MITIGATION OF TCP OUTCAST PROBLEM USING DCTCP

Rajashree[1], Shruthi.M[2], Swathi Pai M[3]
Assistant Professor Dept of Computer Science &Engineering
NMAM INSTITUTE OF TECHNOLOGY, NITTE
Karkala, Karnataka, India

Abstract
As the technology is improving day by day and most of the services used by the mankind are dependable on the internet, we need a datacenter which stores huge amount of data of multiple users and provides high performance, reliability, scalability, fault tolerant, flexibility and availability. Since the Data Center mainly shares the resources among multiple users there exist the problem of throughput degradation in small flows thus provides unfairness between the large flows and small flows which is termed as TCP Outcast Problem. TCP Outcast Problem is reduced by using different queuing mechanisms. In this paper Analysis of TCP Outcast Problem for Data Center is done for Drop Tail, RED and DCTCP using NS2 simulator, and the TCP Outcast Problem is mitigated providing the best results with DCTCP for all types of flows.
Keywords*:* TCP Outcast, RED (Random Early Detection), DCTCP (Data Center Transmission Control Protocol).

I. INTRODUCTION

Internet over the past few years has transformed from an experimental system into a gigantic and decentralized source of information. Large-scale data centers enable the new era of cloud computing [1] and provide the core infrastructure to meet the computing and storage requirements for the cloud based services. In Modern years data centers [2] have emerged as the corner stones of modern communication and computing infrastructure. Data Center is a cluster of large number of servers connected via shallow buffered switches having many to one communication pattern. Data Centers are mainly used to store and process large amount of data of multiple users and also provides high bandwidth, low latency and limited size buffer. Data Centers typically host many different classes of applications that are independently owned and operated by completely different entities- either different customers or different divisions within the same organization. Data Centers must provide the main basic property of agility i.e, the capacity to assign any server to any service. With agility, the data center operator can meet the fluctuating demands of individual services from a large shared server pool, resulting in higher server utilization and lower cost. Large-scale data centers form the core infrastructure support for the ever expanding cloud based services. The main qualities of a Data Center networks are low latency, high bandwidth, high availability, high performance and storage infrastructure with low cost commodity switches. Due to this, cloud computing services make use of data center for large scale general computation, web search and cluster storage.

When dealing with the real time applications with huge amount of data there exist different types of traffic in Data Center Networks. They are:

(i) Mice Traffic or Query Traffic(<100KB)- The volume of Query Traffic that is generated in Data Center Networks is usually less and the majority of the traffic in a data center network is query traffic.(e.g. google search, facebook updates, etc).

(ii) Cat Traffic(100KB-5MB) – The traffic is generated due to control state and co-ordination messages and the volume of the cat traffic is more compared to Mice traffic. (e.g. small and medium sized file downloads, etc).

(iii) Elephant Traffic (>5MB) – Here the volume of the traffic is extremely large which is generated due to large software updates. (e.g. anti-virus updates, movie downloads, etc).

Depending on the users Quality of service the choice of protocol made by Data Center Network is TCP [3]. TCP is the existing technology with good reliability and congestion control features. Though the scale of the internet and its usage is increasing in recent years, TCP has evolved to keep up with the changing network and has proven to be scalable and robust. Still congestion remains the major problem that affects the Internet Service Quality, and the performance of TCP in Data Center network has been a major concern recently because it leads to major challenges such as TCP Incast [4], TCP Outcast, Queue Buildup and Buffer Pressure[5]. One of the problem discussed in this paper is TCP Outcast problem. The major approach to mitigate the TCP Outcast problem is to use different queuing techniques. In this paper RED queuing technique is used to overcome the TCP Outcast problem when compared to Drop Tail and it is almost resolved using DCTCP Technique.

II. LITERATURE SURVEY

In modern era of distributed and parallel computing, data of single user is stored on different servers, so that one can fetch data in less time. Low latency is becoming a major performance requirement for DCNs because of real-time nature of many popular applications, whose performance is critical to service provider industries. For example, in high frequency trading where market data must come with minimal latency (of the order of microseconds), financial service providers rely on high speed networks that must provide low latency to carry these computational transactions. Similarly, in retail web services, single page request may require calling more than 100 services and because these services can be dependent on each other. Low latency is an important factor for user experience. Many data center applications require task completion within their
deadlines. If the deadlines miss, it may result in bad user experience or failure of the job resulting into revenue loss. In DCNs also, data is distributed among multiple servers. When a user fetches the data from multiple servers connected via switch, each server responds to the user query and each server tries to acquire the bottleneck switch buffer. In this race, some of the responders successively get the switch buffer, while some of the packets are dropped due to switch buffer overflow. These responders have to wait for hundreds of RTT of DCNs because of minimum retransmission time out (RTO) in modern operating systems. This phenomenon results in catastrophic throughput loss of DCNs. This scenario is called TCP Incast (Y. Chen, 2009). TCP Incast occurs mainly due to bursty mice traffic in DCNs. TCP Incast is common in modern data centers because of traffic pattern like web searches, when each server responds to query. This problem also arises in data center internal computation applications like map-reduce (Dean and Ghemawat,2008). When computation is done on different servers and results are aggregated on single servers, each server sends the map result simultaneously to common server for aggregation of results. It is not mandatory that TCP Incast arise only on ToR switch. This throughput collapse can happen at any switch layer. There may be two possible ways to mitigate the TCP Incast problem. First, stop the packet drop at switch port and second is fast recovery of dropped packets. To stop the dropping of packets, we need to notify the receiver that there is congestion in the network so that it will narrow down its congestion window accordingly. But traditional TCP does not notify the receiver about any congestion and once it gets to know that packets are dropping, it halves its congestion window. While default Min RTO in modern Linux kernel is 200 ms, that's why responder has to sit idle for long time before retransmission of packets. Buffer Pressure and Queue Build-up in DCNs refer to queues getting filled up or remaining full persistently. The short flows experience delay due to packets of long flows in the queue, this is known as Queue Build-up (Alizadeh et al., 2010).

TCP Outcast problem occurs due to two conditions, (i), When there are two different flows available : small flows and large flows and (ii), switches in DCNs use the droptail queue management scheme. The queuing mechanism at layer-2 switch is the key to improvement in fairness and latency while achieving high throughput. Although a data center usually resides in a single building with very low propagation delay, the high bandwidth and high burstiness still create high queuing delay resulting into high latency in traditional

switches. Moreover, since non-cooperating applications and multi-tenants coexist in data center networks, fairness issue is better to be tackled at switch level through queuing scheme. There are different solutions available to TCP Outcast such as, use Active Queue Management (AQM) scheme for example Random Early Detection (RED) (Floyd and Jacobson, 1993), Stochastic Fair Queuing (SFQ)(McKenney,1990), TCP Pacing (Aggarwal et al., 2000) and equal length routing (Prakash et al., 2012). In RED, packets are marked randomly before the buffer queue is full so that packets get dropped randomly at different ports. RED queue management scheme achieves RTT bias but is unable to provide true fairness in DCNs (Prakash et al., 2012). In equal length routing every packet traverses equal distance rather than shortest available path and thus, provides the fairness among the senders. However, it will not work when the number of flows in a set are different. SFQ gives the better results than RED and equal length routing but SFQ is not commonly available in commodity switches because it is very complex and needs extra overhead to maintaining multiple buckets. TCP pacing reduces the unfairness in the network but consecutive packet drops of small flows are more in TCP pacing and it is also inverse RTT biased. None of the solution of TCP Outcast solves the problem completely and in each case there is unfairness in throughput between different flows and these solutions will not mitigate the another problems in DCNs like incast etc.

III. TCP OUTCAST

In DCN's, Cloud resources such as CPU and storage are shared among multiple users. Thus when two users try to access data at the same time from the common data center network consisting of large no of flows and small set of flows with two different input ports competing for the same output port, the small set of flows obtain lower throughput than the large set of flows. This is termed as TCP Outcast Problem. In a Data Center Network where the enormous data transfer takes place the TCP Outcast problem arises due to two main aspects:

1. Use of Drop Tail Queuing Mechanism.
2. In Many to one Communication between large flows and small flows the smaller flows gets outcasted by the larger flows leading to packet degradation.

Due to the TCP Outcast problem the packet drop takes place which leads to loss of data. Here smaller flows are effected packet more due to the inverse RTT bias method found in the Data Center Network. In the Normal Networking TCP's throughput is inversely proportional to the RTT. Hence low-RTT flows will get higher share of bandwidth than high-RTT flows. But in Data Center Networks RTT bias does not hold true ,and exhibits reverse RTT bias, where small set of flows have lower RTTs then the large set of flows consisting of lower throughput in smaller flows leading to packet drop when compared to higher RTT ones. Avoiding packet drop keeps network bandwidth and permits congestion signals to be propagated faster.

IV. QUEUE MANAGEMENT ALGORITHMS

A. DropTail

Droptail is the simplest form of queuing management and most commercial switches make use of it. It entails packets entering the buffers in a FCFS scheduling technique and after the buffer space of the switch fills up, new packets trying to enter at the tail end are dropped. . In Drop Tail technique, a maximum queue size is set for each queue which is usually the physical buffer size. The arriving packets are accepted since the maximum queue size is not reached yet. Once the maximum queue length is reached, the following incoming packets are dropped until the queue size decreases as a result of the leave of a packet from the head of the queue.It allocates bandwidth according to the packet at the top of the queue which has been placed at the output port by the scheduler. This process seems correct but does not prioritize packets therefore important packets or packet flows whose deadline to reach its destination is near are not given priority in the allocation of bandwidth; which can cause unnecessary retransmission by the senders even though this packets are received, as they are received after the set RTT. This is one of the primary causes of TCP Incast and Outcast whereby packets from different flows are dropped, or even though a packet is delivered or an ACK is received, the packets are still retransmitted because the receiver did not receive this packet before the RTT expired. queue size control. RED detects incipient congestion and provides feedback to end hosts by dropping the packets. The motivation behind RED is to keep the queue size

small, reduce burstiness and solve the problem of global synchronization.

B. Random Early Detection (RED)

RED was originally designed to solve the full queue problems associated with tail drop queues. The main objective of the Random Early Detection algorithm is:

1. To minimize the packet loss and queuing delay.
2. To avoid the global synchronization of sources.
3. To maintain high link utilization.
4. Decrease the packet loss probability and to minimize the overflow.
5. Decreasing the packet dropping Probability.

RED[6] is referred in terms of distinguishing constant congestion from transient upsurges, distinguishing between different congestion levels, and adjusting the packet dropping probability based on the severity of the congestion. This is because of the *Drop Activation Function* that it uses
for congestion detection as well as the *Drop Probability Function* used for congestion notification, and queue size control. RED detects incipient congestion and provides feedback to end hosts by dropping the packets. The motivation behind RED is to keep the queue size small, reduce burstiness and solve the problem of global synchronization. It is based on an average queue length that is calculated using an exponential weighted average of the instantaneous queue length. RED drops packets with certain probability depending on the average length of the queue. The drop probability increases from 0 to maximum drop probability *(maxp)* as average queue size increases from minimum threshold *(min/h)* to maximum threshold *(max/h)'*. If average queue size goes above *(max/h)*, all packets are dropped. for congestion detection as well as the *Drop Probability Function* used for congestion notification, and queue size control.

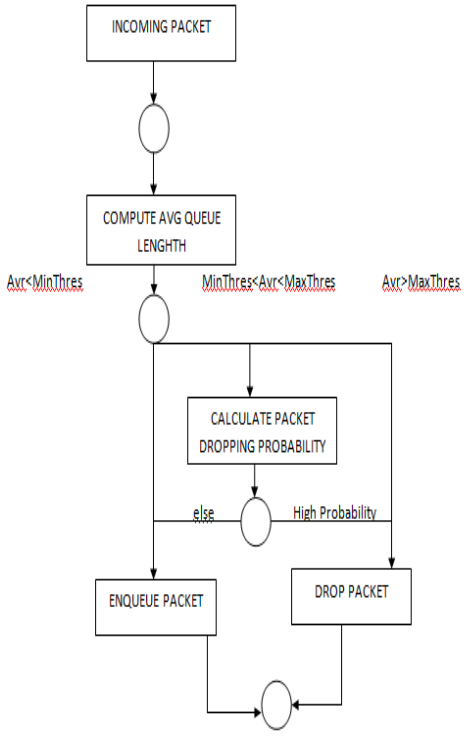

Fig 1: Random Early Detection.

A. DCTCP

DCTCP[7] has been proposed as a TCP Variant for data centers to achieve high burst tolerance, low latency and high throughput. DCTCP achieves full throughput for the smaller flows, compared to TCP. The mechanisms used by DCTCP are a simple active queue management scheme at the switch, based on Explicit Congestion Notification (ECN), and a window control scheme at the source which reacts to ECN marks by reducing the window size in proportion to the fraction of packets that are marked (contrast this with TCP which always cuts the window by half if at least one packet is marked). The performance of DCTCP is determined by two parameters: (i) K, the marking threshold on the queue at the switch above which all packets are marked; and (ii) g, the weight used for exponentially averaging ECN mark values at the source. The fairness properties of DCTCP for flows with diverse RTTs is better than TCP–Drop-tail and achieve a better fairness than TCP–RED.

Algorithm:

A. Switch Side: DCTCP has only a single parameter, the marking threshold, K. An arriving packet is marked with the Congestion

Experienced codepoint, if the queue occupancy is greater than K upon it's arrival, else it is not marked. The RED marking scheme implemented by most of the modern switches can be re-purposed for DCTCP. We simply need to set both the low and high thresholds to K, and mark the packets based on instantaneous, instead of average queue length.

B. Source Side: DCTCP is designed to simultaneously achieve high throughput and very low queue occupancies. It does this by reducing its current window (hence, sending rate) in proportion to the extent of congestion. Specifically, a DCTCP source reduces its window by a factor that is proportional to the fraction of marked packets: the larger the fraction, the larger the decrease factor. This is in contrast to the behavior a TCP source which reacts to marked packets by always halving its window size.

C. ECN-Echo at the Receiver: The only difference between a DCTCP receiver and a TCP receiver is the way information in the Congestion Experienced codepoints is conveyed back to the sender. RFC 3168 states that a receiver sets the ECN-Echo flag in a series of acknowledged packets until it receives confirmation from the sender (through the CWR flag) that the congestion notification has been received. A DCTCP receiver, however, tries to accurately convey the exact sequence of marked packets back to the sender. The simplest way to do this is to ACK every packet, setting the ECN-Echo flag if and only if the packet has a marked Congestion Experienced codepoint. DCTCP receiver uses the trivial two state state-machines to determine whether to set ECN Echo bit. The states correspond to whether the last received packet was marked with the Congestion Experienced codepoint or not. Since the sender knows how many packets each ACK covers, it can exactly reconstruct the runs of marks seen by the receiver.

D. Controller at the Sender: The estimation of the fraction of packets marked, for every window of data (roughly one RTT) is as follows:

$$\alpha \leftarrow (1 - g) \times \alpha + g \times F \text{----------(1)}$$

Where, α is an estimation of the fraction of packets marked.

F is the fraction of packets that were marked in the last window of data.

$0 < g < 1$ is the weight given to new samples against the past in the estimation of α.

Given that the sender receives marks for every packet when the queue length is higher than K and does not receive any marks when the queue length is below K. Equation (1) implies that α estimates the probability that the queue size is greater than K. Essentially, α close to 0 indicates low, and α close to 1 indicates high levels of congestion. DCTCP senders start gently reducing their window as soon as the queue exceeds K. This is how DCTCP maintains low queue length, while still ensuring high throughput.

IV. CONCLUSION

TCP is the heart of Internet since its invention. But it has some shortcomings in DCNs. Lower latency, fairness and higher throughput are the basic requirements of DCNs, which are not fulfilled by traditional TCP variants. Thus DCTCP used in DCNs prove better than the traditional TCP, which gives the fairness to all hosts regardless of it being small or large flows and mitigates the TCP outcast problem compared to DropTail and Red.

V. REFERENCES

[1] Mohiuddin Ahmed, Abu Sina Md. Raju Chowdhury, Mustaq Ahmed, Md. Mahmudul Hasan Rafee ,"An Advanced Survey on Cloud Computing and State-of the-art Research Issues", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012

[2] C. Guo et al. Bcube: High performance, server-centric network architecture for data centers. In SIGCOMM, 2009

[3] A. Kabbani and B. Prabhakar. In defense of TCP. In The Future of TCP:Train-wreck or Evolution, 2008.

[4] Y. Chen, R. Griffith, J. L. A. J. R. H. K. (2009). "Understanding TCP Incast Throughput Collapse in Datacenter Networks". Workshop on Research in Enterprise Networks (WREN'09).

[5] Pawan Prakash, Advait Abhay Dixit, Y Charlie Hu, and Ramana Rao Kompella. "The TCP Outcast problem: Exposing unfairness in DataCenter Networks". In NSDI , pages 413426, 2012.

[6] S.Floyd, "TCP and Explicit Congestion Control",ACM Computer Communication review, V.24 N.5, p.1 0-23, October 1994.

[7] Floyd, S. and Jacobson, V. (1993). Random early detection gateways for congestion

avoidance. Networking, IEEE/ACM Transactions on, 1(4):397 413.

[8] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, JitendraPadhye,Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. "Data Center TCP (DCTCP)". In Proceedings of the ACM SIGCOMM 2010 Conference , SIGCOMM 10, pages 6374. ACM, 2010.

[9] Aggarwal, A., Savage, S., and Anderson, T. (2000). Understanding the performanceof TCP Pacing. In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 3, pages1157–1165. IEEE.

[10] McKenney, P. E. (1990). Stochastic fairness queueing. In INFOCOM'90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE, pages 733–740. IEEE.

[11] Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113.