Journal of Network and Computer Applications 53 (2015) 103–114

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

DC-Vegas: A delay-based TCP congestion control algorithm for datacenter applications

Jingyuan Wang [a,*], Jiangtao Wen [b], Chao Li [a,c], Zhang Xiong [a], Yuxing Han [d]

[a] School of Computer Science and Engineering, Beihang University, Beijing 100191 China
[b] Department of Computer Science, Tsinghua University, Beijing 100084 China
[c] Research Institute of Beihang University in Shenzhen, Shenzhen 518057 China
[d] TCPEngines Inc., Santa Clara, CA 95054, USA

ARTICLE INFO

Article history:
Received 31 August 2014
Received in revised form
31 December 2014
Accepted 24 March 2015
Available online 13 April 2015

Keywords:
TCP
Congestion control
Datacenter
Deployment cost

ABSTRACT

TCP congestion control in datacenter networks is very different from in traditional network environments. Datacenter applications require TCP to provide soft real-time latency and have the ability to avoid incast throughput collapses. To meet the special requirements of datacenter congestion control, numerous solutions have been proposed, such as DCTCP, D^2TCP, and D^3. However, several deployment drawbacks, including significant modifications to switch hardware, the Operating System protocol stack, and/or upper-layer applications, as well as switch ECN requirements, which are not always available in already existing datacenters, limit deployment of these solutions. To address these deployment problems, in this paper, we proposed a delay-based TCP algorithm for datacenter congestion control, namely DC-Vegas. DC-Vegas combines the performance advantages of DCTCP with the deployment advantages of delay-based TCP Vegas. DC-Vegas can meet both soft real-time and incast avoidance requirements of datacenters, requiring minimal deployment modification to existing datacenter hardware/software (with only sender-side update and without ECN requirements). Experimental results obtained using the real datacenter test bed and an ns-2 simulator demonstrate that DC-Vegas has similar performance with the state-of-the-art Data Center TCP algorithm.

1. Introduction

Traditional TCP congestion control algorithms designed for the Internet environments have encountered many problems in datacenter networks. One problem is introduced by Online Data Intensive (OLDI) applications (Meisner et al., 2011), such as web search and social network services, which operate under "soft" real-time constraints so that any need for transmission latency in datacenters is as minimal as possible (Alizadeh et al., 2012a; Lee et al., 2012; Zhang et al., 2014). The other problem is TCP throughput collapse caused by a special network scenario in datacenters, namely incast. TCP flows in incast scenarios may suffer serious throughput losses (Chen et al., 2009). Incast scenarios widely exist in datacenter applications, such as MapReduce (Wang et al., 2014b; He et al., 2014). To address the soft real-time and incast problems, many improved solutions have been proposed, including fine-grained retransmissions (Vasudevan et al., 2009), ECN* (Wu et al., 2012), and ICTCP (Wu et al., 2010) for mitigating the TCP incast problem, HULL (Alizadeh et al., 2012a), D^3 (Wilson et al., 2011), Detail (Zats et al., 2012), and PDQ (Hong et al., 2012) for reducing low transmission latency, as well as DCTCP (Alizadeh et al., 2010), D^2TCP (Vamanan et al., 2012), L^2DCT (Munir et al., 2015), RDT (Jingyuan et al., 2014) and WDCTCP (Wang et al., 2013a) for both latency reduction and incast avoidance. Although these solutions achieved excellent performance in laboratorial datacenter networks, there still exist some deployment barriers when adopting them in "*existing*" datacenters that were already built all over the world. For solutions such as D^3 and PDQ that are based on custom network protocols and not compatible with TCP, all TCP-based application layer software must be replaced, which is impractical in many commercial systems. Algorithms such as DCTCP that rely on ECN (Explicit Congestion Notification) marking for congestion detection, on the other hand, will require ECN support, a capability which is not universally available even today (Stewart et al., 2011; Bauer et al., 2011). In addition, many solutions require significant modifications to network hardware and/or the OS protocol stack; some even require custom ASICs. Therefore, these existing solutions are very promising for new datacenters, but for the datacenters that already exist, they are not suitable.

In this paper, we propose a high-performance and low-deployment-cost datacenter TCP algorithm named DC-Vegas. DC-Vegas was inspired by an analysis of queue length distribution of delay-based TCP in datacenter networks. The analysis revealed why the delay-based TCP

* Corresponding author.
E-mail address: jywang@buaa.edu.cn (J. Wang).

http://dx.doi.org/10.1016/j.jnca.2015.03.010
1084-8045/

Vegas (Brakmo et al., 1994) algorithm performs well for reducing transmission latency but poorly in incast scenarios. Based on this insight, we design the DC-Vegas algorithm, which combines the low-cost deployment advantage of TCP Vegas and the excellent performance of DCTCP in datacenters. We deployed DC-Vegas in a production datacenter test bed. The deployment only needed to update TCP senders and did not require ECN support, application software and/or network modifications. Experiments using the datacenter test bed show that DC-Vegas (1) reduces queuing delays by two orders of magnitude and reduces the deadlines missed rate of OLDI applications by 80%; (2) significantly avoids incast collapse; and (3) maintains graceful fairness and convergence. The ns-2 simulation experiments show that the performance of DC-Vegas is similar to that of DCTCP.

The rest of this paper is organized as follows. Section 2 analyzes why delay-based TCP Vegas performs well for reducing latency but poorly in incast scenarios. Section 3 describes the algorithm details of DC-Vegas and presents some analysis. Experimental results are presented in Sections 4 and 5. The conclusions of this paper are given in Section 7.

2. Delay-based TCP in datacenters

2.1. Performance of TCP Vegas in datacenters

Congestion detection mechanism is an important module of TCP algorithms (Alrshah et al., 2014; Xu et al., 2011). The most widely used congestion detection mechanism is loss-based detection, which is adopted by many popular TCP algorithms such as TCP Reno (Jacobson, 1988) and CUBIC (Ha et al., 2008; Wang et al., 2013b). Because packet losses occur only after packet queues have built up, loss-based TCP in datacenters always introduces very long queuing delay, and therefore causes degradation of user experience.

The most popular congestion detection method in datacenters is ECN (Explicit Congestion Notification), such as DCTCP, ECN*, and D^{2}TCP. However, ECN is not universally supported by datacenter hardware, especially in datacenters currently in operation. According to the report of Bauer et al. (2011), only 60% end-to-end loops in the Internet are ECN enabled, and as reported in Stewart et al. (2011), finding a datacenter switch that supports ECN to test performance of DCTCP is very difficult. Furthermore, ECN is an IP level protocol, which requires all agents in an end-to-end loop to be ECN enabled, so deploying ECN-based TCP in a non-ECN equipped datacenter means a system-wide software and/or hardware updating. Even though deploying DCTCP over an ECN enabled datacenter, modifications on both sender side and receiver side, as well as parameter setup of switches, are still unavoidable. So the time cost and expense of such deployment is unbearable for production datacenters. As reported by Emerson Network Power (State of the data center, 2011), more than 500 thousand datacenters were built before 2011 around the world. Most of these datacenters are waiting for TCP updating because most of ECN-based datacenter TCP solutions were proposed during 2011–2013.

An alternative for ECN is delay-based TCP congestion control (Brakmo et al., 1994; Wei et al., 2006), which uses end-to-end queuing delay as an indication of network congestion. Because delay-based TCPs aim at keeping packet queue length in network buffers under a reasonable level, queuing delay of delay-based TCP is usually shorter than loss-based algorithms (Wang et al., 2014a). Moreover, delay-based TCPs can proactively reduce their throughput before packet losses appear; therefore, they have potential to mitigate the TCP incast problem (Lee et al., 2012; Wu et al., 2010). However, existing delay-based TCP algorithms cannot meet *both* the low latency and incast avoidance requirements of datacenters. In this section, we use TCP Vegas as a representative of delay-based TCP algorithms to analyze its performance in datacenter networks.

We conducted two sets of experiments on a production datacenter of the Computer Network Information Center (CNIC)[1] in the Chinese Academy of Sciences. First, we investigated queuing delay of TCP Vegas in the datacenter. We let ten long-term TCP flows pass between two hosts that were connected by 1 Gbps links in the datacenter. According to Alizadeh et al. (2010), the typical number of concurrent long-term TCP connections for a host in datacenters is less than four; therefore, ten flows represented a fairly crowded setting. Figure 1(a) plots the RTT (Round-Trip Time) variations between the hosts when TCP Vegas and TCP Reno algorithms were used. As shown in the figure, the queuing delay of TCP Vegas was far smaller (by approximately two orders of magnitude) than that of the loss-based TCP Reno algorithm. Second, we investigated the performance of TCP Vegas in an incast scenario. We used 47 hosts connected by a 1 Gbps switch, where one host was designated the receiver, and the others were used as senders. A subset of the senders simultaneously and repeatedly sent 128 KB data blocks to the receiver. Figure 1(b) shows aggregated throughput of the senders. Regardless of whether TCP Vegas or Reno was used, the throughput of the senders collapsed when the number of senders exceeded a threshold; this is a typical TCP incast collapse phenomenon. The threshold for TCP Vegas was the small value of 20, slightly bigger than TCP Reno and not satisfactory in datacenter applications. The two experiments on TCP Vegas in datacenter networks demonstrated that the TCP Vegas algorithm is promising for latency-sensitive datacenter applications, but performs poorly in incast scenarios.

2.2. Analysis of TCP Vegas in datacenters

TCP Vegas uses end-to-end queue length samples to detect congestion. In each RTT, TCP Vegas estimates current queue length as

$$q = \frac{w}{RTT} \cdot (RTT - RTT_{min}), \tag{1}$$

where w is the size of the congestion control window, RTT is the sampled RTT, and RTT_{min} is the minimum observed RTT, which is used as an estimate of network propagation delay. TCP Vegas adjusts the window size in each RTT by comparing q with a threshold K_v. If $q > K_v$, TCP Vegas considers the network congested and reduces the window but instead increases the window if $q < K_v$. In other words, TCP Vegas uses a binary decision to detect network congestion.

The binary congestion detection of TCP Vegas works well for traffic on the Internet but not for datacenter networks because network queue behaviors on the Internet are very different from those in datacenter networks. Figure 2 shows network buffer queue fluctuation of datacenter networks and the Internet. Datacenter queue fluctuation is observed between two servers of the CNIC datacenter using 10 TCP Vegas flows, and the Internet queue fluctuation is observed from an Amazona EC2 server to a computer in our laboratory. For the sake of comparison, we normalize the queue length samples using $Q_s/((\sum_{i=1}^{i=S} Q_i)/S)$, i.e., the ratio between the current queue sample Q_s and the average of all such values of S samples. The value of S is 200,000 in our experiments, among which 1000 samples are plotted in Fig. 2. As shown in Fig. 2(a), there are many impulses in the Internet queue fluctuation. A probable cause of these impulses is network congestion; if so, TCP Vegas can easily distinguish them from non-congestion network states by using simple binary quantization. On the other hand, the queue length variation of the datacenter, which is shown in Fig. 2(b), is very

[1] http://english.cnic.cas.cn/