2012 International Symposium on Cloud and Services Computing

TCP Variants for Data Center Networks: A Comparative Study

Rohit P. Tahiliani
Dept. of Computer Science and Engineering
NMAM Institute of Technology, Nitte
Karnataka, India 574110
rohit.tahil@gmail.com

Mohit P. Tahiliani and K. Chandra Sekaran
Dept. of Computer Science and Engineering
National Institue of Technology Karnataka,
Surathkal, Karnataka, India 575025
tahiliani.nitk@gmail.com, kchnitk@gmail.com

***Abstract*—Transmission Control Protocol (TCP) has been the workhorse of the Internet ever since its inception. The success of the Internet, infact, can be partly attributed to the congestion control mechanisms implemented in TCP. Though the scale of the Internet and its usage increased exponentially in recent past, TCP has evolved to keep up with the changing network conditions and has proven to be scalable and robust. However, the performance of TCP in Data Center Networks has been a major concern recently because it leads to impairments such as TCP Incast, TCP Outcast, Queue build-up and Buffer pressure. With cloud computing becoming an important part of the foreseeable future, it has become extremely important to enhance the performance of TCP in Data Center Networks and overcome these impairments. In this paper, we describe the above mentioned impairments in brief and then compare the TCP variants proposed so far to ovecome these impairments in Data Center Networks. The advantages and shortcomings of every TCP variant are highlighted with respect to its efficacy and the deployment complexity. A few open issues related to TCP's performance in Data Center Networks are also discussed.**

***Index Terms*—Data Center Networks, TCP Incast, TCP Outcast, Queue build-up, Buffer pressure.**

I. Introduction

Internet over the past few years has transformed from an experimental system into a gigantic and decentralized source of information. Data centers form the backbone of the Internet and host diverse applications ranging from social networking to web search and web hosting to advertisements. Transmission Control Protocol (TCP) is one of the most dominant transport protocols and is widely used by a large variety of Internet applications and hence, constitutes majority of the traffic in Data Center Networks [1]. Data center environment, however, is largely different than that of the Internet e.g., the Round Trip Time (RTT) in Data Center Networks can be as less as 250μs in the absence of queuing [2]. The reason is that Data Center Networks are well designed and layered to achieve high-bandwidth and low-latency.

The traffic in Data Center Networks can be classified into three types [1]: (i) Mice traffic - the queries form the mice traffic (e.g. google search, facebook updates, etc). Majority of the traffic in a data center network is query traffic and its data transmission volume is usually less. (ii) Cat traffic - the control state and co-ordination messages form the cat traffic (e.g. small and medium sized file downloads, etc) and (iii) Elephant traffic - the large updates form the elephant traffic (e.g. anti-virus updates, movie downloads, etc). The different traffic types in Data Center Networks, their applications and performance requirements are summarized in Table I.

TABLE I
Data Center Traffic: Applications and Performance Requirements

Traffic Type	Examples	Requirements
Mice traffic (< 100KB)	Google Search, Facebook	Short response times
Cat traffic (100KB-5MB)	Picasa, YouTube, Facebook photos	Low latency
Elephant traffic (> 5MB)	Software updates, Video On-demand	High throughput

Thus, bursty query traffic, delay sensitive cat traffic and throughput sensitive elephant traffic co-exist in Data Center Networks. Therefore, the three basic requirements of the data center transport are high burst tolerance, low latency and high throughput [1]. The state-of-the-art TCP fails to satisfy these requirements together within the time boundaries because of impairments such as TCP Incast [2], TCP Outcast [3], Queue build-up [1] and Buffer pressure [1].

Recently, a few TCP variants have been proposed for Data Center Networks. The major goal of these TCP Variants is to overcome the above mentioned impairments and improve the performance of TCP in Data Center Networks. This paper describes each of the above mentioned problems in brief, followed by a comparative study of TCP variants that aim to overcome these problems. Although a few other transport protocols have been proposed for Data Center Networks, we restrict the scope of this paper to TCP variants because TCP is the most widely deployed transport protocol in modern operating systems.

The rest of the paper is organized as follows: Section II describes the challenges for TCP in Data Center Networks. Section III presents a comparative study of TCP variants designed for Data Center Networks along with their advantages and shortcomings. Section IV summarizes the comparative study. Section V discusses a few open issues related to the performance of TCP in Data Center Networks and Section VI concludes the paper.

978-0-7695-4931-6/12 $26.00 © 2012 IEEE
DOI 10.1109/ISCOS.2012.38

II. Challenges for TCP in Data Center Networks

The four problems described below are the major challenges faced by TCP in Data Center Networks. We start with the most popular one, TCP Incast.

A. TCP Incast

TCP Incast has been defined as the pathological behaviour of TCP that results in gross under-utilization of the link capacity in various many-to-one communication patterns [4], e.g. partition/aggregate application pattern as shown in Fig. 1. Such many-to-one communication pattern is the foundation of numerous large scale applications like web search, MapReduce, social network content composition, advertisement selection, etc [1], [5].

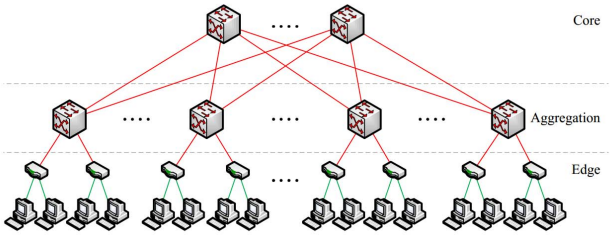

Fig. 1. Partition/Aggregate Application Structure

In such many-to-one communication patterns, an aggregator issues data requests to multiple worker nodes. The worker nodes upon receiving the request, concurrently transmit a large amout of data to the aggregator (see Fig. 2). The data from all the worker nodes traverse a bottleneck link in many-to-one fashion. The probability that all the worker nodes send the reply at the same time is high because of the tight time bounds. Therefore, the packets from these nodes happen to overflow the buffers of Top of the Rack (ToR) switches and thus, lead to packet losses. This phenomenon is also known as "synchronized mice collide" [1]. Moreover, as the number of concurrent worker nodes increases, the perceived application level throughput at the aggregator decreases due to a large number of packet losses. The lost packets are retransmitted only after the Retransmit TimeOut (RTO), which is generally in the order of few milliseconds. It must be noted that "Fast Retransmit" mechanism may not be possible for these applications since the data transmission volume of such traffic is quite less and hence, there are very few packets in the entire flow. As a result, the sender may not get sufficient duplicate acknowledgements (*dupacks*) to trigger a Fast Retransmit.

A lot of solutions, ranging from application layer solutions to transport layer solutions and link layer solutions have been proposed recently to overcome the TCP Incast problem. Ren et al [6] provides a detailed analysis and summary of all such solutions. This paper, instead, focuses mainly on analyzing TCP based solutions to overcome TCP Incast and other performance problems in Data Center Networks.

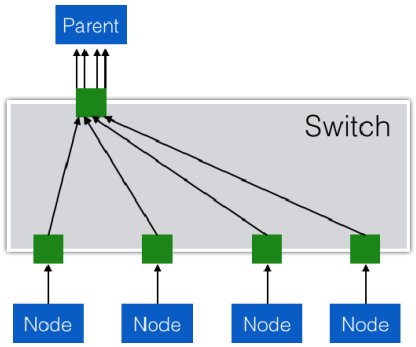

Fig. 2. TCP Incast

B. TCP Outcast

When a large set of flows and a small set of flows arrive at two different input ports and compete for the same bottleneck output port, the small set of flows lose out on their throughput share significantly. This phenomenon has been termed as TCP Outcast [3] and mainly occurs in data center switches that employ drop-tail queues. Drop-tail queues leads to consecutive packet drops from one port and hence, cause frequent TCP timeouts. This property of drop-tail queues is termed as "Port Blackout" [3] and it significantly affects the performance of small flows because frequent timeouts lead to high latencies and thus, poor quality results. Fig. 3 shows an example scenario of port blackout where A and B are input ports whereas C is the common output port. The figures shows that packets arriving at Port B are successfully queued whereas those arriving at Port A are dropped consecutively.

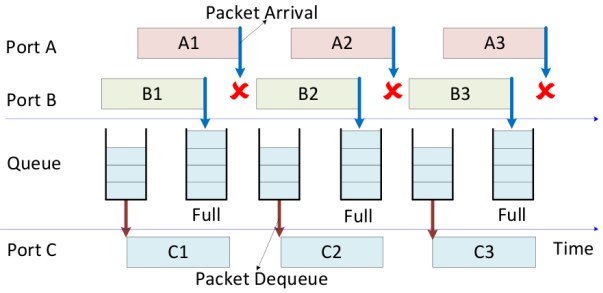

Fig. 3. Example scenario of Port Blackout [3]

It is well known that the throughput of a TCP flow is inversely proportional to the RTT of that flow. This behavior of TCP leads to RTT-bias i.e., flows with low RTT achieve larger share of bandwidth than the flows with high RTT. However, it has been observed that due to TCP Outcast problem in Data Center Networks, TCP exhibits *Inverse RTT-bias* [3] i.e., flows with low RTT are outcasted by flows with high RTT.

The two main factors that cause TCP Outcast are: (i) the usage of drop-tail queues in switches and (ii) many-to-one communication pattern which leads to a large set of flows and a small set of flows arriving at two different input ports and competing for the same bottleneck output port. Both these

factors are quite common in Data Center Networks since majority of the switches employ drop-tail queues and many-to-one communication pattern is the foundation of many cloud applications.

A straightforward approach to mitigate TCP Outcast is to use queuing mechanisms other than drop-tail e.g., Random Early Detection (RED) [7], Stochastic Fair Queue (SFQ) [3], etc. Another possible approach is to minimize the buffer occupancy at the switches by designing efficient TCP congestion control laws at the end hosts.

C. Queue Buildup

Due to the diverse nature of cloud applications, mice traffic, cat traffic and elephant traffic co-exist in a Data Center Network. The long lasting and greedy nature of elephant traffic drives the network to the point of extreme congestion and overflows the bottleneck buffers. Thus, when both mice traffic and elephant traffic traverse through the same route, the performance of mice traffic is significantly affected due to the presence of the elephant traffic [1].

Following are two ways in which the performance of mice traffic is degraded due to the presence of elephant traffic [1]: (i) since most of the buffer is occupied by elephant traffic, there is a high probability that the packets of mice traffic get dropped. The implications of this situation are similar to that of TCP Incast because the performance of mice traffic is largely affected by frequent packet losses and hence, the timeouts. (ii) the packets of mice traffic, even when none are lost, suffer from increased queuing delay as they are in queue behind the packets of elephant traffic. This problem has been termed as Queue build-up.

Queue build-up problem can be solved only by minimizing the queue occupancy in the Data Center Network switches. Most of the existing TCP variants employ reactive approach towards congestion control and hence, fail to minimize the queue occupancy. A proactive approach, instead, is desired to minimize the queue occupancy and overcome the problem of queue build-up.

D. Buffer pressure

Buffer pressure is yet another impairment caused by the long lasting and greedy nature of elephant traffic. When both mice trafffic and elephant traffic co-exist on the same route, most of the buffer space is occupied by packets from the elephant traffic. This leaves a very little room to accommodate the burst of mice traffic packets arising out of many-to-one communication pattern. The result is that large number of packets from mice traffic are lost, leading to poor performance. Moreover, majority of the traffic in Data Center Networks is bursty [1] and hence, packets of mice traffic get dropped frequently because the elephant traffic lasts for a longer time and keeps most of the buffer space occupied.

Like Queue build-up, Buffer pressure problem too can be solved by minimizing the buffer occupancy in the switches.

III. TCP Variants for Data Center Networks

A. Fine grained TCP RTO [2]

The default value of minimum RTO in TCP is generally 200ms. This value of RTO is suitable for Internet like scenarios where the average RTT is in order of hundreds of milliseconds. However, it is significantly larger than the average RTT in a data center which is in the order of a few micro-seconds. Large number of packet losses due to TCP Incast, TCP Outcast, Queue build-up and Buffer pressure result in frequent timeouts and in turn, lead to missed deadlines and significant degradation in the performance of TCP. Phanishayee et al show that reducing the minimum RTO from 200ms to 200μs significantly alleviates the problems of TCP in simulations and improves the overall throughput by several orders of magnitude.

Advantages: The major advantage of this approach is that it requires minimum modification to the traditional TCP and hence, can be easily deployed without any further complexity.

Shortcomings: The real time deployment of fine grained timers is a challenging issue because the present operating systems lack the high-resolution timers required for such low RTO values. Moreover, fine grained RTOs may be not suitable for servers that communicate to clients through the Internet. Apart from the implementations issues of fine grained timers, it must be noted that this approach of eliminating drawbacks of TCP in Data Center Networks is a reactive approach. It tries to reduce the impact of a packet loss rather than avoiding the packet loss in the first place. Hence, though this approach minimizes the implications of TCP Incast, it cannot overcome the problems such as Queue build-up, Buffer pressure or even TCP Outcast.

B. Fine grained TCP RTO + Delayed ACKs disabled [2]

Delayed ACKs are mainly used for reducing the overhead of ACKs on the reverse path. When delayed ACKs are enabled, the receiver sends only one ACK for every two data packets received. If only one packet is received, the receiver waits for delayed ACK timeout period before sending an ACK. This timeout period is usually 40ms. This scenario may lead to spurious retransmissions if fine grained RTO timers (as explained in the previous section) are deployed. The reason is that receiver waits for 40ms before sending an ACK for the received packet and by that time, fine grained RTO which is in order of few microseconds, expires and forces the sender to retransmit the packet. Thus, either the delayed ACK timeout period must be reduced to a few microseconds or must be completely disabled while using fine grained RTOs to avoid such spurious retransmissions. This approach further enhances the TCP throughput in Data Center Networks.

Advantages: It has been shown in [2] that reducing the delayed ACK timeout period to 200μs while using fine grained RTO achieves far better throughput than the throughput obtained when delayed ACKs are enabled. Moreover, completely disabling the delayed ACKs while using fine grained RTO further improves the overall TCP throughput.

Shortcomings: The shortcomings of this approach are exactly similar to that of the previous one because this approach is an undesired side-effect of the previous approach.

C. DCTCP: Data Center TCP [1]

Additive Increase Multiplicative Decrease (AIMD) is the cornerstone of TCP congestion control algorithms. When an acknowledgement (ACK) is received in AIMD phase, the congestion window (*cwnd*) is increased as shown in (1). This is known as Additive Increase phase of the AIMD algorithm.

$$cwnd = cwnd + \frac{1}{cwnd} \tag{1}$$

When congestion is detected either through *dupacks* or Selective Acknowledgement (SACK), *cwnd* is updated as shown in (2). This is known as Multiplicative Decrease phase of the AIMD algorithm.

$$cwnd = \frac{cwnd}{2} \tag{2}$$

DCTCP employs an efficient multiplicative decrease mechanism which reduces the *cwnd* based on the *amount of congestion* in the network rather than reducing it by half. DCTCP leverages Explicit Congestion Notification (ECN) mechanism [8] to extract multi-bit feedback on the *amount of congestion* in the network from the single bit stream of ECN marks. On receiving the congestion notification via ECN, the *cwnd* in DCTCP is reduced as shown in (3).

$$cwnd = cwnd \times (1 - \frac{\alpha}{2}) \tag{3}$$

where α ($0 \leqslant \alpha \leqslant 1$) is an estimate of the fraction of packets that are marked and is calculated as shown in (4). F in (4) is the fraction of packets that are marked in the previous *cwnd* and g ($0 < g < 1$) is the exponential weighted moving average constant. Thus, when congestion is low (α is near 0), *cwnd* is reduced slightly and when congestion is high (α is near 1), *cwnd* is reduced by half, just like traditional TCP.

$$\alpha = (1 - g) \times \alpha + g \times F \tag{4}$$

The major goal of DCTCP algorithm is to achieve low latency (desirable for mice traffic), high throughput (desirable for elephant traffic) and high burst tolerance (to avoid packet losses due to incast). DCTCP achieves these goals by reacting to the *amount of congestion* rather than blindly reducing the *cwnd* by half. DCTCP uses a marking scheme at switches that sets the Congestion Experienced (CE) codepoint [8] of packets as soon as the buffer occupancy exceeds a fixed predetermined threshold (17% as mentioned in [9]). The DCTCP source reacts by reducing the window by a factor that depends on the fraction of marked packets: the larger the fraction, the bigger the decrease factor.

Advantages: DCTCP is a novel TCP variant which alleviates TCP Incast, Queue-up and Buffer pressure problems in Data Center Networks. It requires minor modifications to the original design of TCP and ECN to achieve these performance benefits. DCTCP employs a proactive behavior i.e., it tries to avoid packet loss. It has been shown in [1] that when DCTCP uses fine grained RTO, it further reduces the implications of TCP Incast and also improves the scalability of DCTCP. The stability, convergence and fairness properties of DCTCP [9] make it a suitable solution for implementation in Data Center Networks. Moreover, DCTCP is already implemented in latest Microsoft Windows Server operating systems.

Shortcomings: The performance of DCTCP falls back to that of TCP when the degree of Incast increases beyond 35 i.e., if there are more than 35 worker nodes sending data to the same aggregator, DCTCP fails to avoid Incast and its performance is similar to that of TCP. However, authors show that dynamic buffer allocation at the switch can scale DCTCP's performance to handle upto 40 worker nodes in parallel. Moreover, apart from the scalability issues, it is not clear whether DCTCP can alleviate the problem of TCP Outcast. DCTCP utilizes minimum buffer space in the switches, which infact, is a desirable property to avoid TCP Outcast. However, a few more experiments are required to conclude whether DCTCP can overcome the problem of TCP Outcast.

D. ICTCP: Incast Congestion Control for TCP [5]

Like DCTCP, the main idea of ICTCP is to avoid packet losses due to congestion rather than recovering from the packet losses. It is well known that a TCP sender can send a minimum of advertised window (*rwnd*) and congestion window (*cwnd*) (i.e. *min(rwnd, cwnd)*). ICTCP leverages this property and efficiently varies the *rwnd* to avoid TCP Incast. The major contributions of ICTCP are: (a) The available bandwidth is used as a quota to co-ordinate the *rwnd* increase of all connections. (b) Per flow congestion control is performed independently and (c) *rwnd* is adjusted based on the ratio of difference between expected throughput and measured throughput over expected throughput. Moreover, live RTT is used for the throughput estimation.

Advantages: Unlike DCTCP, ICTCP does not require any modifications at the sender side (i.e. worker nodes) or network elements such as routers, switches, etc. Instead, ICTCP requires modification only at the receiver side (i.e. an aggregator). This approach is adopted to retain the backward compatibility and make the algorithm general enough to handle the Incast congestion in future high-bandwidth, low-latency networks.

Shortcomings: Although authors of ICTCP show that they achieve almost zero timeout and high throughput, the scalability of ICTCP is a major concern i.e., how to handle Incast congestion when there are extremely large number of flows because ICTCP employs per flow congestion control. Another limitation of ICTCP is that it assumes that both the sender and the receiver are under the same switch, which might not be the case always. Moreover, it is not known how much buffer space is utilized by ICTCP. Hence, it is difficult to conclude whether ICTCP can handle Queue build-up, Buffer pressure and TCP Outcast problems in a Data Center Network.

E. IA-TCP: Incast Avoidance algorithm for TCP [10]

Unlike DCTCP and ICTCP which use window based congestion control, IA-TCP uses rate based congestion control algorithm to control the total number of packets injected in the network. The motivation behind selecting rate based congestion control mechanism is that window based congestion control mechanisms in Data Center Networks have limitations in terms of scalability i.e., number of senders in parallel.

The main idea of IA-TCP is to limit the total number of outstanding data packets in the network so that it does not exceed the bandwidth-delay product (BDP). IA-TCP employs ACK regulation at the receiver and like ICTCP, leverages the advertised window (*rwnd*) field of the TCP header to regulate the *cwnd* of every worker node. The minimum *rwnd* is set to 1 packet. However, if large number of worker nodes send packets with respect to a minimum *rwnd* of 1 packet, the total number of outstanding packets in the network may exceed the link capacity. In such scenarios, IA-TCP adds delay, Δ, to the ACK packet to ensure that the aggregate data rate does not exceed the link capacity. Moreover, IA-TCP also uses delay, Δ, to avoid the synchronization among the worker nodes while sending the data.

Advantages: Like ICTCP, IA-TCP also requires modification only at the receiver side (i.e. an aggregator) and does not require any modifications at the sender or network elements. IA-TCP achieves high throughput and significantly improves the query completion time. Moreover, the scalability of IA-TCP is clearly demonstrated by configuring upto 96 worker nodes sending data in parallel.

Shortcomings: Like ICTCP, it is not clear how much buffer space is utilized by IA-TCP. Hence, it is difficult to conclude whether IA-TCP can handle Queue build-up, Buffer pressure and TCP Outcast problems in a Data Center Network.

F. D^{2}TCP: Deadline-aware Datacenter TCP [11]

D^{2}TCP is a novel TCP-based transport protocol which is specifically designed to handle high burst situations. Unlike other TCP variants (DCTCP, ICTCP and IA-TCP) which are deadline-agnostic, D^{2}TCP is deadline-aware. D^{2}TCP uses a distributed and reactive approach for bandwidth allocation and employs a novel deadline-aware congestion avoidance algorithm which uses ECN feedback and deadlines to vary the sender's *cwnd* via a gamma-correction function [11].

D^{2}TCP does not maintain per flow information and instead, inherits the distributed and reactive nature of TCP while adding deadline-awareness to it. Similarly, D^{2}TCP employs its congestion avoidance algorithm by adding deadline-awareness to DCTCP. The main idea, thus, is that far-deadline flows back-off aggressively and the near-deadline flows back-off only a little or not at all.

Advantages: The novelty of D^{2}TCP lies in the fact that it is easily deployable, avoids TCP Incast as well as Queue build-up and has high burst tolerance because it is built upon DCTCP. In addition, it is deadline-aware and reduces the fraction of missed deadlines upto 75% as compared to DCTCP.

Shortcomings: The shortcomings of D^{2}TCP are exactly similar to those of DCTCP. The major concern is scalability in terms of number of worker nodes in parallel. Further experiments are required to conclude whether D^{2}TCP can overcome TCP Outcast problem.

IV. Summary

Table II summarizes the comparative study of TCP variants proposed for Data Center Networks. Apart from the novelty of the proposed TCP variant, the table also highlights the deployment complexity of each protocol. The protocols which require modifications in sender, receiver and switch are considered as hard to deploy. The ones which require modification only at the sender or receiver are considered as easy to deploy.

Apart from the above mentioned parameters, the summary also includes which problems amongst TCP Incast, TCP Outcast, Queue build-up and Buffer pressure are alleviated by each TCP variant. The details regarding the tools used / approach of implementation adopted by the authors are also listed.

V. Open Issues

As discussed throughout the paper, several modifications have been proposed to the original design of TCP. However, there is an acute need to further optimize the performance of TCP variants discussed above. A few open issues are listed below:

- An experimental evaluation and comparison of the above mentioned TCP variants over a wide range of Data Center Network scenarios is highly desired to confirm their suitability and sustainability in Data Center Networks.
- DCTCP, ICTCP and D^{2}TCP have scalability issues i.e., if the number of worker nodes increases beyond 40, these TCP variants fail to avoid TCP Incast. The performance degradation in such scenarios can be significantly minimized if these TCP variants are modified to use fine grained RTO and delayed ACKs are disabled. Though this has been already done by the authors of DCTCP in [1], it has not been carried out with respect to ICTCP and D^{2}TCP. Thus, the robustness of ICTCP and D^{2}TCP in Data Center Networks can be further enhanced by coupling each with fine grained RTO + disabled delayed ACK approach.
- The buffer space utilization while using ICTCP and IA-TCP needs further investigation to ensure these protocols overcome the problems of Queue build-up and Buffer pressure. Moreover, the performance of ICTCP and IA-TCP has not been analyzed when switches employ Active Queue Management (AQM) mechanisms such as RED, SFQ, etc.
- Except D^{2}TCP, all other TCP variants are deadline-agnostic. Meeting deadlines is the most important requirement in Data Center Networks. While D^{2}TCP proposes once approach of meeting deadlines, a few more approaches need to be explored to improve the overall performance of Data Center Networks.

TABLE II
SUMMARY OF TCP VARIANTS PROPOSED FOR DATA CENTER NETWORKS

TCP Variants proposed for Data Center Networks	Modifies Sender	Modifies Receiver	Modifies Switch	Solves TCP Incast	Solves TCP Outcast	Solves Queue build-up	Solves Buffer pressure	Implementation
Fine grained TCP RTO	✓	x	x	✓	x	x	x	*ns*-2
Fine grained TCP RTO + Delayed ACKs disabled	✓	x	x	✓	x	x	x	*ns*-2
DCTCP	✓	✓	✓	✓	x	✓	✓	Testbed and *ns*-2
ICTCP	x	✓	x	✓	x	x	x	Testbed
IA-TCP	x	✓	x	✓	x	x	x	*ns*-2
D^2TCP	✓	✓	✓	✓	x	✓	✓	Testbed and *ns*-3

- A convincing solution to TCP Outcast problem is unavailable. An optimal solution to overcome TCP Outcast must ensure minimal buffer occupancy at the switch and usage of efficient AQM mechanisms rather than simple drop-tail mechanisms.
- All the TCP variants discussed in the paper above have not yet been tested in TCP Outcast scenarios. An experimental analysis of the same is highly desired.

VI. CONCLUSIONS

Data Centers in the present scenario house a plethora of Internet applications. These applications are diverse in nature and have various performance requirements. Majority of these applications use many-to-one communication pattern to gain performance efficiency. TCP, which has been a mature transport protocol of Internet since past several decades, suffers from performance impairments such as TCP Incast, TCP Outcast, Queue build-up and Buffer pressure in Data Center Networks.

In this paper, we have described each of the above mentioned impairment in brief along with the causes and possible approaches to mitigate them. Moreover, we have carried out a comparative study of TCP variants which have been specifically designed for Data Center Networks and the advantages and shortcomings of each TCP variant are highlighted. The study is summarized by briefly listing out the following for every TCP variant: the deployment complexity, efficiency against the above mentioned impairments and the tools used for implementation. A few open issues related to the performance of the TCP variants proposed for Data Center Networks are also discussed.

REFERENCES

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," *SIGCOMM Computer Communications Review*, vol. 40, no. 4, pp. 63–74, Aug. 2010. [Online]. Available: http://doi.acm.org/10.1145/1851275.1851192

[2] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective Fine-grained TCP Retransmissions for Datacenter Communication," *SIGCOMM Computer Communications Review*, vol. 39, no. 4, pp. 303–314, Aug. 2009. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592604

[3] P. Prakash, A. Dixit, Y. C. Hu, and R. Kompella, "The TCP Outcast Problem: Exposing Unfairness in Data Center Networks," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 30–30. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228339

[4] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," in *Proceedings of the 1st ACM workshop on Research on Enterprise Networking*, ser. WREN '09. New York, NY, USA: ACM, 2009, pp. 73–82. [Online]. Available: http://doi.acm.org/10.1145/1592681.1592693

[5] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast Congestion Control for TCP in Data Center Networks," in *Proceedings of the 6th International Conference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 13:1–13:12. [Online]. Available: http://doi.acm.org/10.1145/1921168.1921186

[6] Y. Ren, Y. Zhao, P. Liu, K. Dou, and J. Li, "A survey on TCP Incast in Data Center Networks," *International Journal of Communication Systems*, pp. n/a–n/a, 2012. [Online]. Available: http://dx.doi.org/10.1002/dac.2402

[7] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, August 1993. [Online]. Available: http://dx.doi.org/10.1109/90.251892

[8] K. K. Ramakrishnan and S. Floyd, "The Addition of Explicit Congestion Notification (ECN) to IP," 2001, rFC 3168.

[9] M. Alizadeh, A. Javanmard, and B. Prabhakar, "Analysis of DCTCP: Stability, Convergence and Fairness," in *Proceedings of the ACM SIGMETRICS, Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '11. New York, NY, USA: ACM, 2011, pp. 73–84. [Online]. Available: http://doi.acm.org/10.1145/1993744.1993753

[10] J. Hwang, J. Yoo, and N. Choi, "IA-TCP: A Rate Based Incast-Avoidance Algorithm for TCP in Data Center Networks," *ICC 2012*, 2012.

[11] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware Datacenter TCP (D^2TCP)," *SIGCOMM Computer Communications Review*, vol. 42, no. 4, pp. 115–126, Aug. 2012. [Online]. Available: http://doi.acm.org/10.1145/2377677.2377709