)$$
$$RTO \leftarrow SRTT + 4 \times MDEV$$

Table 1 shows the features of different accessible light-weight TCP implementations that are currently being used in embedded systems and IoT applications are reported in Ref. [22]. Most importantly, it shows that μIP TCP does not implement a sliding window in the sense that it always uses one MSS for its window size. In other words, it can only have one unacknowledged TCP segment per connection. This causes a poor interaction between the sender and a receiver that is using a delayed acknowledgement mechanism. Thereby, resulting in long waiting times at the receiver and thus hindering the sender throughput [15].

In this paper, we seek to adopt a light-weight TCP implementation with a relatively small code size alongside adequate congestion control and other TCP features. The purpose is to avoid being restricted by a single MSS on the window size, and to be able to have at least the slow start, congestion avoidance and fast retransmit/recovery congestion control algorithms. These mentioned features are found in the open source lwIP stack, as shown in Table 1. Therefore, lwIP provides an ideal platform to analyse the congestion control performance in constrained IoT networks, in addition to the fact that it has an optimised memory usage and a compressed code size. Moreover, lwIP is currently being utilised by various leading manufacturers of embedded systems such as Intel, Analog Devices, Xilinx and many others [15, 20].

4 | THE PROPOSED SCHEME

To improve the overall performance of lwIP congestion control mechanism and ultimately increase the total goodput as well as decreasing the total number of retransmissions and congestion in general, multiple parameters and features were adjusted within the CoCoA. Below, we describe a set of novel algorithms, each of which aims at addressing one congestion control feature.

4.1 | lwIP with dynamic RTO back-off

Firstly, the RTO back-off procedure was investigated. By default, upon a retransmission, the RTO is configured to back-off from its estimated initial value by a doubling factor. This ultimately results in an exponential increase in the RTO value. To avoid having excessively large RTOs and inspired by CoCoA of CoAP, a dynamic back-off factor is investigated. The process involved multiple trial and error iterations by adjusting the back-off factor with respect to a specific RTO value until an improved and optimised result was observed. These settings, although quite specific to our settings, are likely to be

TABLE 1 Comparison between different light-weight TCP implementations with their corresponding TCP features [22].

Stack properties		Light-weight TCP implementations			
		Traditional TCP	μIP	lwIP original	RIOT
Code size (kB)		N/A	<5	~9 to ~14	<7
TCP features	Window size (MSS)	Multiple	1	Multiple	1
	Slow start	YES	NO	YES	NO
	Fast retransmit/fast recovery	YES	NO	YES	NO
	Keep-alive	YES	NO	NO	NO
	Window scale	YES	NO	NO	NO
	TCP timestamp	YES	NO	NO	NO
	SACK	YES	NO	NO	NO
	Delayed ACK	YES	NO	YES	NO
	Socket	YES	NO	NO	Optional
	Concurrent connections	YES	YES	YES	YES

Abbreviations: lwIP, light weight IP; MSS, maximum segment size; TCP, transport control protocol.

adequate for general scenarios. In particular, the RTO back-off feature of the lwIP algorithm was adjusted as follows:

$$RTO\_back\_off\_factor = \begin{cases} 2 & when\ RTO \leq 3s \\ 1.3 & when\ RTO > 3s \end{cases}$$

This adjustment helps in avoiding extremely high RTOs resulting in long waiting times when a packet is lost and requires retransmission. This lwIP adaptation is referred to as lwIP back-off in the results of this study.

4.2 | lwIP with congestion window adjustment

This section explores the *cwnd* floor value and its impact on the congestion. Originally, upon a retransmission or upon reaching the *ssthresh*, the *cwnd* is configured to default back to $1 \times MSS$. Such aggressive configuration causes extreme variation in the congestion window value. For instance, the *cwnd* can drop from $10 \times MSS$ down to $1 \times MSS$ with such implementation. This can ultimately impact the total goodput over an extended period. Therefore, we propose to set the *cwnd* to 50% of its *ssthresh* limit instead of resetting it back to $1 \times MSS$ upon a retransmission or upon reaching the ssthresh. This optimal percentage value of 50% was deduced after multiple iterations of trial error involving different configurations of *cwnd*. Besides, this same *cwnd multiplicative* decrease (by 0.5) is similar to that of traditional TCP. This lwIP adaptation is referred to as *lwIP cwnd* in the results of this study.

4.3 | lwIP with back off & cwnd adjustment

Lastly, both dynamic RTO back-off and *cwnd* floor value adjustments were combined in the same implementation with several attempts to optimise the overall performance by tuning both the back-off factor and the *cwnd* configuration. The optimisation process resulted in maintaining the back-off factor and setting the *cwnd* to 50% of its *ssthresh* limit. This lwIP adaptation is referred to as lwIP back-off & cwnd in the results of this study.

It is worth noting that the proposed set of algorithms is an IoT transport layer protocol, not an application layer protocol (such as AMQP [29] and QUIC [30]) or network layer protocol (such 6LoWPAN [16] and RPL [27]). Contrary to common wisdom that TCP based implementations are not suited for IoT scenarios, there is a growing evidence that contradicts this [31]. Our proposal is a step further, towards confirming this evidence. As has been shown in Table 1, our proposed set of algorithms inherits the same communication overhead as that of lwIP, in terms of maintaining multiple (few) MSS as window size, keeping a slow start phase as well as a congestion avoidance and fast retransmit/recovery phases.

5 | EXPERIMENTAL SETTINGS AND RESULTS

This section starts by first describing the simulation settings and then we proceed to the experimental results.

5.1 | Experimental settings

We begin by describing the network topology and associated simulation parameters. We have used the OMNET++ simulator [32] and created a star topology network, as depicted in Figure 2. The network includes 12 clients wirelessly connected to an access point at 2.4 GHz and adhering to the IEEE802.11 Wi-Fi standard. Meanwhile, the access point is connected to a TCP server via an Ethernet connection. The network follows a star topology with all 12 clients evenly spaced and at the same approximate distance from the access point. The star topology was deemed more suitable since it provides a fairer state of communication across all clients and eliminates the hidden node problem. This implementation would be more likely to accentuate the performance evaluation on the congestion control.

The client access-point wireless links run at 2 Mbps, while the access point-server Ethernet link runs at 10 Mbps. After establishing a TCP connection with the server, each client sends a 1024 byte packet to the server, through the access point, every 0.4 s for a total of 1000 packets over a period of 400 s. The server acts as a sink by only responding with an ACK. Table 2 shows a compiled summary of the relevant network and simulation settings. The network simulation is configured to either run with original TCP clients as reference, or lwIP clients. The recorded parameters include RTT, RTO, *cwnd*, *ssthresh*, total goodput and the total number of

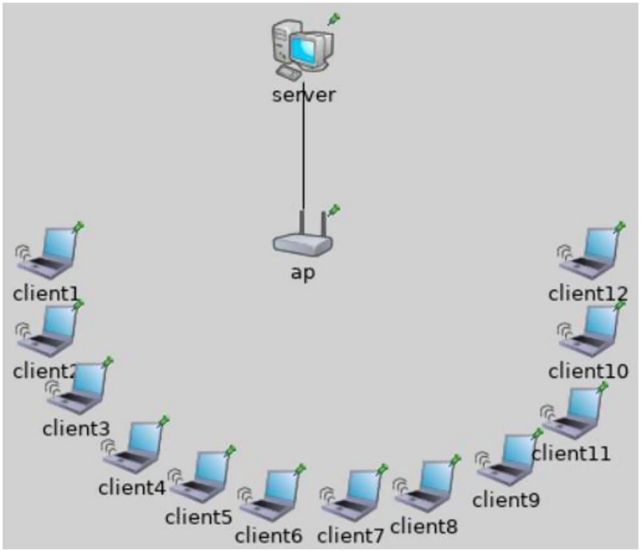

FIGURE 2 Overview of the network used in simulation.

retransmissions. The RTT, RTO, *cwnd*, *ssthresh* and the total number of retransmissions are recorded in accordance with a single client, while the total goodput represents the network's aggregated total bits/seconds received at the input of the access point.

5.2 | Experimental results

Initially, we looked at the comparison between the original full TCP implementation and the original lwIP implementation as reference. Figure 3 demonstrates the *cwnd* changes of both TCP and lwIP over a span of 400 s, averaged overall clients. It can be observed that during the first 50 s the TCP *cwnd* climbs to a much higher *ssthresh* than lwIP's *cwnd*. This is because in TCP the *ssthresh* is initially set to 65,535 bytes, while in lwIP, it is set to 5360 bytes (10 × MSS). Apart from that, the *cwnd* progression over the remaining duration is very similar for both TCP and lwIP since they both have a similar CoCoA. The second experimental result we evaluated is related to RTT effect and that of the RTO. Figure 4 shows the RTT values of a single client and the access point over a period of 400 s. Then, we compared the RTO values of the original lwIP scheme as compared to our proposed lwIP_back_off scheme, as described in Section 4.1. As depicted in Figure 5, a smaller RTO estimation is observed throughout the entire simulation with a maximum of 7 s for lwIP with dynamic RTO back-off and 24 s for the original lwIP. This is attributed to the back off adjustment that dynamically sets the RTO according to its latency. The RTO of values for original lwIP and original TCP are depicted in Figure 6.

We further observed the congestion window behaviour over time of both the original lwIP algorithm and our proposed lwIP_back_off. Figure 7 shows that the lwIP

TABLE 2 Simulation settings.

Network parameters	Values
Topology	Star
Number of packets in MAC/link layer queue	8 entries
Packet size	1024 bytes
Number of packets	1000 packets
Number of clients	12 clients
Frequency	2.4 GHz
Client-access point data rate	2 Mbps
Access point-server data rate	10 Mbps
Server stack	TCP

Abbreviation: TCP, transport control protocol.

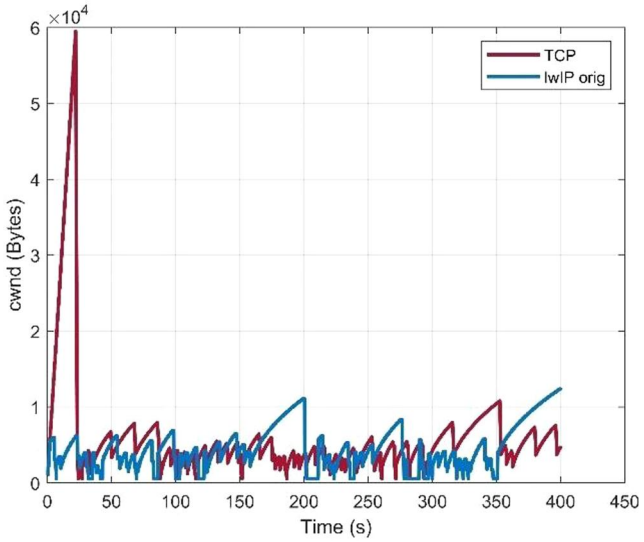

FIGURE 3 Progression of *cwnd* for original lwIP and original TCP implementations. lwIP, light weight IP; TCP, transport control protocol.

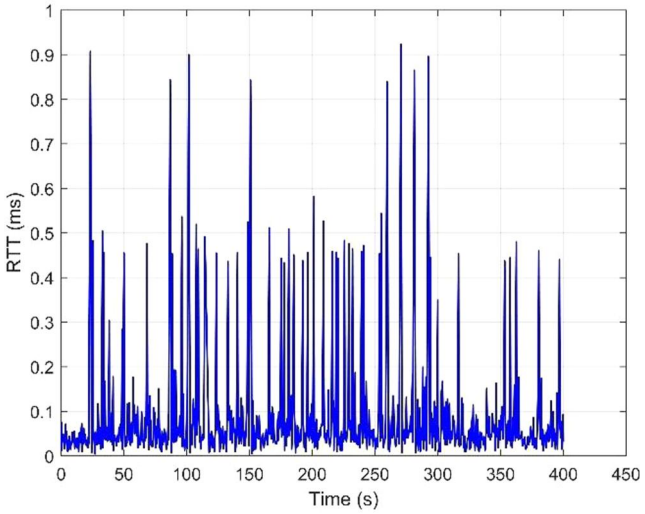

FIGURE 4 RTT estimation between a single client and the access point over 400 s. RTT, round-trip time.

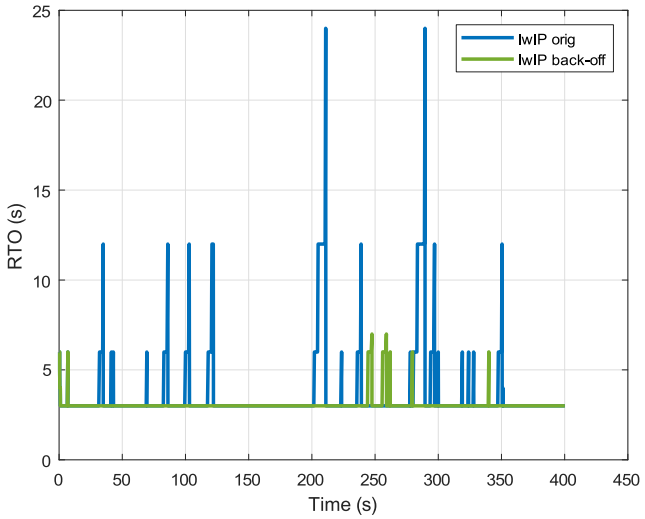

FIGURE 5 Progression of RTO values of lwIP and lwIP_back_off algorithms. lwIP, light weight IP; RTO, retransmission time-out.

implementation with dynamic RTO back-off results in a *cwnd* with less congestion points as compared to the original lwIP implementation. In other words, the *cwnd* tends to climb to higher values with less congestion points represented by retransmissions or reaching the *ssthresh*. Such response depicts less overall congestion within the network and therefore results in higher network utilisation.

A further technique has been to adaptively update the congestion window, as presented in Section 4.2 with the proposed lwIP_cnwd algorithm. We have compared the congestion window evolution over time of both the original lwIP algorithm and that of lwIP_cnwd. As can be seen in Figure 8, the lwIP implementation with the *cwnd* adjustment results in a better *cwnd* behaviour in the sense that *cwnd* drops to a higher floor value upon a retransmission or when the *ssthresh* is reached in comparison to the original lwIP implementation that defaults to one MSS. Moreover, *cwnd* tends to climb to higher values with less congestion points represented. We further observed the congestion window growth of the original and the combined algorithms, as presented Section 4.3. A similar trend is observed in Figure 9, where the network can sustain higher traffic and is able to absorb transient congestion periods resulting in higher network utilisation. Figure 10 summarises the congestion window behaviour of all algorithms and shows the merits of each of them, as discussed above.

Figure 11 shows that the total network goodput of all algorithms. As can be seen from the Figure, the goodput was certainly improved from 22 kbps for original lwIP implementation to 24 kbps when using either of lwIP with dynamic RTO back-off adjustment and/or lwIP back-off & cwnd floor

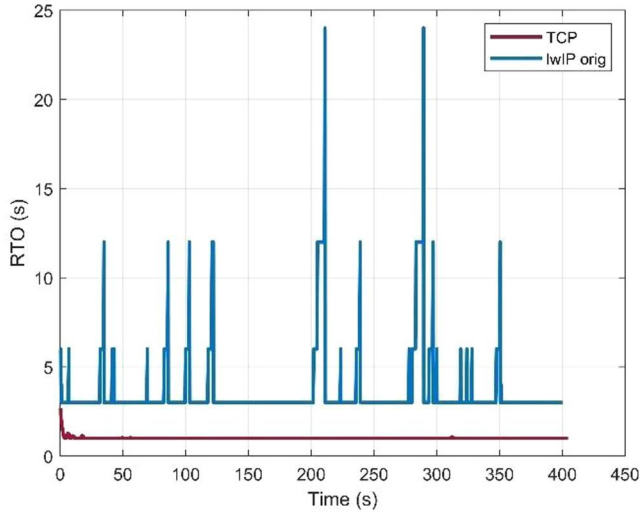

FIGURE 6 Progression of RTO values for original lwIP and original TCP implementations. lwIP, light weight IP; RTO, retransmission time-out; TCP, transport control protocol.

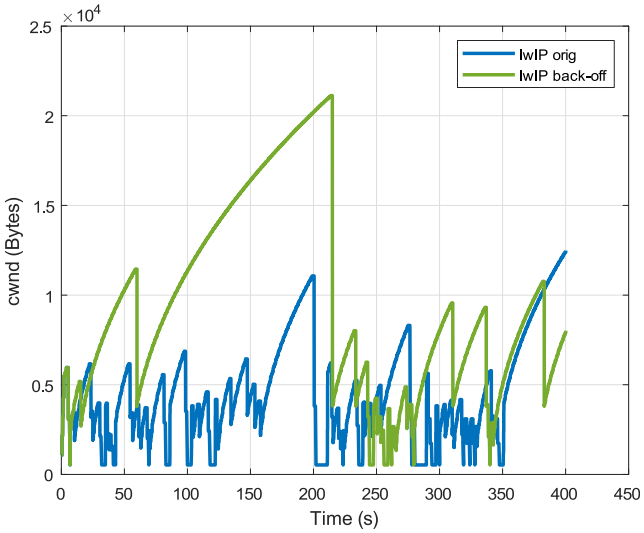

FIGURE 7 Congestion window progress of lwIP and lwIP_back_off algorithms. lwIP, light weight IP.

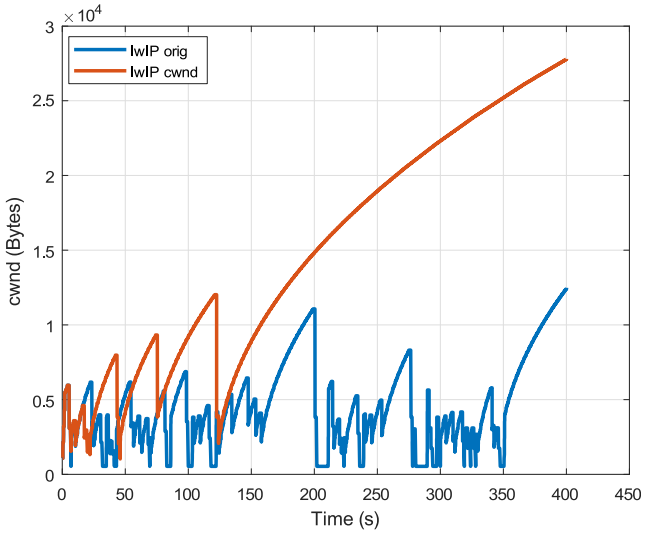

FIGURE 8 Congestion window progress of lwIP and lwIP_cnwd algorithms. lwIP, light weight IP.

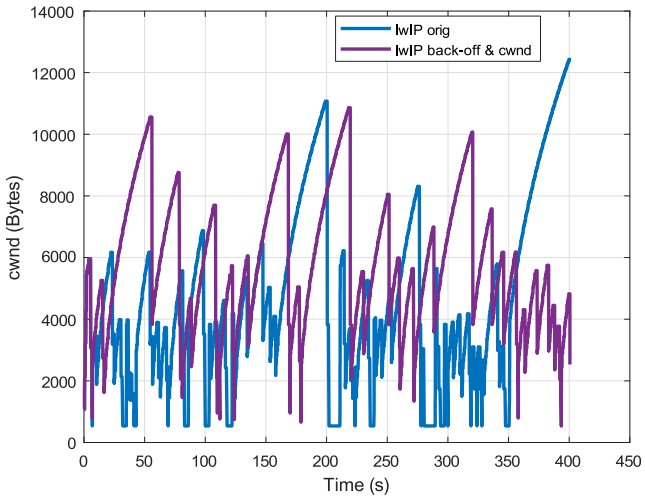

FIGURE 9 Congestion window progress of lwIP and lwIP_back_off_cnwd algorithms. lwIP, light weight IP.

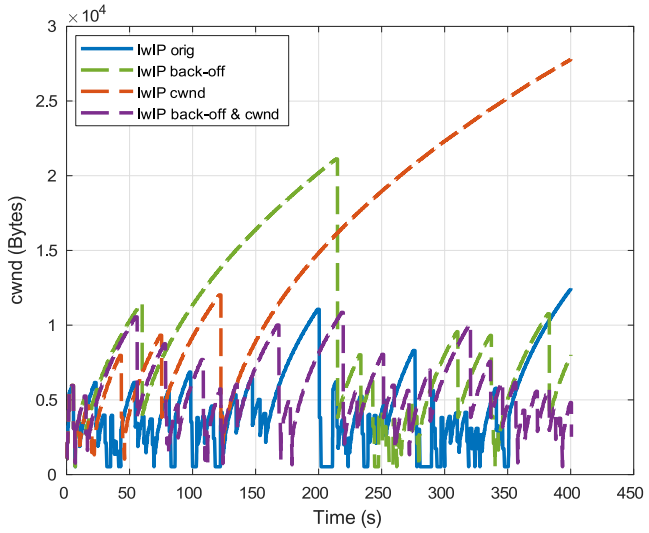

FIGURE 10 Summary of congestion window progress of all algorithms combined.

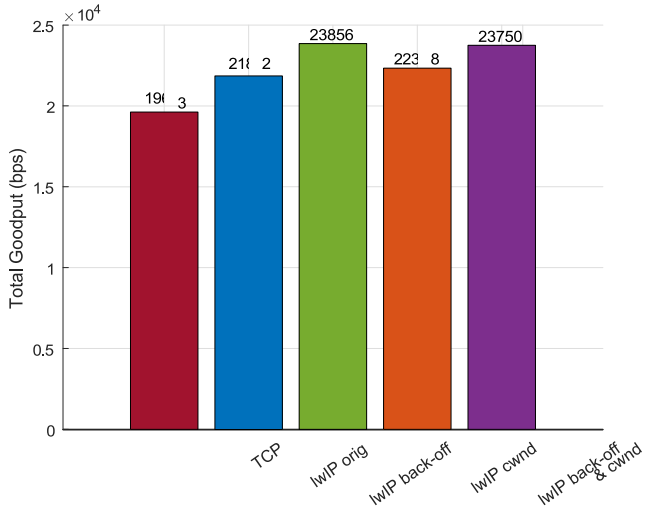

FIGURE 11 Total network goodput for all algorithms.

value adjustment, but it was slightly less than the lwIP with *cwnd* floor value alone.

We have also computed the total number of retransmissions of all algorithms over the simulation time, as depicted in Figure 12. As we can see, the total number of retransmissions significantly decreased from 136 for original lwIP to <27. We can see that this is dropped to 10 for lwIP with both dynamic RTO back-off and cwnd floor value adjustment which is the minimum number of retransmissions across all implementations. Therefore, the final combined implementation provides a slight trad-off on the network goodput when compared to lwIP with dynamic RTO back-off, but reduces the total number of retransmissions to a minimum.

As can be seen from Figures 11 and 12, our proposed algorithms, especially lwIP with both dynamic RTO back-off and cwnd, have better performance than other proposals.

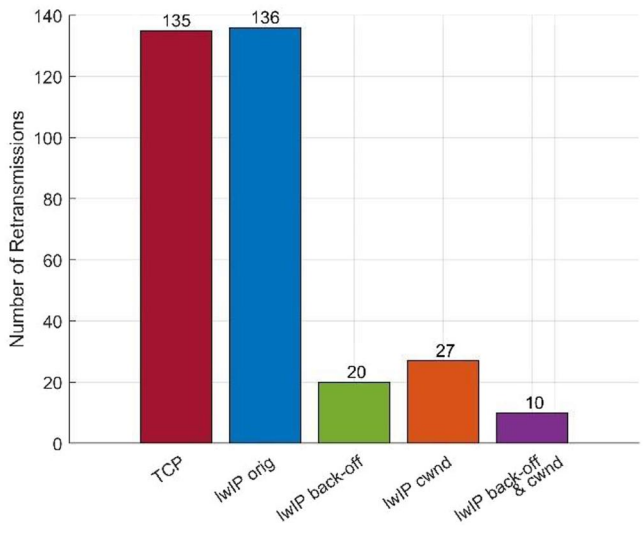

FIGURE 12 Total number of retransmissions of all algorithms.

This has a direct consequence on the overall IoT network efficiency in terms of energy consumption. Less retransmissions translates into more battery life for IoT devices, as more retransmissions would otherwise drain devices batteries faster.

6 | CONCLUSION

IoT devices and networks will be a significant part of the future. They are utilised for multiple applications across various industries and will only keep growing in the foreseeable future, warranting careful consideration to the reliability of the underlying IoT network and it congestion. Light-weight TCP implementations have been considered as suitable candidates for TCP-based IoT networks. This paper proposes a light-weight TCP set of implementations and evaluated their performance merits through simulations of a typical start topology IoT network.

The results show that the proposed novel schemes produced an improvement in terms of total goodput of the IoT network as well as a significant drop in the total number of retransmissions for lwIP clients. In particular, finding the right optimal combination of congestion metrics, such as shown by the lwIP back-off & cwnd algorithm, is the right sailing direction in overcoming congestion in IoT networks. This is considered as a one step towards making ulta-scalable IoT networks that are more reliable both in traffic growth as well as node count. Needless to say, this is an attempt to improve existing congestion control schemes in IoT one-to-one scenarios. We believe that IoT networks should intrinsically account for many-to-one communications type of traffic, to account for partition/aggregate applications. This is a, yet, one more challenge to address in the future which would make the IoT transport protocols ready for next generation intelligent networks.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: study conception and design: Lotfi Mhamdi and Hussam Abdul Khalek; data collection: Hussam Abdul Khalek; analysis and interpretation of results: Lotfi Mhamdi and Hussam Abdul Khalek; draft manuscript preparation: mainly Lotfi Mhamdi. All authors reviewed the results and approved the final version of the manuscript.

CONFLICT OF INTEREST STATEMENT

The authors have no conflict of interest to disclose.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Lotfi Mhamdi https://orcid.org/0009-0000-6492-2088

REFERENCES

1. Lee, I., Lee, K.: The Internet of Things (IoT): applications, investments, and challenges for enterprises. Bus. Horiz. 58(4), 431–440 (2015). https://doi.org/10.1016/j.bushor.2015.03.008
2. Jovanovic, B.: Internet of Things statistics for 2021 –taking things apart (1999)
3. IETF: Terminology for constrained node networks. http://www.ietf.org/rfc/rfc7228.txt (2014)
4. Stankovic, J.A.: Research directions for the Internet of Things. IEEE Internet Things J. 1(1), 3–9 (2014). https://doi.org/10.1109/JIOT.2014.2312291
5. He, Z., et al.: Congestion avoidance in intelligent transport networks based on WSN-IoT through controlling data rate of Zigbee protocol by learning automata. Electronics 12(9), 2070 (2023). https://doi.org/10.3390/electronics12092070
6. Premaratne, U., Warnakulasooriya, S., Nandana, R.: Characterization of event-based sampling encoders for industrial Internet of Things using input–output mutual information. IEEE Trans. Ind. Inf. 17(8), 5495–5505 (2021). https://doi.org/10.1109/TII.2020.3027009
7. Jain, V.K., et al.: Congestion control in Internet of Things: classification, challenges, and future directions. Sustain. Comput. 35, 100678 (2022). https://doi.org/10.1016/j.suscom.2022.100678
8. Vasudevan, V., et al.: Safe and effective fine-grained TCP retransmissions for datacenter communication. Comput. Commun. Rev. 39(4), 303–314 (2009). https://doi.org/10.1145/1594977.1592604
9. Adesanmi, A., Mhamdi, L.: Controlling TCP Incast congestion in data centre networks. In: IEEE International Conference on Communications (ICC), pp. 1827–1832. London, UK (2015)
10. Hung, W.C., Law, K.E.: Simple slow-start and a fair congestion avoidance for TCP communications. In: 2008 Canadian Conference on Electrical and Computer Engineering, pp. 001771–001774. (2008)
11. Al-Fuqaha, A., et al.: Internet of Things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutorials 17(4), 2347–2376 (2015). https://doi.org/10.1109/comst.2015.2444095
12. Dong, X., et al.: NATS-bench: benchmarking NAS algorithms for architecture topology and size. IEEE Trans. Pattern Anal. Mach. Intell. 44(7), 3634–3646 (2022). https://doi.org/10.1109/TPAMI.2021.3054824
13. Sembroiz, D., Ricciardi, S., Careglio, D.: Chapter 10 - a novel cloud-based IoT architecture for smart building automation. In: Ficco, M., Palmieri, F. (eds.) Security and Resilience in Intelligent Data-Centric Systems and Communication Networks, pp. 215–233. Academic Press (2018)
14. Betzler, A., et al.: CoCoA+: an advanced congestion control mechanism for CoAP. Ad Hoc Netw. 33, 126–139 (2015). https://doi.org/10.1016/j.adhoc.2015.04.007
15. Lim, C.: Improving congestion control of TCP for constrained IoT networks. Sensors 20(17), 4774 (2020). https://doi.org/10.3390/s20174774
16. Mulligan, G.: The 6LoWPAN architecture. In: Proceedings of the 4th Workshop on Embedded Networked Sensors, EmNets 2007, Cork, Ireland, 25–26 June 2007
17. Al-Kashoash, H.A.A., et al.: Congestion control in wireless sensor and 6LoWPAN networks: toward the Internet of Things. Wireless Network 25(8), 4493–4522 (2019). https://doi.org/10.1007/s11276-018-1743-y
18. Dunkels, A., et al.: Low-Power IPv6 for the Internet of Things. In: 2012 Ninth International Conference on Networked Sensing (INSS), pp. 1–6. IEEE, Antwerp, Belgium (2012). https://doi.org/10.1109/INSS.2012.6240537
19. Castellani, A.P., Rossi, M., Zorzi, M.: Back pressure congestion control for CoAP/6LoWPAN networks. Ad Hoc Netw. 18, 71–84 (2014). https://doi.org/10.1016/j.adhoc.2013.02.007
20. Dunkels, A.: Design and implementation of the lwIP TCP/IP stack. Swed. Inst. Comput. Sci. 2 (2001)
21. Ahmed, Z., et al.: AD-RDC: a novel adaptive dynamic radio duty cycle mechanism for low-power IoT devices. IEEE Internet Things J., 1 (2022). https://doi.org/10.1109/JIOT.2022.3145017
22. Gomez, C., Crowcroft, J., Scharf, M.: RFC 9006: TCP usage guidance in the Internet of things (IoT) (2021)
23. Dunkels, A.: Full TCP/IP for 8-bit architectures. In: Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys '03), pp. 85–98. Association for Computing Machinery, New York (2003). https://doi.org/10.1145/1066116.1066118
24. IETF: TCP congestion control, IETF, RFC 5681 (2009)
25. Naik, N.: Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP. 2017 IEEE International Systems Engineering Symposium (ISSE), pp. 1–7. IEEE, Vienna, Austria (2017). https://doi.org/10.1109/SysEng.2017.8088251
26. Thomson, C., et al.: Cooja simulator manual (2016). https://doi.org/10.13140/RG.2.1.4274.8408
27. Musaddiq, A., Zikria, Y.B.: Routing protocol for low-power and lossy networks for heterogeneous traffic network. EURASIP J. Wirel. Commun. Netw. 21, 126–139 (2020). https://doi.org/10.1186/s13638-020-1645-4
28. Gomez, C., Arcia-Moret, A., Crowcroft, J.: TCP in the Internet of Things: from ostracism to prominence. IEEE Internet Comput. 22(1), 29–41 (2018). https://doi.org/10.1109/MIC.2018.112102200
29. Vinoski, S.: Advanced message queuing protocol. IEEE Internet Comput. 10(6), 87–89 (2006). https://doi.org/10.1109/MIC.2006.116
30. Langley, A., et al.: The QUIC transport protocol: design and internet-scale deployment. In: ACM SIGCOMM '17, pp. 183–196. Association for Computing Machinery, New York (2017)
31. Kumar, S., et al.: Performant TCP for low-power wireless networks. In: 17th USENIX Symposium on Networked Systems Design and Implementation, pp. 911–932. USENIX Association, USA (2020)
32. Varga, A.: The OMNET++ discrete event simulation system. In: Proc. ESM'2001. 9 (2001)

How to cite this article: Mhamdi, L., Abdul Khalek, H.: Congestion control in constrained Internet of Things networks. IET Wirel. Sens. Syst. 13(6), 247–255 (2023). https://doi.org/10.1049/wss2.12072