International Journal of Advances in Computer Networks and Its Security
Volume 6 : Issue 3 *[ISSN 2250-3757]*
Publication Date : 30 December, 2016

A Delay-Based TCP for Data Center Networks

Yi-Cheng Chan, Wan-Chen Chang

***Abstract*—TCP Incast is a pathological behavior of TCP that results in gross under-utilization of link capacity in certain many-to-one communication patterns. The many-to-one communication is the main transmission mode in data center networks. In this paper, we modify the most famous delay-based TCP, Vegas, and propose a new congestion control mechanism for data center networks. The proposed DCVegas can quickly and appropriately adjust its congestion window size without the special help of the intermediate nodes. Through extensive NS-2 simulations, the results of DCVegas are compared with that of IA-TCP and TCP NewReno. DCVegas outperforms IA-TCP and NewReno in a variety of conditions. The proposed DCVegas maintains the end-to-end approach and effectively avoids the occurrence of TCP Incast in data center networks.**

Keywords*—*data center networks, TCP, Incast

I. Introduction

Nowadays, data center networks have become a key resource to provide online services such as web search, social networking, cloud computing, etc. Many online service providers such as Google, Amazon, Microsoft, Facebook and IBM have vastly invested in building data center networks to support large scale computing power and storage. To ensure the integrity of data transmission, the majority of data centers use TCP as its default transport protocol [6][11]. However, the high-bandwidth and low-latency environment of data centers is different from the TCP original assumptions [11]. When the servers send the requested to the client, there will be a lot of data into the switch at the same time. The switch may suffer buffer overflow and Incast congestion. It will cause lots of packet loss and network collapse, known as TCP Incast.

Delay-based TCP comes with a more sensitive congestion control feature is confirmed performs better than loss-based TCP in homogeneous environments [9]. Due to a data center network can be a homogeneous environment, it's no need to consider the competition problem between different variants of TCP. Therefore, we modify the most famous delay-based TCP, Vegas, and propose a new congestion control mechanism for data center networks. The proposed Data Center Vegas (DCVegas) can quickly and appropriately adjust its congestion window size without the special help of the intermediate nodes.

Yi-Cheng Chan, Wan-Chen Chang
National Changhua University of Education
Taiwan

DCVegas maintains the end-to-end approach and features two improvements: (1) A more positive window adjustment in the slow start phase. DCVegas estimates the amount of extra data that kept in the bottleneck and increases its congestion window size aggressively if the amount of extra data is less than the predefined parameter γ. Thus DCVegas can effectively utilize the bandwidth at the beginning of transmission without packet loss. (2) An improved congestion avoidance mechanism. Traditional TCP adjusts its congestion window linearly in the congestion avoidance phase. It is too sluggish for data center networks. DCVegas employs the history and the estimated amount of extra data to guide the window size changes. The new mechanism can more quickly and appropriately adjust its window size, and utilize the bandwidth.

Through the extensive NS-2 simulations, the results of DCVegas are compared with that of IA-TCP and TCP NewReno. DCVegas outperforms IA-TCP and NewReno in a variety of conditions. The rest of this paper is organized as follows. The related work is described in Section 2. Section 3 presents the proposed mechanism, DCVegas. Section 4 discusses the results of NS-2 simulations. Finally, the conclusions are presented in Section 5.

II. Related Work

In this section we first describe the TCP Incast problem and then depict the TCP Vegas, the base of the proposed DCVegas and finally describe the IA-TCP that is a famous TCP variant designed for data center networks.

A. *TCP Incast*

The TCP Incast problem was first reported by Nagle et al. [1] in the design of scalable storage architecture. Data center networks provides distributed service to satisfy the request of different users. When the client sends the information requested to the servers, the data will be transmitted to the client through the bottlenecks between switch and client, and then formation of many-to-one transmission mode. With the increasing synchronous transmission of the servers, there will be a lot of data into the switch at the same time, and also competing for the same output port. Therefore, the switch will result in a buffer overflow caused lot of packet loss. In the meantime, if the Retransmission Timeout (*RTO*) setting is not suitable [7][10].(The default RTO_{min} of traditional TCP is set to 200 *ms*, while the *RTT* can be less than 250 μs in data center networks [2].) It will cause network collapse and bandwidth utilization decrease, known as TCP Incast.

B. *TCP Vegas*

Vegas adopts a more sophisticated bandwidth estimation scheme that tries to avoid rather than to react to congestion [3]. It uses the measured *RTT* to accurately calculate the amount of data packets that a source can send. Vegas calculates the extra data (Δ) and doubles its congestion window every other *RTT*. Vegas estimates a proper amount

of extra data to be kept in the network pipe and controls the congestion window size accordingly. It records the *RTT* and sets *BaseRTT* to the minimum of ever measured round-trip times. The amount of extra data (Δ) is estimated as follows:

$$\Delta = (Expected - Actual) \times BaseRTT, \quad (1)$$

where *Expected* throughput is the *CWND* divided by *BaseRTT*, and *Actual* throughput represents the *CWND* divided by the newly measured smoothed-*RTT*. When the amount of Δ is greater than γ, Vegas leaves the slow-start phase. The *CWND* is kept constant when the Δ is between two thresholds α and β. If Δ is greater than β, the *CWND* will be reduced. On the other hand, if the Δ is smaller than α, the connection may be under utilizing the available bandwidth. Hence, the *CWND* will be increased. The rule for congestion window adjustment can be expressed as follows:

$$CWND = \begin{cases} CWND+1, & if \quad \Delta < \alpha \\ CWND-1, & if \quad \Delta > \beta, \\ CWND, & otherwise. \end{cases} \quad (2)$$

C. IA-TCP

Hwang et al. proposed a rate based congestion control algorithm, called IA-TCP [2]. This algorithm controls the total number of packets injected into the network pipe to meet the bandwidth delay-product (BDP). IA-TCP regulates the total number of outstanding data packets so that it does not exceed the path BDP. The aggregator controls the window size of the workers and then adds Δ(s) to the RTT_{min} for fine-grained rate control. The data packet rate is set to equation (3):

$$IARate = \frac{\sum_{i=1}^{n} Wi \times MSS}{\mathrm{RTT}_{\min} + \Delta} \le Link\ capacity \quad (3)$$

where MSS denotes the maximum segment size, *n* is the total number of concurrent connections, and w_i is the window size of the *ith* connection. IA-TCP control the window sizes of all connections to be equal to *W* for simplicity and fairness between the connections. Then they have the window size *W* from equation (4) as follows:

$$W = \frac{Link\ capacity / MSS \times RTT_{\min}}{n} \quad (4)$$

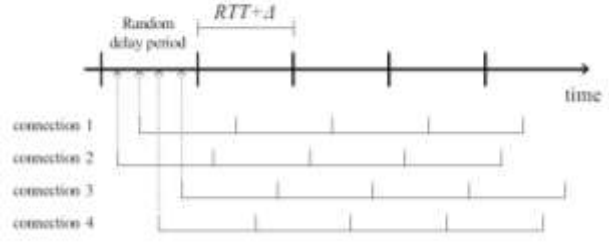


Figure 1. Example of random delay for the first ACK period

Even though the total number of outstanding packets may be maintained, if all the workers send their data in a synchronized fashion, which occurs more often than not, then it may still cause network overflow. Fig. 1 shows an example when a random delay is given for each connection for the first ACK, so that all outstanding packets are timely distributed over the path with n/(RTTmin+Δ) of mean interval from the next period.

III. Proposed Method

A data center network can be a homogeneous network environment in which every node adopts the same TCP version. This is a good opportunity to perform the delay-based TCP that is considered outperforms loss-based TCP in such environments. Therefore, we modify the most famous delay-based TCP, Vegas, and propose a new congestion control mechanism for data center networks, named Data Center Vegas (DCVegas). DCVegas is a sender-sided modification and hence it can be implemented solely at the end host. DCVegas enhances the slow start and congestion avoidance phase of Vegas. The details are described as follows.

A. Slow Start Phase

In slow start phase, TCP Vegas doubles the size of its congestion window only every other *RTT*. Such increasing speed is too conservative for data center networks. To be able to quickly make good use of link bandwidth in the connection initiation phase. The congestion window of DCVegas is adjusted every *RTT* instead of every other *RTT*. Furthermore, DCVegas increases its window size by 1.5 segment when it receives an ACK, as shown in equation (5):

$$CWND = CWND + 1.5 \quad (5)$$

To prevent packet loss in the slow start phase, we adjust γ to 0.5, a more sensitive parameter. When the amount of extra data (Δ) is greater than γ, DCVegas leaves its slow start phase and enters the congestion avoidance phase. Especially, Vegas reduces the congestion window size by 1/8 when it leaves the slow start phase. We believe such reduction is unnecessary to DCVegas. Because the γ is setting to a small value. Therefore, DCVegas keeps the same window size when it enters the congestion avoidance phase.

B. Congestion Avoidance Phase

In data center networks, servers may be configured a big server request unit (SRU). In such situation, a TCP connection may enter the congestion avoidance phase to complete the transmission. Therefore, the congestion avoidance mechanism also needs to be improved.

TCP Vegas updates its congestion window linearly in the congestion avoidance phase, it is too sluggish for a high BDP network [8]. If $\Delta < \alpha$, the connection may be under utilizing the available bandwidth. For the increment of congestion window, DCVegas has the history to guide the window size changes. DCVegas records the number of consecutive increments due to $\Delta < \alpha$ and refers to this value as *succ*. Whenever the *CWND* should be increased due to $\Delta < \alpha$, it is updated as follows:

$$CWND = CWND + (\beta - \Delta) \times succ \quad (6)$$

When first estimation $\Delta < \alpha$, continuous occurrence $\Delta < \alpha$ (*succ*) will be denoted by 1. The congestion window size will be increased by (β - Δ). The next consecutive estimation

of $\Delta < \alpha$, that means network are still have idle bandwidth can be used. The congestion window size will be increased by $(\beta - \Delta) \times 2$, and so on. The *succ* will be reset whenever $\Delta \geq \alpha$. The idea is that if the increment was successful, it might be the case that there is enough bandwidth and it is worthwhile to move to a more aggressive increasing strategy. In addition, to ensure that the *CWND* will not be increased too fast, DCVegas can at most double the size of *CWND* for every estimation of $\Delta < \alpha$.

In aspect of reducing *CWND*, when measured $\Delta > \beta$, we consider that network may beginning congestion. DCVegas will make decrease *CWND* more serious. DCVegas uses the difference of Δ and $(\alpha + \beta) / 2$ as the guide for every estimation of $\Delta > \beta$. The adjustment rule as shown in equation (7):

$$CWND = CWND - (\Delta - ((\alpha + \beta) / 2)) \quad (7)$$

To reduce the bursty effect of increment, the *incr* is served as the increment amount of congestion window after each Acknowledgement Packet is received by a DCVegas source. If Δ between β and $(\alpha + \beta) / 2$, *CWND* will decrease 1, the *incr* and *succ* will be reset to zero. When Δ between α and $(\alpha + \beta) / 2$, the *incr* will be set to $(1 / CWND)$, and *succ* will be reset. When $\Delta = (\alpha + \beta) / 2$, that means network currently used in good, we don't need to adjust *CWND*. Due to data center networks have the characteristic of high-bandwidth and low-latency. The congestion index α and β of DCVegas is set to 1 and 2, respectively.

IV. Performance Evaluation

Figure 2 shows the simulation network topology. Link bandwidth is set to 1 Gbps or 10 Gbps. Each link delay is 25 μs. Depending on different simulation scenarios, the bottleneck buffer can be 256 K, 1 M or 4 M bytes. Packet size is fixed at 1000 bytes. The RTO_{min} of all TCP variants are set to 20 *ms*.

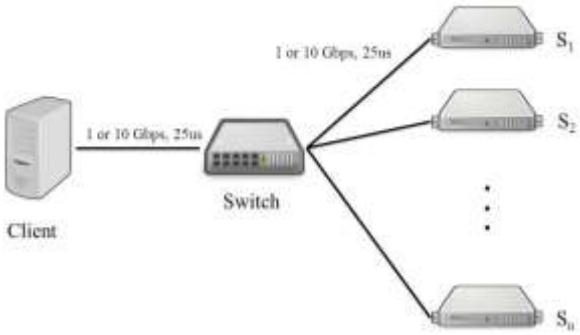


Figure 2. Data center networks topology

TABLE I. DATA CENTER TRAFFIC: APPLICATIONS AND PERFORMANCE REQUIREMENTS

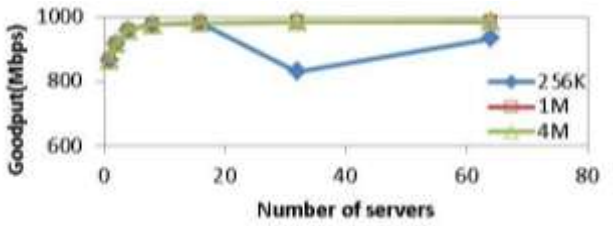
Traffic Type	Example	Requirements
Mice traffic (< 100 KB)	Google Search, Facebook	Short response times
Cat traffic (100 KB-5 MB)	Picasa, YouTube, Facebook photos	Low latency
Elephant traffic (> 5 MB)	Software updates, Video On-demand	High throughput

From the previous literature [4][5], traffic in data center networks is classified mainly into three types: (i) Mice traffic - the queries form the mice traffic. Majority of the traffic in a data center network is query traffic and its data transmission volume is usually less. (ii) Cat traffic – the control state and co-ordination messages form the cat traffic and (iii) Elephant traffic - the large updates form the elephant traffic. The different traffic types in data center networks, their applications and performance requirements are summarized in Table I.

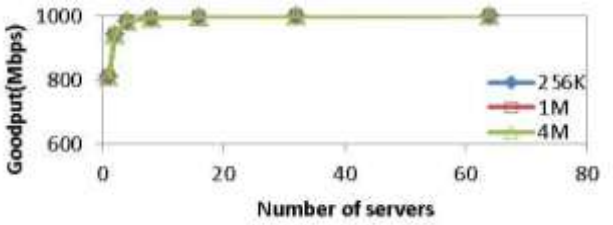
According to these three transmission modes, we present the simulation results of DCVegas and compare that with the results of IA-TCP and TCP NewReno by NS-2.

A. Cat Traffic

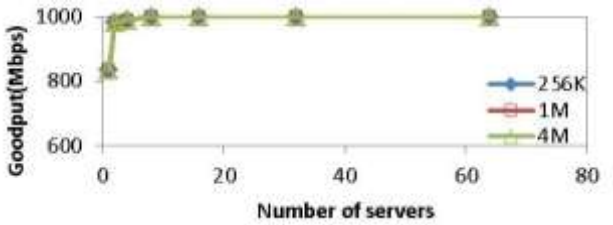
In data center network, the most common scenarios is set SRU to 256 KB [2]. Fig. 3 and Fig. 4 show the goodput of NewReno, IA-TCP and DCVegas.



(a) NewReno



(b) IA-TCP



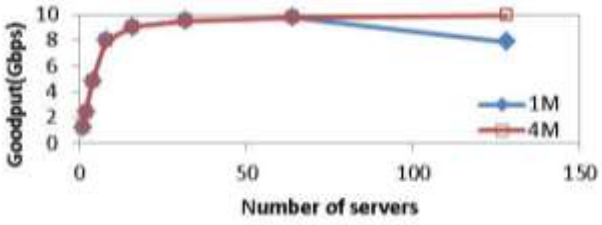
(c) DCVegas

Figure 3. The goodput of three TCP variants. The link bandwidth is 1 Gbps, bottleneck buffer size is 256 K/1 M/4 M bytes and SRU is 256 KB.

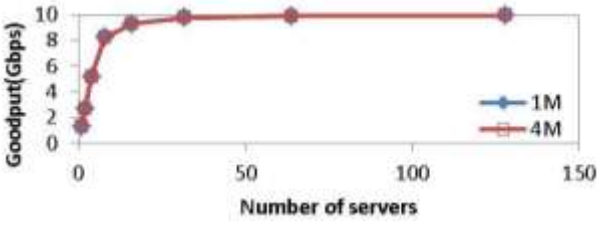
As shown in Fig. 3, the goodput of DCVegas and IA-TCP is better than that of NewReno. Because NewReno may introduce packet loss by itself, so it cannot fully utilize the network bandwidth. With the knowledge of the number of concurrent server, IA-TCP computes the product of bottleneck bandwidth and the minimum *RTT* of connections. It equally distributes the maximum window size to each server. The mechanism effectively avoids the packet loss. However, IA-TCP does not consider the buffer of intermediate node, which also is a considerable network resource. When the number of servers is 1, 2 and 4, the allocated *CWND* of IA-TCP would be too conservative. In such cases DCVegas creates 2.9%, 4.4% and 0.7%

performance improvement as compared with IA-TCP, respectively. The mechanism of DCVegas can quickly and appropriately adjust its window size, and thus utilize the bandwidth without packet loss. The performance of DCVegas is the best.

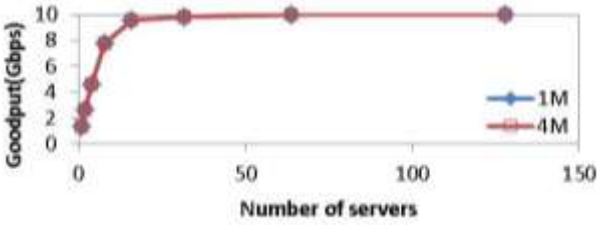
In Fig.4, when the number of servers is 128, buffer size is 1 M bytes, NewReno injects serious packet loss and results in performance collapse. In such situation, both DCVegas and IA-TCP create 22% performance improvement as compared with NewReno. IA-TCP allocates maximum window size for each server in advance, so it will not incur packet loss. Due to the sensitive congestion control and the bursty nature of TCP, DCVegas enters the congestion avoidance phase too early and postpones the increasing of window size. Thus, when the number of servers is 4 and 8, the goodput of IA-TCP is greater than DCVegas. However, when the number of server is 16, DCVegas outperforms IA-TCP about 2.7%. In general, both DCVegas and IA-TCP have a good performance in these simulation scenarios.



(a) NewReno



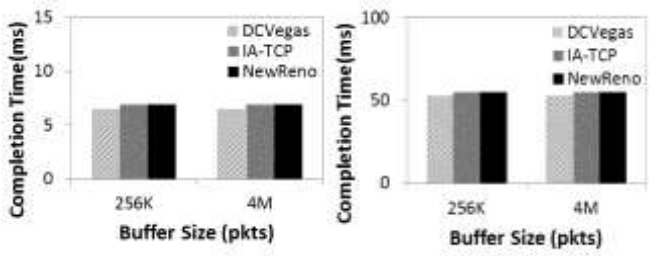
(b) IA-TCP



(c) DCVegas

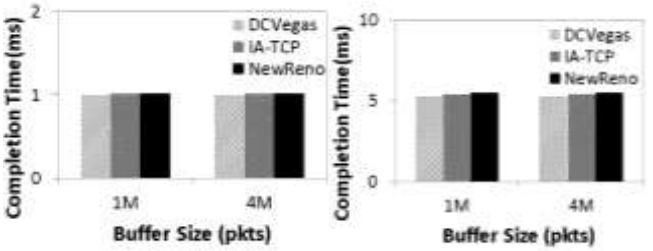
Figure 4. The goodput of three TCP variants. The link bandwidth is 10 Gbps, bottleneck buffer size is 1 M/4 M bytes and SRU is 256 KB.

B. *Mice Traffic*



(a) number of servers = 8 (b) number of servers = 64

Figure 5. The transmission completion time. The link bandwidth is 1 Gbps and SRU is 100 KB.

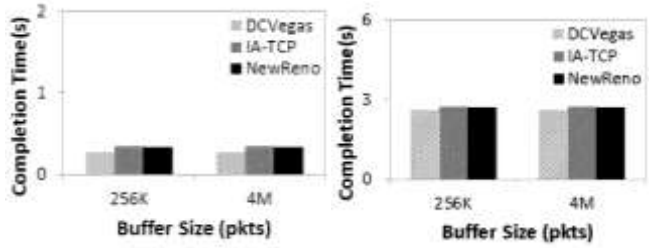


(a) number of servers = 8 (b) number of servers = 64

Figure 6. The transmission completion time. The link bandwidth is 10 Gbps and SRU is 100 KB.

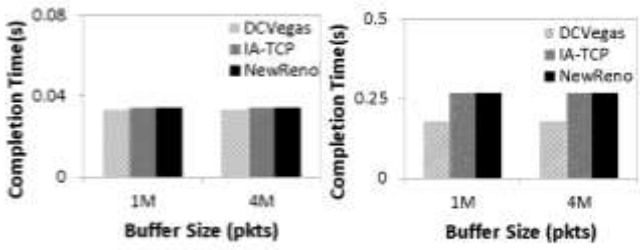
In mice traffic mode, the SRU of each server is set to 100 KB. We compares the transmission completion time of TCP NewReno, IA-TCP and DCVegas. When the SRU is small, all the packet transmission is likely complete in slow start phase. Since DCVegas has a more positive window adjustment scheme in the slow start phase, so the transmission completion time of DCVegas is shorter than that of NewReno and IA-TCP, as shown in Fig. 5 and Fig. 6.

C. *Elephant Traffic*



(a) number of servers = 8 (b) number of servers = 64

Figure 7. The transmission completion time. The link bandwidth is 1 Gbps and SRU is 5 MB.



(a) number of servers = 8 (b) number of servers = 64

Figure 8. The transmission completion time. The link bandwidth is 10 Gbps and SRU is 5 MB.

The SRU in elephant traffic is set to 5 MB. As shown in Fig. 7 and Fig. 8, no matter what number of server, the completion time of DCVegas is shorter than NewReno and IA-TCP. Especially, when the server number is 8, the transmission time of DCVegas is shorter than that of IA-TCP and NewReno about 59%. Because the amount of synchronous transmission data is very large, NewReno will has periodic packet loss, resulting in a long completion time. IA-TCP waits for a random time before starts transmission and does not make good use of switch buffer, caused the transmission completion time longer than DCVegas.

In short, the performance of NewReno relies the buffering ability of switch. When the buffer size is not large enough, the packets prone to be dropped and thus the performance suffered. Although IA-TCP can effectively avoid packet loss, but it needs to know the number of concurrent servers, bandwidth of the bottleneck link, and the minimum *RTT*. Those are not easy information for a transport layer protocol. Furthermore, an IA-TCP connection would wait for a random time before it starts transmission. The waiting intends to alleviate bursty effect when a lots of connections start to transmit at the same time and thus prevents the packet loss occurs. However, the waiting is unnecessary when the number of servers and SRU are small.

v. Conclusions

In this paper, we propose a new TCP variant based on the TCP Vegas - called Data Center Vegas (DCVegas). DCVegas features a positive window size increasing scheme in slow start phase and a sensitive congestion control mechanism in congestion avoidance phase. Therefore, DCVegas presents a good performance in a variety of conditions. Compared to IA-TCP, DCVegas doesn't need cross layer information such as the number of servers, the bandwidth of the bottleneck link. This reduces the complexity of deployment. Overall, the proposed DCVegas can effectively avoid the occurrence of TCP Incast and easy to be adopted in data center networks.

References

[1] D. Nagle, D. Serenyi, A. Matthews, "The Panasas activescale storage cluster: Delivering scalable high bandwidth storage," in Proceedings of the 2004 ACM/IEEE conference on Supercomputing, pp. 53, November 2004.

[2] J. Hwang, J. Yoo, N. Choi, "IA-TCP: A Rate Based Incast-Avoidance Algorithm for TCP in Data Center Networks," in Proceedings of the IEEE International Conference on Communications (ICC), pp. 1292-1296, June 2012.

[3] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," IEEE J. Select. Areas Communications, vol. 13, pp. 1465-1480, Oct. 1995.

[4] M. Alizadehz, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, "Data Center TCP (DCTCP)," SIGCOMM'10, pp. 63-74, August 30-September 3, 2010.

[5] R.P. Tahiliani, M.P. Tahiliani, and K. Chandrasekaran, "TCP variants for Data Center Networks: A comparative study" in proceedings of the Cloud and Services Computing (ISCOS), pp. 57-62, December 2012.

[6] R.P. Tahiliani, M.P. Tahiliani and K. C. Sekaran "TCP Congestion Control in Data Center Networks," Handbook on Data Centers, pp. 485-505, March 2015.

[7] U.U. Hafeez, A. Kashaf, Q. Bajwa, A. Mushtaq , "Mitigating Datacenter Incast Congestion Using RTO Randomization" IEEE Global Communications Conference (GLOBECOM), pp.1-6, December 2015.

[8] Y. Chan, C. Lin, and C. Ho, "Quick Vegas: Improving Performance of TCP Vegas for High Bandwidth-Delay Product Networks," IEICE Transactions on Communications, Vol. E91-B, No. 4, pp. 987-997, April 2008.

[9] Y. Chan, C. Lin, C. Chan, C. Ho, "CODE TCP: A competitive delay-based TCP," Computer Communications, vol. 33, no. 9, pp. 1013–1029, Jun. 2010.

[10] Y. Chen, R. Griffit, D. Zats, R. H. Katz "Understanding TCP Incast and Its Implications for Big Data Workloads," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-40, pp 1-10, April 2012.

[11] Y. Chen, R. Griffith, J. Liu, A. Joseph, R. H. Katz, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," Workshop on Research in Enterprise Networks (WREN'09), pp.73-82, August 2009.