IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 29, No. 3, June, 2024, pp. 1115-1123
Journal homepage: http://iieta.org/journals/isi

Enhancement of Data Center Transmission Control Protocol Performance in Network Cloud Environments

Qusay Kanaan Kadhim, Atyaf Ismaeel Altameemi, Rasha Mahdi Abdulkader, Shaymaa Taha Ahmed*

Department of Computer Science, University of Diyala, Baqubah 32001, Iraq

Corresponding Author Email: ShaimaaAhmed@uodiyala.edu.iq

https://doi.org/10.18280/isi.290329

Received: 23 November 2023
Revised: 9 April 2024
Accepted: 16 May 2024
Available online: 20 June 2024

Keywords:
cloud computing, DCTCP, data center, TCP, Random Early Detection (RED), explicit congestion notification (ECN)

ABSTRACT

The increasing use of cloud services in several areas has led to the growth of data-intensive applications. It is necessary to find ways to enhance the efficiency of communications within the networks of data centers to improve the performance of cloud environments. Explicit Congestion Notification (ECN) is used by Data Center TCP (DCTCP) to enhance congestion control in data center networks. The DCTCP uses ECN to assess the amount of congestion, whereas normal TCP congestion management can simply detect its presence. This paper examines DCTCP using the Random Early Detection (RED) queue management strategy. The evaluation reveals that employing Random Early Detection incurs certain costs. The RED is criticized on the one hand for both short- and medium-term connections due to longer completion time delays compared to typical DCTCP techniques. Because of ECN, DCTCP may maintain small queue sizes. However, because RED uses the average queue size, it penalizes short-lived traffic because it does not reach the bottleneck quickly. An intelligent queue management mechanism with ECN is believed to enhance DCTCP's performance in a cloud-computing environment by predicting sending rates and providing fast feedback on queue length.

1. INTRODUCTION

The concept of cloud computing is expected to affect the future of the Internet significantly. It has emerged because of the Internet's quick development and the rise of client-server computing. Data centers are recognized as the center's hubs and hosts of various applications that are the Internet's foundational configuration. Reliability is frequently referred to as a mission-critical system in data centers, where it must be attained at all costs [1]. Users of popular Social Medias and Websites such as Facebook, and Yahoo rely on timely access to services provided by cloud data centers [2]. The difficulties in scaling and distributing data in cloud data centers are caused by a variety of factors [3] namely: Data growth that is exploding growth without specific localization poses challenges in effectively managing and distributing data across cloud data centers; Legal necessity for geographically distinct data backup sites that are regulatory requirements often mandate the establishment of geographically distributed data backup and replication sites to ensure integrity, availability and reliability of data; the development of mobile and cloud computing for storing or processing date necessitates efficient data scaling and distribution mechanisms [4]. Massive interactive web application requires real-time interactivity and data processing that adds complexity to data distribution across multiple servers and data centers and Constraints on energy are a significant factor in reducing environmental impact poses to scale and distributing data in cloud data centers.

In cloud data centers networks accommodate three different types of traffic [5].

The first type is short-lived traffic that includes transmitting connections at less than 100 KB/per second such as Google searches and Facebook updates [6]. These connections require short response times and emphasize the need for quick data transmission. The second type is medium lifespan traffic. It involves connections that have medium lifespan, such as Facebook photographs and small-to- medium-sized files from YouTube [7]. These connections typically demand low latency and ensuring minimal delay in data transfer for example typically range from 100 KB to 5 MB. The last one is long-lasting traffic that encompasses long-duration task connections such as antivirus upgrades and video-on-demand services [8]. These connections typically involve data transfers exceeding 5 MB and emphasize the importance of strong throughput, ensuring efficient and reliable data transmission over an extended period.

The Round-Trip Time (RTT) in a cloud environment, disregarding queuing delay, can be less than 255 microseconds in certain cases due to the emphasis on achieving high throughput and minimizing latency [9]. In order to optimize costs and performance, usually, the cost and performance levels of data centers are used to categorize them. Requests from incoming users are divided across the various data centers to ensure efficient utilization. The coexistence of various traffic types within cloud networks imposes three fundamental requirements namely: high throughput, low delay, and robust tolerance for bursty traffic. These requirements

play a critical role in determining the quality of service needed for various applications depending on the specific type of traffic [10]. The extent to which cloud data center networks perform successfully is influenced by how well these requirements are met [11].

Transmission Control Protocol (TCP)/Internet Protocol (IP) usage in cloud data center networks that is standard and unmodified, without taking into account the varying demands of multiple coexisting traffic kinds, can lead to performance challenges. The various qualities of the networks in cloud data centers and diverse requirements of applications that TCP fails to meet the three prerequisites described above [12].

Consequently, TCP's congestion control systems encounter difficulties, resulting in several flaws such as TCP, and impact of queue accumulation, buffer stress, as well as phony congestion. In cloud computing environments the partition/aggregate, as shown in Figure 1, is a commonly structure. However, the configuration can lead to TCP missing deadlines and delivering subpar results that violate Service-Level Agreements (SLAs).

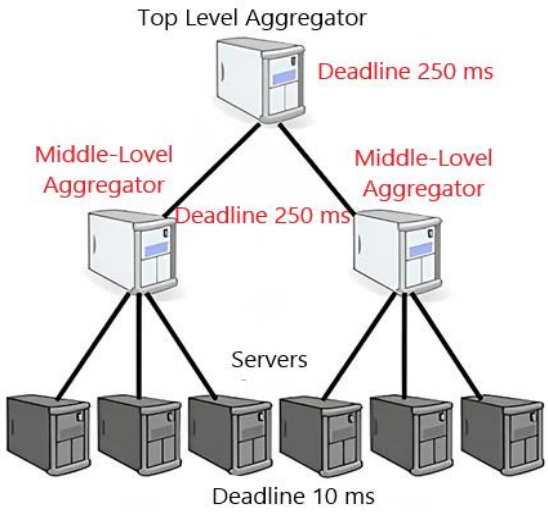


Figure 1. Structure for partition/aggregation in a cloud environment [13]

To prevent packet loss brought to reduce end-to-end latency, the Data Center Transmission Control Protocol (DCTCP) was created as a result of issues using queue building, TCP Incast, and TCP outcast. It the crucial for transient connections over TCP, achieving a high-through, needed for persistent TCP connections, and successfully handling high burst traffic tolerance [14].

Data centers host diverse applications, from batch processing to user-driven workloads. The rise of data-intensive applications and cloud computing have made it necessary to look for alternate strategies to increase the effectiveness of communication in data center networks. Data centers in the cloud coordinate hundreds of thousands of heterogeneous tasks daily, and meeting everyone's requirements in various aspects becomes a very complex problem. To enhance TCP congestion control in data center networks, Data Center TCP (DCTCP) makes use of Explicit Congestion Notification (ECN). DCTCP uses ECN to assess the amount of congestion, whereas normal TCP congestion management can simply detect its presence. With the increasing growth of cloud services in several areas, more and more research is focusing on the performance improvement of cloud environments.

2. TCP ISSUES IN CLOUD NETWORKS

The majority of traffic in cloud data center networks is carried through the widely adopted Transmission Control protocol (TCP) which is a popular transport layer protocol for Internet applications [15]. TCP is known for its, scalable, reliable, and adaptable congestion control methods that can handle variety of network scenarios [16]. However, it has become evident that TCP falls short in meeting the three fundamental requirements mentioned earlier due to the diverse nature of cloud data center networks and the wide range of application demands. These diversities and demands have led numerous difficulties challenges for TCP's congestion control techniques, resulting in several flaws, including TCP Incast TCP Outcast, queue expansion, and the appearance of overcrowding [17]. The subsections that follow highlight the causes of these issues and present suggested remedies.

Issues such as TCP Incast, TCP Outcast, buffer compression, queue backlog, and congestion impact severely influence the performance of traditional TCP variants in cloud environments. As cloud environments become an essential part of the near future, controlling or mitigating these problems is important.

2.1 TCP Incast

In the cloud computing environment, TCP Incast can occur due to the aggregate/partition network architecture of cloud data centers. When a query is made requests are sent by the aggregator to various servers that have the required data. One's a query is answered servers immediately respond with a significant amount of data [18]. This data is then transmitted over a bottleneck connection to the appropriate aggregator as shown in Figures 2 and 3. This cloud aggregate/partition structure communication pattern exemplifies how TCP Incast can occur.

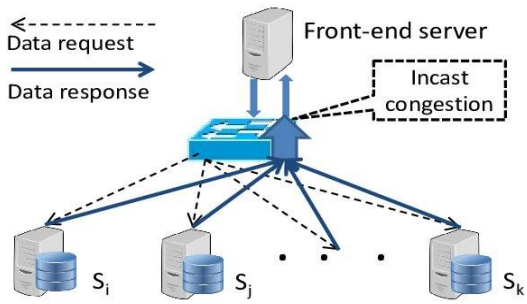


Figure 2. Application partition/aggregate model [19]

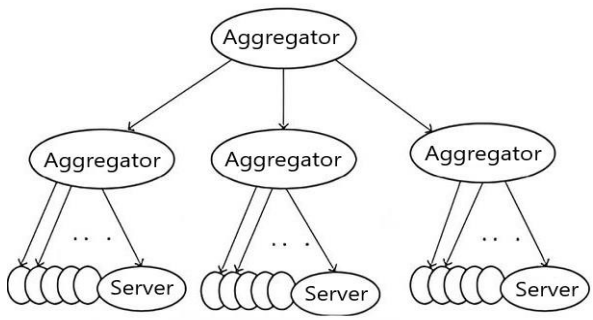


Figure 3. TCP Incast in cloud computing environment [20]

The environment's short, constrained reaction time requirement is the reason behind coordinated and synchronous responses to the demands. Without a use from cloud switches or routers, the resulting data packets would probably overload the queues, leading to packet damage [21]. As shown in Figure 4 this effect illustrates the impact of synchronized and simultaneous fulfillment of aggregator requests. This condition, where servers pause and the coordinated transmission are hindered by something. Broadcasting of upcoming data block while awaiting the provision of the most recent data from all servers is commonly referred to as synchronized mice colliding [22]. Due to the frequent packet losses in such a scenario, significantly decreases as the number of simultaneous servers increases [23].

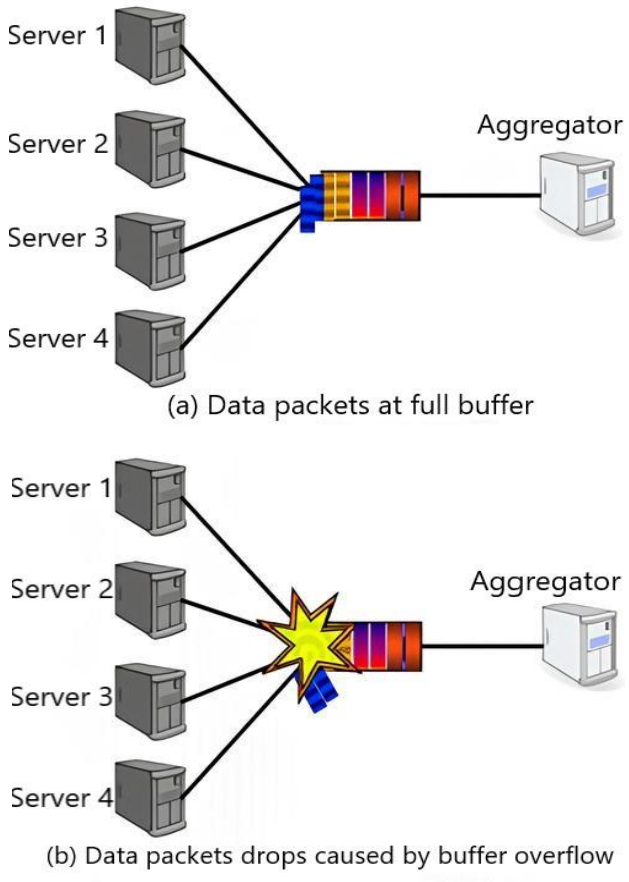


Figure 4. Response to aggregator requests in synchronous

When a Re-Transmission Timeout (RTO) which represents the duration of the transmitting server waits for acknowledgements for previously delivered sent packets expires the lost packets are simply retransmitted [24]. Consequently, the TCP Incast's recurring timeouts have a significantly impact on the effectiveness of transient traffic especially those with stringent delay requirements and the need for quick and precise response time as shown in Figure 5.

2.2 TCP Outcast

The TCP Outcast phenomenon occurs when multiple incoming flows compete with a few flows at a switch's or routers two ports. In a cloud environment, this phenomenon is caused by the Tail-Drop (TD) operation implemented by data center routers and switches [25]. The TD is a simpler and more efficient computational mechanism compared to other queue management systems. It does not, however, ensure that each flow's packet drop rate is proportionate to its bandwidth share. Upon reaching the queue's maximum length, TD removes packets from the end of the queue [26]. This can lead to consecutive packet from a port's queue increases the likelihood of timeouts and adverse global synchronization events. In this situation, known as a port blackout, the performance of small flows is significantly reduced due to the substantial delays caused by numerous timeouts. Data packets arrive to input ports (A and B) in the port blackout scenario shown in Figure 6, where they compete with one output port at (C). Data packets coming at port B are appropriately buffered, as shown in Figure 6, whereas packets arriving at port are lost one after the other.

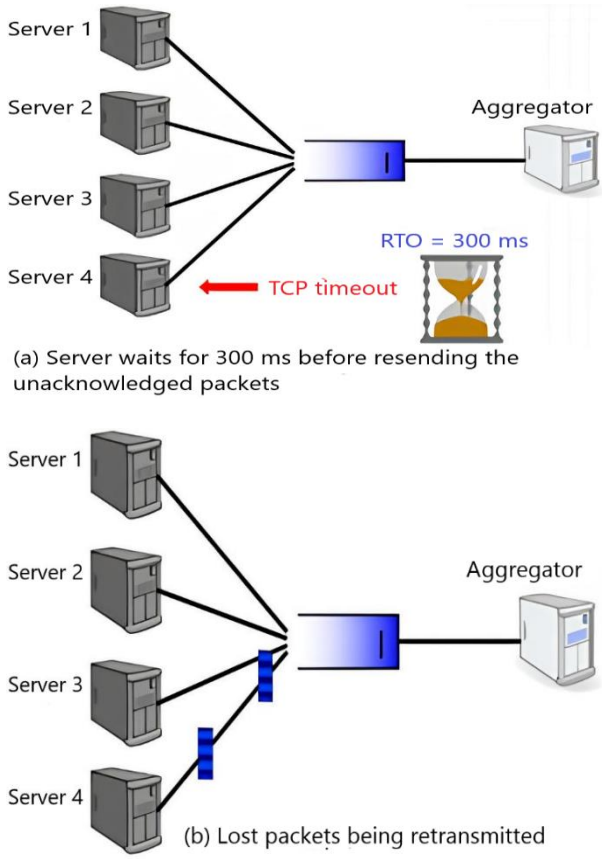


Figure 5. After timeout expiration retransmission

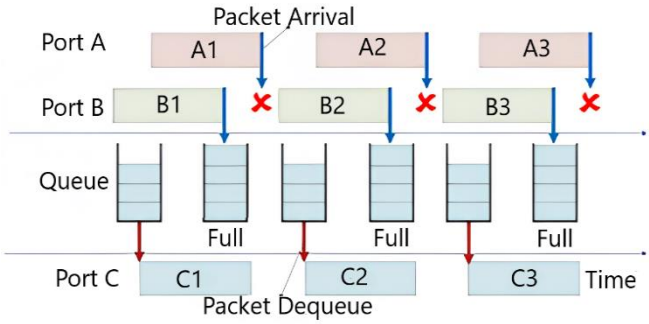


Figure 6. Case of the port blackout [27]

The Round-Trip Tim (RTT) bias can arise from the inverse relationship between a TCP connection's throughput and RTT. This relationship implies that TCP connections with lower RTT can use a greater portion of the available bandwidth than

connections with higher RTT. However, it is observed that TCP Outcast that is caused by the TD queue management mechanism employed in data center routers and switches leads to an inverted RTT Bias. As a result, TCP connections with high RTT tend to overwhelm those with low RTT. The many-to-one communication model servers are the foundation for various applications in a cloud computing environment. The TCP Outcast becomes a major problem in cloud computing because data center networks frequently use TD queue management.

3. QUEUE BUILD-UP PROBLEM

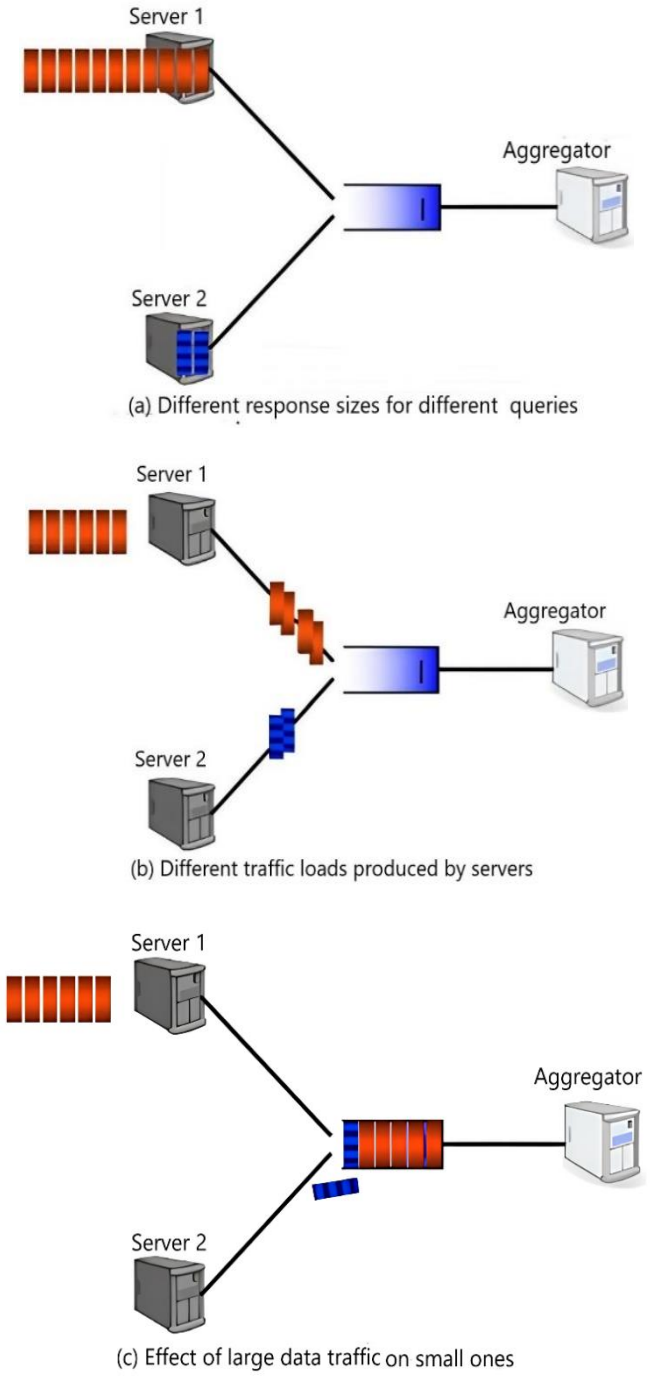


Figure 7. Long-lasting traffic causes queue building

Due to the coexistence of long-lived traffic with variety data center types and traffic patterns for cloud computing the network is compelled to prioritize overflow queues and reduce congestion [28]. Consequently, when short-lived traffic uses the same link as long-lived traffic its performance suffers dramatically. This impact can be observed in Figure 7, where the presence of long-lived traffic consumes as considerable amount of queue space leading to adverse consequences for short-lived traffic. It is highly likely that a larger number of packets from transient traffic will experience drops under such circumstances. This situation bears similarities to challenges posed by TCP Incast as frequent packet loss-induced timeouts have a notable impact on the effectiveness of transient traffic. Furthermore, in situations where the long-lived traffic's packets must first stay digested the queued short-lived traffic packets that arrive later encounter longer queuing times, even in the absence of packet losses which is the typical situation.

The queue building phenomena is what causes it [29] and it can be diminished by intentionally reducing the sending rate of TCP senders within cloud networks, at routers and switches. By using packet drops as a congestion notification signal, TCP senders can be informed of the congestion issue. If the TCP sender only relies on receiving three duplicate Acknowledgements (ACKs) or waits for timeouts, the queue occupancy may not decrease to an acceptable level. Therefore, to solve the issue, proactive action is needed to address the issue of queue accumulation and shorten the backlog's duration, as shown in Figure 7.

4. DATA CENTER TCP

A transport-optimized protocol called DCTCP is specifically designed for networks with early and aggressive Explicit Congestion Notification ECN marking in cloud data centers. This protocol aims to provide low latency and high throughput by enabling TCP senders to generate multi-bit feedback. Moreover, DCTCP is designed to TCP senders should have burst tolerance so they can respond quickly before discarding packets, and the buffer has sufficient available capacity to accommodate bursts.

DCTCP is considered a progressive advancement over TCP aiming to address the issues of TCP Incast and queue accumulating in networks of cloud data centers. To achieve satisfactory performance DCTCP requires minor's modification to the standard Explicit Congestion Notification mechanism used the TCP protocol. ECN serves a purpose of indicating network congestion, thereby helping to maintain a controlled decrease in packet retransmissions and buffers utilization [30]. Packet loss is typically an indication of network congestion caused by buffer overflow. Unlike classic TCP which halves the congestion windows DCTCP senders gradually reduce their capacity window (cwnd) in response to measured congestion levels [31]. By taking a proactive packet loss can be reduced. DCTCP possesses notable such as convergence, equity, and stability making it an ideal protocol for requirements of the cloud environment. The utilization of ECN in DCTCP involves establishing a connection between the DCTCP sender and a compatible receiver, using specific code point in the IP and TCP headers that allocate two for ECN. ECN is triggered where negotiation is done based on certain code points that employ bits from the TCP and IP headers, respectively. Figure 8 demonstrates functioning of ECN and how the knowledge of the network status is leveraged to prevent and manage congestion.

Data centers for cloud computing host a variety of

applications, combining workloads that demand high sustained throughput with workloads that require tiny, predictable latency. In this context, the performance of latency-delivered communication is impacted by the current TCP protocol's failure to meet latency constraints [32]. To solve these problems, we suggest DCTCP; a TCP-like protocol for data center networks. Using the network's explicit congestion notification (ECN), DCTCP gives end hosts multi-bit feedback. Additionally, DCTCP offers low latency and great burst tolerance for brief flows.

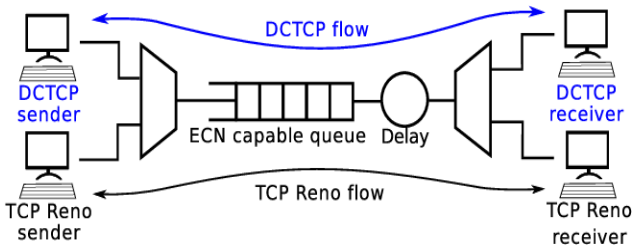


Figure 8. ECN in DCTCP [32]

5. QUEUE MANAGEMENT MECHANISMS IN CLOUD NETWORKS

The networks of cloud data centres employ both reactive and proactive mechanisms for queue size management and congestion control. The commonly used conventional queue management mechanism in cloud networks is Tail-Drop (TD), which is categorized as reactive mechanism. It does not take any action until the buffer becomes flooded, at which point all incoming packets are dropped [33].

In contrast, proactive devices such as Random Early Detection control a queue size by pre-emptively dropping packets before the buffer reaches full capacity, thus protecting against congestion.

Reactive queue management mechanisms suffer from two critical issues [34]:

Lockout: it occurs when certain TCP flows consume a significant portion of the buffer space due to their high transmission rates, leading to excessive packet loss for other TCP flows and unfairness.

Full Queues: they arise when the queue size reaches the maximum buffer capacity without any preventive measures being implemented by the reactive queue management mechanism to control its growth. This situation can persist for extended periods, resulting in significant queuing delays.

The subsequent subsections discuss the advantages and disadvantages of Tail-Drop and Random Early Detection (RED) in addressing these challenges.

5.1 Tail-Drop (TD)

TD (Tail-Drop) employs a packet-dropping strategy where packets are discarded from the tail end of the queue when the defence becomes full, as shown in Figure 9. New incoming packets are only accepted if sufficient space is available in the buffer to accommodate them. However, if the queue is complete, all arriving packets are dropped until packets departing from the queue's head cause the queue to decrease. One advantage from TD is its reduced computational overhead and implementation complexity compared to other queue management mechanisms.

When the line is as full as it can get, the tail drops, it drops newly arrived packets until it has adequate room to take in the incoming traffic [35, 36]. The policy's effect on incoming packets—reducing packet loss by absorbing transient flows of data when routers are unable to handle them—gives origin to its name to send them at that precise moment.

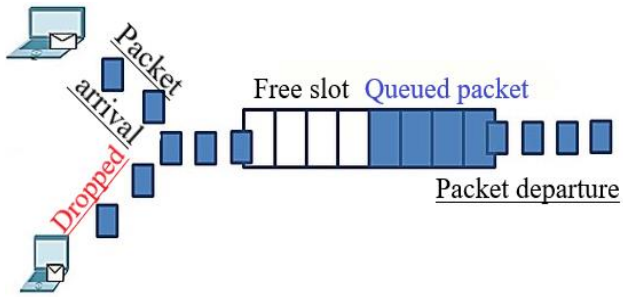


Figure 9. Tail-Drop mechanism

5.2 Random Early Detection (RED)

The Internet Engineering Task Force (IETF) has recommended the use of RED for the Internet routers. Figure 10 illustrates how RED utilizes two thresholds (*Min th, Max th*) to distinguish between various levels of network congestion.

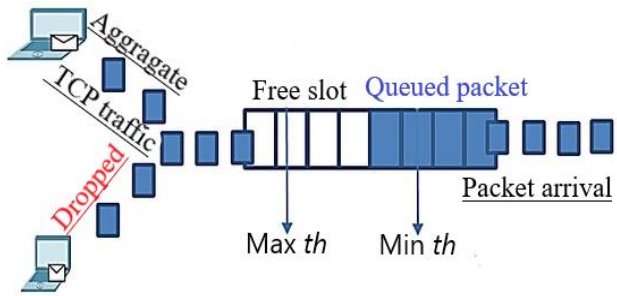


Figure 10. RED mechanism

In case of RED, incoming packets are permitted into the buffer until the queue size reaches the Min *th* threshold value. At this point, RED randomly drops a portion of the incoming packets using a linear distribution function until the queue size reaches the Max *th* threshold value, all incoming packets are dropped.

The RED utilizes the average queue size to inform its congestion control decisions, enabling it to differentiate between temporary and persistent increases in the queue size and accurately detect network congestion. However, while RED effectively addresses the worldwide synchronization issue by discarding packets at regular intervals of time, its reliance on the average queue size results in a slower congestion response. This is because the average queue size is collected over a long period. Consequently, this can lead to significant fluctuation in the queue size, untimely congestion detection and notification, and ultimately reduce the network performance due to high packet loss and prolonged queuing delays.

Proposed a RED mechanism to manage congestion at an early stage and overcome the drawbacks and limitations of Tail-Drop. One common method for preventing network congestion in networks is the RED mechanism.

Certain network conditions and requirements are addressed

by TCP congestion control protocols including Data Center TCP (DCTCP) and ECN Stream Control Transfer Protocol (SCTP). Additionally, the goal of the RED system's network traffic load monitoring is to foresee and prevent congestion. Packet loss monitoring can help identify areas of congestion and troubleshoot network performance issues.

6. METHODOLOGY AND EVALUATION OF DCTCP

Congestion control improves the current performance of flows by maximizing their throughput while maintaining queues in the network. Having small queues is important because it reduces the latency of short flows by reducing the queue delay. It also reduces packet drops and this improves network performance.

The DCTCP is a solution to maintain small queues for data centre switches and reduce dropped packets. The methodology of this paper evaluates DCTCP using Random Early Detection (RED) queue management strategy to improve efficiency in data centre networks to improve the performance of cloud environments. Using Explicit Congestion Notification (ECN) allows DCTCP to keep queue sizes short. It is believed that an intelligent queue management mechanism with ECN can improve the performance of DCTCP in a cloud computing environment by predicting transmission rates and providing quick feedback on the queue length.

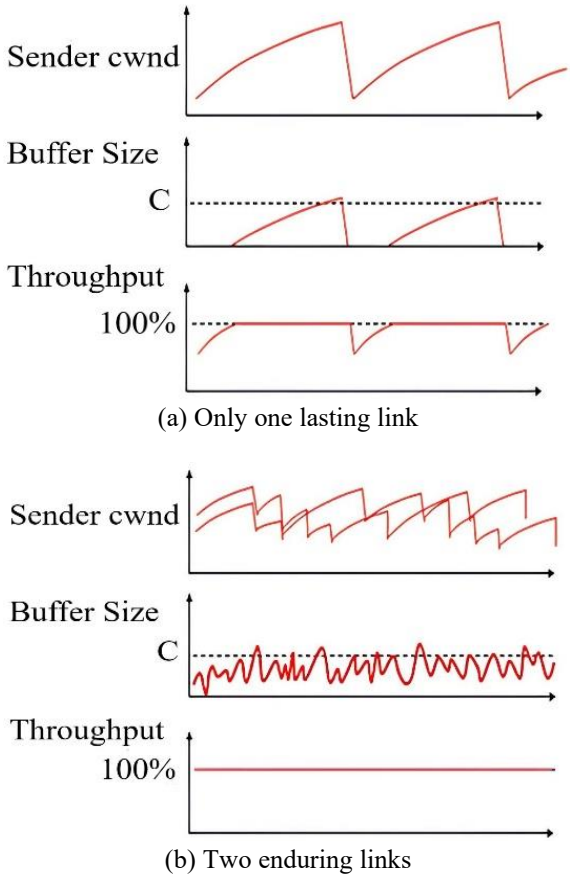
Data Centers TCP host a diverse mix of applications ranging from batch processing to user-driven workloads. The former creates large tail flows that require high-sustained throughput while the latter creates short-latency critical traces with strict completion time requirements. The performance and efficiency of these two types of traffic is extremely important in determining the feasibility of a service in a cloud environment.

To give a clear understanding of how the DCTCP operates effectively in the cloud computing environment the authors demonstrate the requirements for maintaining a short line length while achieving high or respectable throughput. To achieve this goal, it is necessary to multiply the bandwidth and round-trip time of bottleneck connection B by the buffer size E. The buffer size has to equal to otherwise greater than this value the generalization for determining the buffer size is based on the bandwidth delay product, represented by E=RTT. Figure 11 demonstrates that a single connection needs a buffer of size E=RTT×B to achieve accuracy 100% throughput. The buffer size needs to be adjusted to E=(RTT×B)/n. It is evident that insufficient buffer size can result in suboptimal performance. F=Amount of marked ACKs/Total amount of marked ACKs (1).

$$F = \frac{Number\ of\ marked\ ACKs}{Total\ number\ of\ marked\ ACKs} \quad (1)$$

For transmission rate differences, a small buffer size might be adequate.

As a result, the first fundamental principle guiding the Sender's response in DCTCP's congestion-based architecture, as illustrated in Table 1 is that should be proportional to the congestion level. This approach aims to minimize queuing requirements by requirements by reducing transmission rates and mitigating volatility. The second fundamental concept is that the actual queue length should be used to implement the ECN pattern. This provides rapid feedback, allowing for improved management of burst traffic.



(a) Only one lasting link

(b) Two enduring links

Figure 11. Queue size and cwnd are affected by long-lived connections

Table 1. Response of TCP-ECN senders against DCTCP-ECN

ECN	DCTCP	TCP
1010110111	Reduce cwnd by 40%	Reduce cwnd by 50%
0000000001	Reduce cwnd by 5%	Reduce cwnd by 50%

In DCTCP, data packets are labeled for transmission only if the true queue size is q in a switch or router exceeds the threshold K (when q>K). It is important to observe K>1/7(B Round-Trip Time).

The average fraction F from tagged packets is kept for Round-Trip Time (RTT) at the DCTCP sender.

$$\alpha \leftarrow (1-w)\times\alpha + w\times F \quad (2)$$

Eq. (3) shows that the (cwnd) window adaptably loses by a factor between 1 and 2.

$$cwnd \leftarrow (1-(\alpha/2))\times cwnd \quad (3)$$

The DCTCP sender utilizes a stream of marked ACKs to evaluate the percentage of marked packets. When the transmission rate fluctuates smoothly, DCTCP demonstrates effective performance even with small buffers. The fraction F can be obtained from the previous RTT since TCP's self-clocking mechanism aids in determining the next (RTT) based on (cwnd).

Switches use a queue management system to assign congestion to an incoming packet if the queue is long. The ECN sets the ACK packets it sends to the sender for each packet flagged as congested by the router. The sender keeps a

continuous estimate of the percentage of packets lost, and each time, it needs to reduce the size of the congestion window according to the standard TCP protocol. Thus, only a few packets encounter long queues.

7. RESULT AND DISCUSSION

This presents a significant challenge to innovation and limits evaluation to simulation tools because, to conduct practical tests in cloud data centers, there are no available experimental facilities where it is necessary to modify internal mechanisms. To assess the effectiveness of DCTCP in a network setting for cloud computing, Network Simulator 2 (ns-2) was employed. The simulation involved a bottleneck of 20 servers responding to random queries with a 10 Gbps link.The cushion contained 200 packets of 1500-byte size. TCP Newreno and DCTCP were selected with an RTT of 1 ms and a Retransmission Timeout (RTO) of 10 ms . The traffic size and arrival follow a Poisson distribution. For regular DCTCP, the thresholds (K) were set at 40, while for DCTCP over RED; the thresholds were commonly set to a minimum of 5 and a maximum of 15. In Figure 12 the connection durations are depicted for TCP Newnero and DCTCP about their respective traffic sizes.

As depicted in the graphic, DCTCP exhibits lower latency than TCP Newreno -RED for short- and medium-lived connections. However, when coupled with RED and relying on typical queue size for decisions, DCTCP performs worse, resulting in increased delay for such traffic types. The figure also demonstrates that DCTCP achieves its ability to understand and effectively handle burst traffic. Moreover, when RED is utilized, a lower delay can be achieved. However, it is worth noting that when the bottleneck link is primarily congested, TCP-related issues may still arise, making the performance of DCTCP comparable to that of traditional TCP with their fair share of throughput well on short and medium-lived connections with short and medium-lived queue lengths, although DCTCP over RED provides acceptable delay on long-lived connections in comparison.

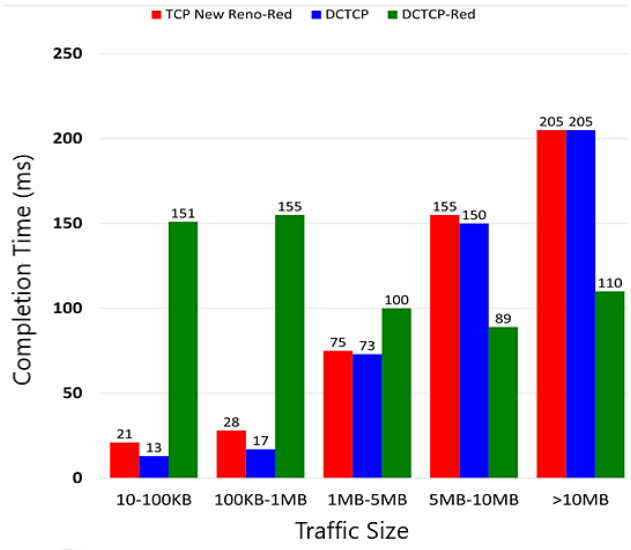


Figure 12. Time to finish DCTCP, TCP Newnero-RED

We repeated this test with up to 500 flows and found that DCTCP was fast enough to check all flows.

Limiting the ACK before the timer expires unless the occurrence of network congestion is received reduces the congestion window size to a minimum and the threshold.

Figure 12 showed that DCTCP flows converge quickly with their fair share of throughput well on short—and medium-lived connections with short—and medium-lived queue lengths, although DCTCP over RED provides acceptable delay on long-lived connections in comparison.

8. CONCLUSIONS

The DCTCP is a popular cloud transport protocol capable of dealing with TCP queue backlog issues. However, its performance deteriorates significantly when dozens of servers respond to one cluster simultaneously. DCTCP is based on Explicit Congestion Notification (ECN), a feature now available on commodity switches. We designed DCTCP to address TCP queue build-up issues. With multiple reactions, DCTCP prepares the traffic volume derived from a series of ECN tags, allowing it to react early to congestion.

Experimental results showed different capacities in five experiments: 10-100MB, 100KB - 1MB, 1MB - 5MB, 5MB - 10MB and the largest 10MB. Evaluation results showed that there is a trade-off between using DCTCP and RED. On the one hand, DCTCP over RED is unsuitable for short and medium-lived connections because it introduces a longer delay in completion time than the regular DCTCP mechanism. On the other hand, it represents an acceptable delay for long-lived connections, but at the expense of throughput, DCTCP achieves its design goals.

The DCTCP is evaluated independently and compared to the Random Early Detection (RED) queue management system and TCP Newreno. Due to its low delays, the evaluation findings demonstrate that DCTCP effectively manages Incast issues and meets deadlines for short- and medium-lived traffic. Furthermore, DCTCP provides high throughput for persistent traffic, albeit with slightly higher latency, compared to when used in conjunction with RED, which significantly reduces delay. DCTCP maintains small queue sizes by utilizing Explicit Congestion Notification (ECN). On the other hand, RED penalizes transient traffic as it cannot quickly and promptly alleviate congestion since it relies on average queue lengths. To enhance DCTCP and the cloud computing environment's performance, an intelligent queue management system capable of anticipating the sending rate should be used in conjunction with ECN.

REFERENCES

[1] Kanniga Devi, R., Gurusamy, M., Vijayakumar, P. (2020). An efficient cloud data center allocation to the source of requests. Journal of Organizational and End User Computing, 32(3): 23-36. https://doi.org/10.4018/JOEUC.2020070103

[2] Nazir, R., Ahmed, Z., Ahmad, Z., Shaikh, N., Laghari, A., Kumar, K. (2020). Cloud computing applications: A review. EAI Endorsed Transactions on Cloud Systems, 6(17): 164667. https://doi.org/10.4108/eai.22-5-2020.164667

[3] Odun-Ayo, I., Eweoya, I., Toro-Abasi, W., Bogle, S. (2019). Networking for cloud and data centre-A systematic mapping study. International Journal of

Engineering Research and Technology, 12(12): 2559-2573.
[4] Katal, A., Dahiya, S., Choudhury, T. (2023). Energy efficiency in cloud computing data centers: A survey on software technologies. Cluster Computing, 26(3): 1845-1875. https://doi.org/10.1007/s10586-022-03713-0
[5] Guo, Z., Liu, S., Zhang, Z.L. (2020). Traffic control for RDMA-enabled data center networks: A survey. IEEE Systems Journal, 14(1): 677-688. https://doi.org/10.1109/JSYST.2019.2936519
[6] Noormohammadpour, M., Raghavendra, C.S. (2018). Datacenter traffic control: understanding techniques and tradeoffs. IEEE Communications Surveys and Tutorials, 20(2): 1492-1525. https://doi.org/10.1109/COMST.2017.2782753
[7] Burgess, S., Sellitto, C., Cox, C., Buultjens, J., Bingley, S. (2017). An innovation diffusion approach to examining the adoption of social media by small businesses: An Australian case study. Pacific Asia Journal of the Association for Information Systems, 9(3): 1-24. https://doi.org/10.17705/1pais.09301
[8] Syed, A.S., Sierra-Sosa, D., Kumar, A., Elmaghraby, A. (2021). Iot in smart cities: A survey of technologies, practices and challenges. Smart Cities, 4(2): 429-475. https://doi.org/10.3390/smartcities4020024
[9] Tamizhselvi, S.P., Muthuswamy, V. (2021). Delay - aware bandwidth estimation and intelligent video transcoder in mobile cloud. Peer-to-Peer Networking and Applications, 14(4): 2038-2060. https://doi.org/10.1007/s12083-021-01134-1
[10] AlZoman, R.M., Alenazi, M.J.F. (2021). A comparative study of traffic classification techniques for smart city networks. Sensors, 21(14): 4677. https://doi.org/10.3390/s21144677
[11] Liang, Y., Lu, M., Shen, Z.J.M., Tang, R. (2021). Data center network design for internet-related services and cloud computing. Production and Operations Management, 30(7): 2077-2101. https://doi.org/10.1111/poms.13355
[12] Hadi, T.H., Kadum, J., Kadhim, Q.K., Ahmed, S.T. (2024). An enhanced cloud storage auditing approach using Boneh-Lynn- Shacham's signature and automatic blocker protocol. Ingénierie Des Systèmes d'Information, 29(1): 261–268. https://doi.org/https://doi.org/10.18280/isi.290126
[13] Kadhim, Q.K., Yusof, R., Mahdi, H.S., Selamat, S.R. (2017). The effectiveness of random early detection in data center transmission control protocol - Based cloud computing networks. International Journal on Communications Antenna and Propagation (IRECAP), 7(October): 1-7. https://doi.org/10.15866/irecap.v7i5.10104
[14] Shah, Z. (2018). Mitigating TCP incast issue in cloud data centres using software-defined networking (SDN): A survey. KSII Transactions on Internet and Information Systems, 12(11): 5179-5202. https://doi.org/10.3837/tiis.2018.11.001
[15] Polese, M., Chiariotti, F., Bonetto, E., Rigotto, F., Zanella, A., Zorzi, M. (2019). A survey on recent advances in transport layer protocols. IEEE Communications Surveys and Tutorials, 21(4): 3584-3608. https://doi.org/10.1109/COMST.2019.2932905
[16] Alsultani, H.S.M., Kanaan, Q., Khudhair, I.Y. (2018). Empirical investigation of TCP incast congestion in Wireless cloud computing networks. Journal of Computer Science, 14(5): 663-672. https://doi.org/10.3844/jcssp.2018.663.672
[17] Ahmad, S.Z., Khalid, S. (2023). Optimizing data transport efficiency in datacenters through traffic shaping of BBR congestion control. Journal of Communications, 18(2): 97–108. https://doi.org/10.12720/jcm.18.2.97-108
[18] Alipio, M., Tiglao, N.M., Bokhari, F., Khalid, S. (2019). TCP incast solutions in data center networks: A classification and survey. Journal of Network and Computer Applications, 146(15): 102421. https://doi.org/10.1016/j.jnca.2019.102421
[19] Wang, H., Shen, H., Liu, G. (2017). Swarm-based incast congestion control in datacenters serving web applications. Annual ACM Symposium on Parallelism in Algorithms and Architectures, Part F1293(September 2018): 217-226. https://doi.org/10.1145/3087556.3087559
[20] Khadhiml, B.J., Ahmed, S.T., Abdulqader, R.M. (2019). Transmission control protocol (TCP) incast and outcast issues in cloud computing. Journal of Advanced Research in Dynamical and Control Systems - JARDCS, 13(May): 2026-2032.
[21] Abdelmoniem, A.M., Bensaou, B., Barsoum, V. (2018). IncastGuard: An efficient TCP-Incast mitigation mechanism for cloud networks. 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, pp. 1-6. https://doi.org/10.1109/GLOCOM.2018.8647878
[22] Kheirkhah, M., Lee, M. (2020). A solution to MPTCP's inefficiencies under the incast problem for Data Center Networks. Computer Communications, 161: 238-247. https://doi.org/10.1016/j.comcom.2020.07.034
[23] Arun Selvi, K., Kumar, K., Ramalakshmi, K., Sathiya, A. (2019). A hybrid framework for TCP incast congestion control in data center networks. International Journal of Recent Technology and Engineering, 8(2): 798-806. https://doi.org/10.35940/ijrte.F2701.078219
[24] Nikzad, M., Jamshidi, K., Bohlooli, A., Faqiry, F.M. (2022). An accurate retransmission timeout estimator for content-centric networking based on the Jacobson algorithm. Digital Communications and Networks, 8(6): 1085–1093. https://doi.org/10.1016/j.dcan.2022.03.006
[25] Gomez, J., Kfoury, E.F., Crichigno, J., Srivastava, G. (2022). A survey on TCP enhancements using P4-programmable devices. Computer Networks, 212: 109030. https://doi.org/10.1016/j.comnet.2022.109030
[26] Addanki, V., Linguaglossa, L., Roberts, J., Rossi, D. (2018). Controlling software router resource sharing by fair packet dropping. 17th International IFIP TC6 Networking Conference, Networking, Zurich, Switzerland, pp. A9-A10
[27] Sarkar, N.I., Ammann, R., Zabir, S.M.S. (2022). Analyzing TCP performance in high bit error rate using simulation and modeling. Electronics, 11(14): 1-20. https://doi.org/10.3390/electronics11142254
[28] Menikkumbura, D., Taheri, P., Vanini, E., Fahmy, S., Eugster, P., Edsall, T. (2023). Congestion control for datacenter networks: A control-theoretic approach. IEEE Transactions on Parallel and Distributed Systems, 34(5): 1682-1696. https://doi.org/10.1109/TPDS.2023.3259799
[29] Shan, D., Jiang, L., Zhang, P., Jiang, W., Li, H., Tang, Y., Ren, F. (2023). Enforcing fairness in the traffic policer

among heterogeneous congestion control algorithms. IEEE/ACM Transactions on Networking, 1-16. https://doi.org/10.1109/TNET.2023.3276410
[30] Astudillo León, J.P., de la Cruz Llopis, L.J., Rico-Novella, F.J. (2023). A machine learning based Distributed congestion control protocol for multi-hop wireless networks. Computer Networks, 231: 109813. https://doi.org/10.1016/j.comnet.2023.109813
[31] Jowkarishasaltaneh, F., But, J. (2022). An analysis of MPTCP congestion control. Telecom, 3(4): 581-609. https://doi.org/10.3390/telecom3040033
[32] Nandhini, C., Gupta, G.P. (2023). Exploration and evaluation of congestion control algorithms for data center networks. SN Computer Science, 4(5): 5-19. https://doi.org/10.1007/s42979-023-02016-4
[33] Bul'ajoul, W., James, A., Shaikh, S. (2019). A new architecture for network intrusion detection and prevention. IEEE Access, 7: 18558-18573. https://doi.org/10.1109/ACCESS.2019.2895898
[34] Tawfeeg, T.M., Yousif, A., Hassan, A., Alqhtani, S.M., Hamza, R., Bashir, M.B., Ali, A. (2022). Cloud dynamic load balancing and reactive fault tolerance techniques: A systematic literature review (SLR). IEEE Access, 10: 71853-71873. https://doi.org/10.1109/ACCESS.2022.3188645
[35] Baklizi, M. (2019). FLACC: Fuzzy logic approach for congestion control. International Journal of Advanced Computer Science and Applications, 10(7): 43-50. https://doi.org/10.14569/ijacsa.2019.0100707
[36] Tsiknas, K.G., Aidinidis, P.I., Zoiros, K.E. (2021). On the fairness of DCTCP and CUBIC in cloud data center networks. 2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, pp. 1-4. https://doi.org/10.1109/MOCAST52088.2021.9493352