Analysis of TCP Issues and Their Possible Solutions in the Internet of Things

Syed Zeeshan Hussain
Department of Computer Science, Jamia Millia Islamia, India
szhussain@jmi.ac.in

Sultana Parween
Department of Computer Science, Jamia Millia Islamia, India
sultana.tech@gmail.com

Abstract: *The Internet of Things (IoT) is widely known as a revolutionary paradigm that offers communication among different types of devices. The primary goal of this paradigm is to implement efficient and high-quality smart services. It requires a protocol stack that offers different service requirements for inter-communication between different devices. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are used as transport layer protocols in IoT to provide the quality of service needed in various IoT devices. IoT encounters many shortcomings of wireless networks, while also posing new challenges due to its uniqueness. When TCP is used in an IoT system, a variety of challenging issues have to be dealt with. This paper provides a comprehensive survey of various issues which arises due to the heterogeneous characteristics of IoT. We identify main issues such as Retransmission Timeout (RTO) algorithm issue, congestion and packet loss issue, header overhead, high latency issue, link layer interaction issue, etc., Moreover, we provide several most probable solutions to the above-mentioned issues in the case of IoT scenarios. RTO algorithm issue has been resolved by using algorithms such as CoCoA, CoCoA+, and CoCoA++. Apart from these, the high latency issue has been solved with the help of a long lived connection and TCP Fast open. Congestion and packet loss issue has been resolved by using several TCP variants such as TCP New Reno, Tahoe, Reno, Vegas, and Westwood.*

Keywords: *RTO, acknowledgment, round trip time.*

Received June 4, 2021; accepted August 29, 2022
https://doi.org/10.34028/iajit/20/2/7

1. Introduction

Transmission Control Protocol (TCP) is the most reliable, connection-oriented protocol that offers reliable end-to-end delivery of data over networks. The reliability of TCP is maintained by the use of sequence numbers (seqno) and Acknowledgment (ACKs). It is referred to as a self-clocking algorithm because it uses the ACK as a clock to send information to the network system and dynamically adjusts its transmission rate according to the available network capacity. It allocates a seqno to every octet (byte) during data transmission, and then encapsulates the octets into segments. The transmission of data is based on the first octet of a segment's data, also known as the segment sequence seqno [3, 35].

Once the data segment reaches the target node, an acknowledgment is sent back to the sender along with the seqno of the next expected data octet number. It ensures the reliability of data segment transmission. If an erroneous sequence of data segments reaches the destination node, it indicates that a data segment has been lost between the previously and presently arrived segments. At that time, the receiver sends out an ACK in order to acknowledge the data segment so that it can begin the process of retransmitting. A Duplicate ACK (DACK) is one that recognizes the same data segment to be retransmitted more than once. After getting three DACKs, the sender acknowledges that the segment was lost and must be resent.

In addition, whenever a segment is transferred, TCP starts a timer to record the timeout occurrence. If the timeout occurs before receiving the ACK, the sender considers that the segment was lost. Then the lost segment must be retransmitted, and TCP initiates the slow start method. A Retransmission timeout is another name for the Timeout interval (RTO) [34]. TCP has an additional receiver-side technique to regulate the amount of data that is sent by the sender. To state each ACK, the receiver defines a window size referred to as the advertised window or receiver window (rwnd). Also, a congestion window (cwnd) indicates the amount of data that a sender can send without getting an ACK from the recipient. The total data transferred over the network by the sender is equal to the minimum of these two windows i.e.

Data transferred = MIN (Cwnd, Advertised window).

The two principal transmission mechanisms of congestion algorithms in TCP are the slow start mechanism and congestion avoidance. In the slow start phase, the cwnd increases exponentially before the ssthresh are achieved. The congestion avoidance process begins afterward and the cwnd is increased to some predefined value by one Maximum Segment Size (MSS).

1.1. Motivation and Objectives

TCP was initially designed to perform well in traditional wired network environments and provide reliable data services. Because of its extensive usage on the internet, it is required that TCP should continue to provide reliable data service for interactions within IoT. Unfortunately, the performance of TCP control mechanisms is inadequate in IoT scenarios. TCP performance degrades due to several issues such as RTO algorithm issue, Congestion, packet loss issue, header overhead, high latency issue, link layer interaction issue, etc., in the context of IoT environment. Several solutions have been proposed in response to these IoT-specific problems and enhance TCP performance in the IoT. The objective of this paper is to present an overview of recent developments and explore some open research issues of TCP in IoT networks and provide probable solutions.

1.2. Research Gaps

Moreover, there are only a few studies that present a complete understanding of TCP current state of IoT. After careful review of the several research papers, the following research gaps have been identified:

- The research works presented here the TCP issues and their possible solutions on the IoT networks. However, till now most of the surveys are done on the TCP issues in the networks such as multi-hop wireless networks, Mobile Ad Hoc Network (MANET), and wireless ad hoc networks.
- The outcome obtained from recently published papers have been incorporated in this paper.
- A significant number of the papers did not provide any suggestions for the conduct of further study or development.
- As a result of this, a gap in the existing research has been investigated and our work is well capable of addressing this research gap.

1.3. Contributions of The Paper

The comprehensive review work focuses on the above-listed gaps in the literature. The originality and contributions of the paper have been highlighted as follows:

- A comprehensive analysis of the state that the existing situation of TCP in the IoT environment.
- The research refining procedure is used to filter out the papers that provide the most valuable information and the solutions of RTO are documented in Table 1 to highlight the comparison.
- A comprehensive evaluation of various issues in different TCP variants for congestion control and packet loss is represented in Table 2.
- A holistic view of various issues such as header overhead, high latency issue, link layer interaction issue, multicast incompatibility issue, and their probable solutions.

The paper is structured into four different sections: section 2 highlights the related work of TCP issues in a different network and also provides a brief explanation of IoT and the significance of TCP in the IoT paradigm. Section 3 describes the several issues related to TCP and their possible solutions in IoT. Finally, section 4 summarizes our conclusions.

2. Related Work and Background

The majority of the research articles chosen are focused on the challenges and solutions of TCP. In this section, we begin with a brief overview of various issues and their probable solutions for TCP in different scenarios. Then, we present an outline of the Internet of Things and the role of TCP in the IoT environment.

Xu and Wu [45], highlighted some of the issues TCP encounters in MANET that are mainly focused on security, cross-layer design, route failure, compatibility and scalability, energy management, and simulation models. These factors provide degradation information of TCP performance in MANET. To overcome the shortcomings of an existing model, a set of robust solutions is suggested.

Leung and Li [23], presented a comprehensive review and recent advancements in TCP for wireless communications. This research provides the challenges of TCP and different typical algorithms that maintain end-to-end semantics are analyzed.

Molia and Kothari [27], focused on TCP variants and MANET losses. The primary purpose of this work is to identify problems with TCP in MANETs and provide future research possibilities for enhancing this protocol.

Goswami and Sultanah [16], reviewed the cross-layer techniques to enhance the TCP performance in wireless ad hoc networks. This research highlighted TCP challenges such as link failure and channel congestion.

Al-Jubari *et al*. [2], addressed the challenges of TCP in multi-hop wireless networks. When TCP is used in multi-hop wireless networks then it cannot handle route failure and wireless problems.

2.1. Internet of Things

In recent years, we have seen the idea of IoT arising, which is an entirely new computing paradigm. This paradigm uses a wide variety of smart devices, so that they can be easily accessed and operated over the Internet either directly by the people who use them (with an operation panel) or by automated programs that contain their behaviors and aims.

This paradigm is trying to shape tomorrow's society and substantially improve urban living through such innovations as smart cities, healthcare, and

transportation. IoT has dramatically transformed the way in which individuals and companies communicate with the physical world to a significant extent. In the last decade, IoT has drastically evolved [31].

As Vermesan *et al*. [44], "IoT is known as a collaboration among the physical and digital worlds. The digital and physical worlds work together using a variety of sensors and actuators. These abilities are used to check the object's state and to change its state if possible. IoT devices are designed with integrated sensors, processors, actuators, and transceivers for this intelligence and interconnection. IoT agglomeration of various technologies that work together."

2.2. TCP in IoT

IoT is the interconnected network platform that brings different objects together, providing them the capacity to exchange information across the network. Smart devices can be sensor nodes used in monitoring everything from the home to industry to healthcare to the environment. According to the Ericsson survey and suggestion, the number of devices that were connected by IoT technologies saw a significant increase in 2021, and it is anticipated that this number will reach nearly 500 million at the end of 2022 [12]. If the number of IoT devices on the Internet grows, more research attention is needed to address future challenges in IoT domain. Smart transport systems, smart homes, industrial monitoring systems, health monitoring, environmental monitoring, smart cities, etc., are the main IoT applications. In order to deploy the IoT, a number of protocols are designed to monitor communication over the Internet [43]. While designing an IoT system, many application protocols may be utilized to enable the exchange of data between several devices. Extensible Messaging and Presence Protocol (XMPP) [38], Message Queue Telemetry Transport (MQTT) [29], and Advanced Message Queuing Protocol (AMQP) [41] are Internet of Things application protocols that employ TCP as the transport layer protocol for data delivery. Yet, another IoT application protocol known as CoAP [40] was originally developed to operate over User Datagram Protocol (UDP) and provide fast communication among devices. Figure 1 illustrates the IoT application protocols that are used to transmit data. It demonstrates that TCP is a fundamental aspect of the IoT application protocol.

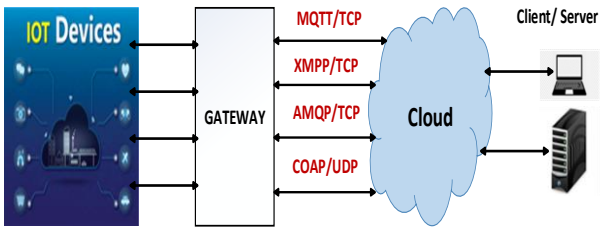

Figure 1. IoT application protocols.

3. Issues and Possible Solutions of TCP in IoT

TCP usage in the IoT environment is expected to grow significantly. Moreover, a properly configured TCP can resolve or alleviate some of the possible issues with TCP in the context of IoT. For these reasons, a standard intended to provide basic metrics on lightweight and appropriate TCP operation in IoT scenarios is being created in the IETF LWIG, with the partnership of IETF CoRE and TCPM WGs [15]. The several issues related to TCP and their possible solutions have been presented in the form of different scenarios as given below:

3.1. RTO Algorithm Issue

One of the main problems facing TCP is that it does not adapt the RTO to unexpected huge delays. Many times the normal RTT, i.e., RTO is achieved before the unexpectedly significant delay occurs, resulting in an early timeout and early retransmissions [33]. These delays can be induced by wireless data link faults such as shadowing, fading, noise, and interference. El-Bazzal *et al*. [11] suggested an end-to-end improvement to the TCP by addressing abrupt huge delays in wireless networks induced by route loss and shadowing. They have done so by dynamically expanding RTO using a variety of approaches in different ways so that early timeouts and early retransmissions caused by these errors are avoided.

The Congestion Control (CC) strategies used in CoAP is insensitive to network condition [32]. It doesn't adapt to the network dynamics. The other improved CC mechanism is CoCoA. It has been developed to provide CC that is adaptive to network dynamics and appropriate for IoT environments. It comprises adaptive RTO calculation, RTO aging mechanism, and Variable Backoff Factor (VBF) to optimize performance. The RTO estimators used in CoCoA are strong and weak. A strong RTO estimator uses only strong Round-Trip Time (RTT) and a weak RTO estimator uses weak RTT. RTT estimations are taken from packets that have received an ACK before the sender is out of retransmission. It is called a strong RTO estimator. RTT estimations are taken from packets that have needed at most two retransmissions. It is called a weak RTO estimator. The $RTO_{overall}$ values are based on strong and weak RTT and RTO. The CoCoA performs better than the CoAP CC mechanism but there is an ambiguity in the RTT_{weak} estimator calculation [42].

Betzler *et al*. [5] offered an alternative solution to the problems encountered when CoCoA is employed as the CC mechanism. The CoCoA+ are presented to resolve the several drawbacks of CoCoA. Due to the ambiguity of RTT_{weak} estimator in CoCoA, there is effects on the $RTO_{overall}$ estimation. CoCoA+ intends to

dampen the effect of RTT_{weak} estimator in the estimation of $RTO_{overall}$ by decreasing the value of 'M' from 4 to 1.

$$RTO_X = RTT_X + M_X * RTTVAR_X \quad (1)$$

In addition, the value of the RTO_{weak} estimator is restricted in the $RTO_{overall}$ calculation.

$$RTO_{overall} = 0.25 * RTO_{weak} + 0.75 * RTO_{overall}$$

Moreover, the CoCoA+limits calculations to first transmission and retransmission only for weak RTT measurements. These solutions help in preventing significant increases in total RTO values. In a burst traffic scenario, CoCoA+ is still not able to choose the proper value of RTO [6].

The new CC technique proposed by Ouakasse takes into account both the RTT and Packet Loss Ratio (PLR) [30]. It provides two ways to configure the loss ratio, based on packet loss rate, and changes the RTO accordingly. Each transmission includes an updated RTO value depending on the PLR, and no aging mechanism is required. An unnecessary calculation of the RTO value adds excessive overhead and may delay transmission.

Context-Aware Congestion Control (CACC) is suggested to handle the issue of identifying packet loss due to Bit Error Rate (BER) or congestion [1]. It considers dynamic network conditions to detect the appropriate RTT of retransmitted message ACK. It utilizes strong, weak, and failed RTT estimators to find the status of the network and offer more adaptable CC. However, the main limitation of CACC is the absence of an aging mechanism for poor RTO, which results in a sudden RTO rise.

Fast-Slow RTO (FASOR) solves high link error rates and performs well in the case of bufferbloat [22]. It divides the RTO computation into two groups. For deep bufferbloat and high congestion, slow RTO calculation is employed while fast RTO computation is utilized to compute unambiguous RTT samples. This reduces unnecessary delays and also helps in the case of link errors by minimizing flow completion. The primary shortcoming of the proposed approach is that it doesn't contain specific logic for senders remaining idle.

Bolettieri *et al*. [7] proposed pCoCoA to address the problems of CoCoA+. Two primary factors underpin the proposed mechanism:

1. A method for precisely matching requests to replies.
2. RTO's estimation algorithm has undergone several improvements. The Transmission Counter (TC) option is used in CoAP precisely for connecting requests to responses. This establishes a connection between each transmission's ACK message and its associated confirm message. Since the TC value is modified during retransmissions, it identifies unnecessary retransmissions as well. Additionally, when spurious transmissions occur, the SRTO estimator rises more rapidly because of the higher weight of RTTVAR, which aids in minimizing further spurious transmissions.

Rathod *et al*. [37] presented CoCoA++, a delay gradient-based congestion management scheme. It addresses congestion management concerns in default CoAP, CoCoA, and CoCoA+. These schemes predict network congestion using per-packet RTT measurements, but these measurements are noisy and inefficient. CoCoA++ depends on CAIA Delay Gradient for predicting network congestion by finding an RTT gradient over time and offers a Probabilistic Backoff Factor (PBF) to manage network congestion. This algorithm eliminates the need for weak and strong RTO estimations by utilizing the delay gradient. The primary disadvantage of this mechanism is that as the average packet sending rate is increased, consecutive retransmissions may take place rapidly, leading to the sender running out of retransmissions.

Table 1. Issues encountered in RTO and their proposed scheme.

Scheme Name	Issues Countered	Adopted Mechanism for RTO	Shortcomings
CoCoA [5]	RTO aging issue and Congestion control issue	VBF and RTT estimations	Complexity in weak RTT estimator
CoCoA+ [6]	Complexity in weak estimator values of CoCoA	VBF and RTO aging mechanism, Modifications of the weak estimator calculations.	In burst traffic, incorrect calculation of retransmitted RTT.
Improved Adaptive Congestion Control [30]	CC in burst traffic, RTO aging issue	RTO calculations are based on PLR.	More burden in computing RTO in every transmission and it is less adaptable.
CACC [1]	To identify the cause of packet loss due to BER or congestion.	RTO estimator, lower bound RTO restriction approach, retransmission count–based smoothed RTT observation.	Weak RTO aging process, vanish of RTTVAR variable for same successive RTT samples.
FASOR [22]	High link error rates and deep bufferbloat condition.	RTO Estimations and Self-adaptive backoff logic.	It doesn't contain specific logic for senders remaining idle.
pCoCoA [7]	Erroneous retransmissions, a diminishing RTTVAR because of similar RTT sampling.	TC, Changes in RTO Estimation and calculation of Max mean deviation of RTO.	Its main limitation is the MDEV method for calculating RTT.
CoCoA++ [37]	Limitations of default CoAP, CoCoA, CoCoA+	Delay Gradient-based estimation and PBF to manage congestion.	Due to the increased packet sending rate and fast retransmissions, the sender runs out of retransmissions.

3.2. Multicast Incompatibility Issue

There are many applications of the IoT in which one sender communicates with several receivers. Lighting controls and firmware upgrades are two examples of smart homes or smart cities. These apps take advantage of multicast's packet economy to conserve energy and bandwidth. Furthermore, in the case of the CoAP group communication, IP multicast is used [36]. Since TCP is a unicast protocol, so it is incompatible with the multicast protocol.

3.3. Protocol Complexity

TCP has frequently been recognized as a complex and challenging protocol in the IoT environment [19]. While it has evolved over time, it still adheres to the RFC 793 specification. It was first implemented in the early 1980s on machines with very restricted processor and memory capabilities. These machines are now classified as constrained devices [8].

3.4. Header Overhead

The TCP header requires at least 20 bytes, which is higher than the UDP header. Additionally, header compression in 6LoWPAN enables efficient UDP header encoding but it does not allow for efficient TCP header encoding. It's worth noting that RFC1144 for TCP header compression [20] is not appropriate for lossy links, and ROHC identifies the former issues, as too large for IoT devices. Actually, Robust Header Compression (ROHC) is one of the most complicated Internet Engineering Task Force (IETF) protocols, making it incompatible for constrained devices with 8- or 16-bit microprocessors and 10 to 50 kB of RAM [8]. The header compression in TCP was once suggested for 6LoWPAN, but it was never completed or standardized. As a result, TCP header compression is still an open issue in the case of IoT scenarios.

3.5. Long TCP Connection Is Not Feasible Due To Sleep Periods

IoT devices can be run with a limited amount of energy, such as a battery. The most energy-consuming factor in such devices is communication, specifically idle listening. To conserve energy, various IoT devices employ an Radio Duty Cycling (RDC) mechanism, in which the radio interface is stayed in an off state, and is activated only for communication under particular circumstances. “Devices can often go into sleep mode; therefore, maintaining a long-lived link in IoT applications is infeasible,” according to Shang *et al*. [39]. RDC techniques, on the other hand, facilitate packet sharing between two energy-constrained devices at the cost of increased latency and buffering requirements. We believe that it is possible to keep long TCP connections if RDC mechanisms are properly setup. RDC methods use a negligible amount of energy to maintain, and hence they enable long TCP connections.

3.6. High Latency Issue

Applications of the IoT that involve low latency include the activation of alarms and human-triggered communication between controllers and actuators. According to some researchers, the necessity for short-lived TCP connections in IoT systems may result in the establishment of a new connection each time new data needs to be sent, rising delay due to connection establishment. On the other hand, a long-lived connection, that is established once and can be used again for data transfers, minimizes delay. TCP Fast Open (TFO) is an alternative that enables data to be embedded in SYN and SYNchronize-ACKnowledgement (SYN-ACK) packets. Hence, saving one RTT over the traditional method, in which the 3-way handshake comes before data exchange [10].

3.7. Link-Layer Interaction Issue

Automatic Repeat request (ARQ) is used by a large number of IoT link-layer protocols. Using link layer ARQ might help to improve TCP performance on end-to-end routes. Link RTT is predicted to be less than end-to-end route RTT, allowing the sender to recover lost packets before sending out an additional packet. If link quality degrades, LL-ARQ mechanisms may execute retry, raising delay in some cases, and resulting in erroneous TCP retransmissions [28]. This issue does not just specific on TCP and will occur on any ARQ-based link layer, like with CoAP (over UDP).

3.8. Lack of Transport Service Flexibility

Application monitoring in IoT frequently accepts just a small percentage of faulty sensor signals. It can be used to conserve power and bandwidth by employing unacknowledged transfer. For example, CoAP (over UDP) provides NON transmission as an option. TCP ensures that all upper layer communications are acknowledged at the transport layer, preventing the application developer from adopting a less secure and inefficient solution.

3.9. Congestion and Packet Loss Issue

In the 1980s, when the majority of Internet connections were still wired, TCP congestion control was developed. Packet losses were assumed to be caused by congestion while corruption in a wired link was highly improbable. To avoid network collapse due to congestion, CC mechanisms were developed. A TCP sender decreases the segment rate whenever a packet loss is identified. On the other hand, packet losses in IoT environment can occur for several reasons other than congestion. First, most IoT connection technologies are wireless, making them vulnerable to

errors. Second, a mesh topology is used by many IoT networks. Route changes, for example, owing to node mobility or transient link quality reduction can result in connection gaps and packet loss.

TCP under non-congestion losses performs suboptimal [18]. An example of a proposed solution is Explicit Loss Notification. It involves determining the cause of packet losses, which is not always feasible. Additionally, these solutions have not been standardized nor extensively used. On the other hand, the incapability to identify the cause of packet losses is not a TCP-specific issue. Indeed, any other ARQ-based method for end-to-end reliability will initiate CC techniques in the event of a packet loss, just like TCP. Different devices in the IoT have different communication speeds, latency, and reliability specifications. CC on the internet is an open issue due to the fast increase of smart devices in IoT [17]. As the number of internet-connected devices grows, so does network congestion.

Thus, the TCP needs modifications as per the IoT specifications in order to initiate better bandwidth connections, update the transmission rate when congestion grows, and provide a constant transmission rate while network components are consistent [26]. According to the CC policies, transport-layer congestion control protocols may be classified into two types:

3.9.1. Loss Based Algorithm

A packet-loss incidence is used as an indicator of network congestion in a loss-based CC mechanism. TCP Tahoe is the loss-based CC mechanism developed by Jacobson in 1998 [21]. It adjusts the window size using the Additive Increase Multiplicative Decrease (AIMD) method. The cwnd is increased by 1 for successful packet delivery. The window is reduced to half if data loss or delay happened only when the first negative ACK is received. In the event of a timeout, it decreases the cwnd to one MSS. It modifies the cwnd size based on packet loss likelihood. The shortcoming of TCP Tahoe is that it does not prevent the communication link from going empty. Due to this, large bandwidth product links may have a high cost.

TCP Reno [4] is unlike TCP Tahoe in terms of congestion avoidance. Whenever three DACKs are received, the congestion window is halved, fast retransmission is performed, and fast recovery is entered. Reno, like TCP Tahoe, will enter a slow-start mode in the case of a timeout. TCP Reno recovers quickly from a single packet loss, but it doesn't perform too well in many packet losses in one window.

TCP New Reno is outperforming Reno in the case of multiple packet loss. It adds a new mechanism called Fast recovery [13]. It enters fast-retransmit whenever it gets numerous duplicate packets. It does not leave fast-recovery mode until all packets are acknowledged that were outstanding when they entered the fast recovery phase. It resolves Reno's issue of successive reductions in Congestion Window (CWD). The main issue with New Reno is that it takes 1 RTT to identify every packet loss.

TCP SACK is an Extension of TCP RENO that addresses the issue of numerous lost packets [14]. It preserves RENO's slow-start and fast-retransmission phases. The major issue with SACK is that the receiver does not support selective acknowledgment. It is a very difficult task to implement SACK.

TCP FACK refers to the development of TCP SACK with Forward Acknowledgement [25]. It is used in a similar manner to SACK, but with a little enhancement. It provides a more efficient method of reducing the window size by half when congestion is detected. The main issue is that it is not feasible to avoid the unnecessary inflation of the congestion window by using the delay sensing method.

3.9.2. Delay Based Algorithm

Delay-based CC mechanism utilizes packet delay as a main cause of congestion. When a delay grows, it suggests the network is experiencing congestion, and when a delay lowers, it suggests the network is less congested. Delay-based CC methods continuously monitor packet RTTs durations and respond to changes in RTT in order to prevent substantial network congestion before it occurs.

The bandwidth estimation technique of TCP Vegas is more efficient than the other delay based CC mechanisms [9]. The difference between expected and actual flow rates is used in this scheme to estimate bandwidth. It utilizes the cwnd, RTT, and RTT_{min} for bandwidth estimation. It has less packet loss and better network usage but still, it suffers from a fairness issue.

The end-to-end bandwidth estimation technique is used by TCP Westwood and it is implemented on the sender side of a TCP connection [24]. It adaptively estimates the available bandwidth and modifies the cwnd and ssthresh to maximize network utilization. Perform badly if it estimates the incorrect bandwidth due to the unpredictable behavior of the TCP Westwood bandwidth estimation algorithm.

Loss-based CC mechanism maximizes utilization of the available bandwidth but still suffers from packet loss and retransmission, whereas delay-based CC mechanism makes an attempt to minimize packet loss and retransmission but was unable to fully use the available bandwidth. They have similar issues with both inter-protocol fairness and data rate adaptation while operating at different transmission rates. TCP variants will be effective depending on the parameters to be considered. Table 2 compares the TCP variant as well as their associated problems.

Table 2. Problems found in different TCP variants for congestion control and packet loss.

Variants of TCP protocol → / Parameter Used↓	Tahoe	Reno	SACK	New Reno	FACK	Westwood	Vegas
Congestion Control Algorithm	Loss–based CC mechanism	Loss-based CC mechanism	Loss-based CC mechanism	Loss-based CC mechanism	Loss-based CC mechanism	Delay-based CC mechanism	Delay-based CC mechanism
Slow Start process	√	√	√	√	√	√	Modified Slow Start process
Fast Retransmit process	√	√	√	√	√	√	√
Fast Recovery process	None	√	Modified version	Slight Modification over Reno	Modified version	√	√
Congestion Avoidance(CA)	√	√	√	√	√	√	Enhanced CA technique
Retransmission Method	Simple retransmission method	Simple retransmission method	Simple retransmission method	Simple retransmission method	Simple retransmission method	Simple retransmission method	New Re-Transmission method
Selective Acknowledgement method	×	×	√	×	×	×	×
Forward Acknowledgement mechanism	×	×	×	×	√	×	×
ACK format required	Cumulative ACK	Immediate ACK	Immediate ACK	Immediate ACK	Immediate ACK	Immediate ACK	Immediate ACK
Packet loss detection	Single packet loss	Single packet loss	Loss of several packets	Loss of several packets	Loss of several packets	Loss of several packets	Loss of several packets
Problem	It doesn't prevent the communication link from going empty. High cost	It cannot effectively detect several packet losses.	Implementing selective acknowledgment is a very difficult task. Energy consumption is high.	It takes 1 RTT for detecting packet loss.	It's not possible to prevent the unnecessary inflation of the congestion window by using the delay sensing method.	Performs poorly if it estimates incorrect bandwidth.	It has a problem when packets don't follow the same route and when large delays are present.

4. Conclusions

The findings of this study not only gave a comprehensive knowledge of TCP and IoT but also viewpoints on various issues and the probable solutions of TCP in the IoT network have been addressed. In the IoT network paradigm, TCP has traditionally been ignored, but recent trends indicate that TCP can be widely deployed in IoT environments. The significant challenges with the TCP protocol are RTO algorithm issue, header overhead, congestion and packet loss issue, high latency, and multicast inappropriateness. The outcome of this paper is that it has elaborated several issues and their possible solutions which have been highlighted in Tables 1, and 2. It has also been concluded how well the solutions employ TCP Fast Open which has been developed for latency-based applications that are sensitive to TCP's initial connection setup delay. Additionally, the CoCoA algorithm may be used to fix the issue with the RTO Algorithm. As a result, TCP may be properly configured to behave like unicast end-to-end reliability approaches, that are widely used in the IoT.

References

[1] Akpakwu G., Hancke G., and Abu-Mahfouz A., "CACC: Context-aware Congestion Control Approach for Lightweight Coap/UDP-Based Internet of Things Traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, pp. 1-19, 2020.

[2] Al-Jubari A., Othman M., Ali B., and Abdul Hamid N., "TCP Performance in Multi-Hop Wireless ad Hoc Networks: Challenges and Solution," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1-25, 2011.

[3] Al-Khatib W. and Gunavathi K., "A Novel Mechanism to Improve Performance of TCP Protocol Over Asymmetric Networks," *The International Arab Journal of Information Technology*, vol. 5, no. 1, pp. 66-74, 2008.

[4] Allman M., Paxson V., and Stevens W., "RFC 5681: TCP Congestion Control, 2009.

[5] Betzler A., Gomez C., Demirkol L., and Paradells J., "Coap Congestion Control for the Internet of Things," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 154-160, 2016.

[6] Betzler A., Gomez C., Demirkol L., and Paradells "Cocoa+: an Advanced Congestion Control Mechanism for Coap," *Ad Hoc Networks*, vol. 33, pp. 126-139, 2015.

[7] Bolettieri S., Tanganelli G., Vallati C., and Mingozzi E., "Pcocoa: A Precise Congestion Control Algorithm for Coap," *Ad Hoc Networks*, vol. 80, pp. 116-129, 2018.

[8] Bormann C., Ersue M., and Keranen A., "Terminology for Constrained-Node Networks," RFC 7228, 2014.

[9] Brakmo L. and Peterson L., "TCP Vegas: End-to-End Congestion Avoidance on A Global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, 1465-14, 1995.

[10] Cheng Y., Chu J., Radhakrishnan S., and Jain A.,

"Tcp Fast Open," *RFC 7413*, 2014.

[11] El-Bazzal Z., Ahmad A., Houssini M., El Bitar I., and Rahal Z., "Improving the Performance of TCP Over Wireless Networks," *in Proceeding of 6th International Conference on Digital Information, Networking, and Wireless Communications*, Beirut, pp. 12-17, 2018.

[12] Ericcson., "Ericsson Mobility Report-November 2022," https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook, Last Visited, 2020.

[13] Floyd S., Henderson T., and Gurtov A., "The Newreno Modification to TCP's Fast Recovery Algorithm," *Internet Request for Comments*, pp. 2582,1999.

[14] Floyd S., Mahdavi J., Mathis M., and Romanow A., "TCP Selective Acknowledgment Options," *No. rfc2018*, 1996.

[15] Gomez C., Moret A., and Crowcroft J., "TCP in the Internet of Things: from Ostracism to Prominence," *IEEE Internet Computing*, vol. 22, no. 1, pp. 29-41, 2018.

[16] Goswami C. and Sultanah P., "A Study on Cross-Layer TCP Performance in Wireless Ad Hoc Network," *in proceeding of International Conference on Intelligent Data Communication Technologies and Internet of Things*, Coimbatore, pp. 56-70, 2018.

[17] Hussain S. and Parween S., "Comparative Study of TCP Congestion Control Algorithm in IoT," *in Proceeding of 3rd International Conference on Advances in Computing, Communication Control and Networking*, Greater Noida, pp. 1428-1431, 2022.

[18] Inamura H., Montenegro G., Ludwig R., Gurtov A., and Khafizov F.,"TCP Over Second (2.5 G) and Third (3G) Generation Wireless Networks," Rfc3481, 2003.

[19] Jacobson V., Braden R., and Borman D., "TCP Extensions for High Performance," *Internet Request for Comments*, vol. 1323, pp. 1-37, 1992.

[20] Jacobson V., "Congestion Avoidance and Control," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 14-3, 1988.

[21] Jacobson V., "Compressing TCP/IP Headers for Low-Speed Serial Links," No. rfc1144, 1990.

[22] Jarvinen I., Raitahila L., Cao Z., and Kojo M., "FASOR Retransmission Timeout and Congestion Control Mechanism for CoAP," *IEEE Global Communications* Conference, Abu Dhabi, pp. 1-7, 2018.

[23] Leung K. and Li V., "Transmission Control Protocol (TCP) in Wireless Networks: Issues, Approaches, and Challenges," *IEEE Communications Surveys and Tutorials*, vol. 8, no. 4, pp. 64-79, 2006.

[24] Mascolo S., Gerla M., Sanadidi M., Casetti C., and Wang R., "TCP Westwood: End-to-end Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks," *Wireless Networks*, vol. 8, no. 5091294, pp. 467-479, 2002.

[25] Mathis M. and Mahdavi J., "Forward Acknowledgement: Refining TCP Congestion Control," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 4, pp. 281-29, 1996.

[26] Mishra N., Verma L., Srivastava P., and Gupta A., "An Analysis of IoT Congestion Control Policies," *Procedia Computer Science*, vol. 132, pp. 444-450, 2018.

[27] Molia H. and Kothari A., "TCP Variants for Mobile Adhoc Networks: Challenges and Solutions," *Wireless Personal Communications*, vol. 100, no. 4, pp. 1791-1836, 2018.

[28] Montenegro G., Grossman D., Touch J., Mahdavi J., Bormann C., Karn P., Reiner L., Fairhurst G., and Wood L., "Advice for Internet Subnetwork Designers," No. rfc3819, 2004.

[29] OASIS, "Information Technology--Message Queuing Telemetry Transport (MQTT) v3.1.1," ISO/IEC 20922:2016; *International Organization for Standardization (ISO)*, 2016.

[30] Ouakasse F. and Rakrak S., "An Improved Adaptive Coap Congestion Control Algorithm," *International Journal of Online and Biomedical Engineering*, vol. 15, no. 3, pp. 96-109, 2019.

[31] Parween S., Hussain S., and Hussain M., "A Survey on Issues and Possible Solutions of Cross-Layer Design in Internet of Things," *International Journal of Computer Networks and Applications*, vol. 8, no. 4, pp. 311-333, 2021.

[32] Parween S. and Hussain S., "A Comparative Analysis Of Coap Based Congestion Control In Iot," *in Proceeding of 4th International Conference on Recent Trends in Computer Science and Technology*, Jamshedpur, pp. 321-324, 2021.

[33] Parween S. and Hussain S., "Cross-Layer based TCP Performance Enhancement in IoT Networks," *International Journal of Advanced Computer Science and* Applications, vol. 13, no. 6, pp. 3830-396, 2022.

[34] Paxson V. and Allman M., "RFC 2988: Computing TCP's Retransmission Timer Internet RFC," 2000.

[35] Postel J., "Transmission Control Protocol," IETF RFC 793, 1981.

[36] Rahman A. and Dijk E., "Group Communication for the Constrained Application Protocol (CoAP)," 2014.

[37] Rathod V., Jeppu N., Sastry S., Singala S., and Tahiliani M., "Cocoa++: Delay Gradient Based Congestion Control for Internet of Things," *Future Generation Computer Systems*, vol. 100,

pp. 1053-1072, 2019.
[38] Saint-Andre P., "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 6120, 2011.
[39] Shang W., Yu Y., Droms R., and Zhang L., "Challenges in IoT Networking via TCP/IP Architecture," NDN Technical Report NDN-0038, 2016.
[40] Shelby Z., Hartke K., and Bormann C., "Constrained Application Protocol (CoAP)," Orgiddraftietfcorecoap01 Txt 0807, 2010.
[41] Standard O., "Oasis Advanced Message Queuing Protocol (Amqp) Version 1.0," *International Journal of Aerospace Engineering Hindawi*, 2018.
[42] Tariq M., Khan M., Khan M., and Kim D., "Enhancements and Challenges in Coap—A Survey," *Sensors*, vol. 20, no. 21, pp. 1-29, 2020.
[43] Verma L. and Kumar M., "An IoT based Congestion Control Algorithm," *Internet of Things*, vol. 9, pp. 100157, 2020.
[44] Vermesan O., Friess P., Guillemin P., et al. "Internet of Things Strategic Research Roadmap," *Internet of Things-Global Technological and Societal Trends*, 2011.
[45] Xu W. and Wu T., "TCP Issues in Mobile ad Hoc Networks: Challenges and Solutions," *Journal of Computer Science and Technology.*, vol. 21, no. 1, p. pp. 72-8, 2006.

Syed Zeeshan Hussain is presently working as a Professor in the Department of Computer Science, Jamia Millia Islamia (A Central University), New Delhi (India). He has an experience of over 20 years of Teaching and research in the area of Computer Science and its Applications. He has supervised twelve Ph.D. scholars so far, out of which seven scholars have already been awarded Ph.D. Degree. A total number of more than 50 research papers have been published by him in the proceedings of National/International Conferences and refereed Journals. His area of research is Computer Networks and Security.

Sultana Parween is a Research scholar in the Department of Computer Science, Jamia Millia Islamia (A Central University), New Delhi, India. Her area of Research is Computer Networks, Internet of Things, and Wireless Sensor Networks.