Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Adaptive congestion control in IoT networks: Leveraging one-way delay for enhanced performance

Lal Pratap Verma [a], Gyanendra Kumar [b,*], Osamah Ibrahim Khalaf [c], Wing-Keung Wong [d], Abdulsattar Abdullah Hamad [e], Sur Singh Rawat [f]

[a] *Department of Computer and Communication Engineering, Manipal University Jaipur, Jaipur, 303007, India*
[b] *Department of IoT and Intelligent Systems, Manipal University Jaipur, Jaipur, 303007, India*
[c] *Department of Solar, Al-Nahrain Research Center for Renewable Energy, Al-Nahrain University, Jadriya, Baghdad, Iraq*
[d] *Asia University, Taiwan*
[e] *University of Samarra, College of Education, Department of Physics, Iraq*
[f] *Department of Computer Science and Engineering, J.S.S. Academy of Technical Education, Noida, 201301, India*

A R T I C L E I N F O

Keywords:
Congestion control
IoT
TCP
Transport layer
Congestion window

A B S T R A C T

With the exploding number of IoT devices generating vast data volumes, there is a growing risk of significant performance degradation without efficient congestion management. To tackle this challenge, efficient regulation and supervision are essential for managing congestion in IoT networks. This research work introduces a One-Way-Delay (OWD)-based congestion control (CC) method that estimates data transmission delays of the communication path and adjusts network traffic accordingly. The proposed method enhances IoT device performance by continuously monitoring OWD along the transmission path to identify and mitigate congestion. Comparative analysis with existing methods demonstrates that the proposed approach more effectively utilizes network resources, reduces congestion, and improves throughput while ensuring fairness and reliability within the IoT infrastructure. The experimental simulations show that the proposed OWD-based method outperforms well-established TCP variants such as BBR, TCP Cubic, HTCP, and New Reno, achieving average throughput improvements ranging from 4.1 % to 22.7 %. The proposed method also maintains fairness in mixed-traffic environments and effectively manages congestion in complex network topologies.

1. Introduction

The Internet of Things (IoT) is a rapidly expanding network of interconnected devices that communicate via the Internet, often equipped with sensors and actuators for data collection, remote monitoring, and control [1]. IoT technology is widely applied across various industries, including healthcare, agriculture, automotive manufacturing, home automation, and smart cities, to enhance efficiency, productivity, and decision-making through data analysis. According to the latest IoT analytics report [2], there were 19.2 billion connected IoT devices globally in 2024, and this number is expected to reach 29.7 billion by 2027 (Fig. 1). This surge in

* Corresponding author.
E-mail addresses: er.lpverma1986@gmail.com, lalpratap.verma@jaipur.manipal.edu (L.P. Verma), maurya.gyanendra@gmail.com, gyanendra.kumar@jaipur.manipal.edu (G. Kumar), usama81818@nahrainuniv.edu.iq (O.I. Khalaf), wong@asia.edu.tw (W.-K. Wong), a.a.hamad@uosamarra.edu.iq (A.A. Hamad), sur.rawat@jssaten.ac.in (S.S. Rawat).

https://doi.org/10.1016/j.heliyon.2024.e40266
Received 13 July 2024; Received in revised form 31 October 2024; Accepted 7 November 2024
Available online 9 November 2024

connected devices has resulted in a substantial increase in data traffic across communication network, both wired and wireless, necessitating effective congestion management strategies to maintain performance and reliability.

In communication networks, the Transport Layer protocol, Transmission Control Protocol (TCP) [3] is responsible for reliable data transmission between nodes. TCP provides an end-to-end, connection-oriented service over IP networks. In 1981, Jacobson [3] proposed mechanisms such as slow start, congestion avoidance, fast retransmission, and recovery to manage congestion during data transmission. These innovations significantly enhanced TCP's congestion control, reducing data packet loss and improving network bandwidth utilization, setting a benchmark for future TCP versions.

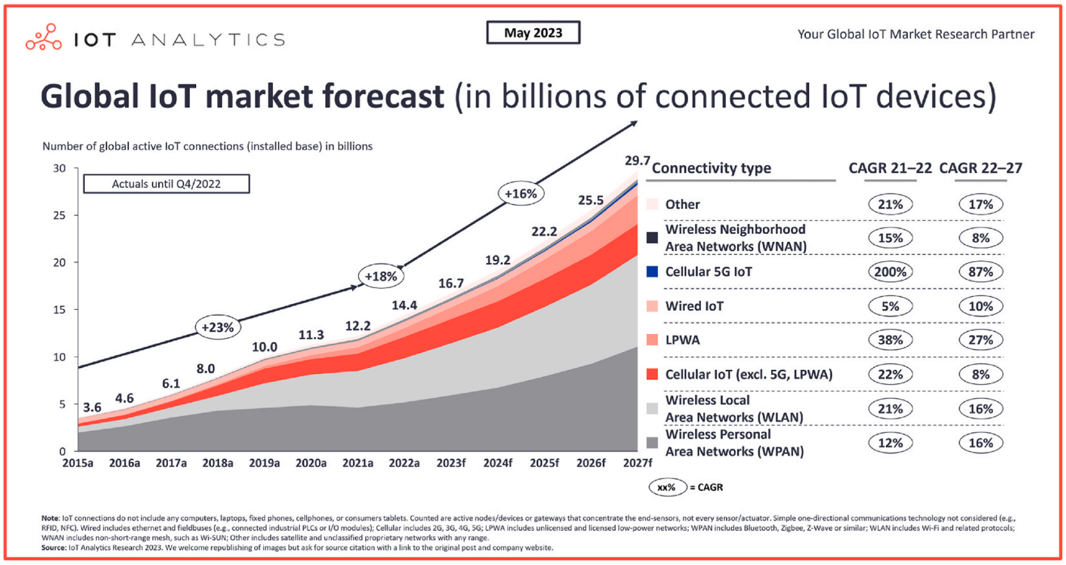
Cubic [4] and Compound [5] are the most widely deployed versions of TCP over the Internet and IoT networks [43–46]. Cubic, globally adopted and used in Linux and Android operating systems, is a congestion control approach designed to enhance TCP performance in high-bandwidth networks. It utilizes a cubic function to regulate the data transmission rate based on network conditions. However, Cubic's aggressive nature limits its performance in IoT networks, where most devices communicate through low bandwidth channels. Compound, developed by Microsoft and used in Windows Vista and later versions, aims to improve TCP performance in high-speed and high-delay networks. It adapts the transmission rate to the current congestion window (cwnd) and smooth round-trip time (RTT). While the Compound performs well in high-speed and long-delay networks, it requires further enhancements for optimal performance in IoT networks due to its reaction time.

Therefore, we analyzed the performance of Cubic [4] and Compound [5] to identify the positive and negative aspects of both protocols. The analysis was performed using NS-2.35 to simulate both protocols in a dumbbell topology. In this simulation setup, the bottleneck link bandwidth was set at 1.5 Mbps, the queue size at 50 packets, and the simulation time at 100 s. This topology included one TCP and two UDP connections. One UDP flow (200 Kbps) ran parallel to TCP, while the other UDP flow (1000 Kbps) was in the reverse direction of TCP. We compared the TCP variants' throughput and packet loss ratio with RTT varying from 150 ms to 400 ms. The choice of a 50-packet queue size and a bottleneck bandwidth of 1.5 Mbps was made based on empirical data from related work and the typical characteristics of constrained IoT environments. Many IoT networks, particularly in resource-constrained deployments, tend to have low-bandwidth links and limited buffer sizes, making these parameters representative of real-world conditions.

Fig. 2 illustrates the goodput of both TCP variants. TCP Cubic's goodput increases (to some extent) as RTT increases, while Compound TCP's goodput decreases with increasing RTT. The behavior of both TCP variants is opposite to each other during the simulation. Compound TCP achieves better goodput until RTT reaches 250 ms, after which TCP Cubic outperforms Compound TCP. Thus, Compound TCP is suitable for low-delay networks, whereas TCP Cubic is better suited for high-delay networks. Fig. 3 depicts the packet loss ratio for TCP Cubic and Compound TCP. TCP Cubic experiences more packet losses than Compound TCP. As RTT increases, the packet loss in TCP Cubic decreases, whereas Compound TCP consistently exhibits lower packet loss compared to TCP Cubic.

1.1. Motivation

Traditionally, TCP has been widely deployed over the Internet to provide reliable data transfer between devices. However, concerns have been raised about the suitability of TCP for IoT networks due to the unique characteristics and constraints of IoT devices [6]. Studies by Gomez et al. [6] (2018), Jain et al. [7] (2022), and Anitha et al. [46] (2023) on IoT congestion control have highlighted that network service quality suffers due to factors such as network lifespan, connection quality, control overhead, end-to-end delay, and the presence of heterogeneous devices in the network. As IoT is envisioned as a crucial part of the future internet, spanning various



Connectivity type	CAGR 21–22	CAGR 22–27
Other	21%	17%
Wireless Neighborhood Area Networks (WNAN)	15%	8%
Cellular 5G IoT	200%	87%
Wired IoT	5%	10%
LPWA	38%	27%
Cellular IoT (excl. 5G, LPWA)	22%	8%
Wireless Local Area Networks (WLAN)	21%	16%
Wireless Personal Area Networks (WPAN)	12%	16%

Note: IoT connections do not include any computers, laptops, fixed phones, cellphones, or consumers tablets. Counted are active nodes/devices or gateways that concentrate the end-sensors, not every sensor/actuator. Simple one-directional communications technology not considered (e.g., RFID, NFC). Wired includes ethernet and fieldbuses (e.g., connected industrial PLCs or I/O modules); Cellular includes 2G, 3G, 4G, 5G; LPWA includes unlicensed and licensed low-power networks; WPAN includes Bluetooth, Zigbee, Z-Wave or similar; WLAN includes Wi-Fi and related protocols; WNAN includes non-short-range mesh, such as Wi-SUN; Other includes satellite and unclassified proprietary networks with any range.
Source: IoT Analytics Research 2023. We welcome republishing of images but ask for source citation with a link to the original post and company website.

Fig. 1. IoT Analytics forecast of connected IoT devices (Source IoT Analytics [2]).

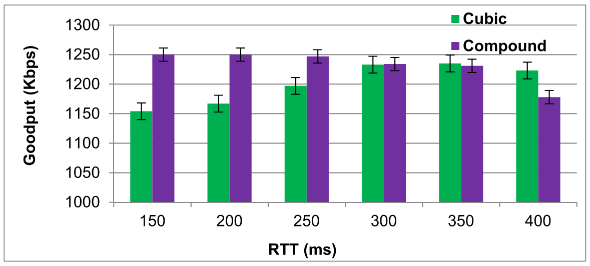


Fig. 2. RTT Vs Goodput: Under dumbbell topology with bottleneck link of 1.5 Mbps, queue size 50 packets, forward and backward UDP 200Kbps and 1000Kbps respectively, CBR, RTT varies from 150 to 400 ms, simulation time 100 ms.

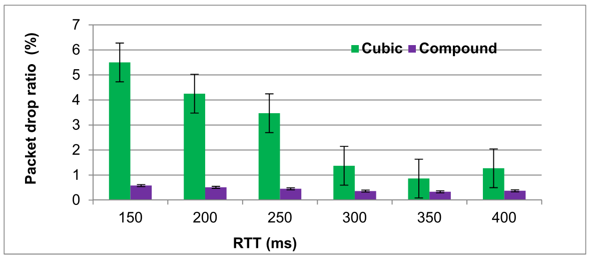


Fig. 3. RTT vs. packet drop ratio Under dumbbell topology with bottleneck link of 1.5 Mbps, queue size 50 packets, forward and backward UDP 200Kbps and 1000Kbps respectively, CBR, RTT varies from 150 to 400 ms, simulation time 100 ms.

domains, resolving congestion issues is imperative to facilitate resource sharing and performance, fostering IoT development in the future. These studies also argue that TCP will continue to play a significant role in the growth of IoT due to its congestion control, reliability, and end-to-end semantics, which are important for various IoT applications.

The large number of IoT devices will generate enormous amounts of data, necessitating a suitable congestion control approach to regulate internet traffic. Several innovative approaches have been proposed to address this issue in recent years. For instance, deep reinforcement learning-based congestion control algorithms have shown promise in optimizing transmission rates and improving network stability under varying traffic conditions [47,48]. Similarly, fuzzy logic-based methods have been employed to provide energy-efficient solutions for congestion management in resource-constrained environments like WSNs [13,49,50]. Hybrid approaches that combine traditional TCP mechanisms with adaptive algorithms have also been explored to enhance throughput and fairness [18–21]. Despite these advancements, many of the existing solutions struggle with the unique challenges posed by IoT networks, particularly in terms of handling asymmetric delays and maintaining fairness in mixed-traffic environments.

The following points motivated us to design a new congestion control approach for IoT networks.

- ✓ Real-Time Monitoring and Control: IoT applications, such as smart city traffic management and healthcare, rely on minimizing forward delay to ensure timely updates and real-time responses. For instance, in traffic management, forward delay-based congestion control helps maintain the flow of critical information, while in healthcare IoT, it ensures the real-time transmission of vital signs for immediate intervention [1,6].
- ✓ Asymmetric Networks in IoT: IoT devices in remote areas often operate in asymmetric network environments, such as satellite or long-range wireless networks. For example, in rural monitoring applications, forward delay-based congestion control optimizes data transmission from sensors to servers, ensuring timely delivery even with slower return paths [7,46].
- ✓ Traffic Engineering for IoT: Efficient traffic management is critical in IoT networks. In Industrial IoT (IIoT), congestion control based on forward delay ensures time-sensitive data is prioritized, maintaining process stability. Similarly, during firmware updates or data delivery to large IoT networks, forward delay management prevents congestion and ensures efficient transmission [46].

On the other hand, most delay-based congestion control approaches use RTT as a delay indicator. When the source computes the RTT, it includes the time a packet takes to reach the destination and the time for the acknowledgment (ACK) to return to the source. Our communication system utilizes separate channels for forward and backward traffic to implement full-duplex communication. Therefore, these channels operate independently, resulting in potentially different delays for each channel.

1.2. Contributions

The key contributions of this paper are as follows.

- ➢ It leverages a one-way delay estimator at the receiver end and includes this information with acknowledgments sent to the sender.

➢ It introduces an OWD Analyzer system to measure and analyze OWD in the network for accurately predicting the traffic intensity of the path.
➢ It also proposed an adaptive congestion control mechanism to adjust the transmission rate based on current traffic intensity.
➢ An experimental evaluation of the proposed approach is performed, and results indicate improvement in the performance regarding device lifetime, throughput, and fairness.

These contributions collectively advance the understanding and implementation of congestion control mechanisms tailored specifically for IoT networks, offering enhancements in performance and network management.

The rest of the paper is organized as follows: Section 2 describes the existing literature on IoT and TCP. In section 3, we proposed a new congestion control approach that utilizes a one-way path delay to adjust the data transfer rate. Section 4 presents the experimental analysis of the proposed approach compared with the existing approach. Section 5 concludes the overall performance of the proposed approach.

2. Related work

This section reviews existing research on CC mechanisms for IoT networks, with a focus on TCP variants, to assess their suitability and identify gaps for our proposed Delay-based Adaptive CC Algorithm.

2.1. Related survey on congestion control

Due to limitations in smart devices, congestion is a significant issue in IoT networks, leading to data loss and performance degradation. The authors of [7] review end-to-end and hop-by-hop CC strategies for reliable communications, emphasizing rate optimization and traffic engineering within the IoT network stack. Another survey work [8] provides an overview of CC algorithms in the transport layer, highlighting the importance of CC for critical IoT applications and addressing current difficulties in adjusting to network conditions.

Machine learning applications in broad-sensing Internet CC and avoidance are analyzed in Ref. [9], discussing specific procedures and technologies for network state acquisition. The authors of [10] focus on improving TCP CC to accommodate emerging technologies, using delay signals (RTT or one-way delay) to detect congestion early. Apart from these, the work of [11] examines CC mechanisms in multipath environments, identifying research gaps and suggesting future directions.

2.2. TCP variants for IoT

TCP variants have been specifically developed to address the distinct challenges faced by IoT networks, including resource constraints, intermittent connectivity, and the need for efficient data transmission. For instance, LLN TCP is designed for low-power, lossy networks, employing trickle timers and adaptive retransmission to manage packet loss effectively [12]. However, it may struggle with rapidly changing network conditions, potentially leading to delays in data transmission. TCP-WSN incorporates energy-efficient algorithms tailored for Wireless Sensor Networks, addressing the limited resources of sensor nodes [13]. Despite this, it can face difficulties in managing diverse application requirements and varying traffic loads. Similarly, TCP-IoT optimizes the protocol for managing short data packets in low-bandwidth environments, enhancing communication efficiency [14]; yet it may not handle larger packet sizes well, leading to potential bottlenecks. The CoAP-TCP variation minimizes overhead and resource usage for efficient interactions with the Constrained Application Protocol [15], but its effectiveness can be hindered in highly congested networks. Furthermore, Compound TCP for IoT is adapted to manage low bandwidth and variable latency, enhancing performance in fluctuating network conditions [16]; however, it may still encounter challenges with fairness and throughput during extreme congestion scenarios. Lastly, TCP Cubic with Optimized Small Packets includes enhancements for handling small data packets in resource-constrained devices [17], but it may be sensitive to specific network conditions, impacting its overall performance. Despite these advancements, issues such as ongoing packet loss, latency variations, and the inherent complexity of these protocols remain, necessitating continuous research and refinement to ensure optimal performance in diverse IoT scenarios.

2.2.1. IoT-specific CC mechanisms

Several congestion control methods aim to reduce network congestion, but challenges remain. The Effective Loss-Based Scheme adjusts the congestion window based on packet loss, but it may still react too aggressively to transient losses, leading to bandwidth underutilization [19]. DDNN-PSO, designed for intelligent transportation systems, improves delivery ratios and reduces latency, but it still faces scalability issues and may not be well-suited for highly dynamic environments with frequent topology changes [20]. EWT-IoT enhances TCP performance without requiring AQM changes, but fairness and throughput may degrade in dense IoT environments where many flows compete for limited resources [21].

2.2.2. Delay-based CC approaches

Verma et al. proposed a delay-based adaptive method named DACC which adjusts transmission rates based on queuing latency but can misinterpret high latency caused by other factors as congestion, leading to unnecessary reductions in transmission rates [23]. In another work, Su et al. [24] outlined DVPTCP, which improves bandwidth utilization in high-latency networks, but its reliance on RTT can lead to inaccurate congestion detection in asymmetric networks, causing inefficiencies in traffic management. DA-BBR addresses fairness concerns in BBR, but it may still experience performance degradation in high-latency or high-loss environments due to its reliance on RTT-based control [25]. Modular Data CC reduces packet loss in enterprise cloud systems but can face difficulties in complex, multi-tenant environments where network conditions change rapidly [26]. CoCoA++ improves congestion detection in low-power IoT devices, but it may still suffer from slow convergence times in highly variable network conditions, affecting the timeliness of congestion control [27].

2.2.3. *Adaptive CC for resource-constrained devices*

In the work proposed by Xu et al. [28], the adaptive approach PHAQM, which uses MPC theory and Hebbian learning, outperforms traditional AQM methods but requires significant computational resources, making them less suitable for highly constrained devices. Adaptive Aggregation Strategies reduce packet and header sizes, improving transmission efficiency, but these methods can still lead to excessive data abstraction, impacting the granularity of information sent across the network [29]. QTCP, which uses Q-learning to manage congestion, improves throughput and reduces delays, but it still faces challenges in environments with highly unpredictable traffic patterns [30]. Lastly, LACC improves vehicle-to-vehicle communication by optimizing message delivery. Still, it may struggle in large-scale vehicular networks with high-density traffic, where maintaining low latency becomes increasingly difficult [31].

2.2.4. *Machine learning-based CC for IoT*

Machine learning-based CC methods have emerged to optimize performance in IoT networks by leveraging predictive algorithms and adaptive strategies. Leon et al. [32] proposed a Cat Boost-Based Approach that predicts packet delivery based on the network's traffic state, which leads to improvements in both packet delivery ratio and network transit time, enhancing overall efficiency in IoT networks. In another work proposed by Xie et al. [33], adaptive Fuzzy Prescribed Time CC uses a saturation Fuzzy time prescribed function to ensure user-defined tracking performance, providing flexibility and enhancing system responsiveness under varying network conditions. DRL-AQM applies deep reinforcement learning to manage congestion adaptively, outperforming classic AQM methods, especially in complex network scenarios where traditional approaches struggle [34]. Congestion Control using Learning Automata combines learning automata with IoT environments, efficiently eliminating congestion and improving QoS in lossy networks, particularly where packet loss is prevalent [35].

In more specialized cases, OAC-TCPCC focuses on addressing long-term delays in the Internet of Distributed Smart Things communications, significantly enhancing TCP throughput and reducing file transfer times, making it suitable for distributed systems [36]. Hou et al. [47] proposed the CC method for lossy networks like 6LoWPAN, it uses a Deep-Reinforcement-Learning-Aided Loss-Tolerant mechanism that adjusts transmission rates based on network state feedback, improving packet delivery ratios and throughput in constrained environments. In another work Luo et al. [48] proposed inverse reinforcement learning with parallel training optimization-based congestion control decisions by applying inverse reinforcement learning, which increases training efficiency through parallel processing, leading to better network performance in highly dynamic scenarios. In smart grid networks, a reinforcement learning algorithm dynamically adjusts transmission rates, optimizing data flow and enhancing network stability in complex and large-scale smart grid environments [49]. Finally, the efficient fuzzy methodology for CC in wireless sensor networks uses fuzzy logic to manage congestion, improving energy efficiency while reducing computational complexity, which is crucial for resource-constrained wireless sensor networks [50].

Machine learning-based CC approaches show significant promise in addressing congestion and improving performance across diverse IoT environments. By leveraging predictive models and adaptive mechanisms, these methods enhance packet delivery, optimize throughput, and improve network responsiveness. However, challenges still exist, particularly related to the high computational demands required for real-time decision-making and adaptability. As IoT networks grow more complex and dynamic, the need to balance performance gains with resource constraints, especially in low-power, lossy networks, remains a critical area for further research and development.

2.2.5. *Cross-layer CC for IoT*

Cross-layer CC techniques utilize data from multiple layers of the communication stack to enhance network performance, but several challenges still persist. QCCP, designed for IoT in the medical industry, prioritizes critical data and reduces packet loss, delay, and power consumption. However, the method can still struggle with scalability in large networks, especially under heavy traffic loads, where prioritization alone may not be sufficient to prevent congestion [38]. Rank-Based CC adjusts transmission rates based on successful transmissions, which improves throughput in multipath environments. Yet, this method can suffer from inefficiencies in highly dynamic environments, where fluctuating network conditions may result in inaccurate transmission rate adjustments, leading to congestion or underutilization [37].

Mast et al. [39] proposed a cross-layer solution for TCP in Wireless Ad Hoc Networks, that improves throughput and fairness by reducing retransmissions. But, it faces challenges in managing the overhead of cross-layer communication, which can become complex and resource-intensive in larger, more mobile networks. In another work, Mishra et al. [40] outlined the IoV cross-layer strategy to improve network performance in the Internet of Vehicles networks by notifying the sender about buffer space and link usage. Despite these improvements, the approach can be limited by the rapid mobility and high density of vehicles in urban environments, leading to delays in communication that may not fully alleviate congestion.

While cross-layer approaches improve congestion management and resource utilization, but on the cost of complexity, overhead, and compatibility issues across different devices and protocols, limit their scalability and efficiency in larger, more heterogeneous IoT and vehicular networks.

The key variants TCP Cubic [4], New Reno [42], HTCP [41], and BBR [51] are designed to enhance performance in diverse network conditions, each addressing specific challenges inherent to traditional TCP. Cubic optimizes congestion control for high-bandwidth and long-distance networks by employing a cubic function to adjust its congestion window, which improves throughput but can be sensitive to sudden changes in network conditions, leading to potential performance degradation. An improved variant of classical Reno, New Reno, addresses issues related to packet loss and retransmission in scenarios with multiple losses, yet it may still struggle in environments with high delay and frequent packet loss, impacting overall efficiency. A more aggressive variant HTCP introduced to adjust the congestion window based on observed network conditions, effectively enhancing performance in high-bandwidth-delay product environments; however, its performance can diminish in highly dynamic or unpredictable network settings. Lastly, BBR (Bottleneck Bandwidth and Round-trip propagation time) aims to optimize throughput and minimize latency by estimating the bottleneck bandwidth and RTT, yet it has been criticized for its potential to create unfairness among competing flows and its dependency on accurate bandwidth estimation. As IoT and high-speed networks continue to evolve, future research may focus on hybrid approaches that combine the strengths of these TCP variants while addressing their limitations, such as enhancing adaptability in rapidly changing environments, improving fairness in multi-flow scenarios, and optimizing performance.

2.3. Research gap

From the literature study and presented comparative summary of existing congestion control schemes for IoT outlined in Table 1, it is evident that the advancement of congestion management is inevitable to address the requirement of resource-constrained networks like IoT and M2M communication. The study reveals the following listed research gaps that must be addressed to fulfil the current requirements.

- ✓ Real-time adaptation in highly dynamic IoT environments represents a critical research gap that necessitates focused attention to fulfil performance requirements related to device lifetime, throughput, and fairness.
- ✓ Scalability issues represent a significant challenge that must be addressed in the context of congestion management for resource-constrained networks like IoT and M2M communication.
- ✓ Despite the continuous push of standardization and interoperability in enhancing congestion management for IoT networks, there is a lack of solutions that effectively address the resource-constrained requirement within current congestion control mechanisms.
- ✓ The QoS awareness to be incorporated into congestion management strategies is crucial for effective resource utilization.
- ✓ To satisfy the changing needs of M2M and IoT communication, a unique research need that has to be explored is the dynamic adjustment of delay thresholds.

3. Proposed OWD congestion control

The key goal of designing the proposed method is to enhance the performance of IoT devices. The OWD is an important factor that plays a vital role in various IoT applications where real-time communication, control, and decision-making are essential. High delay may lead to longer RTT, potentially impacting TCP throughput and efficiency. When utilizing RTT as a congestion indicator, the source calculates the total time for a packet's journey from the source to the destination and back. This computation includes the time for the data packet to reach the destination and the time for the ACK packet to return to the source.

Consider a scenario where the backward path, responsible for transmitting acknowledgment packets, encounters higher traffic than the forward path. This congestion on the backward path results in delays or increased queuing times for acknowledgment packets. If the source relies on RTT and detects delays based on the backward path, it might interpret this delay as a sign of overall network congestion. Consequently, the source may mistakenly assume that the entire network is congested and respond by reducing its transmission rate. This reduction in transmission rate aims to mitigate perceived congestion based on the RTT measurement.

The challenge here lies in the fact that, in this particular situation, the congestion is confined to the backward path handling acknowledgment packets. The forward path, responsible for sending data packets from the source to the destination, may be relatively uncongested. Consequently, reducing the overall transmission rate based on RTT could be an unnecessary and suboptimal response, as it impacts both the forward and backward paths.

To tackle this issue, the proposed approach recommends using forward delay (One-Way Delay, OWD) as a congestion indicator. By concentrating on the delay specifically in the forward path, the source can make more precise and targeted adjustments to its transmission rate, effectively adapting to the actual congestion conditions within the network. Based on Fig. 4, the proposed approach has been structured into four distinct modules: the OWD Estimator, OWD Analyzer, Traffic Intensity Predictor, and Transmission Rate Controller. The process flow of the proposed OWD congestion control method is shown in Fig. 5. It depicts how different modules interact with each other during the transmission of data (see Table 1).

3.1. Notation and abbreviation

The description of notations and abbreviations used in the proposed method and its evaluations are listed in Table 2.

3.2. One-way delay estimator

The proposed approach first calculates the OWD at the destination and then sends it to the source using acknowledgment. Here's the representation of Algorithm 1 for OWD.

Table 1
A summary of key congestion control methods.

Method	Year of Publication	Working Approach	Type (Delay, Loss, or Hybrid)	Role in IoT	Issue if Used in IoT Network
Dynamic Cubic Function for Congestion Window Adjustments [4]	2008	Uses a cubic function to dynamically adjust the congestion window based on network conditions.	Hybrid	Enhances high-speed and long-distance networks by optimizing window sizes.	Sensitivity to rapidly changing conditions, leading to performance degradation.
Priority Queuing for IoT-Based Technologies [18]	2023	Implements priority queuing for assigning priority levels to traffic.	Delay	Improves IoT traffic management and resource allocation.	Complex configuration, and challenges in diverse IoT applications.
Loss-Based Congestion Management for IoT [19]	2022	Manages congestion by leveraging loss-based mechanisms.	Loss	Outperforms standard algorithms in congestion management.	Performance varies in dynamic and real-world IoT environments.
Particle Swarm Optimization in Wireless Sensor Networks for ITS [20]	2023	Combines PSO with a Dynamic Deep Neural Network to tackle congestion.	Hybrid	Improves congestion control in Intelligent Transportation Systems (ITS).	Limited performance beyond experimental setups; context-dependent.
Early Window Tailoring for TCP in Home Networks [21]	2021	Introduces early window tailoring to enhance TCP performance without modifications.	Delay	Reduces transfer time, improves fairness, and increases goodput in home IoT networks.	Lacks explicit details on challenges or limitations in implementation.
Lightweight TCP for IoT (lwIP) [22]	2022	Enhances lwIP TCP by addressing constraints and scalability in IoT networks.	Hybrid	Improved performance metrics such as RTOs and congestion behavior in IoT.	Lacks real-world testing to identify potential limitations.
Queuing Delay Variation-Based Adaptive TCP [23]	2020	Adapts transmission rates based on queuing delay variation.	Delay	Reduces packet loss and retransmissions, enhances fairness and goodput.	May perform poorly in rapidly changing or unpredictable environments.
RTT-Based Dynamic Adjustment of Virtual Parallel Streams [24]	2019	Adjusts parallel streams based on RTT to optimize performance.	Delay	Enhances bandwidth utilization and TCP friendliness.	Limited efficacy in heterogeneous, dynamically changing IoT networks.
DA-BBR (Delay-Aware BBR) [25]	2021	Addresses fairness in Google's BBR by considering RTT.	Delay	Reduces persistent queues and improves fairness indices.	Requires more diverse evaluations to ensure broader applicability.
Delay-Tolerant Data Congestion Avoidance (Modular Computing) [26]	2018	Uses modular computing for delay-tolerant congestion avoidance.	Delay	Reduces packet loss in large enterprise cloud systems.	Efficiency in processing needs improvement.
CoAP Congestion Control (Delay Gradients and Probabilistic Backoff) [27]	2019	Utilizes delay gradients and probabilistic backoff.	Delay	Provides a more accurate congestion measure and improves packet-sending rates.	Complex implementation, unnecessary congestion window reduction.
MPC-Based Adaptive AQM with Hebb Learning [28]	2020	Applies MPC theory with Hebb learning rules for adaptive queue management.	Hybrid	Faster convergence rate, reduces queue length fluctuations.	May perform poorly in real-world applications.
QTCP (Reinforcement-Based Q-Learning for TCP) [30]	2021	Uses Q-learning to allow senders to learn optimal congestion control policies.	Hybrid	Improves throughput and reduces transmission latency in IoT networks.	Performance varies in dynamic or unpredictable environments.
Adaptive CC for Vehicle-to-Vehicle Communication [31]	2019	Implements adaptive congestion control with greedy routing for vehicular networks.	Hybrid	Enhances communication by reducing delays and improving message delivery.	May impose unnecessary burden in handling interruptions.
Adaptive Fuzzy Congestion Control for HSTCP/AQM Systems [33]	2021	Utilizes fuzzy logic with a novel shifting function for congestion control.	Hybrid	Improves convergence and reduces queue length in adaptive AQM systems.	Complex implementation and requires frequent adjustments.
CCCLA (Learning Automata for IoT) [35]	2019	Uses learning automata for congestion control in lossy networks.	Loss	Improves QoS, delay, reliability, and throughput in IoT networks.	Performs poorly in real-world networks due to complexity.
OAC-TCPCC (Dynamic Congestion Windows for IoDST) [36]	2020	Dynamically adjusts congestion windows for optimal throughput.	Delay	Improves TCP throughput in networks with long propagation delays.	Limited to high-delay networks, may struggle in others.
QCCP (Priority-Based CC for IoMT) [38]	2023	Prioritizes traffic in Internet of Medical Things (IoMT) networks.	Delay	Reduces latency, packet loss, and improves throughput.	Limited to medical IoT, may underperform in other scenarios.
Buffer-Aided Congestion Control for IoV [40]	2021	Uses two extra bits in ACKs to convey buffer and link status for dynamic congestion window adjustment.	Delay	Optimizes bandwidth and prevents congestion in vehicular networks.	Additional overhead and complexity in real-time ACK processing.

(continued on next page)

Table 1 (*continued*)

Method	Year of Publication	Working Approach	Type (Delay, Loss, or Hybrid)	Role in IoT	Issue if Used in IoT Network
H-TCP for High Bandwidth-Delay Products [41]	2006	Adapts congestion control for paths with high bandwidth-delay products.	Delay	Enhances performance in networks with high BDP.	Focuses mainly on congestion avoidance, leaving slow-start unchanged.
Fast Retransmit and Recovery Without TCP SACK [42]	2002	Responds to partial ACKs during Fast Recovery to improve performance without SACK.	Loss	Improves performance in scenarios where SACK is not available.	Takes longer to detect loss and can unnecessarily reduce the window size.
DRL-Aided Loss Tolerance for 6LoWPAN [47]	2023	Uses deep reinforcement learning to tolerate losses and adjust rates in 6LoWPAN.	Loss	Enhances throughput and delivery in lossy IoT networks.	High computational complexity increases energy consumption.
Inverse Reinforcement Learning for Parallel Congestion Control [48]	2023	Optimizes congestion control through parallel training with IRL.	Hybrid	Increases training efficiency and improves network performance.	Requires a large amount of training data, leading to initial delays.
RL-Based Congestion Management in Smart Grids [49]	2022	Dynamically adjusts transmission rates in smart grid networks using RL.	Hybrid	Improves real-time decision-making and stability in smart grids.	High computational overhead may be unsuitable for resource-limited devices.
Fuzzy Logic for Congestion Control in Wireless Sensor Networks [50]	2021	Uses fuzzy logic for congestion control in resource-constrained WSNs.	Hybrid	Improves energy efficiency and lowers computational demands.	Limited adaptability in highly dynamic networks due to predefined rules.

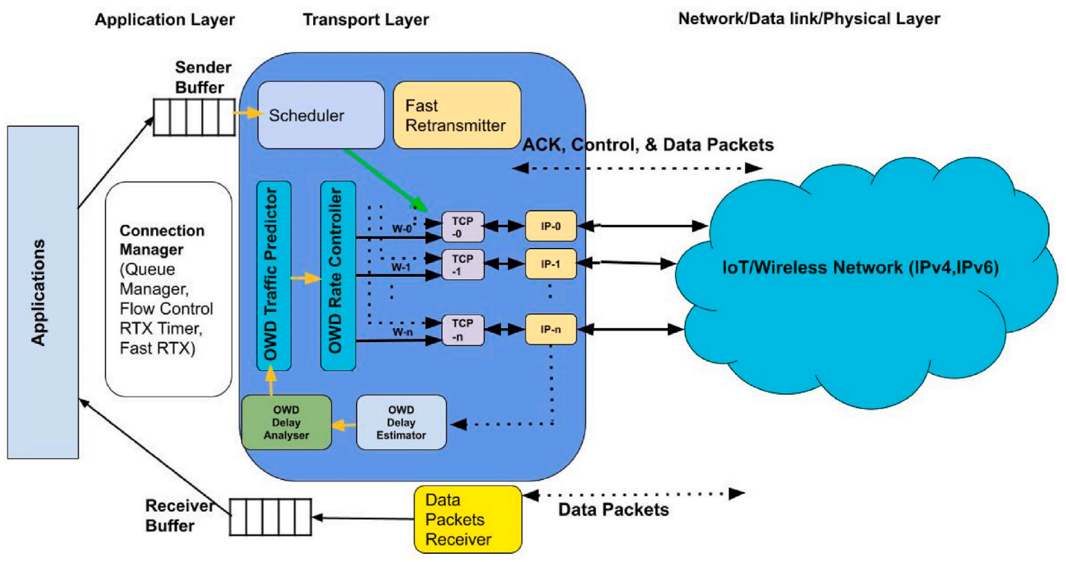


Fig. 4. Proposed OWD framework.

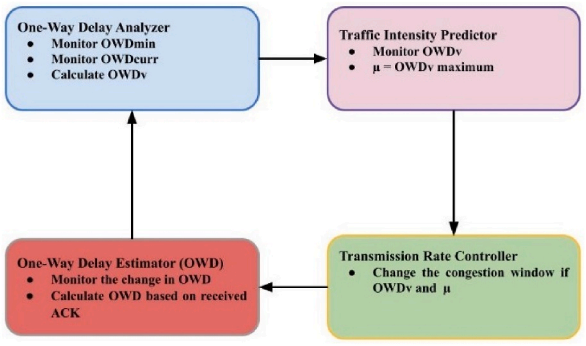

One-Way Delay Analyzer
Monitor OWDmin
Monitor OWDcurr
Calculate OWDv
Traffic Intensity Predictor
Monitor OWDv
μ = OWDv maximum
Transmission Rate Controller
Change the congestion window if OWDv and μ
One-Way Delay Estimator (OWD)
Monitor the change in OWD
Calculate OWD based on received ACK

Fig. 5. Process flow of proposed method.

Table 2
Notation and Abbreviations used in the proposed method.

Notation	Description
PS_{time}	Packet Sent Time
PR_{time}	Packet Receive Time
OWD	One Way Delay
OWDv	OWD variation
OWDmin	Minimum OWD
OWDcur	Maximum OWD
μ	The threshold value based on the maximum OWDv
cwnd	Congestion Window Size
$w_s(t)$	Congestion window size at time t
$D_s(t)$	Observed one-way delay at time t
d_s	Estimated propagation delay

```
Algorithm 1: One-way delay Estimator
Declaration: $PS_{time}$, $PR_{time}$
Output: OWD
Begin
For (Each Packet transmitted) do:
        $PS_{time}$ = Current time
        Put $PS_{time}$ in the packet in the optional field.
        Send the packet.
End Loop
For (Each Packet Received) do:
        $PR_{time}$ = Current time
        Extract $PS_{time}$ from the received packet
        OWD= $PR_{time}$ - $PS_{time}$
        Send OWD in the ACK packet.
End Loop
End
```

3.3. One-way delay Analyzer

The OWD Analyzer is a module proposed specifically to measure and analyze the OWD of a path within a network. Its primary objective is to evaluate and monitor the delay characteristics of the network, providing valuable insights into its overall performance.

This module employs an OWDv approach to predict the traffic intensity along the path. OWDv is calculated by measuring the deviation of the OWDcur from the path's OWDmin. An increase in OWDv indicates a higher degree of variability or fluctuation in the OWD, which could signify congestion or other network-related issues. The OWDv is determined using the following equation (1):

$$OWDv = \frac{OWD_{Cur} - OWD_{min}}{OWD_{min}} \tag{1}$$

OWDv represents the variation relative to OWDmin. OWDv serves as an indicator of the dynamic behaviour exhibited by the network. A rising OWDv signifies increased variability in delay, which may be attributed to shifts in traffic intensity or alterations in network conditions.

The proposed approach involves adapting the traffic intensity along the path based on the observed changes in OWDv. By monitoring OWDv over time, the system can make informed decisions regarding traffic management to mitigate congestion and optimize network performance.

3.4. *Traffic Intensity Predictor*

During the slow start phase, the values of OWDv, OWDmin, and μ (the threshold value based on the maximum OWDv) are calculated. These values form the basis for assessing traffic intensity and network conditions. As TCP transitions into the congestion avoidance phase, the calculated values of OWDv, OWDmin, and μ are used to adjust the transmission rate. In this phase, the method continuously monitors OWDv and μ to detect any signs of increasing congestion. When OWDv exceeds the threshold μ, it indicates that traffic intensity is high on the network path. Thus TCP needs to adjust the transmission rate along the path. The value of μ is continuously updated based on the maximum observed OWDv, ensuring that the method adapts to evolving network traffic intensity in real-time.

3.5. *Transmission Rate Controller*

This module dynamically adjusts the transmission rate along the network path in response to fluctuations in path traffic intensity. To determine if the path's traffic intensity has reached or exceeded its capacity, the method uses OWD and threshold μ, serving as a congestion indicator. When the traffic volume surpasses the path's capacity, the source must reduce its transmission rate by reducing congestion window.

```
Algorithm 2: Transmission Rate Controller at Source
Input: $cwnd_i$, $OWD_v$, μ
Output: $cwnd_{i+1}$
Begin
For (Each ACK received) do:
 If (RTOs expires) then: // Time Out (RTO)
      cwndi+1 = MTU
 Else If (Receive 3-SACKs(duplicate)) then:
       cwndi+1 ≔ cwndi/2
 If ($OWD_v > 2\mu$)
     $cwnd_{i+1}=cwnd_i-(1/cwnd_i)$
 Else If (OWDv > μ)
     $cwnd_{i+1}=cwnd_i$
 Else If ($cwnd_i<\mu$)
     $cwnd_{i+1}=cwnd_i+(1/cwnd_i)$
 Else
     $cwnd_{i+1}=cwnd_i+1$
End
```

The OWDv plays a crucial role as a congestion predictor. If the OWDv exceeds the threshold μ, indicating potential congestion on the path, the source modifies its transmission rate. The reference value OWDv_max is utilized to compute the threshold μ. This dynamic adjustment of μ ensures an accurate reflection of the network status at the source, enabling swift responses to changes in traffic conditions and facilitating efficient congestion management.

Equation (2) represents a set of conditions and corresponding actions aimed at dynamically adapting the congestion window size in a network protocol. These adjustments are based on the observed OWDv and the threshold value μ. When OWDv exceeds the threshold 2μ, indicating potential congestion, the rate controller reduces the congestion window size to mitigate congestion and promote efficient data transmission. Similarly, if OWDv is within the acceptable range (OWDv >μ), the congestion window size remains unchanged to maintain optimal network performance. However, if OWDv is less that μ means that network path still having available bandwidth. Thus, TCP increase the transmission rate in a linear fashion.

$$cwnd_{i+1} = \begin{cases} cwnd_i - \left(\dfrac{1}{cwnd_i}\right) & \text{if } OWD_v > 2\mu \\ cwnd_i & \text{if } OWD_v > \mu \\ cwnd_i + \dfrac{1}{cwnd_i} & \text{if } cwnd_i < \mu \\ cwnd_i + 1 & \textit{if } cwnd_i \langle ssthresh < \end{cases} \tag{2}$$

In essence, the proposed method introduces a novel approach to congestion control in IoT networks, leveraging OWDv, dynamic transmission rate adjustment, and advanced congestion prediction techniques to optimize network performance and reliability. The following are the novelty of the proposed approach.

- ✓ The suggested solution reduces the cwnd according to equation (2) by subtracting 1/cwnd if OWDv> 2μ. It means TCP flow needs to lower the transmission rate when the network reaches capacity. Nevertheless, these kinds of notions are absent from conventional TCP variations. Therefore, this modification is applied when the observed OWDv is greater than twice the product of the μ.
- ✓ On the other hand, if OWDv is greater than μ, the proposed method maintains the study state and does not change the value of cwnd. Unlike the traditional TCP, the proposed method will not exceed the network capacity when the network has limited resources.

The proposed approach uses the same slow start and congestion avoidance method to start the TCP and utilize the available network capacity. This complete method is shown in Algorithm 2.

Additionally, the time and space complexities of the proposed congestion control method, along with several widely used TCP variants, are as follows: TCP Reno and TCP New Reno both have a time complexity of O(n) and a space complexity of O(1), meaning they require linear time to process data but constant space regardless of the data size. Similarly, TCP H-TCP and TCP BBR also exhibit O (n) time complexity and O(1) space complexity. In contrast, TCP Cubic has a more efficient time complexity of O(log n), though it maintains the same space complexity of O(1). The proposed OWD-based method matches the linear time complexity of O(n) and the constant space complexity of O(1) seen in the other protocols, making it computationally efficient while requiring minimal memory resources.

4. Analysis of OWD over RTT

To mathematically demonstrate that OWD is a better indicator of congestion than RTT, we have modeled the relationship between these metrics under various congestion scenarios. Let, OWD_{AB} be an OWD from node A to node B and OWD_{BA} be a One-Way Delay from node B to node A. Let, RTT be the time it takes for a packet to travel from node A to node B and back to node A. Initially, without congestion RTT should be two times of OWD as shown in equation (3).

$$RTT = OWD_{AB} + OWD_{BA} \tag{3}$$

If RTT is equal to double of OWD, and OWD is denoted as d, then RTT will be calculated using equation (4):

$$RTT = 2 \times d \tag{4}$$

If the network is congested in only one direction (from A to B), Then OWD_{AB} is calculated by equation (5).

$$OWD_{AB} = d + \Delta \tag{5}$$

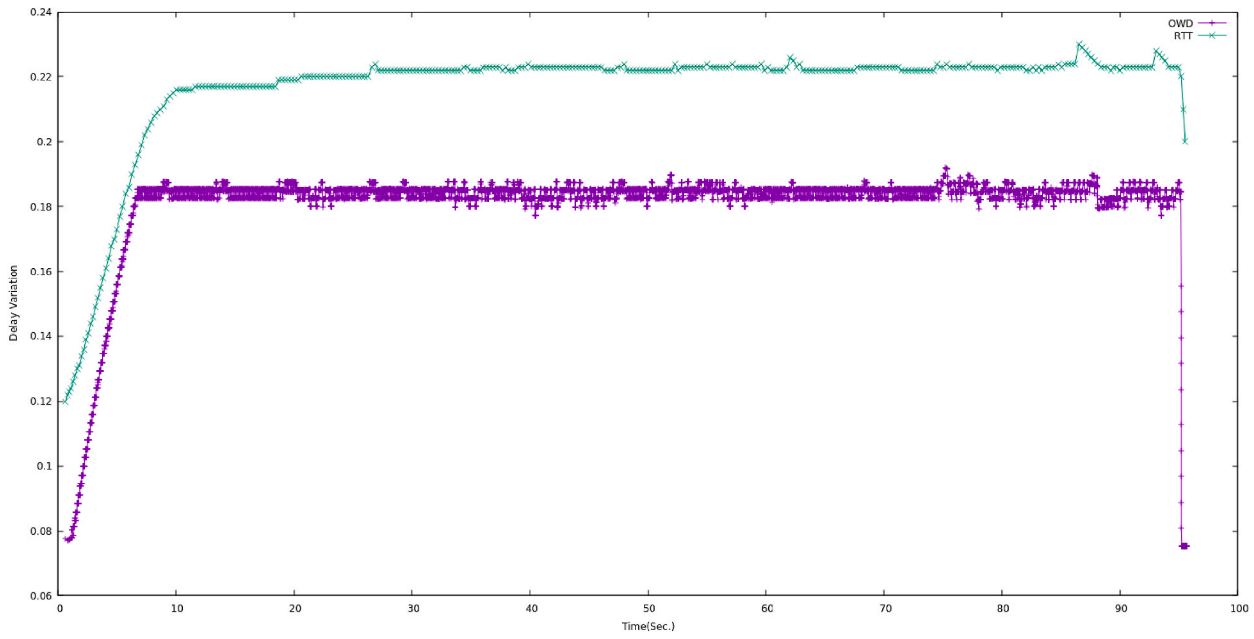


Fig. 6. Sensitivity analysis with heavy congestion on the forward path and less congestion on the backward path.

Where, Δ an additional delay was introduced due to congestion on the path from A to B. However, the backward path B to A is not congested. Therefore, RTT is calculated using equation (6).

$$OWD_{BA} = d$$

$$RTT = OWD_{AB} + OWD_{BA}$$

$$RTT = (d + \Delta) + d$$

$$RTT = 2d + \Delta \tag{6}$$

4.1. *Sensitivity analysis*

The relative and percentage changes for OWD and RTT can be computed as follows:
Relative change in OWD: Δ/d, Percentage change in OWD: (Δ/d) × 100.
Relative change in RTT: Δ/(2d), Percentage change in RTT: (Δ/2d) × 100.
To illustrate the sensitivity difference, we use d = 50 ms and Δ = 20 ms:
OWD Sensitivity: Relative change = 20/50 = 0.4 (40 %)
RTT Sensitivity: Relative change = 20/(2 × 50) = 0.2 (20 %)
This analysis shows that OWD is twice as sensitive to congestion as RTT. The greater sensitivity of OWD means it can detect changes in network conditions more quickly, making it a more effective metric for congestion control in real-time environments, especially in scenarios where rapid detection of congestion is crucial.

To validate the claim that OWD is twice as sensitive to congestion as RTT, we conducted additional simulations to verify this sensitivity. These simulations were performed under two scenarios. In the first scenario, the forward path experienced heavy traffic, while the backward path was less congested. The observed results of delay sensitivity are plotted in Fig. 6. In the second scenario, the forward path had low traffic intensity, while the backward path was highly congested. The corresponding sensitivity results are shown in Fig. 7.

These results reinforce the conclusion that OWD is more sensitive to forward-path congestion, whereas RTT predicts similar behaviour in both scenarios. So, we can conclude that the proposed OWD method is more effective at detecting delays and congestion than RTT, making it a more sensitive and precise metric for congestion indicators.

4.2. *Congestion window adjustment model using OWD*

The Congestion window adjustment $(w_s(t+1))$ is managed by equation (7).

Given: $w_s(t+1) = w_s(t) + \frac{1}{D_s(t)}\left(\frac{d_s}{D_s(t)} - 1\right)$

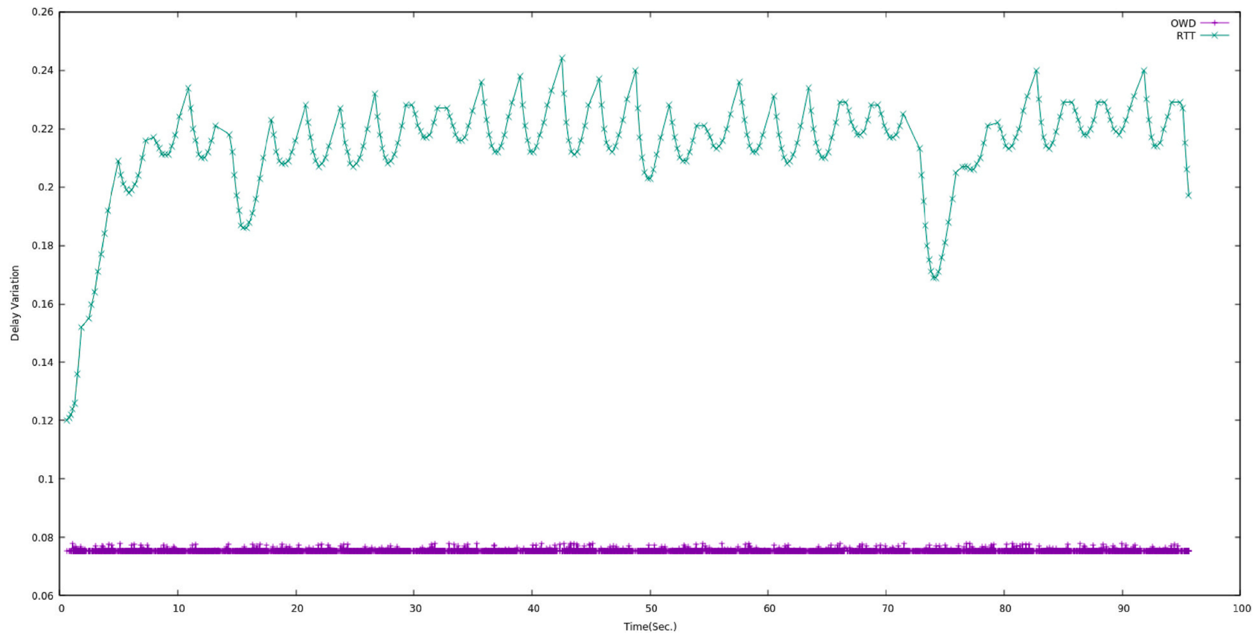


Fig. 7. Sensitivity analysis with low congestion on the forward path and heavy congestion on the backward path.

$$w_s(t+1) = w_s(t) + \frac{d_s - D_s(t)}{D_s(t)^2} \tag{7}$$

4.2.1. *Derivative Analysis*

To understand the dynamics of the congestion window adjustment, we can take the derivative of w_s(t+1) concerning time. This gives us the rate of change of the window size, indicating how quickly the window adjusts to changes in delay and it is calculated using equation (8).

$$\frac{dw_s(t+1)}{dt} = \frac{d}{dt}\left(w_s(t) + \frac{d_s - D_s(t)}{D_s(t)^2}\right) \tag{8}$$

Let's rewrite the adjustment term using equation (9).

$$\Delta w_s(t) = \frac{1}{D_s(t)}\left(\frac{d_s}{D_s(t)} - 1\right) \tag{9}$$

Combining the fractions, we get equation (10).

$$\Delta w_s(t) = \frac{d_s - D_s(t)}{D_s(t)^2} \tag{10}$$

So, the full update formula is represented using equation (11).

$$w_s(t+1) = w_s(t) + \frac{d_s - D_s(t)}{D_s(t)^2} \tag{11}$$

4.2.2. *Interpretation of the adjustment term*

When $D_s(t) \approx d_s$: The observed delay is close to the estimated propagation delay, indicating low congestion.

$$\Delta w_s(t) = \frac{d_s - d_s}{D_s(t)^2} = 0$$

The window size remains almost unchanged.

When $D_s(t) > \rangle d_s$: The observed delay is greater than the estimated propagation delay, indicating high congestion.

$$\Delta w_s(t) = \frac{d_s - D_s(t)}{D_s(t)^2} < 0$$

The window size decreases to reduce congestion.

When $D_s(t) < d_s$: This scenario is less common but can occur due to measurement anomalies or dynamic network conditions.

$$\Delta w_s(t) = \frac{d_s - D_s(t)}{D_s(t)^2} > 0$$

The window size increases, although this situation should be carefully monitored.

Derivative Analysis.

To understand how the window size changes over time, we can take the derivative of the adjustment term concerning time.

Derivative of the Adjustment Term

$$\Delta w_s(t) = \frac{d_s - D_s(t)}{D_s(t)^2}$$

Differentiate this concerning time t we can represent congestion window adjustment using equation (12).

$$\frac{d}{dt}(\Delta w_s(t)) = \frac{d}{dt}\left(\frac{d_s - D_s(t)}{D_s(t)^2}\right) \tag{12}$$

Use the chain rule for differentiation, equation (13) is obtained.

$$\frac{d}{dt}\left(\frac{d_s - D_s(t)}{D_s(t)^2}\right) = \frac{d}{dt}(d_s - D_s(t)) \cdot \frac{1}{D_s(t)^2} + (d_s - D_s(t)) \cdot \frac{d}{dt}\left(\frac{1}{D_s(t)^2}\right) \tag{13}$$

Since d_s is a constant:

$$\frac{d}{dt}(d_s - D_s(t)) = -\frac{dD_s(t)}{dt}$$

$$\frac{d}{dt}\left(\frac{1}{D_s(t)^2}\right) = -2\cdot\frac{1}{D_s(t)^3}\cdot\frac{dD_s(t)}{dt}$$

Combine these results, equation (14) is derived.

$$\frac{d}{dt}\left(\frac{d_s - D_s(t)}{D_s(t)^2}\right) = -\frac{dD_s(t)}{dt}\cdot\frac{1}{D_s(t)^2} + (d_s - D_s(t))\cdot -2\cdot\frac{1}{D_s(t)^3}\cdot\frac{dD_s(t)}{dt}$$

$$\frac{d}{dt}\left(\frac{d_s - D_s(t)}{D_s(t)^2}\right) = -\frac{dD_s(t)}{dt}\left(\frac{1}{D_s(t)^2} + \frac{2(d_s - D_s(t))}{D_s(t)^3}\right) \tag{14}$$

Thus, the derivative of the adjustment term is represented using equation (15).

$$\frac{d\Delta w_s(t)}{dt} = -\frac{dD_s(t)}{dt}\left(\frac{1}{D_s(t)^2} + \frac{2(d_s - D_s(t))}{D_s(t)^3}\right) \tag{15}$$

Full congestion window update model with derivatives including the rate of change of the congestion window:

$$\frac{dw_s(t+1)}{dt} = \frac{dw_s(t)}{dt} - \frac{dD_s(t)}{dt}\left(\frac{1}{D_s(t)^2} + \frac{2(d_s - D_s(t))}{D_s(t)^3}\right) \tag{16}$$

Equation (16) shows that the rate of change of the congestion window w_s at the next time step t+1 is adjusted based on two factors: the current rate of change of w_s and the rate of change of Ds. Specifically, the correction term involving $\frac{d\,D_s(t)}{dt}$ adjusts the rate of change of w_s at time t. This indicates that the evolution of w_s over time is influenced by how Ds changes. Therefore, w_s will vary in response to changes in Ds, as Ds is the dynamic one-way delay factor that evolves and affects the behavior of w_s.

5. Experimental setup and results

The simulation setup employed in this study is detailed in this section, along with the analysis of results focusing on throughput and fairness. The entire experiment was conducted using NS-2.35, a widely used network simulator.

The simulation setup utilized a dumbbell network topology, as shown in Fig. 8. The topology consisted of one TCP IoT source and two UDP IoT sources connected to Router-1. On the other side, Router-2 was connected to two UDP IoT sources, one TCP IoT destination, and two UDP destinations. A bottleneck link was created by interconnecting Router-1 and Router-2.

All nodes in the network were connected via links with varying bandwidths and propagation delays along the path. The experimental configuration incorporated a drop-tail queuing policy, with the default queue size set to 50 packets. Specifically, the TCP source was associated with an FTP traffic generator, while the UDP source was linked to a Constant Bit Rate (CBR) traffic generator.

This setup allowed for the evaluation of network performance metrics such as throughput and fairness, providing insights into the effectiveness of the proposed congestion control method in managing network congestion and optimizing data transmission in IoT environments.

Figs. 9–13 illustrate the performance of the proposed congestion control method in terms of throughput, comparing it against existing algorithms such as Cubic [4], HTCP [41], and New Reno [42]. These figures provide a visual representation of how the throughput varies with the delay of the network path.

5.1. Throughput under variable RTT

Fig. 9 illustrates the performance of the proposed method within a variable RTT environment, where the RTT of the bottleneck fluctuates between 60 ms and 150 ms. The depicted graph demonstrates how the throughput of different TCP variants evolves in

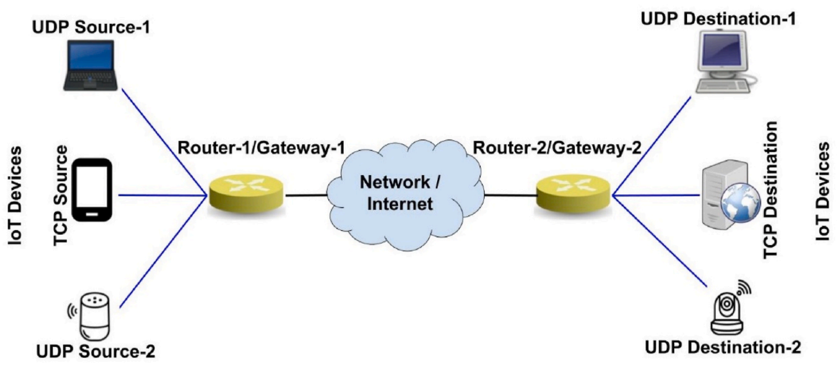


Fig. 8. Dumble topology used for simulation.

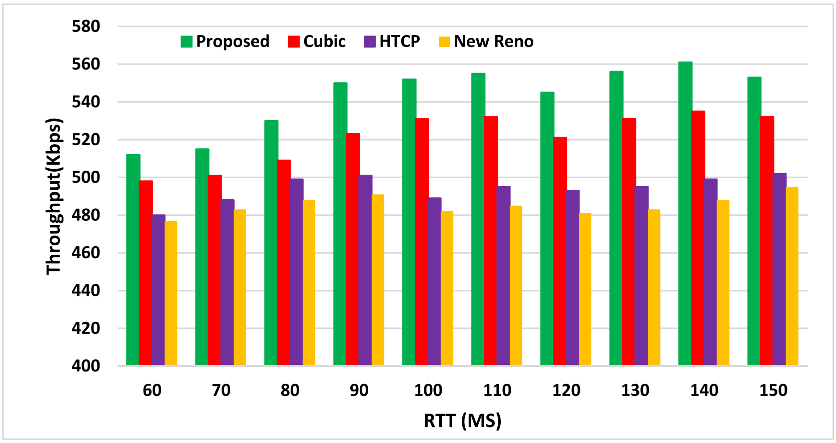


Fig. 9. Performance analysis of the proposed method in the presence of variable RTT.

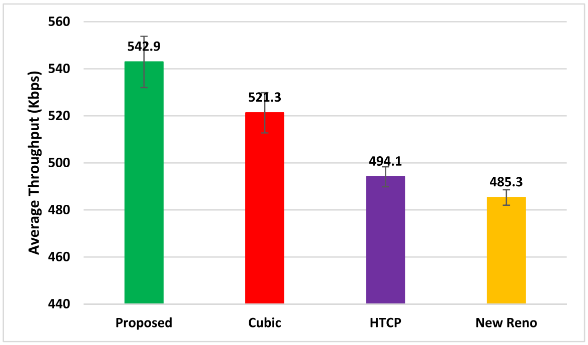


Fig. 10. Average throughput.

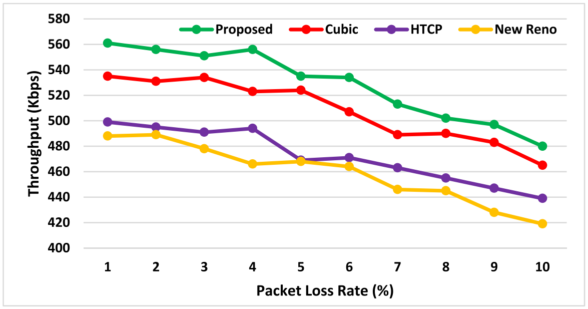


Fig. 11. Performance analysis of proposed method in presence of variable packet loss rate.

response to changes in the RTT of the communication path. Notably, TCP NewReno consistently exhibits the lowest throughput compared to HTCP and Cubic. By comparison, the suggested method exhibits better throughput performance. This benefit stems from using OWD as a congestion indicator, which provides a more accurate assessment of the traffic volume on the communication line. The suggested approach is able to better detect and react to real traffic circumstances by depending on OWD, enabling it to modify the congestion window dynamically. Thanks to its quick flexibility, the suggested technique can traverse and perform better under different path conditions. In summary, incorporating OWD as a congestion indicator enhances the proposed method's ability to swiftly and accurately adapt to changing network dynamics, leading to improved throughput compared to traditional TCP variants like NewReno, HTCP, and Cubic.

Fig. 10 depicts the evaluation of the simulation's average throughput. The results show that the suggested technique regularly achieves 542.9 kbps of average throughput. By contrast, the average throughput of classic TCP variations like Cubic, HTCP, and New Reno is 521 kbps, 494 kbps, and 485 kbps, respectively. The suggested approach guarantees a more precise knowledge of network

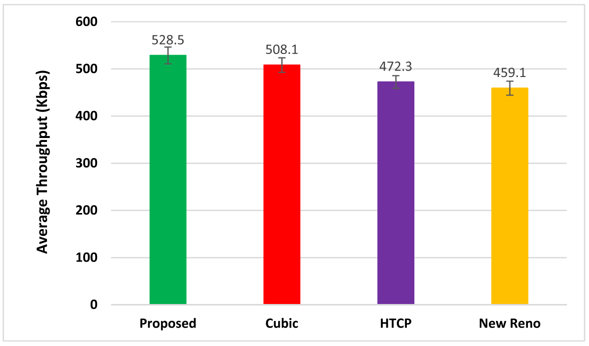


Fig. 12. Average throughput.

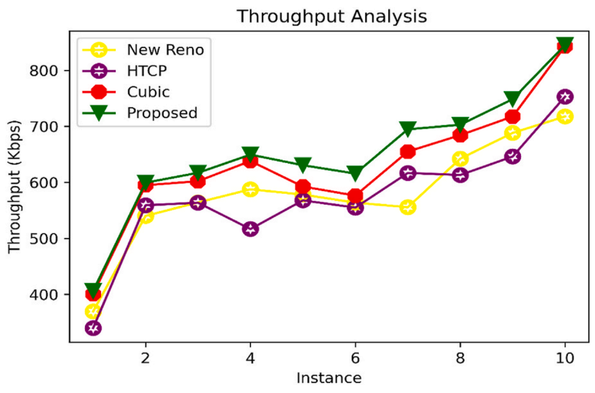


Fig. 13. Throughput under IoT scenario.

circumstances by utilizing precise OWD estimations, allowing it to adjust the transmission rate for maximum efficiency dynamically. The suggested approach performs 14.1 %, 20.5 %, and 22.7 % better than Cubic, HTCP, and New Reno, respectively.

5.2. *Throughput under variable loss rate*

Fig. 11 showcases the experimental results of the OWD method under simulated conditions with varying packet loss rates. The simulation encompasses a spectrum of packet loss rates ranging from 1 % to 10 %.

The trend observed in Fig. 8 illustrates a consistent decline in throughput across all TCP variants as the packet loss rate increases. Notably, TCP New Reno consistently exhibits the weakest performance compared to Cubic and HTCP. This trend highlights the impact of packet loss on TCP performance and underscores the effectiveness of the proposed OWD method in managing congestion and maintaining throughput under adverse network conditions.

In contrast, the OWD method showcases a more robust throughput despite the increasing packet loss rates. This improved performance can be attributed to the method's data rate adaptation policy, which relies on OWD as a fundamental parameter. By utilizing OWD, the proposed method can dynamically adjust its data rate in response to varying network conditions, enabling it to mitigate the impact of packet loss more effectively than traditional TCP variants.

The average performance of different TCP variations with varying packet loss is shown in Fig. 12. The measured throughput for TCP New Reno, Cubic, HTCP, and the suggested approach are, in order, 459.2 Kbps, 472.3 Kbps, 508.1 Kbps, and 528.5 Kbps.

The throughput of New Reno is 13.0 % less than that of the suggested approach, while the throughput of Cubic is 10.6 % less. HTCP performs better than New Reno and Cubic, but it is still not up to par with the suggested approach, with a throughput that is 4.1 % lower. These percentage differences highlight the enhanced performance of the suggested technique in the face of varying packet loss situations.

5.3. *Throughput in sporadically changing environment*

The rapid growth in the number of IoT devices poses significant challenges for network scalability. As the number of devices and traffic volume increase, maintaining low latency, efficient congestion control, and high throughput becomes critical. While protocols like TCP Fast Open and TCP BBR have effectively addressed some scalability issues in traditional TCP networks, the proposed OWD-based method takes a distinct approach. By focusing on real-time congestion management and dynamically adjusting the congestion window based on OWD, the proposed method is designed to rapidly adapt to network congestion in large-scale IoT environments.

To thoroughly analyze the scalability of the OWD-based method, we evaluated its performance under varying traffic volumes and device counts in a highly dynamic environment. We simulated this by configuring the TCP source and destination in scenarios with sporadically changing network conditions. First, we introduced a variable, heavy background of UDP traffic to simulate the massive scale typically found in IoT environments. Second, we accounted for network heterogeneity, incorporating factors such as fluctuating latency, packet loss, and bandwidth constraints.

As bandwidth increases, the network can handle more data, but it also increases the risk of congestion under high traffic. The proposed OWD method adjusts the transmission rate based on real-time traffic intensity using forward delay, preventing overloading the network even at higher bandwidths. High RTT values indicate longer network paths or more congestion. In asymmetric networks, RTT-based approaches can misinterpret the forward and reverse path delays. OWD's focus on one-way traffic allows for more precise congestion control, particularly in scenarios with high RTT. Packet loss directly impacts network performance by triggering retransmissions. The OWD method dynamically adjusts the transmission rate to minimize packet loss, especially in environments where the loss rate is as high as 10 %.

As seen in Table 3, throughput was evaluated across varying network conditions. Bandwidth ranged from 1 Mbps to 10 Mbps, loss rates from 1 % to 10 %, and RTT from 50 ms to 350 ms. Background UDP traffic was adjusted according to the maximum bandwidth available for each instance. The results, illustrated in Fig. 13, demonstrate that the TCP source using the proposed OWD method outperforms other referenced schemes in these dynamic environments. Traditional RTT-based congestion control approaches often struggle with performance in scalable IoT networks, where forward and reverse paths may experience significant differences in conditions. By contrast, the OWD-based approach decouples the forward path from the return path, significantly reducing the complexity of congestion estimation. This decoupling allows the method to perform more efficiently in scalable IoT networks, as each TCP source device only needs to monitor delay in a single direction, resulting in faster and more accurate congestion management. While every congestion control method may face bottlenecks as network size increases, the proposed OWD method is well-positioned to mitigate many common issues, such as computation overhead, increased queuing delays, and memory requirements. However, the biggest challenge for the OWD method is clock synchronization between the source and destination, which is critical for accurately measuring OWD. Fortunately, ongoing research into efficient clock synchronization mechanisms, such as the Network Time Protocol (NTP), can address this challenge, making the method more robust for large-scale implementations.

The scalability of the proposed method is particularly relevant for high-density IoT applications, such as smart cities, industrial IoT, and healthcare systems. In smart city environments, where devices like traffic sensors and surveillance cameras are continuously transmitting data, the OWD-based method ensures that critical data flows are maintained without congestion, even as the number of connected devices grows. Similarly, in IIoT environments, where real-time monitoring and control of machinery are essential, the method's ability to minimize delays ensures stable operations as more devices are added.

5.4. *Fairness Analysis*

Fig. 14 visually represents the fairness among different TCP variants operating concurrently on the same communication path. TCP Cubic, a widely deployed variant in Android and Linux operating systems, serves as the benchmark for fairness measurement in this simulation. In this scenario, three instances of TCP Cubic are actively transmitting on the path, while another variant attempts to achieve a fair share of the bandwidth alongside these three TCP Cubic flows. The figure clearly demonstrates that the proposed approach exhibits superior fairness compared to TCP New Reno, Cubic, and HTCP. Despite the simultaneous operation of three TCP Cubic flows, the proposed method achieves equitable distribution of bandwidth with the existing traffic. This improvement in fairness can be attributed to the proposed approach's sophisticated procedures, which likely involve adaptive modifications based on OWD and intelligent congestion control strategies.

Overall, Fig. 14 highlights the enhanced fairness achieved by the proposed method in managing multiple TCP flows concurrently, showcasing its effectiveness in optimizing bandwidth utilization and ensuring equitable access to network resources.

5.5. *Throughput analysis under heterogeneous environment*

To further validate the performance of the proposed OWD method, we conducted additional simulations using a more complex heterogeneous topology to demonstrate its robustness. This setup is designed to better reflect real-world IoT networks, where multiple paths, varying link qualities, complex topologies, and different traffic levels are common. The simulation topology, depicted in Fig. 15, consists of a wireless IoT node configured as a TCP source and a wired node serving as the TCP destination (sink node). Additionally, two UDP traffic generators are deployed at the TCP endpoints—one wireless and one wired. Both TCP nodes are connected through gateways: one gateway connects to the network via a single path, while the other connects through multiple paths, both wired and wireless. Moreover, several other IoT devices are connected at the wireless end, adding further complexity to the network.

In this configuration, the bottleneck link is set with a bandwidth of 1.5 Mbps and a propagation delay of 50 ms, using a drop-tail queue policy with a queue size of 50 packets. The UDP source at **gateway-1** generates traffic at 1.4 Mbps, while the UDP source at **gateway-2** generates traffic at 1 Mbps. Additionally, the IoT devices connected to the network generate data at rates ranging from 10 to 25 Kbps.

Fig. 16 presents the throughput results under the heterogeneous topology shown in Fig. 15. The results demonstrate that the proposed OWD-based method continues to outperform traditional TCP variants, such as TCP Cubic, HTCP, BBR and New Reno, even in more complex environments. The method adapts effectively to varying link qualities and multiple paths, maintaining higher

Table 3
Parameter setting in a sporadically changing environment.

Instance	Bottleneck links parameters				
	Bandwidth (Mbps)	RTT (ms)	Loss Rate (%)	Impact on OWD	Comparison with Existing Studies
1	1	340	1	Low bandwidth and high RTT introduce significant delays in detecting congestion. The OWD-based method dynamically adjusts the transmission rate to handle these delays.	Most existing studies do not evaluate performance under such high RTTs, making our method's adaptability unique.
2	2	320	2	Increased bandwidth reduces overall delay, but higher loss rates challenge the method's ability to maintain low packet loss. OWD still provides more precise congestion signals than RTT.	Other studies often rely on RTT-based mechanisms that perform poorly with such high loss rates.
3	3	350	3	High RTT and loss rates test the method's robustness in extreme conditions. The OWD metric helps reduce unnecessary congestion window reductions.	Previous work typically focuses on ideal conditions with lower RTT and loss rates, highlighting the resilience of our approach.
4	4	280	4	As bandwidth increases, the method maintains a high throughput, but OWD continues to be more effective in adjusting the transmission rate than RTT in environments with high loss rates.	Studies like [47] focus primarily on optimizing RTT, but fail to address such high-loss environments.
5	5	240	5	A balance between bandwidth and RTT shows how the OWD method can handle diverse network conditions. Increased packet loss still poses challenges, but OWD helps maintain fairness.	Limited studies test conditions with such a combination of high bandwidth and loss, demonstrating the novel contributions of this study.
6	6	200	6	Higher bandwidth enables better congestion control, but the OWD method needs to adapt quickly to prevent high packet loss under these conditions.	Existing research primarily targets either low-loss or low-latency environments, which limits their applicability to real-world IoT scenarios.
7	7	170	7	Increasing loss rate becomes a critical factor. The OWD-based method adjusts transmission to prevent high congestion while balancing packet delivery time.	Few studies have addressed both high-loss and high-RTT scenarios effectively, further supporting the novelty of the proposed method.
8	8	130	8	Low RTT and high bandwidth offer optimal conditions for OWD-based control. The method achieves high throughput with minimal delay in detecting congestion.	Compared to studies focusing only on ideal conditions, our approach demonstrates versatility across varied environments.
9	9	90	9	Extremely high loss rates challenge congestion control mechanisms. The OWD method continues to dynamically adjust to maintain network performance.	Most existing approaches struggle in such high-loss environments, while our method maintains a more stable performance.
10	10	50	10	High bandwidth, low RTT, and very high loss rates test the upper limits of the method. The OWD-based approach still outperforms RTT-based methods in controlling congestion effectively.	Existing studies often fail to accommodate such scenarios with such high packet loss and bandwidth combinations, highlighting the strengths of the proposed method.

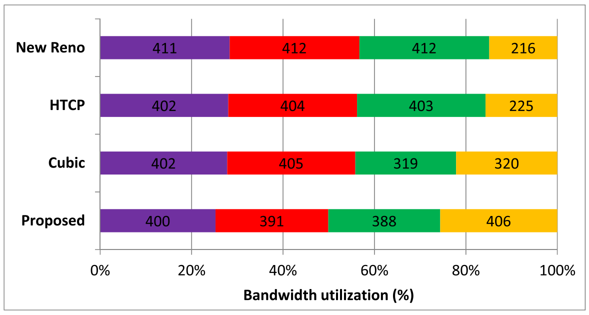


Fig. 14. Fairness Analysis of the proposed method.

throughput and fairness across different conditions.

The proposed OWD-based method outperforms traditional TCP variants like TCP Cubic, HTCP, BBR, and New Reno due to its ability to more accurately detect and respond to congestion in real-time by monitoring one-way delay. Unlike Cubic and HTCP, which may struggle with responsiveness in dynamic IoT networks, the OWD method adapts quickly to varying link qualities and congestion. Compared to BBR, which relies on RTT and can misinterpret asymmetric paths, OWD focuses on forward-path delay, offering more precise congestion detection. Furthermore, while New Reno uses traditional AIMD mechanisms that react slowly, OWD allows faster, more accurate transmission rate adjustments, ensuring better throughput and fairness in complex, multi-path IoT environments.

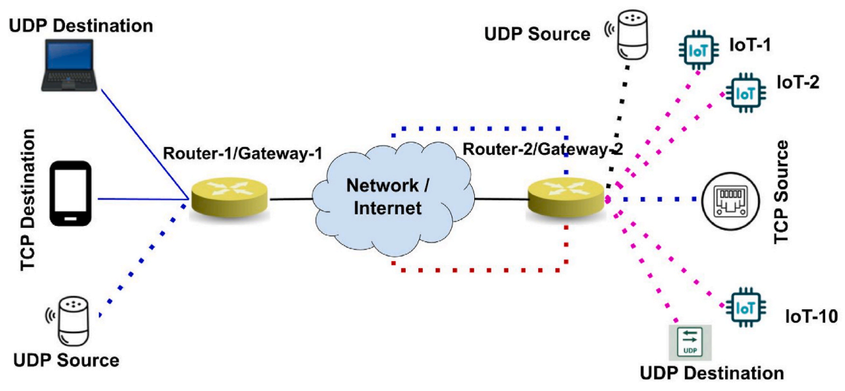


Fig. 15. Heterogeneous environment used for simulation.

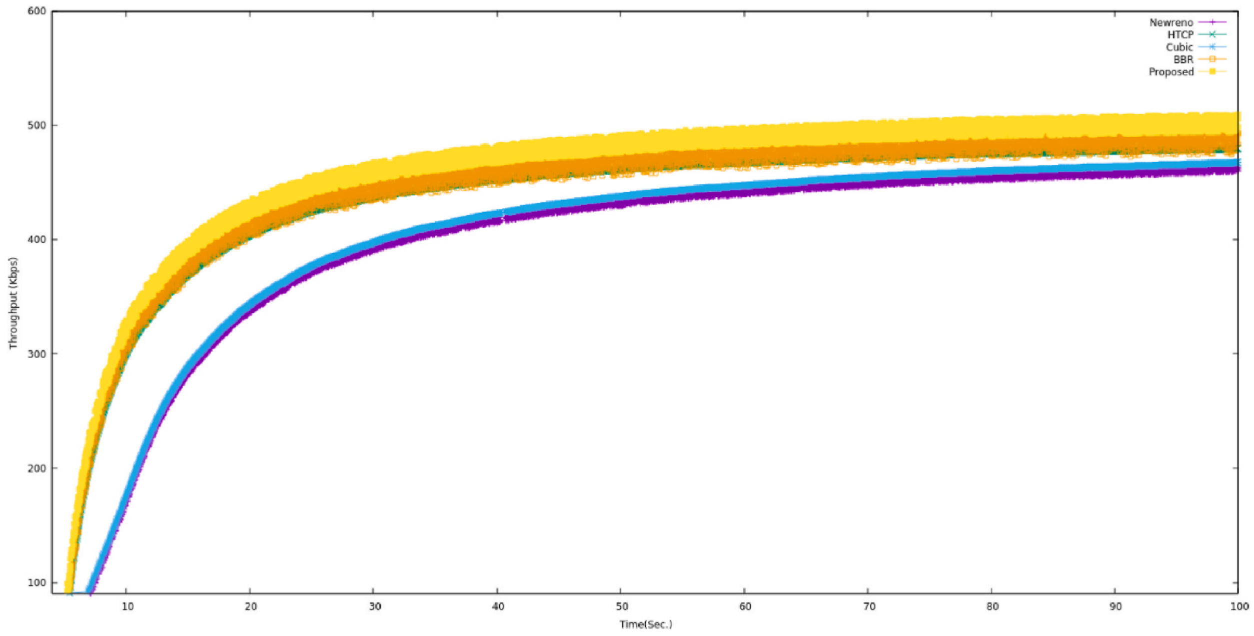


Fig. 16. Throughput under a heterogeneous topology environment.

This simulation demonstrates that the proposed OWD-based method effectively manages mixed-traffic environments, maintaining fairness and stable throughput even when non-TCP (UDP) traffic is present. The method dynamically adjusts TCP transmission rates in real-time, preventing excessive aggression from TCP flows and ensuring that UDP traffic is not adversely impacted. This confirms the method's ability to provide a balanced and efficient solution for IoT networks with both TCP and UDP traffic.

5.6. *Discussion and limitations*

Experimental analysis reveals that the proposed OWD-based approach outperforms well-established TCP variants such as TCP Cubic, HTCP, and New Reno, with improvements in average throughput ranging from 4.1 % to 22.7 %. These results suggest that the OWD-based method has significant potential to enhance both efficiency and fairness in IoT communications. However, One of the main challenges in using OWD for congestion control, particularly in TCP contexts, is the difficulty of accurately estimating OWD in real time. Unlike RTT, which is easier to measure as it considers the time taken for a packet to travel to the receiver and for an acknowledgment to return, OWD only measures the delay in one direction (from sender to receiver). Accurately measuring OWD in real-time requires precise synchronization between the sender's and receiver's clocks. Without this synchronization, it becomes challenging to correctly estimate how long it takes for a packet to reach the receiver. Additionally, OWD measurements may be also affected by network asymmetry, where the forward and reverse paths experience different delays. In such cases, using OWD alone for congestion control may lead to inaccurate conclusions about network congestion, potentially resulting in underutilization or over-utilization of network resources.

To mitigate this, further work may use hybrid approaches that combine OWD with other metrics, such as RTT or packet loss rate, which could offer a more robust solution. A dual-metric system could be explored, where OWD remains the primary metric for congestion control, providing real-time forward path delay information. However, in cases of significant path asymmetry, the system could fall back on RTT to account for both forward and return path delays. Additionally, packet loss rate could serve as a supplementary metric, especially in cases where high packet loss indicates congestion that might not be detected by OWD alone.

6. Conclusions

The paper introduces a novel adaptive congestion control approach based on OWD, which estimates path traffic intensity by calculating OWD during data transmission. This method dynamically adjusts the transmission rate according to traffic intensity, aiming to maximize network performance. Experimental analysis reveals that the proposed approach outperforms established TCP variants such as TCP Cubic, HTCP, and New Reno, with average throughput improvements ranging from 4.1 % to 22.7 %. The OWD-based method shows great potential for enhancing efficiency and fairness in IoT communication, due to its ability to adapt to changing network dynamics, accurately estimate congestion, and ensure equitable bandwidth distribution. However, despite its promise, several limitations must be considered, such as challenges in scaling to very large networks, the resource-constrained nature of IoT devices, and the need for precise clock synchronization. Future research should focus on optimizing the method for large-scale networks, adapting it for devices with limited processing power and memory, improving clock synchronization mechanisms, and further validating its performance across diverse network scenarios. Additionally, exploring further optimizations to enhance the methodology for practical IoT implementations is essential.

CRediT authorship contribution statement

Lal Pratap Verma: Writing – review & editing, Writing – original draft, Visualization, Formal analysis, Data curation, Conceptualization. **Gyanendra Kumar:** Writing – review & editing, Writing – original draft, Formal analysis, Data curation, Conceptualization. **Osamah Ibrahim Khalaf:** Supervision, Resources, Project administration, Investigation, Funding acquisition. **Wing-Keung Wong:** Validation, Supervision, Resources, Project administration, Investigation, Formal analysis. **Abdulsattar Abdullah Hamad:** Validation, Supervision, Project administration, Investigation, Funding acquisition. **Sur Singh Rawat:** Writing – review & editing, Writing – original draft, Formal analysis, Data curation, Conceptualization.

Data and code availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A survey on Internet of Things: architecture, enabling technologies, security and privacy, and applications, IEEE Internet Things J. 4 (2017) 1125–1142.

[2] IoT Analytics. Number of Connected IoT Devices. Available online: https://iot-analytics.com/number-connected-iot-devices/(accessed on June 2024).

[3] V. Jacobson, Congestion avoidance and control, Comput. Commun. Rev. 18 (1988) 314–329.

[4] Rhee, I.; Xu, L.; Ha, S.; Zimmermann, A. CUBIC: CC mechanism for TCP. RFC 8312. Available online: https://tools.ietf.org/html/rfc8312.

[5] D. Wei, Q. Cao, X. Zhou, Y. Qian, Compound TCP: a new TCP congestion control for high-speed and long-distance networks, in: Proceedings of the IEEE International Conference on Network Protocols (ICNP), 2007, pp. 230–239.

[6] C. Gomez, A. Arcia-Moret, J. Crowcroft, TCP in the internet of Things: from ostracism to prominence, IEEE Internet Computing 22 (2018) 29–41, https://doi.org/10.1109/MIC.2018.112102200.

[7] V.K. Jain, A.P. Mazumdar, P. Faruki, M.C. Govil, Congestion control in Internet of Things: classification, challenges, and future directions, Sustainable Computing: Informatics and Systems 35 (2022) 100678.

[8] D. Fonović, S. Sovilj, N. Tanković, A survey of end-to-end congestion mechanisms in the field of IoT, in: Proceedings of the 46th MIPRO ICT and Electronics Convention, MIPRO), 2023, pp. 1684–1689.

[9] H. Huang, X. Zhu, J. Bi, W. Cao, X. Zhang, Machine learning for broad-sensed internet congestion control and avoidance: a comprehensive survey, IEEE Access 9 (2021) 31525–31545, https://doi.org/10.1109/ACCESS.2021.3060287.

[10] R. Al-Saadi, G. Armitage, J. But, P. Branch, A survey of delay-based and hybrid TCP congestion control algorithms, Commun. Surveys Tuts. 21 (2019) 3609–3638, https://doi.org/10.1109/COMST.2019.2904994.

[11] H. Jiang, Q. Li, Y. Jiang, G. Shen, R. Sinnott, C. Tian, M. Xu, When machine learning meets congestion control: a survey and comparison, Comput. Network. 192 (2021) 108033.

[12] A. Ayadi, P. Maillé, D. Ros, TCP over low-power and lossy networks: tuning the segment size to minimize energy consumption, in: Proceedings of the 4th IFIP International Conference on New Technologies, Mobility and Security, 2011, pp. 1–5, https://doi.org/10.1109/NTMS.2011.5720608.

[13] X. Luo, K. Zheng, Y. Pan, Z. Wu, A TCP/IP implementation for wireless sensor networks, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2004, pp. 6081–6086, https://doi.org/10.1109/ICSMC.2004.1401352.

[14] C. Gomez, A. Arcia-Moret, J. Crowcroft, TCP in the internet of Things: from ostracism to prominence, IEEE Internet Computing 22 (2018) 29–41, https://doi.org/10.1109/MIC.2018.112102200.

[15] C. Lim, Improving congestion control of TCP for constrained IoT networks, Sensors 20 (2020) 4774, https://doi.org/10.3390/s20174774.

[16] S.R. Pokhrel, C. Williamson, Modeling Compound TCP over WiFi for IoT, IEEE/ACM Trans. Netw. 26 (2018) 864–878, https://doi.org/10.1109/TNET.2018.2806352.
[17] P. Bruhn, M. Kuehlewind, M. Muehleisen, Performance and improvements of TCP CUBIC in low-delay cellular networks, Comput. Network. 224 (2023) 109609.
[18] S.S. Oyewobi, K. Djouani, A.M. Kurien, Using priority queuing for congestion control in IoT-based technologies for IoT applications, Int. J. Commun. Syst. 34 (2021) e4709, https://doi.org/10.1002/dac.4709.
[19] H.H. Hasan, Z.T. Alisa, Effective IoT congestion control algorithm, Future Internet 15 (2023) 136, https://doi.org/10.3390/fi15040136.
[20] T. Kavitha, N. Pandeeswari, R. Shobana, V.R. Vinothini, K. Sakthisudhan, A. Jeyam, A.J.G. Malar, Data congestion control framework in Wireless Sensor Network in IoT enabled intelligent transportation system, Measurement: Sensors 24 (2022) 100563.
[21] M. Talau, T.A. Herek, M. Fonseca, E.C.G. Wille, Improving TCP performance over a common IoT scenario using the Early Window Tailoring method, Comput. Network. (2023) 109875.
[22] H.A. Khalek, L. Mhamdi, Light-weight congestion control in constrained IoT networks. GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 2022, pp. 6265–6270, https://doi.org/10.1109/GLOBECOM48099.2022.10000648.
[23] L.P. Verma, V.K. Sharma, M. Kumar, D. Kanellopoulos, A novel delay-based adaptive congestion control TCP variant, Comput. Electr. Eng. 101 (2022) 108076.
[24] B. Su, X. Jiang, G. Jin, A. Ma, DVPTCP: a delay-driven virtual parallel TCP for high-speed and lossy networks, IEEE Access 7 (2019) 99746–99753, https://doi.org/10.1109/ACCESS.2019.2930139.
[25] G.-H. Kim, Y.-Z. Cho, Delay-aware BBR congestion control algorithm for RTT fairness improvement, IEEE Access 8 (2020) 4099–4109, https://doi.org/10.1109/ACCESS.2019.2962213.
[26] Yuan, F. Wang, Z. Marszalek, A delay-tolerant data congestion avoidance algorithm for enterprise cloud system based on modular computing, Mobile Network. Appl. 27 (2022) 617–627, https://doi.org/10.1007/s11036-021-01826-1.
[27] V. Rathod, N. Jeppu, S. Sastry, S. Singala, M.P. Tahiliani, CoCoA++: delay gradient based congestion control for Internet of Things, Future Generat. Comput. Syst. 100 (2019) 1053–1072.
[28] Q. Xu, G. Ma, K. Ding, B. Xu, An adaptive active queue management based on model predictive control, IEEE Access 8 (2020) 174489–174494, https://doi.org/10.1109/ACCESS.2020.3025377.
[29] A.S. Ibrahim, K.Y. Youssef, A.H. Eldeeb, M. Abouelatta, H. Kamel, Adaptive aggregation based IoT traffic patterns for optimizing smart city network performance, Alex. Eng. J. 61 (2022) 9553–9568.
[30] W. Li, F. Zhou, K.R. Chowdhury, W. Meleis, QTCP: adaptive congestion control with reinforcement learning, IEEE Transactions on Network Science and Engineering 6 (2019) 445–458, https://doi.org/10.1109/TNSE.2018.2835758.
[31] A.K. Sangaiah, J.S. Ramamoorthi, J.J.P.C. Rodrigues, M.A. Rahman, G. Muhammad, M. Alrashoud, LACCVoV: linear adaptive congestion control with optimization of data dissemination model in vehicle-to-vehicle communication, IEEE Trans. Intell. Transport. Syst. 22 (2021) 5319–5328, https://doi.org/10.1109/TITS.2020.3041518.
[32] J.P.A. León, L.J. de la Cruz Llopis, F.J. Rico-Novella, A machine learning based Distributed Congestion Control Protocol for multi-hop wireless networks, Comput. Network. 231 (2023) 109813.
[33] H. Xie, Y. Jing, J. Chen, Adaptive fuzzy practical prescribed time $H\infty$ congestion control for network systems with input saturation and external disturbance, Expert Syst. Appl. (2023) 120930.
[34] H. Ma, D. Xu, Y. Dai, Q. Dong, An intelligent scheme for congestion control: when active queue management meets deep reinforcement learning, Comput. Network. 200 (2021) 108515.
[35] S. Gheisari, E. Tahavori, CCCLA: a cognitive approach for congestion control in Internet of Things using a game of learning automata, Comput. Commun. 147 (2019) 40–49.
[36] A. Masood, T. Ha, D.S. Lakew, N.-N. Dao, S. Cho, Intelligent TCP congestion control scheme in internet of deep space Things communication, IEEE Transactions on Network Science and Engineering 10 (2023) 1472–1486, https://doi.org/10.1109/TNSE.2022.3212534.
[37] P. Tomar, G. Kumar, L.P. Verma, Path-rank-based data chunk scheduling for concurrent multipath transmission, Comput. J. 66 (9) (September 2023) 2254–2264, https://doi.org/10.1093/comjnl/bxac074.
[38] R. Buenrostro-Mariscal, P.C. Santana-Mancilla, O.A. Montesinos-López, M. Vazquez-Briseno, J.I. Nieto-Hipolito, Prioritization-driven congestion control in networks for the internet of medical Things: a cross-layer proposal, Sensors 23 (2023) 923, https://doi.org/10.3390/s23020923.
[39] N. Mast, S. Khan, M.I. Uddin, Y.Y. Ghadi, H.K. Alkahtani, S.M. Mostafa, A cross-layer solution for contention control to enhance TCP performance in wireless ad-hoc networks, IEEE Access 11 (2023) 24875–24893, https://doi.org/10.1109/ACCESS.2023.3244888.
[40] T.K. Mishra, K.S. Sahoo, M. Bilal, S.C. Shah, M.K. Mishra, Adaptive congestion control mechanism to enhance TCP performance in cooperative IoV, IEEE Access 11 (2023) 9000–9013, https://doi.org/10.1109/ACCESS.2023.3239302.
[41] B. Bruhn, M. Kuehlewind, M.H.-T.C.P. Muehleisen, TCP congestion control for high bandwidth-delay product paths, Available online: https://tools.ietf.org/html/draft-leith-tcp-htcp-03. (Accessed 12 June 2023).
[42] S. Floyd, T. Henderson, A. Gurtov, The NewReno modification to TCP's fast recovery algorithm, Available online: https://tools.ietf.org/html/rfc3782. (Accessed 12 June 2023).
[43] M. Ahmad, U. Ahmad, M. Ngadi, M. Habib, N. Faisal, N. Mahmood, ARFC: advance response function of TCP CUBIC for IoT based applications using big data, Concurrency Comput. Pract. Ex. 33 (2021).
[44] Y. Song, C. Sun, W. Feng, H. Luo, B. Sun, Cubic-PermutationPolynomial-based sliding window network coding algorithm in IoT, IEEE Internet Things J. 10 (21) (2023) 19234–19243.
[45] H.H. Hasan, Z.T. Alisa, Effective IoT congestion control algorithm, Future Internet 15 (4) (2023) 1–14.
[46] P. Anitha, H.S. Vimala, J. Shreyas, Comprehensive review on congestion detection, alleviation, and control for IoT networks, J. Netw. Comput. Appl. 221 (2024) 103749.
[47] Y. Hou, H. He, X. Jiang, S. Chen, J. Yang, Deep-reinforcement-learning-aided loss-tolerant congestion control for 6LoWPAN networks, IEEE Internet Things J. 10 (21) (2023), https://doi.org/10.1109/JIOT.2023.3281482.
[48] Pengcheng Luo, Yuan Liu, Zekun Wang, Jian Chu, Genke Yang, A novel Congestion Control algorithm based on inverse reinforcement learning with parallel training, Comput. Network. 237 (2023) 110071.
[49] A.R. Andrade-Zambrano, J.P.A. León, M.E. Morocho-Cayamcela, L.L. Cárdenas, L.J. de la Cruz Llopis, A reinforcement learning congestion control algorithm for smart grid networks, IEEE Access 12 (2024) 75072–75092, https://doi.org/10.1109/ACCESS.2024.3405334.
[50] N. Mazloomi, M. Gholipour, A. Zaretalab, Efficient fuzzy methodology for congestion control in wireless sensor networks, J. Franklin Inst. (2024 Jun 13) 107014.
[51] N. Cardwell, Y. Cheng, T.S. Ng, BBR: congestion-based congestion control, Comput. Commun. Rev. 46 (3) (2016) 57–66.