International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Congestion Control Mechanism for TCP in Data Centered Networks Using Multithreading

Tejashri P. Mane[1], Snehal Kanade[2]

[1]Computer Engineering Department, SKNSITS, Lonavala, India

[2]Professor, Computer Engineering Department, SKNSITS, Lonavala, India

Abstract: ***Transport Control Protocol (TCP) incast congestion happens in high-bandwidth and low-latency networks once multiple synchronized servers send data to identical receiver in parallel.TCP congestion could severely degrade the performance in many-to-one traffic pattern applications like MapReduce and search. The main limitation of TCP is that congestion in network which resulted into huge data loss, more waiting time etc. However existing methods focuses on either on the reduction of the waiting time for packet loss recovery using the faster retransmissions or controlling switch buffer occupation to prevent overflow with ECN as well as modified TCP on both the sender and receiver sides. This system is mainly focusing on presenting efficient solution which can be able to prevent packet loss before the incast congestion condition in network rather than packet loss recovery after actual packet loss in existing cases. This proposed method presented in this system is nothing but congestion avoidance in TCP based networks. The system presents Cloud Incast Congestion Control for TCP (CICT) scheme on the receiver side using the concepts of multi threading to manage parallelism concepts of data communication in data center networks. This method not only reduces the costs required for router or switch modifications but also minimizes the working on sender and receiver nodes.***

Keywords: TCP, Incast Congestion, Data Center Networks.

1. Introduction

TCP is a end-to-end protocol. To achieve the reliable transmission the retransmit timers are used: for the segments sent each time, the sender expects an ACK from the receiver before the timer expires. If ACK is not received within time, then some segments considered to be lost, due to the network congestion and will be retransmitted at some appropriate instant later. The Transmission Control Protocol (TCP) is used as the transport-layer protocol for reliable data transfer in data center networks, similar to the Internet.

Todays, data center applications and web search generally having the Partition/Aggregate communication architecture. Whenever a request is arrived at the node, it is partitioned and sent to a number of worker nodes. And then, the response data is generated by the workers and transmitted to a common node for aggregation which is known as aggregator node. Such type of traffic may cause network congestion due to the multiple workers send the response data to the same aggregator at the same time. The data center networks commonly accommodate applications and web search that shows the incast communication pattern; multiple senders simultaneously transmit TCP data to a single aggregator.This pattern causes TCP performance degradation in terms of goodput and request completion time, as a result of the severe packet loss at Top of Rack (ToR) switches. The TCP senders aggressively transmit packets which causes throughput degradation even though the network capacity that is bandwidth-delay product, is very small.

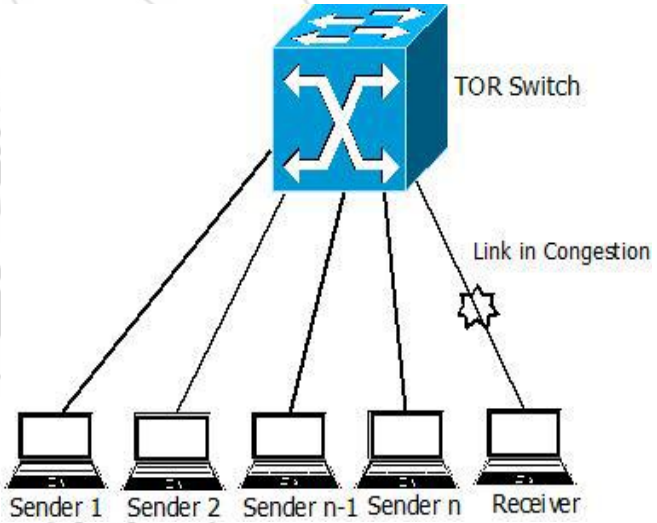

Figure 1: General Scenario of Data Centered Networks

Reason for the congestion is many synchronized servers send data to one receiver in parallel.This performance degradation of many-to-one TCP connections is called TCP incast congestion [1].Sender side TCP uses a congestion window to do congestion avoidance. The congestion window has the vast amount of data which is to be sent out on a connection without acknowledgement.TCP knows the congestion when it doesn't receive an acknowledgement for a packet within the expected timeout.

TCP incast collapse occurs due to the highly bursty traffic of multiple TCP connections which overflows the Ethernet switch buffer in a short period of time, causing intense packet loss and thus TCP retransmission and timeouts. Previous solutions focused on either reducing the wait time for packet loss recovery with faster retransmissions [3], or controlling switch buffer occupation to avoid overflow by using ECN and modified TCP on both the sender and receiver sides.

2. Literature Review

The TCP incast problem was reported first by D. Nagle et al. in the design of a scalable storage architecture[2].They found that the concurrent traffic between a client and many storage devices overows the network as the number of storage devices increases. This results in multiple packet losses and timeout. To mitigate the incast congestion, they reduce the clients receive socket buffer size to under 64kB.They also suggest to tune at the TCP level such as reducing the duplicate ACK threshold and disabling the slow-start to avoid retransmission timeout. However, they do not address the fundamental incast problem.

Two main approaches that address the incast problem have been proposed:

1) The First approach reduces the RTOmin from a millisecond to a microsecond granularity. This solution is very effective for cluster-based storage systems where the main performance metric is to enhance TCP throughput[2]. Nonetheless, it is not adequate for soft real-time applications such as web search because it still induces high queuing delay.
2) The second approach is to employ congestion avoidance before the buffer overflows. RTT is usually a good congestion indicator in a wide area network, so that a delay based congestion avoidance algorithm such that TCP Vegas[5] may be a good candidate. However,it is well known that the microsecond granularity of RTT in data centers may be too sensitive to distinguish the network congestion from the delay spikes caused by the packet/forwarding processing overhead. DCTCP[4] uses the Explicit Congestion Notification (ECN) to explicitly detect network congestion, and provides fine-grained congestion window based control by using the number of ECN marks.

A. Phanishayee, E. Krevat, V. Vasudevan, D. Andersen,G. Ganger, G. Gibson, and S. Seshan,in their paper studied TCP *Incast* which occurs when a client simultaneously receives a short burst of data from multiple sources, overloading the switch buffers associated with its network link such that all original packets from some sources are dropped. When this occurs, the clients receives no data packets from those sources and so sends no acknowledgement packets, requiring the sources to timeout and then retransmit. Often, the result of these TCP timeouts is an order of magnitude decrease in goodput. This paper discusses various TCP implementations such as reduced duplicate ACK threshold, disabling TCP slow start and SACK(Selective Acknowledgement).But this cannot eliminate incast congestion. SACK overcomes the problem of detection of multiple lost packet. ACK has block which describes which segments are being acknowledged. It initializes the variable pipe to determine how much data is outstanding in network and set congestion window to half of the current window size. SACK is not provided by receiver and hard to implement.

V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D.Andersen, G Ganger, G. Gibson, and B. Mueller,in their method described the another method to avoid incast congestion in data center networks. This paper presents an effective solution which enables microsecond granularity timeouts. It is sender side only approach. Faster retransmission is possible. It focuses on only how to mitigate impact of packet loss. It doesnot avoid the packet loss that is it is reactive measure of congestion control. It reduces millisecond granularity to microsecond granularity. But this method doesnot provide satisfactory solution to avoid incast congestion.

Data center TCP(DCTCP),TCP-like protocol for data center networks. DCTCP uses Explicit Congestion Notification (ECN), a feature already available in modern commodity switches. DCTCP combines Explicit Congestion Notification (ECN) with a novel control scheme at the sources. It extracts multibit feedback on congestion in the network from the single bit stream of ECN marks. Sources estimate the fraction of marked packets, and use that estimate as a signal for the extent of congestion. DCTCP operates with very low buffer occupancies while still achieving high throughput. The DCTCP algorithm has three main components:

- Simple Marking at the Switch:
  DCTCP employs a very simple active queue management scheme. There is only a single parameter, the marking threshold, K. An arriving packet is marked with the CE codepoint if the queue occupancy is greater than K upon its arrival. Otherwise,it is not marked. This scheme ensures that sources are quickly notified of the queue overshoot.
- ECN-Echo at the Receiver:
  The only difference between a DCTCP receiver and a TCP receiver is the way information in the CE codepoints is conveyed back to the sender.RFC 3168 states that a receiver sets the ECN-Echo flag in a series of ACK 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 CE codepoint.
- Controller at the Sender:
  The sender maintains an estimate of the fraction of packets that are marked, which is updated once for every window of data (roughly one RTT).

TCP Vegas is a TCP implementation which is a modification of Reno. It overcomes the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule. Also it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and it also suggest a modified slow start algorithm which prevent it from congesting the network. It does not depend solely on packet loss as a sign of congestion. It detects congestion before the packet losses occur. It does not use the loss of segment to signal that there is congestion. It determines congestion by a decrease in sending rate as compared to the expected rate, as result of large queues building up in the routers. Thus whenever the calculated rate is too far away from the expected rate it increases transmissions to make use of the available bandwidth, whenever the calculated rate comes too close to the expected

value it decreases its transmission to prevent over saturating the bandwidth.

ICTCP:Incast Congestion control for TCP in data center networks,this paper focuses on avoiding packet loss before incast congestion. It performs incast congestion avoidance at the receiver side by preventing incast congestion. The receiver side is a natural choice since it knows the throughput of all TCP connections and the available bandwidth[1].The receiver side can adjust the receive window size of each TCP connection, so the aggregate burstiness of all the synchronized senders are kept under control. This paper presented the design, implementation, and evaluation of ICTCP to improve TCP performance for TCP incast in data-center networks. In contrast to previous approaches that used a fine-tuned timer for faster retransmission, they focussed on a receiver-based congestion control algorithm to prevent packet loss. ICTCP adjusts the TCP receive window based on the ratio of the difference of achieved and expected per-connection throughputs over expected throughput and the available bandwidth.

3. Motivation

The many-to-one traffic pattern is common in today's most of the data-center networks. Data centered networks may consist of applications like MapReduce and search, where many-to-one traffic pattern is used. It uses distributed file systems, where the files are stored on multiple servers. When we want to use any file or access the data stored in files, it may be possible that multiple blocks of file are stored on multiple servers. When we are trying to fetch a multiple files of blocks stored on multiple servers at the same time, incast congestion happens. So it is essential to provide a congestion control mechanism to avoid the congestion.

4. Proposed Approach

Proposed system consist of mainly focusing on presenting efficient solution which can prevent packet loss before the incast congestion condition occurs in network rather than packet loss recovery after actual packet loss in existing cases.

This proposed system is nothing but congestion avoidance in TCP based networks. This proposed approach presented to prevent incast TCP congestion over data center networks. We are presenting Cloud Incast congestion Control for TCP scheme on the receiver side using the concepts of multi threading to manage parallelism concepts of data communication in data center networks. This method not only reduces the costs required for router or switch modifications but also minimizes the working on sender and receiver nodes. Basically proposed method adjusts the TCP receive window proactively before packet loss occurs in network. The data center networks are independent TCP senders and receivers connected through the same ToR switch or cloud servers.

In this system we are going to more real-time to claim the efficiency of proposed approach using the real time clouds through which TCP sender and receiver communicates. The performance of proposed approach is measured using goodput.

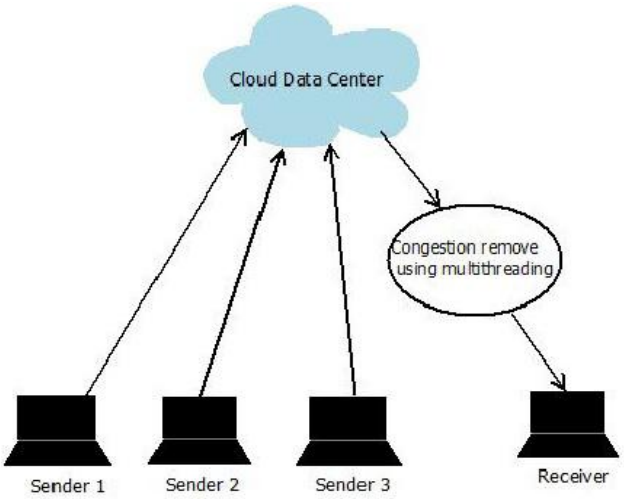

Figure 2: System Architecture

4.1 Algorithm

Input
k: No. of client's request
n: No. of allowed request
a[client][throughput]=[k=1 to k=n][throughput]
p[client][priorities]=[k=1 to k=n][0]
Algorithm
Step-1. for(k=0;k<=n;k++)
Server accept request from client
Step-2. end for
Step-3. Scheduling and serving by server to client
Step-4. use sorting algorithm to set P[][] in descending order
Step-5. for(i=0;i≤ lengthof(p);i++)
server send data to receiver.
Step-6. end for
Step-7. Receiver send acknowledgement to server in response of data reception.
Step-8. Halt.

Algorithm explains the general steps of how the system works when there is possibility of congestion. First, multiple clients/senders send the data/request to the server/receiver. Server accepts these requests and apply multithreading to it such that one thread is assigned to each request. Whenever multiple request are there, one request is served at a time. So it avoids the congestion situation before it happens. System performance is measured in terms of goodput. Goodput is nothing but the application level throughput and calculated as no of useful information bits delivered by the network to a certain destination per unit of time. For solving the scalability issue data is stored at the cloud according to the priority and it is downloaded from the cloud server.

5. Results

Results show that goodput achieved in proposed system is better than the existing system. With no of senders increasing, goodput also increases. It drops for some connections due to the TCP timeout. Goodput is the application level throughput that is number of useful

information bits delivered by the network to a certain destination per unit of time. As the amount of servers increasing, system achieves higher goodput.

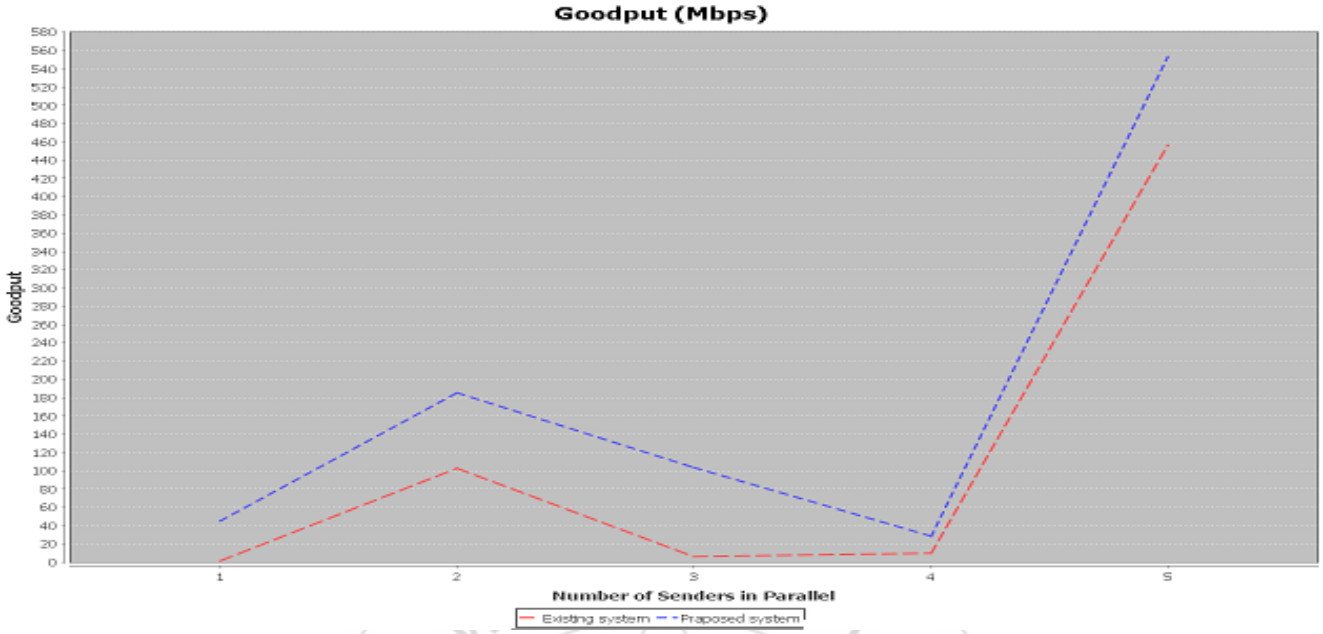

Figure 3: Result Graph

6. Conclusion

This system presents the congestion avoidance mechanism in TCP based networks. This proposed approach presented to prevent incast TCP congestion over data center networks. The system presents Cloud Incast Congestion Control for TCP (CICT)scheme on the receiver side using the concepts of multi-threading to manage parallelism concepts of data communication in data center networks. This method not only reduces the costs required for router or switch modifications but also minimizes the working on sender and receiver nodes. System achieves the better goodput or higher goodput with number of servers increasing.

7. Acknowledgement

The authors would like to thank the researchers as well as publishers for making their resources available and teachers for their guidance. We also thank the college authorities for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

References

[1] Haitao Wu,Zhenqian Feng, Chauanxiong Guo, Yongguang Zhang,"ICTCP:Incast congestion control for TCP in data center networks",IEEE/ACM Transaction On Networking,Vol. 21,No. 2,April 2013..

[2] A. Phanishayee, E. Krevat, V. Vasudevan, D. Andersen, G. Ganger, G. Gibson, and S. Seshan, "Measurement and analysis of TCP throughput collapse in cluster-based storage systems", inProc. USENIX FAST, 2008, Article no. 12.

[3] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. Andersen, G. Ganger, G. Gibson, and B. Mueller, "Safe and effective fine-grained TCP retrainsmissions for datacenter communication", inProc. ACM SIGCOMM, 2009, pp. 303-314.

[4] M.Alizadeh, A.Greenberg, D.Maltz, J.Padhye ,P.Patel, B.Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)", inProc. SIGCOMM, 2010, pp. 63-74.

[5] L. Brakmo and L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet",IEEE J. Sel. Areas Commun., vol. 13, no. 8, pp. 1465-1480, Oct. 1995.

[6] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements analysis", inProc. IMC, 2009, pp. 202-208.

[7] E. Krevat, V. Vasudevan, A. Phanishayee, D. Andersen, G. Ganger, G. Gibson, and S. Seshan, "On application-level approaches to avoiding TCP throughput collapse in cluster-based storage systems", inProc. Supercomput., 2007, pp. 14.

[8] JC.Guo,H.Wu,K.Tan,L.Shi,Y.Zhang,andS.Lu,"DCell:A scalable and fault tolerant network structure for data centers", inProc. ACM SIGCOMM, 2008, pp. 75-86.

[9] V. Jacobson, R. Braden, and D. Borman, `TCP extensions for high performance", RFC1323, May 1992.

[10] Y.Chen, R.Gri_th, J. Liu, R. Katz, and A. Joseph, "Understanding TCP incast throughput collapse in datacenter networks", inProc. WREN, 2009, pp. 73-82.

[11] P. Mehra, A. Zakhor, and C. Vleeschouwer, "Receiver-driven bandwidth sharing for TCP"', inProc. IEEE INFOCOM, 2003, vol. 2, pp.1145-1155.

[12] Balveer Singh,"A Comparative Study of Different TCP Variants in Networks",in International Journal of Computer Trends and Technology(IJCTT),volume 4, Issue 8,August 2013.