The Nucleus ***58****, No. 1-4 (2021) 1-8*

https://doi.org/10.71330/thenucleus.2021.1165

www.thenucleuspak.org.pk

The Nucleus

ISSN 0029-5698 (Print)
ISSN 2306-6539 (Online)

Analysis of Transmission Control Protocol Incast over Large-scale HPC Clusters

S. Khalid*, H.M. Abdullah, S.Z. Ahmad

Computational Intelligence Group, Management Information System Division (MISD), PINSTECH, Islamabad, Pakistan

ABSTRACT

The lifecycle of large-scale applications executing on High-Performance Computing (HPC) clusters involves massive use of transmission control protocol (TCP) while performing orchestration for job completion on multiple compute resources. As the HPC clusters involve large local area network communication for distributing jobs over compute and data nodes, the core network fabric in cluster architecture faces heavy workloads of TCP sessions; causing more than average packet drop events. This results in the poor TCP throughput; thus reducing the overall performance indices of the cluster. In this article, we have analyzed the TCP behavior at nominal, average, and heavy transmission load in a cluster environment for assessing various alternatives to solve the problem. We have also analyzed the cumulative queuing behavior of multiple TCP sessions at the contention switch and used a fine-grained configuration at the network fabric to improve the TCP performance. The simulation results show that the smaller set of data flow suffers a significant throughput collapse. The performance of TCP variants tested indicates that the congestion control mechanism of these protocols plays a significant role in performance degradation and needs a scalable solution to improve TCP performance indices. In this paper, different versions of TCP are employed for an HPC compute cluster and data storage to cater to the TCP Incast problem and simple solutions are presented. It has been observed that none of the classical, as well as newer TCP variants, perform consistently under heavy fan-in workload but a better queue management system at the network fabric greatly simplifies the problem and improves the cluster performance.

Keywords: *Data Transport Protocol, Congestion control, Network Fabric, TCP Incast, HPC clusters, Queue Management*

1. Introduction

The transmission control protocol is responsible for 90% of the internet traffic around the globe. It is a transport layer network communication protocol in the Defecto TCP/IP stack, designed to send data packets over the internet in connection-oriented mode. TCP works in collaboration with the Internet Protocol (IP) to establish connection-oriented sessions between the sender and receiver. The IP defines the logical locations of the sender and receiver, whereas the TCP manages reliable data transport over then unreliable connectionless networks through session management between the sending and receiving sides. TCP ensures error-free, end-to-end delivery of data between the sending and receiving sides using the control-plane signaling that performs session initiation, data transport, congestion control, and session close operations over the internet infrastructure. Before starting data transmission, TCP creates a connection between the source and destination nodes and keeps it alive until any of the sender or receiver sends a connection close request to the other party. While the connection is alive, TCP breaks large data into smaller packets and also ensures that the data integrity is intact once it is reassembled at the destination node.

Despite its phenomenal success on the internet, TCP has shown some serious limitations in handling the data transport in high-performance computing (HPC) clusters and data center (DC) environments concerning throughput degradation and poor transmission resource utilization. The prime reason behind this performance degradation is the convergence of the heavy inflow of TCP sessions at the network edge behind that the HPC compute resources or data center storage systems are operating. The phenomena are known as TCP Incast or fan-in problem (while handling internal sessions; behind the network edge) and TCP outcast or fan-out (while handling sessions from outside the network edge). The very fast resource depletion at the edge switch is attributed as the key cause of the TCP incast and outcast problems that result in the packet drops. The congestion control algorithm of most TCP variants is highly sensitive about packet drop events during a session.

TCP Incast or fan-in problem is described in the literature as the collapse of throughput far below the link capacity that occurs in many-to-one communication patterns. This happens when a host places a request to multiple hosts for data exchange. When multiple receivers get the request simultaneously, all of them respond by sending multiple TCP data streams to the host almost synchronously. This burst of data may overflow the buffer of the host's switch and causes data packets to drop. TCP Incast problem especially affects computing paradigms in which distributed processing cannot progress until all parallel threads in a stage are completed. The throughput collapse occurs mainly due to frequent timeouts and full window loss. Full window loss occurs when there are no feedback ACKs for the data packets sent and the whole window of data packets is lost. In such a case, TCP does not go into fast retransmission and fast recovery mode because there are no acknowledgments to trigger those algorithms. This is also called a full-loss timeout. Another scenario is when there are not enough ACKs to trigger retransmission is called lack-ACK timeout. In such a case, TCP can get stuck in the slow-start phase due to timeouts and lack of enough ACKs to trigger recovery mode. This majorly reduces the throughput of the TCP as well as of the whole network as reported previously [1].

The HPC clusters and data centers support a multitude of services and applications that are based on transport protocols; primarily the TCP. The data transport is used for large-scale computing, storage, web searches, and transactional systems. In clusters and associated storage elements, many nodes send the request to a host

*Corresponding author: khalidsadia92@gmail.com

simultaneously. Similarly in web searches or web-based applications, a data query is sent to retrieve many data objects simultaneously from different cloud data servers. Further, in the MapReduce model, intermediate key-value pairs from many mappers are sent to appropriate reducers [2]. In all inter-cluster communications, the average round trip time in the inner nodes of HPC is significantly less than that of a typical WAN and the retransmission time out (RTO) value in such cases is in the range of a few milliseconds.

In this paper, we have studied to find out exact contributors to the TCP Incast problem in a cluster. For that purpose, we have conducted a detailed analysis of major TCP variants under moderate and severe congestion scenarios at the adjacent network switch and recorded the impact under varying traffic loads. We have then studied the queuing behavior of burst arrival from multiple sessions through M/G/1 queuing model for finding asymptotic behavior under adverse congestion scenarios. It has been found that the buffering resource constraint leads to a larger number of packets drop events that cause lower performance metrics for TCP. We then propose a modified queuing management system with an M/G/1/B model at the edge switch to mitigate incast without modifying TCP implementation.

2. Related Work

The ever-growing role of distributed computing and large-scale data storage through computer clusters, data centers, and cloud infrastructure has maintained the interest of researchers in the area of efficient resource management. Data communication resources are no exception with TCP playing a pivotal role. A comprehensive review of major work to handle TCP Incast and associated problems is presented by P. Sreekumari et al. [3]. It covers major contributions in the analysis and synthesis of the problem along with suggestions proposed by the authors. The study thoroughly analyzes various TCP parameters and algorithms that contribute to the adverse performance of TCP in the many-to-one type of synchronous transport sessions. The survey concludes that the contributing factors in TCP performance degradation can be traced back to the receiver's window size, time-out value, and use of explicit congestion notification (ECN). The survey provides a comparative study of the currently existing transport protocols to help readers understand the specific modifications required for the implementation of major TCP variants. Wu et al. [4] presented a solution with proactive adjustment of receiver window to mitigate the TCP Incast problem. The proactive adjustment of the receiver window may provide reasonably good results in known behaviors of segment arrival, however; in the case of diverse patterns of job distributions, the proactive intelligence may not timely converge to properly adjust receiver window size in real-time. Xu et al. [5] have also proposed receiver-oriented congestion control to cater to the TCP Incast problem. The authors have used oscillatory behavior of queue size at the switch to generate ECN for adjustment of receiver window accordingly. The approach is simple and has a low processing overhead but requires customized implementation of TCP.

Luo et al. [6] presented an analytical model to study the TCP Incast problem. The model estimates probabilities of the TCP Incast problem with respect to the number of sessions and network environment. An optimization technique is used to achieve the best possible throughput under constraints of the above-mentioned parameters. Finally, they used a cross-layer approach to improve the TCP Incast problem by configuring TCP parameters at the application layer. The solution provides better results in a very limited set of scenarios and lacks a comprehensive system-wide approach. The convergence of optimization overhead may also exceed the threshold for timely control triggers.

In one of the more recent works by Xu et al [7], they have exploited network virtualization and software-defined networking (SDN) techniques to propose Retransmission Timeout Avoidance by Packet Intelligent Discarding (RAPID) to control fan-in at software switch. RAPID uses selectively packet discard of each incast flow to force fast retransmit & fast recovery algorithm rather than the invocation of RTO for each packet loss epoch. They also proposed a low-complexity heuristic version of RAPID named RAPID-early detection (ED) that combines features of RAPID and early detection. The proposed scheme though tries to solve the incast problem with a simple approach but the selection criteria of packet drop is not subtle. The fast retransmit may also result in overhead in case of the short RTO values of a few milliseconds as found mostly in local area networks (LANs).

Thiruvenkatam and Mukeshkrishnan [8] in their recent works have proposed a two-step solution that uses a k-means algorithm at the sender to make a priority queue for each group to reduce the complexity of queue management at the core switch. In the next step, they use Pro-Acknowledgement (Pro-Ack) control that acknowledges bulk data receipt to reduce the control signal plane for achieving lesser timeouts and retransmissions epoch. The results show good throughput gains but the complexity of k-means may be high in the case of diverse nature of flows; resulting in the high overhead of prioritized queue management. Similarly, Zou et al. [9] have proposed an adaptive pacing (AP) mechanism to dynamically adjust bursty inflows according to their flow concurrency. The proposed solution may add delays over small RTO values during a large number of flows causing packet losses.

The methodologies used in the above-mentioned work motivated us to apply a clean slate approach by analyzing the problem with the standard queuing theory. Instead of changing any operations of TCP, the queuing behavior of the edge network devices is studied and tailored to get better results. The queuing behavior is studied under a wide spread of traffic load at the edge of the network. This approach saves heavy proactive processing costs to get marginal performance gains. The literature review also highlights that the incast control is achieved through modifications in the TCP stack whereas we have proposed infrastructure level configuration management that provides space to all TCP variants to capitalize their strength and leave the option of selection with the end-user and applications.

3. Analysis of Incast Behavior in TCP

The literature review presented in the preceding section highlights two important aspects for the solution space of incast problem. Firstly, the customized solutions may work well under assumed conditions but a general comprehensive solution may still be needed to cover a larger set of applications. Secondly; predominant proposed solutions tailor the TCP stack causing reduced universality of TCP. In this section, we present an analysis of the TCP Incast behavior based on the well-known TCP parameters that describe the streamflow and congestion control operations. The major focus of examination includes the life cycle of a TCP flow, basic congestion control algorithms with some explicit modifications such as fast retransmit, congestion avoidance, and fast retransmit & recovery algorithms. The behavior of the transport start mechanism, commonly known as Slow Start (SS) is also analyzed for the sack of completeness.

3.1. Life Cycle of a TCP Flow

TCP data transport function starts after successful handshake operations that establish the necessary data structures and control parameters. The transmission starts on a defensive note with a single frame sent to the receiver and the sender waits for its acknowledgment. This is called the SS phase of the session. On arrival of the Acknowledgement, the next frames are transmitted based on the congestion algorithm in use and described in the next paragraph. The control parameters like Round Trip Time (RTT), flow control window, receiver window, and SS threshold (ssthresh) are set on the arrival of the first acknowledgment. The TCP session arbitrates between various states like SS, Congestion Avoidance (CA), congestion control, fast retransmit, and fast recovery phases, etc. These phases do not apply to all flows as the selection of congestion control algorithm decides what subset to be used from the above-mentioned states.

3.2. Congestion Control Algorithm

TCP congestion control protocols have been actively studied for decades for their revision to cope with evolving network technologies. Despite its long-lived revision history, it remains an interesting topic for research in the area of internet transport. Traditionally, TCP congestion control algorithms follow an Additive Increase, Multiplicative Decrease (AIMD) approach for its data transmission over unreliable networks. For each successful transmission of a packet from source to destination, the Congestion Window (CWND) increases by one frame. The CWND has a maximum size before a packet drop is detected by either a timeout (TIME_OUT) event or three duplicate acknowledgments (DUP_ACKs). In the case of CA, the frame transmission rate is slowed to linear increase from exponential increase mode whereas, in the case of three DUP_ACKs, a fast retransmit & recovery algorithm is activated. At this point, TCP reduces the CWND size to half and starts increasing again on receiving ACKs in linear mode. The initial TCP versions like TCP Tahoe, TCP-Reno, TCP-New Reno, TCP-Vegas, and BIC-TCP, etc. are closely resembling clones of the above-mentioned approach. The newer versions like TCP-CUBIC, Bottleneck Bandwidth, and Round-trip propagation time (BBR), however, use a more aggressive approach for increasing CWND (in cubic mode). The process continues till a congestion epoch occurs that results in reduced CWND. Since BBR is not a loss-based algorithm, we have not included it in this study to be consistent with the congestion epoch. The TCP-CUBIC has shown more promising results for utilizing available bandwidth but the fairness in sharing bandwidth with other competing flows still needs to be improved, particularly fairness to flows with other TCP variants may not be guaranteed [10]. All these TCP variants (except BBR and Elastic-TCP) are sensitive to packet loss events. Since packet drop event occurs due to high buffer occupancy at one of the routers en route to the destination; it is considered a key indicator of congestion at some node on the network path and the transmission rate needs to be reduced to help in solving this issue.

In conventional communication infrastructures with limited bandwidth, the efficiency of the fast retransmit and recovery algorithm is measured higher in case segment loss rate is very low (one), but in case of multiple segments drop the efficacy of the algorithm reduces. Hence, the selection of the right algorithm makes a significant difference in diverse operating environments. Since incast has a specific operating environment constraint by the low RTT, a higher number of sessions, and a higher contention rate; the selection of the right congestion control algorithm is crucial for better resource utilization and increased systems throughput in clusters and data centers.

4. Throughput Analysis using NS-2

The prime objective of analyzing TCP throughput under incast or many-to-one (also known as fan-in) scenarios is to find suitable configurations within the infrastructure to keep TCP implementation consistent for all applications. We have analyzed the TCP Incast using the ns-2 simulator. Ns-2 is a discrete event simulator that is used for network protocol and network traffic analysis research. Modeling of TCP incast scenario is an interesting topic that may need many permutations on a host of parameters including traffic generation model, network fabrics and delay models, queue models, and link reliability models. For throughput analysis of various versions of TCP in scenarios where many-to-one communication pattern is being observed, the following topology has been used, as shown in Fig. 1.

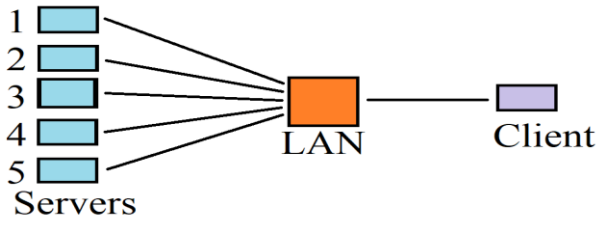

Fig. 1: Simulation topology of the many-to-one communication pattern.

We use the term server or session alternatively to represent one TCP traffic generator. We have used all major TCP variants that use loss-based congestion control algorithms. The choice of these variants is based on their deployment ranking for the sake of consistency and standardization. To understand the effect of several servers, the traffic load is increased gradually while the rest of the topology remains the same. To understand the effect of several switches and buffers, they are also increased one by one but Fig. 1 shows the simplest topology where only one switch is used. The values of some other parameters used for simulation in ns-2 are given in Table 1.

Table 1: Simulation parameters used for NS-2 simulation.

Parameters	Values
The Capacity of Duplex-links	10Mbps
Propagation Delay	1ms
The Capacity of links for LAN	20Mbps
Propagation Delay for LAN	100ms
Packet Size	600 bytes
Simulation Time	20 seconds
Traffic Generator Type	Exponential
Queuing Model	Drop-Tail
Application	FTP

The simulation parameters are set to identify the maximum load on queues. The propagation delay plays a pivotal role as a higher propagation delay shall cause a slower transmission rate. In case, the inter-packet arrival rate in terms of bit/s is higher than the transmission rate the queue built-up is expected and in case of long-tailed burst arrival, the queues may get full and packet drop occurs. Packet drop starts much earlier than the complete depletion of the queues due to Early Congestion Notification (ECN) intended to the sender to slow-down sending rate. On the contrary, if the transmission rate is higher than the packet arrival rate, the queues are not expected to deplete and an even smaller buffer size of two or three packets may be sufficient. The simulation scenarios are created to highlight these aspects of queue management.

4.1. Simulation Results

For different TCP variants, the average throughput per server/session is plotted with respect to the number of servers/sessions. Average throughput per server decreases as the number of servers is increased. This is due to multiple factors like increased communication overhead, increased delay at the node, and that now the same file is distributed among more servers and the data to be sent per server becomes less. It is noticeable that the graphs show a significant decreasing trend with an increased workload. To analyze the small differences in throughput of different TCP variants, differences in the congestion control or flow control mechanisms of these variants are attributed. The increase of the number of servers also causes fairness issues resulting in some of the sessions facing lower contribution of overall bandwidth and causing activation of CA or CC modes.

Fig. 2 shows simulation results for the average throughput of TCP, TCP Reno, TCP New Reno, and TCP Vegas when one switch is used and the number of servers is gradually increased. It can be seen that the performance of all TCP variants is poor at the high server load. The main cause of this degradation is a higher retransmission rate due to timeouts. The basic TCP performs defensively at the SS state causing below-average performance even at the lower server loads.

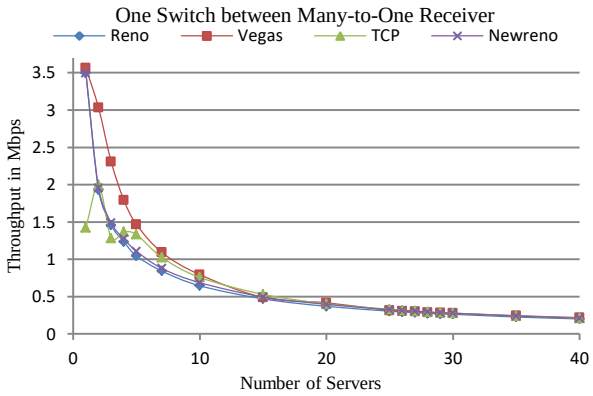

Fig. 2: Average throughput per server for one switch.

Fig. 3 shows simulation results for throughput of TCP, TCP Reno, TCP New Reno, and TCP Vegas when two switches are used and the number of servers is gradually increased. The use of multiple switches signifies an increase in processing rates at the queues.

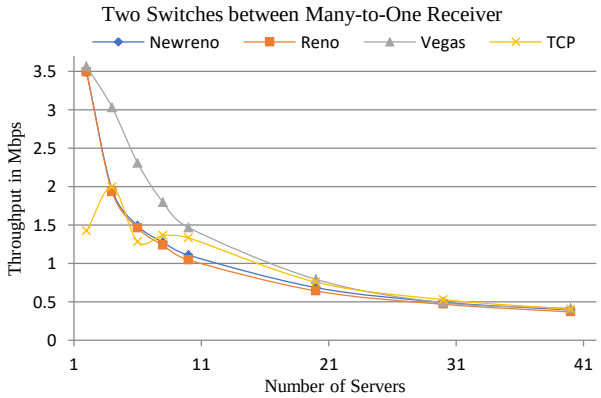

Fig. 3: Average throughput per server for two switches.

Fig. 4 shows simulation results for throughput of TCP, TCP Reno, TCP New Reno, and TCP Vegas when three switches are used and the number of servers is gradually increased. It is noticeable that the performance of the flows has significantly improved while sufficient switching capacity is available in terms of alternate paths and available buffers through adding new resources in the network fabric.

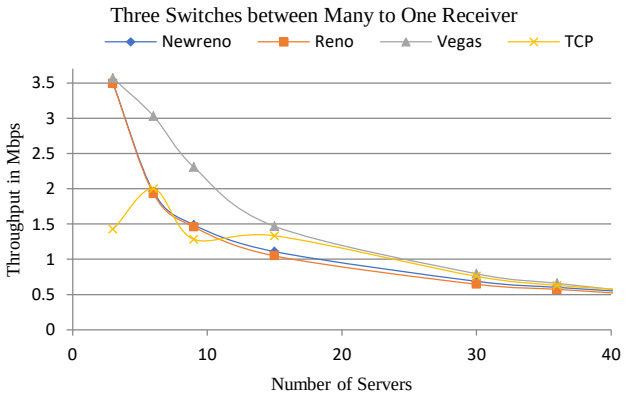

Fig. 4: Average throughput per server for three switches.

The three simulation results motivate for a simpler but suitable infrastructural level configuration that can boost the TCP and its variants performance at the fan-in switch. The increase in the number of switches to deal with the arrival traffic in a load balancing mode reduces the timeout-related congestion epoch. The link capacity and propagation delays can determine the suitable buffering capacity that shall be needed in the network fabric to reduce packet drop probability to the bare minimum level.

5. Proposed Solution for the TCP Incast Problem

The throughput analysis of TCP variants discussed in the preceding section has highlighted two key contributors during incast conditions. These are the buffering capacity of the network fabric that faces fan-in and the propagation delay of the link. It has been discussed in [11], [12] that if the propagation delay is lesser than the average inter-packet arrival time, a buffer size of holding maximum segment size i.e., 64KB may prove sufficient to keep packet drop probability to very low values. In case of propagation delay is larger, the contention-based media access protocols may face serious problems as the buffering capacity can quickly deplete. Therefore, the proposed solution is based on an in-depth buffer occupancy analysis of the network switch that converges all the flows to a single port.

The basic constraint in deploying modified TCP implementation lies in maintaining its standard behavior as all the devices connected to the Internet need to use it seamlessly. Instead of modifying the TCP stack, our approach is to not modify the TCP implementation but we focus on minimizing the chances of packet drop events by adding sufficient resources at the edge of the network where all the in-flows converge. Though customization of internet fabric is out of the question, the intra-network fabric in a cluster of data storage networks can be easily modified. Hence, the following are the stages of our work that is carried out:

i. Buffer Occupancy Analysis at the edge switch using queuing theory.
ii. Analysis of the effects of burst-mode transmission and steady-state transmission.
iii. Segregation of inflows based on their mode (Burst mode or periodic packet injection).
iv. Study of the TCP packet drop events at higher load at the edge devices.

In the succeeding section, we study the behavior of the above-mentioned four scenarios and their related aspects through standard queuing theory. The findings and observations of this study are tested with the simulation study in ns-2.

5.1. Study of Buffer Occupancy using standard queuing theory

The queuing behavior at the routers and switches plays a significant role in the performance studies of TCP. The packet drop events generally occur due to partial or complete depletion of queues at these devices. Hence, a thorough study of buffer occupancy at these devices reveals significant knowledge about the causes of poor throughput of TCP sessions in a multitude of operating environments. The real challenge in studying queuing behavior of TCP sessions lies in the modeling of arrival and service processes at the queue that may be too trivial using common processes such as Poisson and exponential distributions. Consequently, finding an accurate closed-loop quantitative model with known network parameters such as router capacity, propagation delay, and buffer size is not a trivial problem as it has convolutional contributors to deal with. Furthermore; as mentioned above, packet losses occur predominantly as buffer overflows or near overflow states, hence it is a natural choice to take packet loss probability as the blocking probability in a single server queue. That forms the basis of linking packet loss models and queuing models.

The packet loss events may be either modeled in stateless or stateful modes. The stateless model assumes queue length at packet arrival event as Independent and Identically Distributed (I.I.D) random variables. This assumption works well in long chains of routers with a sufficient set of resources and fast service rates but may not be accurate enough in the case of single Local Area Network (LAN) switches. The stateful model assumes a long-range correlation between queue length and the arrival process. Therefore, the queue length is included in our TCP queuing model. The packet loss probability is expressed as a function of random variables which depend on the length of the queue. The M/D/1/B model is considered a useful approximation of stateless TCP queue analysis. The main advantage of using this model is its ready availability of explicit analytical expressions for most quantities of interest, including the blocking probability. Models based on M/M/1/B queues have been considered good for fluid model approximation. These models have, however, depicted inaccuracies in acquiring the complex statistical structure of TCP traffic [13].

Altman et al. [14] studied a particular case where the flows contain a single packet drop signal and a useful observation is made in showing that flow control can be reformulated in terms of an equivalent M/G/1 queue, where the transmission rate is translated into the workload of the queue. The congestion signals correspond to packet drop at the queue according to the Poisson process. This transformation is also valid in our study of TCP congestion behavior during incast. We have worked with the window size rather than the transmission rate. In our model, the data rate of a congestion window-based flow control mechanism is equal to the window size divided by the RTT of the session. Assume W denotes the maximum window size; the limitation on the window size is due to the congestion control algorithm of TCP. Though variants of TCP have slightly different congestion control mechanisms, the primary reason for congestion control activation is a packet drop event and in case of no packet drop or duplicate ACKs with the window size smaller than W, the congestion window of the protocol increases linearly. We assume that in a sequence of ACKs; in the case of the stateless mode, there may be a random number of packet drop events as per the Poisson process while in the case of stateful mode, the same is based on standard Gaussian distribution. The second important classification is based on steady-state transmission and burst transmission. The steady-state TCP session packet arrival is modeled by the standard Poisson process whereas the burst arrival is modeled by the standard exponential distribution. The average transmission rate can be described as follows:

$$X = \lim_{T\to\infty} \frac{1}{T}\int_0^T X(t)dt \; = \; \frac{E(W)}{RTT} \tag{1}$$

where E(W) is the mean or expected value of congestion window length during a short interval of time.

Eq. (1) indicates that in the case of small RTT such as in the case of LAN environment, the transmission rate shall be generally higher and the only restricting factor is the size of the congestion window that is controlled by packet drop events.

Based on Eq. (1), we use TCP flow behavior using the fluid model to find out the rate of change in modeling throughput as follows [15]:

$$\frac{dx(t)}{dt} = \frac{1}{RTT^2} - (x(t-RTT).p(t-RTT).\frac{x(t)}{2} \tag{2}$$

where $p(t)$ is the packet loss rate at time t. The TCP throughput $x(t)$ and desired throughput $\bar{x}(t)$ (with known packet drop rate) satisfy the following relation:

$$\frac{dx(t)}{dt} = \frac{Log2}{RTT^2} - (\bar{x}(t) - x(t)) - p(t-RTT).x(t-RTT).\beta x(t) \tag{3}$$

where β is a constant describing the variation in the packet arrival process. Based on Eq. (3) the expected throughput rate under a given tolerable packet drop rate is given by Eq. (4).

$$\frac{d\bar{x}(t)}{dt} = 1 - (p(t-RTT)^{k_1\, x(t-RTT)}\, k_2\, x(t) - p(t-RTT).x(t-RTT)(x(t)-x(t)) \tag{4}$$

where k_1 & k_2 are constants.

Eq. (1) to Eq. (4) suggest that the desired throughput is specifically dependent on the packet drop rate provided the other parameters are in their normal ranges. If we assume N flows are competing for switch resources, then the buffer size B becomes an important parameter to be estimated to reduce packet drop rate to a minimum value. Hence, the value of B is estimated as follows [16]:

$$B \cong N.\bar{x}(t) \tag{5}$$

Eq. (5) forms the basis of our simulation studies for an $M/G/1$ type queue for minimizing the TCP Incast problem. In the case of a minimum desired rate per flow, the queue model is tailored to $M/G/1/B$ for constraint stateful behavior of the queue, where B is the maximum buffer size. Using the above queue model and the TCP congestion control dynamics, the following behavior was observed to explain the TCP Incast problem.

5.2. Model Description and Evaluation

We have calculated the value of B based on the number of contending sessions and the average arrival rate of sessions as modeled in Eq. (5). The number of sessions is increased to much higher numbers to monitor the impact of contention due and propagation delay. This provides a base value to the appropriate link speed and helps to accommodate all traffic loads within a very low probability of packet drop rate. It also helps to reduce queuing delay in the system in case of deviation from average transmission rates particularly in case of burst mode traffic arrival. The burst mode traffic arrival process with packet drop events is modeled through exponential distribution. The choice of this model is based on increasing the reliability of queuing systems under heavy load and the possibility of packet drop events due to an arrival rate much higher than the average rate of arrival.

The underlying assumption of the proposed solution is to minimize packet loss events and keep competing TCP sessions in congestion avoidance mode rather than triggering RTO or fast retransmit mode. In the case of internet routers, the propagation delay is generally high causing queue built-ups whereas in the case of a data center core switch the RTO values are generally very small causing retransmissions even in case the packet is not lost but waiting in the queuing system due to heavy traffic load. Therefore, excessive buffering can easily solve the problem by accommodating all traffic arrivals and adaptively increasing the RTO values to reduce retransmission timer-based events.

Based on this hypothesis, we observe packet drop rate at a given traffic load in both average and burst mode and then evaluate throughput of the TCP sessions with sufficient buffering space at the core switch. This approach

provides support to all TCP variants in maintaining their operating curve near congestion avoidance mode. In the case of long streamflow fairness, the issue may also arise and some of the flows may be taking a larger share of bandwidth than the others. But in the case of shorter flows, the proposed model also ensures better fairness amongst the competing flows due to lesser wait time in the queue. This applies to all TCP sessions working with legacy congestion control algorithms.

Fig. 5 shows a plot of buffer occupancy at different TCP session loads is presented. It is observed that with a large buffer capacity of 10000 segments, the queue remains under-utilized throughout the simulation period. In the case of a traffic load of 100 TCP sessions, the queue saturates near the end of simulation time. It's worth mentioning that the study is carried out in burst mode. The queue gets quickly fully occupied in the case of 1000 (Nos) and 10000 (Nos) simultaneous TCP sessions increasing the packet drop probability.

Fig. 5: Buffer occupancy at varying TCP session loads with respect to packet arrival at the edge device.

In

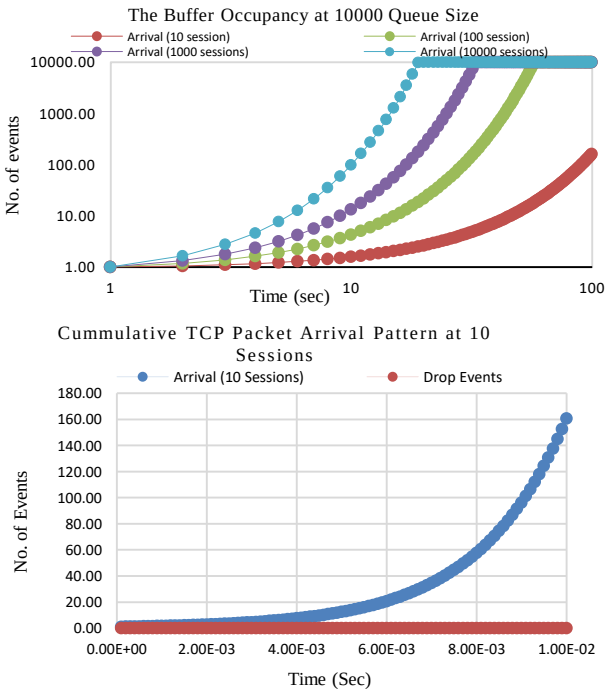

Fig. 6, a plot of packet drop events for ten simultaneous TCP sessions is presented. It is noticeable that the packet drop event is zero throughout the simulation run. Fig. 7 to Fig. 9 show a significant number of packet drop events at 100, 1000 & 10000 simultaneous sessions, respectively. The queue gets filled during some state of simulation and the packet drop events are much more frequent at higher loads. The higher packet-drop events in this area explain the reasons behind poor TCP performance at higher loads leading to the TCP Incast problem.

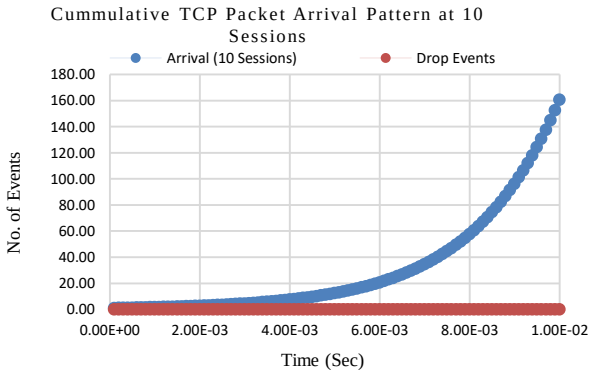

Fig. 6: Packet drop events at 10 simultaneous TCP sessions in burst mode.

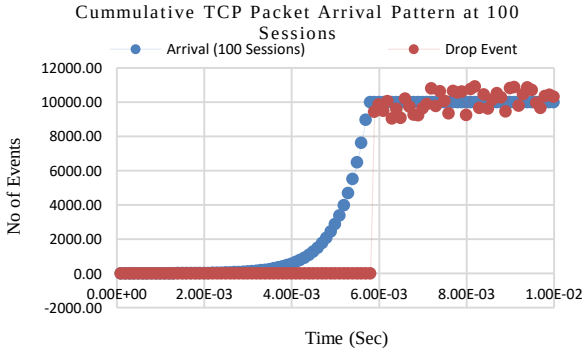

Fig. 7: Packet drop events at 100 simultaneous TCP sessions in burst mode.

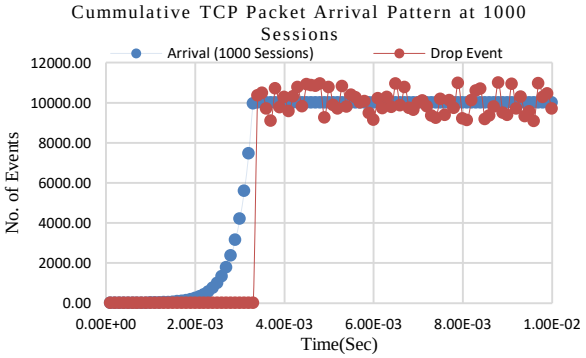

Fig. 8: Packet drop events at 1000 simultaneous TCP sessions in burst mode.

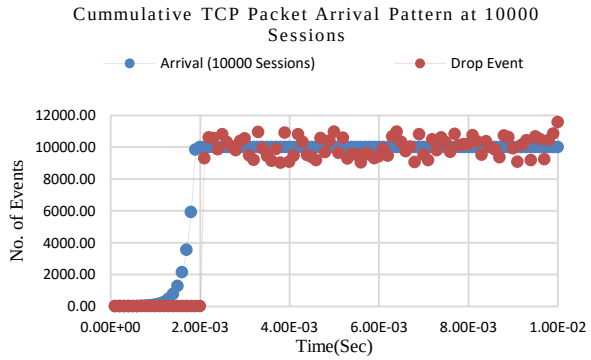

Fig. 9: Packet drop events at 10000 simultaneous TCP sessions in burst mode.

5.3. *Multiple Queue Model*

The multiple queues models like $M/G/n$ and $M/G/n/B$ also need more study to get further insight into the TCP behavior under multiple queue management schemes. Incast may be one special case in such a study. In using a leaf-spine topology at the network edge instead of standard 3-tier topology, the cascaded buffering analysis is reduced to a lesser number of flows per switch, resulting in better resilience to the TCP Incast problem. In the simulation study, we have analyzed these scenarios. In real scenarios, the leaf-spine topology is used with specialized switching fabric like Infiniband switches, etc.

6. Results and Discussions

The overall gain achieved with the proposed scheme, in all loss-based congestion control variants of TCP is plotted in Fig. 10. It can be seen that after increased buffering in the network fabric, the throughput of all the TCP variants has shown improvement. The best result can be noticed in the case of TCP-CUBIC. The main cause of this increase is an increase in congestion window aggressively to use maximum available bandwidth to get better throughput. Similarly, all other variants have also shown significant throughput gain. Fig. 11 shows the same result in percentage (%) values making the gain more visible in the graph.

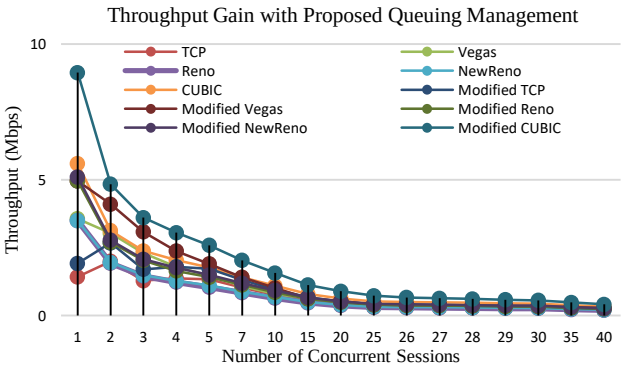

Fig. 10: Throughput gain for various TCP variants with the proposed scheme.

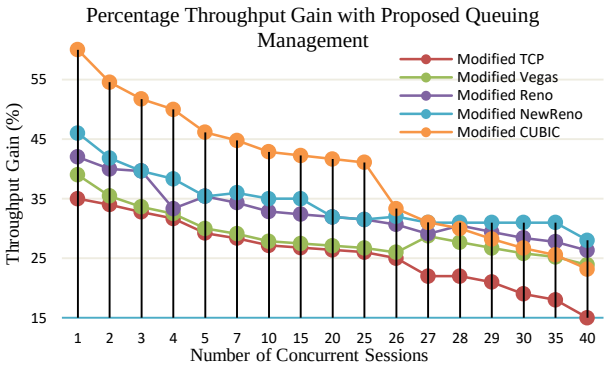

Fig. 11: Percentage gain of major loss-based TCP variant with the proposed scheme.

The performance bottleneck of TCP in highly critical computing infrastructures like data centers, clusters, and clouds greatly hinder the leverage of such technologies in their business processes, resulting in reduced earning and reduced cost-benefit margins. The rectification of this issue is essential for the low-cost use of conventional technologies. Although other options like InfiniBand switches with the inner translation of TCP to native protocols are available, the widespread use of TCP and its standardized stature need to be preserved for better utilization of already deployed global computing and communication infrastructure. In this paper, we have analyzed the TCP Incast problem in a cluster environment and found limiting factors that cause poor resource utilization and higher job completion time on clusters with respect to TCP that otherwise has been accepted and appreciated as one of the most trusted transport protocols. The study finds some problems in the congestion control mechanism of various TCP variants. The key role of packet drop events has also been studied through queuing theory and we conclude that a scalable queuing system at the edge can greatly improve performance at significantly higher traffic loads. The study of the dynamics of the multiple queue model in improving TCP performance has been identified as a future work direction.

References

[1] W. Chen, F. Ren, J. Xie, C. Lin, K. Yin and F. Baker, "Comprehensive understanding of TCP Incast problem", IEEE Conf. Comp. Comm. (INFOCOM), Kowloon, Hong Kong, pp. 1688-1696, 2015.

[2] Y. Chen, R. Griffith, J. Liu, R.H. Katz and A.D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks", Proc. ACM work. Res. enter. net., Barcelona, Spain, 2009.

[3] P. Sreekumari and J. Jung, "Transport protocols for data center networks: a survey of issues, solutions, and challenges", Photo. Net. Comm., vol. 31, no. 1, pp. 112-128, 2015.

[4] H. Wu, Z. Feng, C. Gu and Y. Zhang, "ICTCP: Incast congestion control for TCP in data-center networks", IEEE/ACM Trans. Net. (ToN), vol. 21, no. 2, pp. 345-358, 2013.

[5] L. Xu, K. Xu, Y. Jiang, F. Ren and H. Wang, "Throughput optimization of TCP Incast congestion control in large-scale data center networks", Comp. Net., vol. 124, pp. 46-60, 2017.

[6] J.T. Luo, J. Xu and J. Sun, "Modeling TCP Incast Issue in Data Center Networks and an Adaptive Application-Layer Solution", J. Elect. Sci. Tech., vol. 16, no. 1, pp. 84-91, 2018.

[7] Y. Xu, S. Shukla, Z. Guo, S. Liu, A.S. Tam, K. Xi and H.J. Chao, "RAPID: Avoiding TCP Incast Throughput Collapse in Public Clouds with Intelligent Packet Discarding", IEEE J. select. are. comm., vol. 37, no. 8, pp. 1911-1923, 2019.

[8] B. Thiruvenkatam and M. Mukeshkrishnan, "Optimizing data center network throughput by solving TCP Incast problem using k-means algorithm", Int. J. Comm. Sys., 2020.

[9] S. Zou, J. Huang, J. Wang and T. He, "Flow-aware adaptive pacing to mitigate TCP incast in data center networks", IEEE/ACM Trans. Net., pp.134-147, 2020.

[10] K. Sasaki, M. Hanai, K. Miyazawa, A. Kobayashi, N. Oda and S. Yamaguchi, "TCP fairness among modern TCP congestion control algorithms including TCP BBR", IEEE Int. conf. clou. Net. (CLOUDNET), pp. 1-4, 2018.

[11] H. Wang, "Trade-off queuing delay and link utilization for solving buffer bloat", ICT Exp., vol. 6, no. 4, pp. 269-272, 2020.

[12] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet", Proc. USENIX Symp. Net. Sys. Des. Imp. (NSDI), pp. 329-342, 2018.

[13] H. Hisamatu, H. Ohsaki and M. Murata, "Modeling a heterogeneous network with TCP connections using fluid flow approximation and queuing theory", Proc. Perf. Cont. Nex. Gen. Comm. Net., vol. 5244, 2003.

[14] E. Altman, K. Avrachenkov, C. Barakat and R. Núñez-Queija, "State-dependent M/G/1 type queueing analysis for congestion control in data networks ", Proc. IEEE INFOCOM Ann. Conf. Comp. Comm. Soc., vol. 3, pp. 1350-1359, 2001.

[15] G. Raina and D. Wischik, "Buffer sizes for large multiplexers: TCP queuing theory and instability analysis", Nex. Gener. Inter. Networks, pp. 173-180, 2005.

[16] A. Dhamdhere, H. Jiang and C. Dovrolis, "Buffer Sizing for Congested Internet Links", Proc. IEEE Ann. Conf. IEEE Comp. Comm. Soc., vol. 2, pp. 1072-1083, 2005.