ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.4, Pages : 66-71 (2014)
Special Issue of ICCEIT 2014 - Held on September 01, 2014 in The Solitaire Hotel, Bangalore, India

Overcoming TCP Incast with LTTP in Many to One Communications

Jeenu Alexander
Department of Computer Science
KMEA Engineering College
Ernakulam , Kerala, India
jeenalex@gmail.com

Joish George
Department of Computer Science
KMEA Engineering College
Ernakulam,Kerala,India

***Abstract*—**The Transmission Control Protocol (TCP) is one of the core transport layer protocols which ensures reliable data delivery. In many to one communications there arise a problem known as TCP Incast which is actually a drop in throughput. This arises due to the overflow of output buffer of switch and TCP's Retransmission Timeout. Earlier solutions included controlling switch buffer or updating OS/hardware .In this work we improve the UDP-based transmission with FEC and CRC. Through ns2 based simulations we can observe that the newly designed protocol will not degrade the throughput in many to one communications when the number of servers are increased. LTTP when compared with DCTCP, performs better in maintaining overall goodput of the many to one communications.

***IndexTerms*—**TCP Incast, reliable, bandwidth, goodput

INTRODUCTION

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite (IP), and is so common that the entire suite is often called TCP/IP.TCP is known for its reliable data delivery, congestion control and error flow control. TCP entertains connection oriented communications. Cloud computing realizes the dream of "computing as a utility". People outsource their computing and software capabilities to cloud providers and pay for the service usage on demand[1].Cloud data centers run both online services and back-end computations. Since most distributed computations in data centers are bandwidth-hungry they propose to increase network capacity. They thus require congestion free reliable data transmission.

As for data transmission between servers, TCP is widely used in today's data center networks, since it has been proven a great success in the Internet for both reliable delivery and congestion control. However, the specific application pattern and network environment in data centers pose new challenges to TCP to work smoothly. In such conditions a problem called TCP Incast arises, this is mainly due to the drop in the overall throughput of the communication. The actual condition here is that a client sends requests to multiple servers and wait to get reply from each of them The client waits indefinitely until it gets all the requested blocks from the servers to which it had send requests .The client will never more send requests until it has attained all the requested ones. [1]TCP Incast causes goodput collapse for two reasons. Firstly, when servers simultaneously send response packets back to the client, the response packets will overflow the output buffer of the switch which directly connects the client. Secondly, the default value of TCP's Retransmission Timeout (RTO) is 200 milliseconds in most operating systems. It means that once a timeout occurs, the TCP connections will be idle for quite a long time period before the servers retransmit the dropped packets, since the RTT (Round-Trip Time) is only hundreds of microseconds in data center networks. After the retransmission timer timeouts, the servers will again simultaneously send the response packets, which causes switch buffer overflow and retransmission for a new round, so and so forth.[1]

The goodput degradation in many-to-one communications will significantly delay the task finish time of distributed computations, which is further translated to the violation of SLA. Since the root cause for TCP Incast is the shallow buffer in switches as well as the mismatch between RTO and RTT.TCP incast has risen to be a critical problem recently in data center networks due to its catastrophic goodput collapse. In incast communication pattern, multiple servers concurrently transmit data blocks to a single client and any server can not send another data block until all the servers finish transmitting the current data block. When the number of server increases, the goodput of the receiver will become lower than the capacity of the bottleneck link in one or even two orders of magnitudes. The incast communication pattern exists in many popular applications.

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.4, Pages : 66-71 (2014)
Special Issue of ICCEIT 2014 - Held on September 01, 2014 in The Solitaire Hotel, Bangalore, India

To avoid the performance deterioration of TCP incast, lots of attempts have been made to find the causes of TCP incastand the methods to solve it.TCPincast problem attracts increasing attention since the client suffers drastic goodput drop when it simultaneously strips data over multiple servers. Lots of attempts have been made to address the problem through experiments and simulations, few solutions can solve it fundamentally at low cost.

TCP incast has risen to be a critical problem recently in data center networks due to its catastrophic goodput collapse. In incast communication pattern, multiple servers concurrently transmit data blocks to a single client and any server can not send another data block until all the servers finish transmitting the current data block. When the number of server increases, the goodput of the receiver will become lower than the capacity of the bottleneck link in one or even two orders of magnitudes. The incast communication pattern exists in many popular applications. To avoid the performance deterioration of TCP incast, lots of attempts have been made to find the causes of TCP incast and the methods to solve it. TCP incast problem attracts increasing attention since the client suffers drastic goodput drop when it simultaneously strips data over multiple servers. Lots of attempts have been made to address the problem through experiments and simulations. However, to the best of our knowledge, few solutions can solve it fundamentally at low cost.

We propose a new transport protocol to support many-to-one communications in data centers, which is called LTTP (LT-code based Transport Protocol). Since TCP's timeout is the root cause of low link utilization and goodput deterioration in TCP Incast, LTTP improves UDP-based LT (Luby Transform) code [12] for reliable delivery, which depends on FEC (Forward Error Correction) [13] with data redundancy. Since UDP cannot fairly share bandwidth with other protocols (such as TCP),TFRC (TCP Friendly Rate Control) [14] is also applied to adjust the data sending rates at servers for congestion control.

The intuition behind LTTP's design is that the rate-based congestion control scheme of TFRC ensures that the sender can still send data at an appropriate rate even in face of congestion, instead of stopping sending data for a relatively long time. In addition, LT code can restore the original data without requesting for retransmission as long as the number of packet losses/errors falls into a reasonable range. Each of the two schemes is used to overcome the other's limitations: TFRC maintains reasonable bandwidth utilization, while UDP based LT code ensures reliable data delivery[1].

Using ns2 simulations, we can show that LTTP can maintain high goodput for many to one communications in different topologies, no matter what the number of servers is. Our improvement on the decoding algorithm of LT code effectively improves the goodput of LTTP, and controls the bandwidth overhead of LT-code.

MOTIVATION

In this work, we mainly concentrate on achieving a reliable UDP-based protocol though we know that UDP is unreliable.Through the designing of such a system we get a goodput guaranteed in many to one communications also in case where number of servers are increased.

The TCP Incast problem [1] which is actually a drop in the overall goodput or a goodput collapse is overcome in the design to support any increase in number of servers.However, because many-to-one communication is common in both online services and back-end computations, LTTP shows its great promise, especially when the number of servers is large.Thus as per the proposed system a throughput oriented reliable protocol for many to one communications is to be obtained.

PRIORWORKS

In past years there have been many solutions to the TCP Incast problem mainly ICTCP and DCTCP ,others are FQCN [9] and AF-QCN [10].In ICTCP,TCP incast is studied in detail by focusing on the relationship among TCP throughput, round trip time (RTT) and receive window. The idea was to design an ICTCP (Incast congestion Control for TCP) scheme at the receiver side. In particular, the method adjusts TCP receive window proactively before packet drops occur. The implementation and experiments in the testbed demonstrated that zero timeout and high goodput for TCP incast was achieved. The implementation and evaluation of ICTCP, was to improve TCP performance for TCP incast in data center networks. Main focus was on receiver based congestion control algorithm to prevent packet loss. ICTCP adaptively adjusts TCP receive window based on the ratio of difference of achieved and expected per connection throughputs over expected ones, as well as the last-hop available bandwidth to the receiver. A light-weighted, high performance Window NDIS filter driver was made to implement ICTCP. Compared with directly implementing ICTCP as part of the TCP stack, the driver implementation can directly support virtual machines, which prevail in data centers. ICTCP was effective to avoid congestion by achieving almost zero timeout for TCP incast, and it provides high performance and fairness among competing flows.

DCTCP, a TCP-like protocol for data center networks was designed. DCTCP leverages Explicit Congestion Notification (ECN) in the network to provide multi-bit feedback to the end hosts.ECN allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network.DCTCP delivers the same or better throughput than TCP, while using much lesser buffer space. Unlike TCP, DCTCP also provides high burst tolerance and low latency for short flows. A new variant of TCP, called Data Center TCP (DCTCP) was designed. The work was motivated

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.4, Pages : 66-71 (2014)
Special Issue of ICCEIT 2014 - Held on September 01, 2014 in The Solitaire Hotel, Bangalore, India

by detailed traffic measurements from a thousands of server data center cluster, runningproduction soft real time applications. Several performance impairments were noticed, and linked these to the behavior of the commodity switches used in the cluster. It was found that to meet the needs of the observed diverse mix of short and long flows, switch buffer occupancies need to be persistently low, while maintaining high throughput for the long flows. DCTCP met these needs. DCTCP relies on Explicit Congestion Notification (ECN), a feature now available on commodity switches. DCTCP succeeds through use of the multi-bit feedback derived from the series of ECN marks, allowing it to react early to congestion.

Quantized Congestion Notification (QCN) was been developed for IEEE 802.1Qau to provide congestion control at the Ethernet Layer or Layer 2 in data center networks (DCNs) by the IEEE Data Center Bridging Task Group. One drawback of QCN is the rate unfairness of different flows when sharing one bottleneck link. In FQCN, an enhanced QCN congestion notification algorithm, called fair QCN (FQCN), to improve rate allocation fairness of multiple flows sharing one bottleneck link in DCNs was proposed. FQCN identifies congestion culprits through joint queue and per flow monitoring, feedbacks individual congestion information to each culprit through multi-casting, and ensures convergence to statistical fairness. The stability and fairness of FQCN via Lyapunov functions was measured and the performance of FQCN was evaluated through simulations in terms of the queue length stability, link throughput and rate allocations to traffic flows with different traffic dynamics under three network topologies. Simulation results confirmed the rate allocation unfairness of QCN, and validate that FQCN maintains the queue length stability, successfully allocates the fair share rate to each traffic source sharing the link capacity, and enhances TCP throughput performance in the TCP Incast setting.

The QCN(Quantized Congestion Notification) algorithm was designed to be stable, responsive, and simple to implement. However, it does not provide weighted fairness, where the weights can be set by the operator on a per-flow or per-class basis. Such a feature can be very useful in multi-tenanted Cloud Computing and Data Center environments. AFQCN addresses this issue. Specifically, we develop an algorithm, called AF-QCN (for Approximately Fair QCN), which ensures a faster convergence to fairness than QCN, maintains this fairness at fine-grained time scales, and provides programmable weighted fair bandwidth shares to flows/flow-classes. It combines the QCN algorithm ,and the AFD algorithm. AF-QCN requires no modifications to a QCN source (Reaction Point) and introduces a very light-weight addition to a QCN capable switch (Congestion Point). The results obtained through simulations show that AF-QCN retains the good congestion management performance of QCN while achieving rapid and programmable (approximate) weighted fairness.

AF-QCN, an algorithm that adds a programmable bandwidth partitioning component based on AFD to the QCN Congestion Point mechanism. No changes are needed at a QCN Reaction Point.AF-QCN achieves weighted fairness at the granularity of a few milliseconds. This enables Data Center operators to provide programmable differential bandwidth allocation for flows or flow classes, a feature very useful in multi-tenanted Cloud Computing and Data Center environments. The results obtained via simulations and a hardware implementation showthat AF-QCN retains the good properties of QCN (stability ,responsiveness, and simplicity), while achieving rapid and programmable bandwidth partitioning.

SYSTEM MODEL

Every ns2 based project starts with the normal wireless or wired procedure. Here we design a wired many to one communication. A simulation environment of 10-11 nodes which are wired is created.T he system starts with designing a network topology of the described kind as depicted in Figure1.

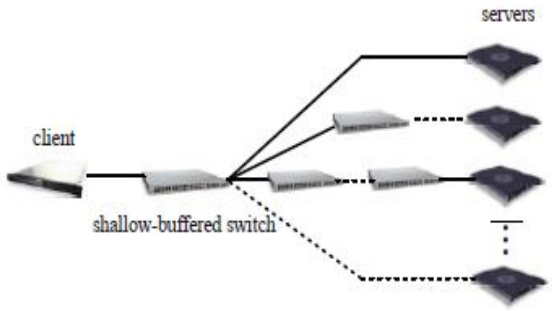

Fig 1: TCP Incastsetup[1]

The nodes are created and the duplex connections are made between the nodes. The corresponding agents are attached and the flows id's are also given.The complete framework of LTTP to support many-to-one communication in data centers includes two parts, i.e., the data channel from each server to the client, and the control channel between the client and each server. In the data channel, we improve LT code for reliable data transport, and adopt TFRC for controlling the traffic sending rate at servers. The control channel is employed by the client to issue data requests to servers and send terminating signals to the servers as soon as the requested da ta have been restored.

The servers also use the control channel to send decoding parameters to the client. The decoding parameters include the original data size and block size, which are used by the client to execute the decoding process. For the control channel messages, the data size is small enough to be put into a single packet. Hence, it is unnecessary to employ coding for transmission. Instead, we establish a TCP connection for each

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.4, Pages : 66-71 (2014)
Special Issue of ICCEIT 2014 - Held on September 01, 2014 in The Solitaire Hotel, Bangalore, India

client-server pair to deliver the control channel messages reliably.[1] The workflow in LTTP is described as follows. First, the client establishes control channels (TCP connections) to all the servers. Second, the client sends requests to all the servers simultaneously through the control channel, asking the servers to start sending the data. Third, once receiving the request, the servers use control channel to send the decoding parameters back to the client. Meanwhile, each server starts to employ LT code to produce and send encoding packets continually.

TFRC is used by both servers and the client to control the sending rate. Finally, as soon as the original data is successfully restored, the client sends a terminating signal through control channel back to the corresponding server, which informs the server to stop encoding.

When all senders adopt digital fountain based protocols and act as selfish players to inject data in network as fast as they can, a Nash equilibrium can be reached eventually. At this equilibrium state, the throughput of each flow is similar to that when all the senders use TCP. However, in typical many-to-one communication pattern when TCP Incast occurs, the transferred data volume is very small, and it is of high probability that the Nash equilibrium cannot be reached before all the data have been transferred. So we still have to spend extra efforts to deal with congestion control in LTTP.

In our implementation, the upper applications on both the server side and the client side are responsible for making decisions that when to send data and which channel to be used. For example, when the application on the server side receives a request, it calls the interface of LTTP to send decoding parameters back to the client through control channel. Next, the server starts the encoding process to generate encoding data and calls the interface of LTTP to transport the encoding data to the client side through data channel. The application on the client side calls the interface of LTTP to receive encoding data and restore the original data. Once the original data is successfully restored, the application invokes the LTTP to send the terminating signal to the server through the control channel.

The complete framework of LTTP to support many-to-one communication in data centers includes two parts, i.e., the data channel from each server to the client, and the control channel between the client and each server. In the data channel, we improve LT code for reliable data transport, and adopt TFRC for controlling the traffic sending rate at servers. The control channel is employed by the client to issue data requests to servers and send terminating signals to the servers as soon as the requested data have been restored. The servers also use the control channel to send decoding parameters to the client. The decoding parameters include the original data size and block size, which are used by the client to execute the decoding process .

For the control channel messages, the data size is small enough to be put into a single packet. Hence, it is unnecessary to employ coding for transmission. Instead, we establish a TCP connection for each client-server pair to deliver the control channel messages reliably.

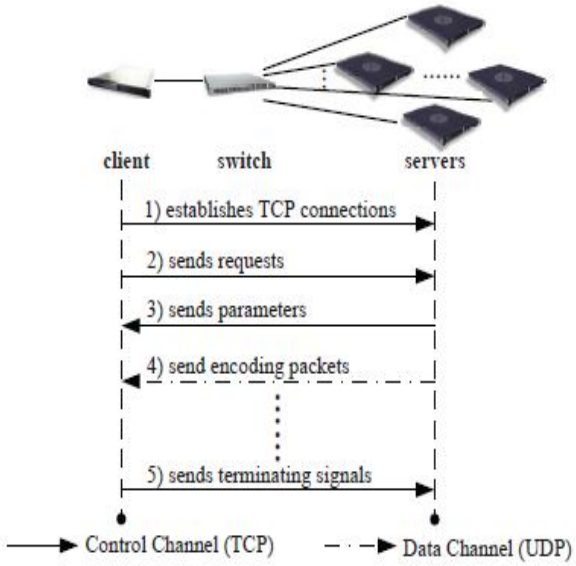

Fig 2: Workflow of the protocol [1]

The working of the simulation is similar to the UDP simulation but the number of packets reaching the destination is much more than normal UDP. This simulation includes an error correction method FEC(forward error correction).The evaluation and comparisons can be made and can be inferred that the throughput and end to end delay is much enhanced in the developed scenario.

The UDP protocol when used along with a forward error correcting code increased the overall output obtained but the packet loss happening is high .So we employ the cyclic redundancy check to control errors and packet loss. The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted.

The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission. The technique is also sometimes applied to data storage devices, such as a disk drive. In this situation each block on the disk would have check bits, and the hardware might automatically initiate a reread of the block when an error is detected, or it might report the error to software. The material that follows speaks in terms of a “sender” and a “receiver” of a “message,” but it should be understood that it applies to storage writing and reading as well.

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.4, Pages : 66-71 (2014)
Special Issue of ICCEIT 2014 - Held on September 01, 2014 in The Solitaire Hotel, Bangalore, India

The Cyclic Redundancy Check is the most powerful of the redundancy checking techniques , the CRC is based on binary division. In CRC a sequence of redundant bits, called the CRC or the CRC remainder is appended to the end of a data stream. The resulting data becomes exactly divisible by a second, predetermined binary number. At its destination, the incoming data is divided by the same number. It protects the data with a checksum or cyclic redundancy check.[14].

EVALUATION RESULTS

By using ns2 simulations we evaluate the performance of the developed protocol.The client is connected to many servers through a single switch.The client sends requests to multiple servers.The parameters used in the project are throughput calculation ,the end to end delay and the packet drops are also calculated.

Table1 shows the different parameters that are being compared when different protocols are taken into account. The protocol which uses Forward Error Correction has more packet loss than compared with the Cyclic redundancy check. The throughput acquired by using LTTP with CRC is considerably high.

Protocol	Throughput	End to end delay	Packet drop
TCP	677.2 KB	38.211 ms	156
UDP	3018 KB	31.68 ms	11228
LTTP with FEC	3043.8 KB	34.3186 ms	30130
LTTP with CRC	3052 KB	37.9703 ms	23502

Table 1 . Comparison Table

The graph has been plotted for the packet loss happening against time. The packet drop in LTTP using the CRC method. The x-axis shows the passage of time and the y-axis shows the number of packet dropped at each period of time

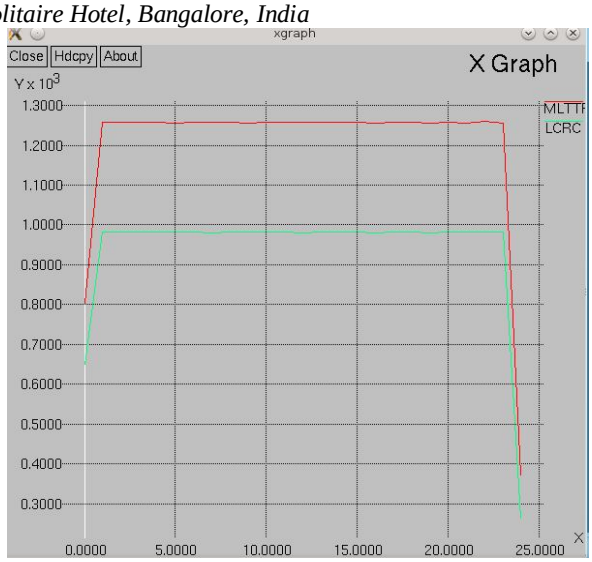

Fig 3 . Graph plotted to compare the protocols

CONCLUSION

Throughput collapse is a severe problem faced in data centers in many to one communications. In this paper the LTTP protocol is developed additionally employing CRC and FEC [14].Through ns2 based simulations the average throughput is increased by overcoming the problem of TCP Incast. The packet loss at the same time can be observed.

REFERENCES

[1]Changlin Jiang, Dan Li, Member, IEEE, and MingweiXu, Member, IEEE, "LTTP: An LT-code based Transport Protocol for Many-to-One Communication in Data Centers".2014

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar,S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in Proceedings of the ACM SIGCOMM 2010, New York, NY, USA, 2010,pp. 63–74.

[3] S. Ghemawat, H. Gobioff, and S.-T.Leung, "The google file system,"in Proceedings of SOSP '03, New York, NY, USA, 2003, pp. 29–43.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in Proceedings of the ACM SIGCOMM 2008, New York, NY, USA, 2008, pp. 63–74.

[5] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in Proceedings of the ACM SIGCOMM 2009, New
York, NY, USA, 2009, pp. 51–62.

[6] D. Nagle, D. Serenyi, and A. Matthews, "The panasas ActiveScale storage cluster: Delivering scalable high

bandwidth storage," in Proceedings of the 2004 ACM/IEEE conference on Supercomputing, Washington, DC, USA, 2004, pp. 53–62.

[7] 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," in Proceedings of SIGCOMM '09, 2009, pp. 303–314.

[8] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: incast congestion control for TCP in data center networks," in Proceedings of the Co-NEXT '10, New York, NY, USA, 2010.

[9] Z. Yan and N. Ansari, "On mitigating tcpincast in data center networks,"in Proceedings of the IEEE INFOCOM 2011, Apr. 2011, pp. 51–55.

[10] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar, "Afqcn: Approximate fairness with quantized congestion notification for multi-tenanted data centers," in IEEE 18th Annual Symposium on High Performance Interconnects (HOTI), 2010, pp. 58–65.

[11] 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 WREN '09, New York, NY, USA, 2009, pp. 73–82