A survey of end-to-end congestion mechanisms in the field of IoT

Dalibor Fonović*, Siniša Sovilj* and Nikola Tanković*

*Juraj Dobrila University of Pula, Pula, Croatia

dalibor.fonovic@unipu.hr

ABSTRACT

The Internet of Things (IoT) is a global network of devices capable of communicating and exchanging data with other devices and systems, mainly via the Internet, using specific communication protocols. The number of such devices that can be connected to the Internet is constantly growing due to progress in various technological areas (such as communication technologies, microelectronic circuits, sensors, embedded systems, smartphones, etc.). This leads to network congestion due to the large amount of data exchanged between devices. In addition, IoT devices are resource-constrained, further exacerbating network congestion. Network congestion leads to additional communication delays, low bandwidth, and waste of computer resources. This is one of the important aspects when some IoT applications exchange critical information (e.g., monitoring the patient's health condition in the application of IoT in smart healthcare). As the number of IoT applications increases, so does the need to modify or introduce new protocols to deal with the problems of adapting to network conditions. In the IoT layered architecture, the transport layer plays an important role in managing the end-to-end connection to the services of the upper application IoT layer. An important function of the transmission layer of the IoT layer is congestion control. This paper presents an overview of related research on the congestion control mechanism of the IoT architecture's transport layer, advantages, disadvantages, and current transport layer problems in IoT applications.

Keywords: Internet of Things; Congestion control; TCP protocol; CoAP protocol; Machine learning.

1. INTRODUCTION

IETF**1** defines the Internet of Things as a system with devices often constrained in communication and computation capabilities, becoming more commonly connected to the Internet or at least to an IP network and to various services built on top of the capabilities these devices jointly provide. Such development is expected to usher in a more machine-to-machine Internet communication with no active human mediation [1].

IoT enables the interconnection of different devices to transfer data over the network. Major applications of IoT are smart homes, smart cities, smart grids, industrial monitoring systems, health monitoring systems, environmental monitoring systems, etc.

As the number of connected IoT devices increases, Internet congestion also increases. Also, as new technologies and new communication networks develop, the complexity of network transmission has also increased. This led to new challenges in the design of network protocols. Congestion control (hereinafter referred to as CC) is one of the fundamental components of computer networks. CC plays an important role when improving the utilization of the network to achieve better IoT application performance. As future networks become increasingly complex, conventional congestion control approaches based on pre-defined rules become increasingly ineffective.

2. OVERVIEW OF THE IOT STACK APPLICATION LAYER PROTOCOLS

In the layered architecture of IoT, the application layer provides various communication protocols and acts as an interface between the application and end users or M2M communication.

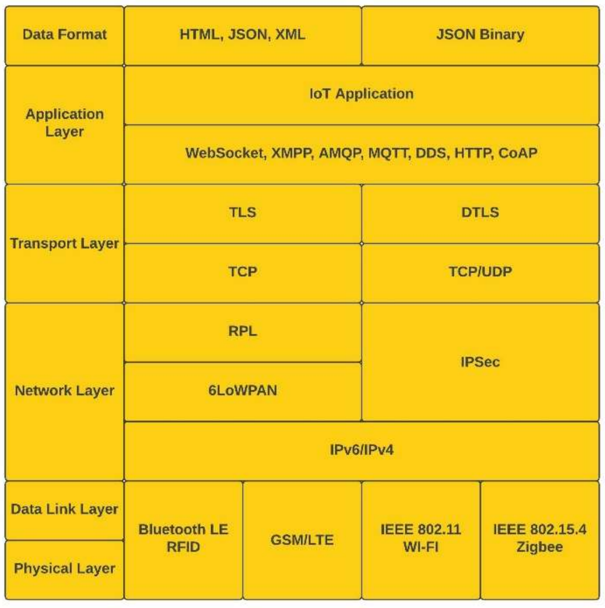

Figure 2.1: Architecture of IoT stack [21]

---

1 IETF - The Internet Engineering Task Force, standards development organization for the Internet.

2.1. Request/response protocols

The request-response model is one of the basic two-way communication models on the Internet, where the calling process sends a request for data, and the responding process issues a response to the request. RESTful HTTP and CoAP protocols are often applied in IoT applications as the request/response communication pattern implementation.

2.2. MQTT

MQTT is a message transfer protocol based on publish/subscribe architecture. MQTT v3.1 was adopted by the OASIS consortium in 2013 and certified by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (ISO/IEC) in 2016. It is developed for resource-constrained devices, based on open source, reliable is also a simple protocol. Four years after MQTT 3.1 became the OASIS standard, the MQTT 5 protocol was released, bringing significant improvements and upgrades to the MQTT 3.1 protocol. In March 2019, MQTT 5.0 became the new OASIS standard.

3. REQUIREMENTS FOR NETWORK IN IOT

IoT devices have limited computing capabilities (so-called constrained devices) and limited network bandwidth. In addition, IoT devices most often use wireless networks in which (due to the nature of this type of communication) packet losses often occur (lossy networks) and have limited energy consumption (e.g., battery-powered IoT devices). Also, IoT devices have different requirements regarding communication speed, delay, and reliability according to the application method.

Another important aspect of some IoT applications is exchanging sensitive and critical information. For example, the application of IoT in smart health deals with the transfer of essential and time-critical data, which includes, for example, monitoring the health status of patients where packet loss or delay is not acceptable. That's why the goal is to reduce losses and packet delay.

TCP/IP was not originally designed to run on limited devices. The current congestion control algorithms are primarily designed for the web. They are only sometimes suitable for device-constrained environments, where nodes are limited in CPU and memory. The network is characterized by high packet loss, low bandwidth, or frequent topology change. Thus, deploying and integrating congestion control mechanisms on limited devices creates new challenges.

3.1. Congestion control

Congestion is a condition that occurs in a network when data traffic is so high that it slows down the network's response time. Congestion negatively affects the performance of IoT applications. It leads to the retransmission of packets, thereby increasing energy consumption, delay, and packet loss while reducing bandwidth and packet delivery ratio (PDR). Therefore, congestion control plays an important role in meeting the performance requirements of network communication.

Congestion control is a technique that controls the rate of sent data packets to optimize the use of network infrastructure. It is designed to avoid the degradation of network performance caused by congestion. A simplified congestion control principle is that when a packet loss is detected, the sender reduces its sending rate. Most IoT technologies use wireless (radio) connections. Wireless radio links generally have a higher transmission error rate than wired links due to their susceptibility to interference. They generally have a lower transmission speed.

An example of how congestion management can affect an IoT application's performance is a camera forest fire detection system with built-in machine learning (computer vision) models. Regardless of successful detection, sending a picture of the fire itself is still necessary. The limitation of such a system can be the communication connection (for example, we use NB-IoT technology, characterized by low bandwidth and the limited amount of traffic that the telecommunications operator allows for this type of communication). In this case, optimizing congestion management would lead to improved system performance. An example of such a system is described in the paper [2].

4. END TO END CONGESTION CONTROL IN THE IOT AREA

Traditional CC algorithms for congestion control (CC in further text) can be categorized into end-to-end CC and network-assisted CC. End-to-end approaches require only the cooperation of the sender and receiver. This category does not rely on explicit congestion information obtained from the network. Conversely, network-assisted approaches require congestion information from network devices (like routers).

Traditional end-to-end CC algorithms implemented so far rely on predefined rules. Based on these rules, the congestion window variable (CWND) and the retransmission timeout (RTO) - which determines how long the transport engine waits for an acknowledgment - are recalculated continuously. Depending on the congestion indicator, they can be classified as loss-based, delay-based, and hybrid (including both congestion indicator mechanisms). Many algorithms of one of these types have already been developed and implemented. However, modern networks are complex, diverse, and heterogeneous in their structure. Applying strict rules in a multi-parameter and dynamic environment does not perform equally well in different network conditions; rules that work nearly optimally in one environment may not work well in other environments or when network conditions change rapidly.

Considering the problem space and the variety and variability of the input parameters, new approaches based on machine learning have recently attracted much research attention. One such study was conducted by Wei et al. [3]. Unlike traditional CC algorithms, ML-based patterns rely on real-time network states to make decisions instead of predetermined rules.

Transmission Control Protocol (TCP) is the fundamental protocol of the Internet. However, IoT applications use specific communication patterns that TCP cannot effectively support. For example, some IoT devices may go into sleep mode, which causes the connection to terminate. In such cases, it is impossible to keep the connection active (keep-alive), which can be one of the requirements in IoT applications (e.g., using the MQTT protocol). Due to situations like the above, an interest in building new reliable transport protocols on top of the unreliable UDP transport protocol has risen.

4.1. Congestion control mechanisms

Congestion management is based on the control of the size of the sending window (window size). The mechanism adjusts the window size to accommodate congestion. The window size can be considered the number of segments sent to the network, representing the number of segments and ACK[2] in transit or queued.

4.1.1. Phases of congestion control

Slow start: at the beginning, we start sending segments slowly, then exponentially increasing the speed of sending data until the congestion window (CWND) value reaches the threshold value (*ssthresh*).

Congestion Avoidance: After reaching the threshold, the sender linearly increases the size of the congestion window to avoid congestion. Each time an acknowledgment is received, the sender increases the congestion window size by 1.

Congestion Detection: there are two methods by which packet losses can be identified over the communication path: (1) by timeout (RTO) and (2) by receiving duplicate acknowledgments (*dupACK*).

4.2. Packet loss approach

When a sender has not received an ACK for a sent packet for a certain period, this usually indicates packet loss. The packet loss-based approach adjusts the sending rate when packet loss occurs. An example of an algorithm based on packet loss is TCP Vegas [4].

4.3. Delay-based approach

The delay-based approach predicts and reacts to congestion before packet loss occurs. This mechanism relies on detected network-induced transmission delays. An example of a delay-based algorithm is *TCP NewReno* [5]. Compared to loss-based approaches, delay-based approaches are more suitable for fast networks. Calculating the exact transmission delay is still a challenge. For example, a small change in packet processing time at the host can cause deviations in the measured transmission delay, which causes wrong decisions when sending. Hybrid approaches have been proposed to take advantage of both loss and delay approaches. TCP Copa is one such example [6].

4.4. ML-based approach

The complexity of today's network architectures is a significant challenge for CC. Designing a generic CC mechanism that will work on all network scenarios is challenging. The dynamic nature of even the same network can make CC operation incorrect.

Because of that, ML-based CC algorithms have been proposed to solve the above-mentioned problems. Traditional CC algorithms rely on using predetermined rules. ML-based mechanisms rely on real-time network states to make congestion control decisions. This allows better adoption of CC mechanisms to dynamic and complex network scenarios.

5. CC MECHANISMS IN THE IoT

5.1. Traditional CC mechanisms

RTO estimation. Retransmission timeout (RTO) determines how long the transport mechanism waits for confirmation (ACK) of the sent segment. The segment is considered lost if confirmation is not received within this time. An important part of the RTO calculation is determining how long it takes for the segment to go to the receiver and for the ACK to return from the receiver to the sender. This is called RTT or Round-Trip Time.

TCP uses the RTO (Retransmission Timeout) estimation algorithm defined in RFC6298 [11]. TCP adaptively determines the RTO by applying the EWMA (exponentially weighted moving average) algorithm to RTT samples.

CC mechanisms based on TCP protocol. Verma [7] et al made CC adapted to IoT. They made a comparative study comparing the new mechanism with other TCP CC mechanisms, such as TCP Cubic, TCP New Reno, and TCP BBR, simulating the IoT environment (devices with limited resources and limited network bandwidth).

The RTO estimation algorithm defined in RFC6298 is not designed to consider IoT environment scenarios. In research [8], the expected RTO algorithm defined in [9] outperformed state-of-the-art algorithms designed to enhance RFC 6298 for TCP in terms of PDR.

CoAP-based CC mechanisms. CoAP is a simple protocol, and the basic specification offers a default CoAP CC mechanism that uses an RTO with binary exponential offset (BEB) for lost packets. Since lost packets are retransmitted in an exponentially increasing time, the default CoAP CC is very simple and insensitive to dynamic network conditions. Consequently, the default CoAP CC has lower performance.

Due to UDP communication, using CoAP with existing network infrastructures (e.g., the use of NAT using firewalls) may lead to certain limitations. This is one of the reasons why there is also a CoAP over TCP specification [14].

To improve the performance of the default CoAP CC, CoAP Simple Congestion Control/Advanced CoCoA is proposed. CoCoA uses round trip time (RTT) measurements and adaptive RTO. This algorithm responds to congestion with a lower sending rate.

[2] ACK - short for "acknowledgement.". In TCP protocol an ACK packet is any packet that acknowledges receiving a sent packet.

Table 5.1: REVIEW OF RECENT RESEARCH ARTICLES ON TCP BASED CC MECHANISMS

Article	Protocol	Description
[10]	TCP Cubic	One of the most common used TCP CC mechanisms. It is supported by most of today OS.
RFC 6582[11]	TCP New Reno	Improvement of performances compared to TCP Reno
[8]	TCP uIP with CoCoA CC mechanism	Performance improvement in the IoT environment compared to RFC6298
Verma [7]	IoT based congestion control algorithm	A comparative study where the mentioned new TCP mechanism is compared in an IoT environment with other known TCP CC mechanisms. Showed the best performance, except in RTT where only TCP Cubic is better

Table 5.2: REVIEW OF RECENT RESEARCH ARTICLES ON CoAP BASED CC MECHANISMS

Article	Protocol	Description	Advantages	Disadvantages
[10]	CoCoA+	Uses RTT for RTO estimation and VBF backoff	Reduction of retransmission compared to the default CoCo model	Inconsistency in expected RTO
[12]	CoCoA++	Uses CAIA Delay Gradient for RTO estimation and PBF backoff	Low RTO with minimal latency and high packet transfer rate in various IoT scenarios	Increased end-to-end latency and computationally intensive
[13]	pCoCoA	Modification of the RTO estimation compared to the CoCoA+ algorithm	Improvement of congestion control (reduced packet loss and thus retransmission)	Increased end-to-end latency
[14]	CoAP over TCP	CoAP over TCP	Enables the use of TLS and WebSocket protocols	Weaker performance

Table 5.3: REVIEW OF RECENT RESEARCH ARTICLES ON ML BASED CC MECHANISMS

Article	Protocol	ML method	Advantages	Disadvantages
Xiao [15]	TCP - Drinc	Reinforcement learning	Increasing performance in complex and dynamic network environments	
Falahatraftar[17]	Predikcija zagušenja GRNN	GRNN model compared to SVM, decision tree, regression models	The GRNN model shows higher accuracy, reliability and stability among the forecasting methods considered	
Demir[16]	mlCoCoA	Uses SVM for dynamic RTO calculation	Performance improvement over CoCoA	Computationally intensive
Sander[18]	DeePCCI	RNN model	Performance improvement over traditional TCP CC mechanisms	
Najm[19]	C4.5 DT	Using a decision tree model for congestion control in a 5G IoT environment	Significant performance improvement over other ML machine learning algorithms	

5.2. CC mechanisms based on ML algorithms.

Xiao et al [15], proposed TCP-Drinc for smart congestion management for TCP variants. The authors use Deep Reinforcement Learning. Congestion is controlled by adjusting the window size. TCP-Drinc is compared against different versions of TCP: TCP-New Reno, TCPCubic, TCP-Hybla, TCP-Vegas and TCP-Illinois. TCP-Drinc provides maximum throughput and the second lowest round trip time (RTT).

Demir et al [16] proposed mlCoCoA, ML-based improvement in CoCoA, mlCoCoA adaptively sets CoCoA retransmission timeout (RTO) estimation parameters using an ML method.

Sander, et al [18] proposed DeePCCI for congestion detection based on packet arrival time in traffic flow.

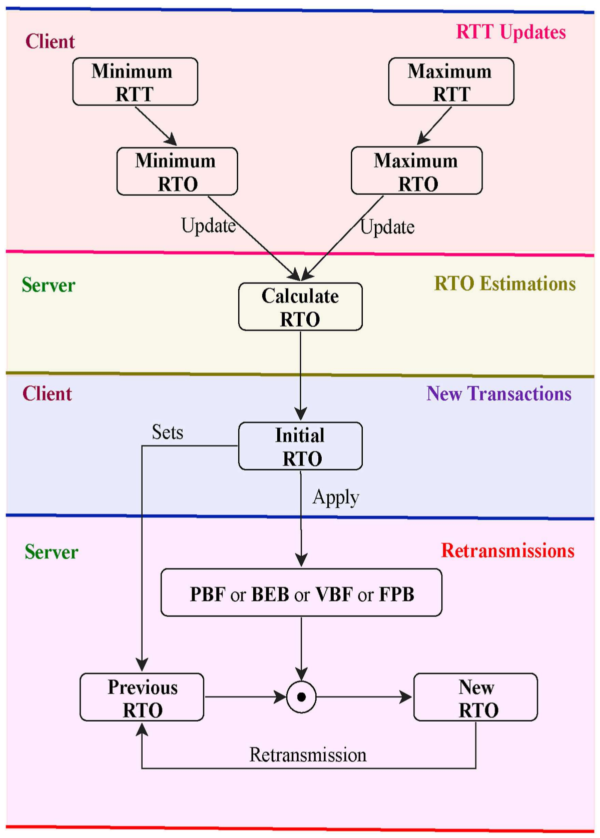

Figure 5.1: Example of RTO estimation strategy for CoAP and its improvements [21].

6. Conclusion

Network communication speed and reliability with low resource utilization in resource-constrained devices are key requirements for IoT.

Congestion negatively affects performance. Congestion leads to the retransmission of packets, thereby increasing energy consumption, delay, and packet loss while reducing the throughput and packet delivery ratio (PDR). This can have negative consequences for the end IoT application. Congestion control should be an important issue when working in IoT networks. Traditional end-to-end CC algorithms implemented so far rely on predefined rules.

There are two main ways to control congestion. In the first case, the application layer uses the CC mechanism of the TCP protocol. Examples are the HTTP and MQTT protocols that build on the TCP protocol for congestion control. The CC mechanism is implemented in the application layer in the second case. An example is the CoAP protocol which uses the CC mechanism in the application layer of the IoT model and is based on the UDP protocol.

In this paper, a literature review of existing CC mechanisms was made with an emphasis on the IoT field of application. A large number of research deal with improving the CC mechanism of the TCP protocol. There are also many works where machine learning has been used to improve the performance of the TCP CC mechanism. However, only a small amount of research is focused on applying TCP congestion control mechanisms to the IoT application area. As for the CoAP protocol, many works are investigating the improvements of the CC mechanism. A smaller number of studies use the application of machine learning for the CC mechanism of the CoAP protocol.

Acknowledgments

This paper acknowledges the support of the Croatian Science Foundation research project CSF no. HRZZ-IP-2019-04-4216 “Reliability and Safety in Complex Software Systems: From Empirical Principles towards Theoretical Models in View of Industrial Applications (RELYSOFT)” and the support of the Erasmus+ Key Action 2 (Strategic partnership for higher education) project No. 2020–1–PT01–KA203–078646: “SusTrainable - Promoting Sustainability as a Fundamental Driver in Software Development Training and Education”.

This paper has also been supported by European Regional Development Fund (ERDF) project SUNSET IoT under grant agreement number KK.01.2.1.02.0116, by Croatian Science Foundation under the project UIP-201705-9066.

References

[1] The internet of things at the IETF. Available at: https://www.ietf.org/topics/iot/ (accessed: January 22, 2023).

[2] G. Peruzzi, A. Pozzebon, and M. Van Der Meer, “Fight fire with fire: Detecting forest fires with embedded machine learning models dealing with audio and images on low power IoT devices,” Sensors, vol. 23, no. 2, 2023, doi: 10.3390/s23020783.

[3] W. Wei, H. Gu, and B. Li, “Congestion control: A renaissance with machine learning,” IEEE Network, vol. 35, no. 4, pp. 262–269, 2021, doi: 10.1109/MNET.011.2000603.

[4] L. S. Brakmo and L. L. Peterson, “TCP vegas: End to end congestion avoidance on a global internet,” IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, pp. 1465–1480, 1995, doi: 10.1109/49.464716.

[5] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida, “The NewReno Modification to TCP’s Fast Recovery Algorithm.” RFC 6582; RFC Editor, Apr. 2012. doi: 10.17487/RFC6582.

[6] V. Arun and H. Balakrishnan, “Copa: Practical delay-based congestion control for the internet ,” in ANRW ’18:Proceedings of the Applied Networking Research Workshop, Jul. 2018, pp. 19–19. doi: 10.1145/3232755.3232783.

[7] L. P. Verma and M. Kumar, “An IoT based congestion control algorithm,” Internet of Things, vol. 9, p. 100157, 2020, doi: https://doi.org/10.1016/j.iot.2019.100157.

[8] C. Lim, “Improving congestion control of TCP for constrained IoT networks ,” Sensors, vol. 20, no. 17, 2020, doi: 10.3390/s20174774.

[9] C. Bormann, A. Betzler, C. Gomez, and I. Demirkol, “CoAP Simple Congestion Control/Advanced,”Internet Engineering Task Force; Internet Engineering Task Force, Internet-Draft draft-ietf-core-cocoa-03, Feb. 2018. Dostupno: https://datatracker.ietf.org/doc/draft-ietf-core-cocoa/03/

[10] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, “CoCoA+: An advanced congestion control mechanism for CoAP,” Ad Hoc Networks, Apr. 2015, doi: 10.1016/j.adhoc.2015.04.007.

[11] V. Paxson, M. Allman, J. Chu, and M. Sargent, Computing TCP’s retransmission timer. RFC 6298, 2011.

[12] V. Rathod, N. Jeppu, S. Sastry, S. Singala, and M. P. Tahiliani, “CoCoA++: Delay gradient based congestion control for internet of things,” Future Generation Computer Systems, vol. 100, pp. 1053–1072, 2019, doi: https://doi.org/10.1016/j.future.2019.04.054.

[13] S. Bolettieri, G. Tanganelli, C. Vallati, and E. Mingozzi, “pCoCoA: A precise congestion control algorithm for CoAP,” Ad Hoc Networks, vol. 80, pp. 116–129, 2018, doi: https://doi.org/10.1016/j.adhoc.2018.06.015.

[14] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, and B. Raymor, “CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets.” RFC 8323; RFC Editor, Feb. 2018. doi: 10.17487/RFC8323.

[15] K. Xiao, S. Mao, and J. K. Tugnait, “TCP-drinc: Smart congestion control based on deep reinforcement learning,” IEEE Access, vol. 7, pp. 11892–11904, 2019, doi: 10.1109/ACCESS.2019.2892046.

[16] A. K. Demir and F. Abut, “mlCoCoA: a machine learning-based congestion control for CoAP,” TurkishJournal of Electrical Engineeringand Computer Sciences, vol. 28, no. 5, 2020, doi: 10.3906/ELK-2003-17.

[17] F. Falahatraftar, S. Pierre, and S. Chamberland, “An intelligent congestion avoidance mechanism based on generalized regression neural network for heterogeneous vehicular networks,” IEEE Transactions on Intelligent Vehicles, pp. 1–13, 2022, doi: 10.1109/TIV.2022.3180665.

[18] C. Sander, J. Rüth, O. Hohlfeld, and K. Wehrle, “DeePCCI: Deep learning-based passive congestion control identification,” in Proceedings of the 2019 workshop on network meets AI & ML, 2019, pp. 37–43. doi: 10.1145/3341216.3342211.

[19] I. A. Najm, A. K. Hamoud, J. Lloret, and I. Bosch, “Machine learning prediction approach to enhance congestion control in 5G IoT environment,” Electronics, vol. 8, no. 6, 2019, doi: 10.3390/electronics8060607.

[20] R. Al-Saadi, G. Armitage, J. But and P. Branch, "A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms," in IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3609-3638, Fourthquarter 2019, doi: 10.1109/COMST.2019.2904994.

[21] C. Bayılmış, M. A. Ebleme, Ü. Çavuşoğlu, K. Küçük, and A. Sevin, “A survey on communication protocols and performance evaluations for internet of things,” Digital Communications and Networks, vol. 8, no. 6, pp. 1094–1104, 2022, doi: https://doi.org/10.1016/j.dcan.2022.03.013.