TCP-do: Enhancing TCP Performance with Delay Oscillation Frequency Analysis

Sueun Lee, Chaeyeong Lee, Jeongyeup Paek

Department of Computer Science & Engineering, Chung-Ang University, Seoul, Republic of Korea
{sueun1015, cxaexeong, jpaek}@cau.ac.kr

***Abstract*—As demand for ultra-high-speed Internet rises, the flaws in traditional congestion control methods based on packet loss are becoming clear. In wireless networks, packet loss often results from interference and collision, not actual congestion, causing inefficiencies. Delay-based congestion control algorithms which typically use round-trip time (RTT) as a key metric have been proposed to address these challenges. However, RTT alone cannot fully capture queuing delays, and small variations in delay measurements can cause significant fluctuations in the sending rate, leading to packet loss or inefficient link utilization. This study introduces *TCP-do*, a novel protocol that shifts the focus from static delay measurements to analyzing *delay oscillation frequency*. *TCP-do* monitors dynamic delay fluctuations and precisely adjusts the congestion window (cwnd), greatly enhancing delay-based congestion control especially in environments with frequent delay variations.**

***Index Terms*—TCP, Delay-based congestion control, RTT analysis, Delay oscillation frequency**

I. Introduction

As the demand for high-speed, reliable internet connections continues to grow, the limitations of traditional congestion control methods are becoming increasingly apparent. TCP CUBIC [1], a widely adopted loss-based congestion control algorithm, exemplifies these methods by using packet loss as a primary signal for congestion. However, in environments where network conditions are dynamic and unpredictable, such as modern wireless networks, relying on packet loss as a primary indicator of congestion often leads to suboptimal performance. Packet loss in these networks is frequently caused by signal interference rather than actual network congestion, resulting in a misinterpretation that can significantly degrade network efficiency.

To address these challenges, the networking community has explored alternative approaches, particularly those based on network delays rather than packet loss. *Delay-based congestion control* algorithms offer a promising solution, adjusting transmission rates by analyzing network delay, typically using round trip time (RTT) as a key metric. However, RTT measurements can be influenced by a multitude of non congestive factors, making it difficult to isolate the impact of actual congestion. Furthermore, the broad range of speeds over which congestion control algorithms must operate exacerbates this problem, as small inaccuracies in delay measurement can lead to large fluctuations in transmission rates, thereby reducing overall efficiency of the network.

Recognizing these issues, recent research has shifted towards analyzing delay trends, such as the rate of RTT increase, to better understand and manage congestion. However, these approaches often overlook the importance of dynamic delay fluctuations, which are critical for accurately assessing network conditions in real time. We thus introduce *TCP-do*, a novel protocol that leverages delay oscillation frequency to dynamically respond to changing network conditions. By shifting the focus from static delay measurements to the frequency of delay oscillations, this new approach provides a more detailed understanding of congestion, enabling more precise control of transmission speeds and ultimately enhancing the performance of delay based congestion control algorithms. In our evaluations, the proposed algorithm consistently outperformed existing algorithms by achieving higher throughput while maintaining lower and more stable RTTs. Its ability to effectively manage dynamic network conditions demonstrates its suitability for environments with varying traffic patterns and delays.

II. Related Work

This section reviews delay based congestion control algorithms in TCP networks, categorizing them into static delay based approaches and trend based delay approaches that analyze delay patterns and variations.

A. Static Delay Based Congestion Control Algorithms

Early delay based congestion control algorithms used instantaneous RTT values to assess network state and directly adjust transmission rates. TCP NewReno [2] is an example of such an early algorithm. It combines RTT measurements with packet loss information to adjust transmission rates during congestion recovery. TCP Vegas [3] introduced the concept of using the difference between expected and actual RTT to detect congestion earlier. TCP FAST [4] advanced this approach for high speed network environments by utilizing absolute

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A4A5034130 & No. RS-2024-00359450), and also by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-RS-2022-00156353) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation)

RTT values to rapidly adjust transmission rates and prevent congestion.

However, these algorithms struggle with accurately measuring congestion induced delay using RTT. Directly mapping RTT to transmission rates can lead to significant fluctuations due to small measurement errors, and this issue becomes particularly pronounced when managing a wide range of transmission speeds, ultimately leading to inefficient use of network resources [5].

B. Trend Based Delay Congestion Control Algorithms

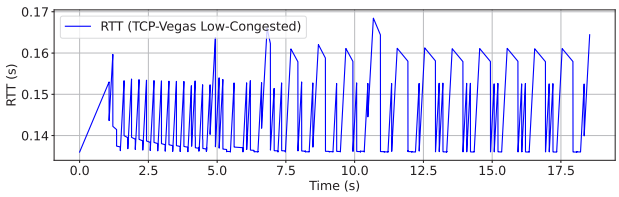
Trend based congestion control algorithms analyze patterns of delay variation and the rate at which it increases to more accurately detect congestion and adjust transmission rates accordingly. For example, TCP Flexis [6] operates by continuously monitoring the speed and direction of RTT changes, dynamically adjusting the transmission rate when a sharp increase in RTT is detected as a signal of congestion. Similarly, TCP Copa [7] focuses on detecting congestion by emphasizing the rate of RTT increase, aiming to improve the precision of delay based congestion control. TCP BBR [8] combines RTT trends with bandwidth estimation to simultaneously address delay and throughput, offering a comprehensive approach to congestion control.

Recently, the analysis of RTT oscillation frequency has gained attention due to its increasing use in enhancing the accuracy of congestion detection. The emergence of dynamic TCP algorithms, such as D-TCP [9], which adaptively adjust congestion control based on channel variability, further underscores the importance of analyzing network fluctuations. Building on these studies, we have also aimed to incorporate delay oscillation frequency analysis into *TCP-do* to better reflect trends in delay.

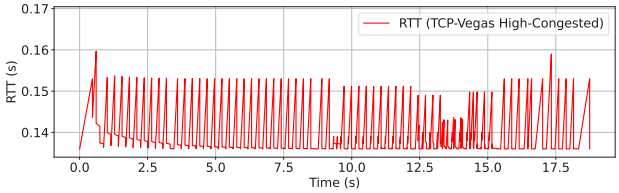
III. Design

Delay-based congestion control algorithms monitor the network's RTT to detect congestion, with increased RTT fluctuations and oscillations occurring under heavier congestion. As shown in Fig. 1, which includes both low-congestion and high-congestion scenarios (Fig. 1a and Fig. 1b, respectively), RTT changes and oscillation frequencies vary significantly depending on the level of congestion. Recognizing that these oscillation frequencies reflect congestion accurately, we devised a method to use them for real-time congestion detection. Based on this approach, *TCP-do* analyzes the magnitude of oscillation frequency to dynamically adjust the cwnd.

The algorithm described in Alg. 1 monitors variations in RTT to calculate the oscillation frequency. Each time a packet is acknowledged (ACK), it measures the difference between the current RTT and the previous RTT. The oscillation count is increased only if this difference exceeds 0.01 ms, effectively filtering out minor fluctuations and focusing on significant changes that indicate actual congestion. The recorded oscillation count is then used to compute the frequency over a 100 ms window. If this frequency surpasses a predefined congestion threshold, the network is deemed to be congested.



(a) Low-Congested



(b) High-Congested

Fig. 1: RTT Oscillation using TCP-Vegas

Algorithm 1 Calculate Oscillation Frequency

Init: $osc\_cnt \leftarrow 0$, $last\_rtt \leftarrow 0$, $start \leftarrow cur$
if $|cur\_rtt - last\_rtt| > 0.01$ ms **then**
    $osc\_cnt \leftarrow osc\_cnt + 1$
end if
$last\_rtt \leftarrow cur\_rtt$
if $cur - start \geq 100$ ms **then**
    $osc\_freq \leftarrow osc\_cnt/100$, $osc\_cnt \leftarrow 0$, $start \leftarrow cur$
end if
Update RTT history, calculate w_avg
if $osc\_freq > cong\_thresh$ **then**
    Mark as congested
end if

Using weighted averages for predicting network performance is a common practice in the field [10]. By applying this technique to recent RTT values, *TCP-do* more accurately captures sustained patterns, enabling reliable real-time monitoring of the network's congestion state.

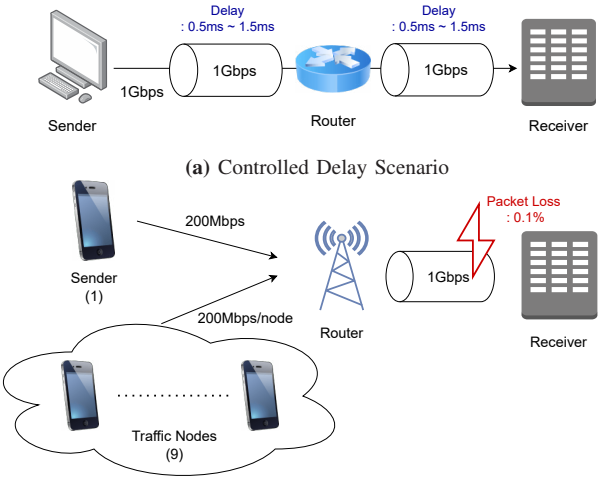
When congestion is detected, the algorithm dynamically adjusts the cwnd based on the ratio of oscillation frequency to the congestion threshold. The cwnd is reduced according to the severity of congestion:

$$\textit{reductionFactor} = \max(0.7, 1.0 - \textit{severity} \times 0.1) \quad (1)$$

The greater the congestion, the more significantly the cwnd is reduced, by up to 30%, to alleviate network load. In cases of mild congestion, the reduction is minimized to maintain throughput as much as possible After congestion subsides, the congestion threshold is temporarily increased by a factor dependent on the severity of congestion to allow for faster recovery of the cwnd:

$$\textit{recoveryFactor} = \min(1.5, 1.0 + \textit{severity} \times 0.2) \quad (2)$$

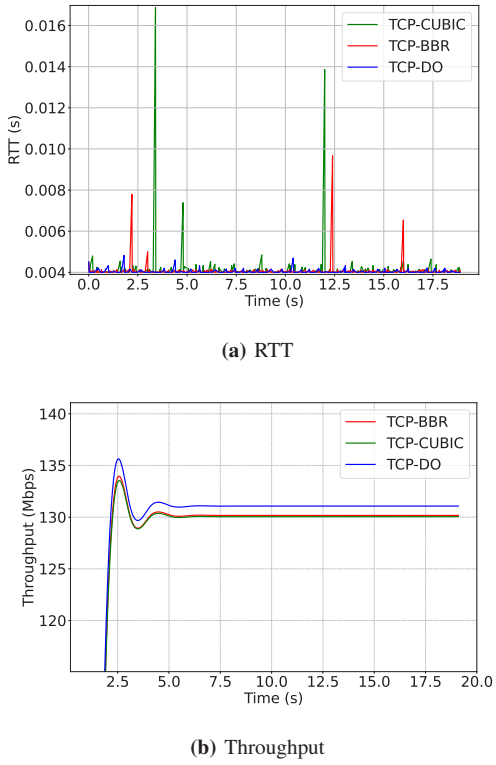
Conversely, if no congestion is detected or RTT oscillations are minimal, *TCP-do* increases the cwnd more aggressively



(a) Controlled Delay Scenario

(b) Dynamic Traffic Scenario

Fig. 2: Network Topologies Used in Experimental Setup



(a) RTT

(b) Throughput

Fig. 3: RTT and Throughput Comparison in Controlled Delay Scenario

by 7 times the segment size at a time and temporarily lowers the congestion threshold by 2% to induce change. This approach enables the system to quickly adapt to various network conditions and maintain stability, even in highly variable environments.

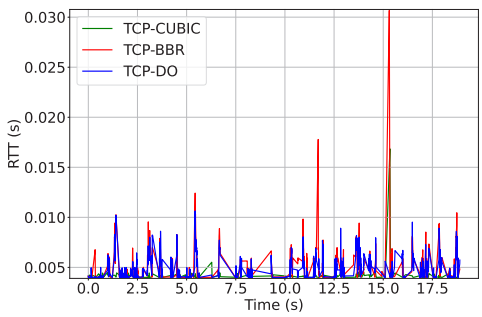
IV. Evaluation

In this paper, we used the ns-3 network simulator to compare the performance of the proposed algorithm with TCP-BBR and TCP-CUBIC. These algorithms were selected as they represent the most popular delay-based and loss-based congestion control methods, respectively. The experiments were conducted using two scenarios, focusing on RTT and throughput as the primary performance metrics.

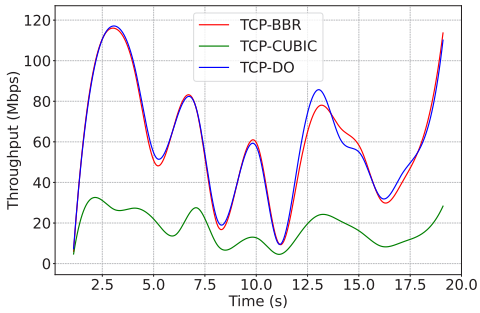
The experimental scenarios are summarized in Fig. 2. In the controlled delay scenario (Fig. 2a), a 1 Gbps link connects the sender, router, and receiver. Random delays between 0.5 ms and 1.5 ms are applied to introduce slight variability, simulating a wired network with real-world imperfections but no competing traffic. The sender transmits data at 200 Mbps using a constant On/Off pattern with both on-time and off-time set to 0.1 seconds, adding minor variability. In the dynamic traffic scenario (Fig. 2b), the setup includes a sender, a router, and a receiver, along with 9 additional traffic nodes. Both the sender and traffic nodes connect to the router, which forwards all traffic through a shared 1 Gbps link to the receiver. Each traffic node, including the sender, transmits at 200 Mbps using a random On/Off pattern, with a mean on-time of 0.1 seconds and an off-time of 0.2 seconds, starting randomly within the first second. The shared link introduces a 0.1% packet loss rate, simulating wireless network variability and congestion.

The results presented in Fig. 3 demonstrate that *TCP-do* consistently achieves superior performance in the controlled delay scenario. As illustrated in Fig. 3a, the proposed algorithm maintains the lowest RTT, averaging 0.004029 seconds, outperforming both TCP-BBR and TCP-CUBIC. Furthermore, Fig. 3b shows that *TCP-do* also excels in terms of throughput. It achieves an average throughput of 124.01 Mbps, surpassing TCP-BBR (123.09 Mbps) and TCP-CUBIC (122.94 Mbps). These results underscore the effectiveness of the proposed approach in enhancing network efficiency by maintaining low latency and high throughput, even in environments with controlled delay.

The results in Fig. 4 demonstrate that the proposed algorithm performs effectively in the dynamic traffic scenario. As shown in Fig. 4a, the algorithm maintains relatively stable RTT values with fewer spikes compared to TCP-BBR. These spikes occur in TCP-BBR because it estimates available bandwidth and misinterprets transient delay increases as signs of network congestion, leading to unnecessary rate adjustments. In contrast, the proposed algorithm more effectively manages delay variations, reducing RTT fluctuations. In terms of throughput, the new method achieves performance comparable to TCP-BBR, as shown in Fig. 4b. Specifically, it maintains an average throughput of 60.17 Mbps, slightly surpassing TCP-BBR's 59.45 Mbps, indicating a 1.22% improvement. While TCP-CUBIC struggles with significantly lower throughput, averaging only 17.21 Mbps due to its sensitivity to packet loss in dynamic conditions, the proposed method sustains high throughput. This balance of maintaining stable RTT and achieving high throughput makes it well-suited for environments with dynamic traffic patterns and varying network



(a) RTT



(b) Throughput

Fig. 4: RTT and Throughput Comparison in Dynamic Traffic Scenario

conditions.

V. Conclusion

Recent advancements in congestion control have focused on incorporating delay trends to better handle dynamic network environments. In this study, we introduced a novel approach that leverages delay oscillation frequency to enhance congestion detection. This method is particularly effective in scenarios where RTT remains low despite underlying congestion, such as during brief traffic spikes. It also addresses cases where interference causes minor packet delays without significant RTT changes. This research highlights the potential to improve the accuracy and stability of congestion control, making a valuable contribution to the field. For future work, we plan to test our approach in real-world environments using advanced simulators and explore integrating machine learning to further enhance its performance.

References

[1] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[2] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC 2582*, 1999.

[3] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, 1994, pp. 24–35.

[4] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, 2006.

[5] V. Arun, M. Alizadeh, and H. Balakrishnan, "Starvation in end-to-end congestion control," in *SIGCOMM '22: Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 177–192.

[6] Q. Li, "TCP FlexiS: A New Approach to Incipient Congestion Detection and Control," *IEEE/ACM Transactions on Networking*, vol. 32, no. 2, pp. 1245–1260, 2024.

[7] V. Arun and H. Balakrishnan, "Copa: Practical Delay-based congestion control for the internet," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 329–342.

[8] N. Cardwell, Y. Cheng, C. S. Gunn, S. Hassas Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[9] M. R. Kanagarathinam, S. Singh, I. Sandeep, H. Kim, M. K. Maheshwari, J. Hwang, A. Roy, and N. Saxena, "NexGen D-TCP: Next Generation Dynamic TCP Congestion Control Algorithm," *IEEE Access*, vol. 8, pp. 164 482–164 496, 2020.

[10] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer TCP throughput," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 145–156, 2005.