DBTCP: A Transmission Mechanism of the Backup for Disaster Recovery in DCNs

Chen-Yue Wang, Zhan Shi, Wei Su, Qi-Li Wen, Hua-Chun Zhou

School of Electronic and Information Engineering, Beijing Jiaotong University, China
1195326758@qq.com, novshz@126.com, wsu@bjtu.edu.cn, 295438550@qq.com, hchzhou@bjtu.edu.cn

Abstract

End-to-end communication has been widely used in all aspect of our daily lives, and are these applications are mostly rely on data center networks (DCNs), which are served as the core infrastructure for storing and processing a large number of data. Disaster backup, different from general data transmission, has some unique characteristics and requires low latency, high throughput, burst robustness, tight deadline, massive loads and easy deployment, where mass data in DCNs nodes need to be transferred as much as possible and the tight deadline should be meet after the alarm for a potential nature extreme phenomena is sound. This paper first introduces and summarizes the existing transmission mechanisms deployed for DCNs worldwide. Further, we explain the unique characteristics of disaster backup. Then our paper discusses the design idea and proposes the detailed design of the DCN backup TCP (DBTCP), through simulation and evaluation, the results of which displayed in the form of graphs. We are confirmed that the DBTCP designed is capable of making almost all of flows meet the tight deadline under high load and high burst. By carefully investigations implemented in both NS2 and Linux kernel, we notice that DBTCP nearly always outperforms previous presented mechanisms, like DCTCP, D^{2}TCP. In typical scenarios, DBTCP can reduce missing deadline rate and improve both throughout and robustness when flows burst, keeping the outstanding easy deployment feature of former mechanisms at the same time. The DBTCP proposed in this paper provides a reliable and effective solution, and has a certain reference value for future research.

Keywords: DCN, Backup for disaster recovery, Transmission mechanism, Deadline

1 Introduction

Nowadays, end-to-end communication generates more and more data in a really short time, thus cloud computing technology and data centers becoming the hot focus of both scientific and public attention. Real-time communication applications, as a representative manner of end-to-end communication which have been widely used in all aspect of our daily lives, are required to provide stringent data transmissions with a given deadline. These services and applications are mostly rely on data center networks (DCNs), which are served as the core infrastructure for storing and processing a large number of data [1-2].

However, due to the geographic distribution characteristic of DCNs devices, it is unavoidably vulnerable to natural disasters, such as typhoons, tsunamis, earthquakes, and floods. Therefore the performance of the mechanism considering missed deadlines and throughput can be significantly affected. To avoid large scale data loss and meet strict transmission deadline, network protocols need to incorporate new tailor traffic management and various transport schemes for DCNs, serving users in a timely manner. In other words, DCNs are required more effective and reliable mechanisms in backup for nature disaster, to ensure that the data in nodes of DCNs which may be destroyed, can be backed up as many as possible to other security nodes within a tight deadline, when a potential disaster is detected.

The mechanisms of the backup for disaster recovery in DCNs' involve network topology optimization [3-4], multipath transmission for load balancing [5-6], optimal destination node and transmission path selection, end-to-end transmission mechanism, and so on. There are indeed other transmission mechanisms which have been used widely in DCNs, such as DCTCP [7] and D^{2}TCP [8] which attain good features as general modes of transportation. But when a sudden disaster comes, they often fail to deliver as much data as protocol especially designed for disaster recovery within the hard response deadlines. For instance, DCTC does not take deadline into consideration, apparently causing large missed data. Besides, the performance of D^{2}TCP which is deadline-aware, will degrade to DCTCP when the network load is too heavy, while there must be a huge burst of data to transfer in the case of disaster backup. In addition, other transmission mechanisms in DCNs, such as PDQ [9] and so on, are difficult to deploy incrementally, as they require large hardware and software changes of hosts

*Corresponding Author: Chen-Yue Wang; E-mail: 1195326758@qq.com
DOI: 10.3966/160792642019072004007

and switches, and cannot coexist with legacy transmission mechanism.

Based on D^2TCP, this paper is aimed at proposing a deadline-aware flow rate control scheme on condition of shallow packet buffers in switches, the principle of which is as followed: flows with shorter deadline or smaller load should take up more bandwidth. By carefully investigations implemented in both NS2 and Linux kernel, we notice that DBTCP nearly always outperforms previous presented mechanisms, like DCTCP, D^2TCP. In typical scenarios, DBTCP can reduce missing deadline rate and improve both throughout and robustness when flows burst, keeping the outstanding easy deployment feature of former mechanisms at the same time.

To demonstrate the effective performance of DBTCP, this paper first introduces and summarizes the existing transmission mechanisms deployed for DCNs worldwide. Further, we explain the unique characteristics of disaster backup. Then our paper discusses the design idea and proposes the detailed design of the DCN backup TCP (DBTCP), through simulation and evaluation, the results of which displayed in the form of graphs. We are confirmed that the DBTCP designed is capable of making almost all of flows meet the tight deadline under high load and high burst.

The rest of this paper is organized as follows. Section 2 analyzes and summarizes the existing domestic and overseas transmission mechanisms of DCNs. Then in section 3, the paper discusses the characteristics and requirements in data transmission of the backup for disaster recovery in DCNs, and proposes DBTCP. Simulation and evaluation of DBTCP are given in section 4. Finally, section 5 is a summary of our paper.

2 Relevant Work

The ever-increasingly developed ideas designed for DCNs have become hot research propositions, while they still face severe network congestions and missing data when nature disaster occurs.

The two principal design ideas for the current transmission mechanisms in DCNs are respectively called Earliest Deadline First (EDF) and Shortest Job First (SJF). In EDF scheduling discipline, each flow is assigned a deadline, where the flow with shorter deadline is allocated bandwidth preferentially, so that more flows can be transmitted within their deadline. As for SJF scheduling discipline, the smaller flow has wider bandwidth, and thus the average flow completion time (AFCT) is shorten dramatically. Some of the existing transmission mechanisms of DCNs, implement only EDF, such as D^2TCP and D^3 [10], or simply SJF, such as L^2DCT [11]. Others conform to both of EDF and SJF, such as PDQ and LPD [12]. As for congestion control in the transmission rate, the related algorithms are divided into two categories. One monitors congestion in the link and adjusts the transmission window passively, the other allocates transmission rate proactively.

In this section, existing transmission mechanisms in DCNs are classified by their manner of regulating the transmission rate, on the basic conception of which we compare the advantages and disadvantages of these transmission mechanisms.

2.1 Transmission Mechanisms Adjusting the Transmission Rate Passively

Among many TCP-like transmission mechanisms for DCNs, DCTCP uses shallow buffers and marks packets once queue length exceeds a preset threshold, by which way the congestion window is adjusted accordingly and the rate of queue occupancy in switches is limited. DCTCP, deployed explicit congestion notification (ECN) [13] mechanism, achieves low latency, high throughput, and high burst tolerance and can easily coexist with TCP. But the weakness of DCTCP is that it does not consider the strict deadline in applications such as web search and chat networking.

D^2TCP proposes a gamma-correction function on the basis of DCTCP, which contributes to the deadline-awareness and higher transport completion rate. In this way, D^2TCP regulate the congestion window according to both deadline and congestion level, correcting small and temporary oversubscribing. However, the performance of D^2TCP degrades to DCTCP on condition of heavy network load.

L^2DCT also uses the ECN and can avoid overloading without knowing the size of lows. It adjusts the congestion window based on the estimated congestion level and flow size, where shorter flows should has shorter completion time thus reducing the mean transport time. In addition, as L^2DCT need not modify switches, it can be easy to deploy incrementally on the basis of TCP. However, L^2DCT also ignores deadline.

pFabric [14] is designed to complete transmission as simple as possible. Switches drop and send every packets selectively and accordingly on the reference of priority set by the sending ends. At the beginning, flows are all sent with the maximal link rate, only under high and persistent packet loss the allocated bandwidth throttled back, therefore flows with higher priority have a shorter completion time. Besides, information of flows is useless for switches, whose buffers should be small. Unfortunately, pFabric is vulnerable to congestion.

HULL [15] also achieves ultra-low latency at the cost of little bandwidth loss. Based on DCTCP, it uses phantom queues as well as packet pacing in order to tolerate burst. But HULL can only be employed under the condition of great change to hosts and switches.

2.2 Transmission Mechanisms Adjusting the Transmission Rate Proactively

D^3 similarly achieves low latency, high bandwidth utilization, and burst tolerance. It allocate bandwidth for next RTT at the routers based on real-time remaining data quantity and transmission time. Nevertheless, due to its greedy approach that allocate bandwidth to far-deadline requests arriving ahead of near-deadline requests, D^3 faces priority inversions and missed deadlines. Moreover, D^3 has the same problem as HULL.

ICTCP is also deadline-agnostic, while promises high throughput, low latency and easy deployment. In this mechanism, the receiving ends are responsible for receive window adjustment according to incast congestion and the aggregate burst control of the sender operating in unison.

In contrast to L^2DCT, PDQ uses the information of flows to decide if the flows should be paused and how much the bandwidth can be allocated. Regardless of the difficulty in deployment, PDQ has fast completion time, high bandwidth utilization and multipath version. In addition, it can also maintain performance under the inaccurate information of the flows, and prevent deadlock.

3 Design of DBTCP

Given network topology, terminal type and transport link, this section explains why the disaster backup mechanisms need special design and how DBTCP ensure more stability and efficiency when extreme nature phenomena appears.

3.1 Characteristics and Requirements of the Transmission for Backup

When nature disaster comes, the duration between detection of possible phenomena and expected data transmission time can be really short, which means, before disasters cause regional damage to DCNs, the completion interval for data backup is extremely tight and strict and transmission loads of these nodes is often huge. Meanwhile, in order to transfer more data within shorter time, there can be massive burst once emergency mechanism is initiated from daily circumstance after the alarm of a potential disaster is detected. In summary, the major characteristic of disaster backup is the tight deadline to send mass data in a burst manner.

When we design the mechanism for disaster recovery, low latency, high throughput, burst robustness, tight deadline, massive loads and easy deployment are all taken into consideration. DBTCP, which studies the present transmission protocols such as TCP, DCTCP and D^2TCP, has the advantages of previous mechanisms and overcomes their weakness, showing superior properties under the typical scenarios as well as the extreme circumstance. The features of DCNs should also be considered, such as the equal ability to back up data at each nodes. In any cases, the mechanism must avoid significant changes of every part like hosts and switches in order to implement incremental deployment.

3.2 Design Principle of DBTCP

DBTCP considers the deadline as well as the size of flows equally, which results in the priority of each flow to satisfy the EDF and the SJF scheduling disciplines in the same time. Within these two factors, when a disaster occurs, the transport completion plays apparently a more major role, so that we give deadline greater weight as it has greater impact on the priority of flows, or even play a decisive role. Note that the flows having the tighter deadline are given the higher priority, so that as many flows as possible can meet their deadline. Flows with smaller size have the higher priority when the different flows possess the same deadline, to reduce the mean flow completion time.

In DBTCP, congestion control mechanism of hosts is based on the same ECN as DCTCP and D^2TCP. The sending ends can not only know when congestion has occurred, but also be aware of the degree of congestion, according to the received ACK packets. Furthermore, the same ECN can contribute to coexisting with legacy transmission mechanisms used widely, reducing the modifications of the current hosts and switches, and incremental deployment of DBTCP.

The senders adjust the congestion window of each flow independently, based on the priority of flows and the current degree of congestion. For a given flow, the severer the congestion is, the more the congestion window is reduced. While at the same degree of congestion, the higher priority the flow possesses, the less the congestion window decreases; and in the absence of congestion, the higher priority the flow has, the more the congestion window increases. So that DBTCP is able to alleviate congestion effectively, and minimize the loss of throughput caused by the reduction of congestion window at the same time. In addition, bandwidth can be preempted by the flows with the higher priority.

Due to the likely burst caused by the backup for disaster recovery in DCNs, the sending ends should use the packet pacing to smooth the transmission rate and reduce the burst proactively. The senders adopt the hardware pacer to support the sub-microsecond scheduling granularity and operate the packets after the segmentation by LSO. Rather than used at the beginning of the transmission, the pacing only plays a role in the flows with the relatively low priority when congestion occurs. Besides, when congestion is mitigated, the sending ends no longer use the pacing for the flows with the high priority comparatively which have already adopted the pacing before. Therefore, DBTCP can not only reduce the throughput

loss caused by the pacing, but also preempt bandwidth for flows with the high priority.

Switches decide how to send and drop packets, according to the priority of the flows to which these packets belong. If a switch receives a packet when its queue is full, it drops the packet arriving last in the flow with the lowest priority. And when the switch wants to send a packet, the packet is sent which arrive first in the flow possessing the highest priority. Because the switch uses the queue to store packets, the packet arriving early is in front of the queue, while the packet arriving late is at the end of the queue. As a result, there is no overhead of the extra space to preserve the arrival orders of packets. In this way, the flows having the higher priority can complete as quickly as possible, and the head-of-line blocking caused by the out-of-order packets can be alleviated. Hence, DBTCP is capable of reducing the average flow completion time.

In addition, because switches have to frequently traverse and compare the information of the packets saved, such as the identification and the priority of the flows to which the packets belong, they had better store these information separately in an extra queue. When a switch compares the priority of the flows to which the packets belong, it simply need to traverse the queue which only preserves the information of these packets instead of whole packets. As a result, DBTCP can lessen the time for traversal and comparison, to improve the performance of switches of processing packets, thereby reducing latency.

The other mechanisms of DBTCP are the same as TCP [16], to ensure that DBTCP is able to coexist with legacy transmission mechanisms used widely at present such as DCTCP and D^{2}TCP, and minimize the changes of hosts and switches. So it can deploy incrementally in DCNs.

Note that DBTCP also adjusts the transmission rate passively. But it overcomes shortcomings of the other transmission mechanisms which modify transmission rate with similar manner. The comparisons between DBTCP and the others similar is shown in Table 1.

Table 1. Comparisons between DBTCP and others similar

Scheme	Degree of congestion	Deadline	Size	Deploy -ment
TCP				√
DCTCP	√			√
D^{2}TCP	√	√		√
L^{2}DCT	√		√	√
pFabric		√	√	
HULL	√		√	
DBTCP	√	√	√	√

3.3 Design Details of DBTCP

We start with the definition of p as the priority imminence factor, knowing that a higher p means a greater priority. Especially, when it comes to the value of p is 0, it implies the corresponding flow does not has a certain deadline, then the window size gets characteristics similar to DCTCP. We maintain p as follows:

$$p = e^{\frac{t_{max}}{t}} \times \frac{s_{max}}{s},\ p \geq 0 \quad \textbf{(1)}$$

Here t is the deadline set for flows with t_{max} is the upper limit of t; the size of the flow is s, and s_{max} is the upper limit of s. When the node sends a packet, the information of the p of the flow is carried in the header of this packet. Then the packet is transferred to switches where two queues is maintained at the same time, one of which stores the whole packet accepted while the other simply keeping the information of each packet like the identification and p.

In the case a packet arrives and the queue of the switch saving the whole packet is not full currently, the packet will be kept. Accordingly, its information in the corresponding queue is stored. However, if the packet arrives when the queue to save packets has already been filled at that time, whether the packet is stored is decided by the comparison of p. If every p of the packets accepted is larger than or equal to the p carried by the packet just received, this packet received should be dropped. While if there is a packet accepted whose p is less than the p of the packet has received, switch will keep the packet which is arrived and destroy the packet whose p is larger with latest arrival time as well as its information. In other words, the packet received can be accepted, its information stored in the corresponding queues.

Furthermore, we can put it this way that when a switch has the opportunity to send a packet, it will traverse the information queue and compare these numerical value until the switch gets the flow with the highest p and then sends the packet first in this flow.

Before sending a packet, we expect the switch is ECN-capable which implies when the ECN field of the packet header is set by the sending node, the switch can configured to mark CE bits when the packet buffer occupancy exceeds a certain threshold. In detail, after a packet is stored in the queue leaving occupancy ratio of the packet queue greater than a fixed threshold called K, the switch should mark the header of this packet with the CE code point. The procession and transmission of the packet with ACK by the receiving ends is the same as DCTCP and D^{2}TCP.

Sending ends use α, a weighted average, to present the estimation of the degree of congestion. As the fraction of the packets marked in the last window of data, F in the formula is estimated according to the received ACK packets. And g is the weight given to the new samples whose range is 0 to 1. α is updated once for every window of data, roughly one RTT, as follows:

$$\alpha = (1-g)\times\alpha + g\times F \quad \textbf{(2)}$$

Note that α has the range of 0 to 1, and the greater α becomes, the severer congestion appears. Based on α and p, the penalty function f is defined as follows:

$$f = \alpha^p,\ f \in [0,1] \quad \textbf{(3)}$$

When the packet starts transmission, the sending nodes employs TCP slow start mechanism [17]. Only if $f > 0$, the sending ends adjusts the congestion window in order to avoid the congestive collapse and severe wave of the window size. And the key effect on f of the corresponding flow is the ACK packet that the sender may receive. We resize the congestion window *cwnd* as follows:

$$cwnd = cwnd\times(1-\frac{f}{2}) \quad \textbf{(4)}$$

If f is measured zero, it is believed that no congestion appears in this flow. Then, if the flow has *cwnd* less than or equal to the slow start threshold, we call the flow in the slow start phase whose congestion window is adjusted according to the principle referred as slow start in TCP mechanism; and if the *cwnd* is greater than the slow start threshold now, the flow is in the stage of congestion avoidance, so the sender should adjust the *cwnd* as follows, where *MSS* is its maximum segment size:

$$cwnd = cwnd + p\times MSS \quad \textbf{(5)}$$

Note that if a flow have no deadline, when congestion occurs its *cwnd* is reduced by half like TCP, and remains still or even decreases if corresponding flow is in the congestion avoidance phase.

We must notice that pacing is not used for all flows at the start. If a flow does not adopt pacing, the sending node can sends a packet immediately, while the packet has to be placed in a token bucket rate limiter, which means it has to be sent with a fixed rate, as the flow employs pacing. Whenever receiving an ACK packet whose header's ECE flag is set to 1, there will be a new flow without pacing adjusted to adopt the principle, whose priority is lowest currently. While if no ACK packet whose ECE flag equals 1 is received after the interval T from last ACK packet with the ECE flag set to 1, the flow with the highest priority should give up pacing

Other mechanisms such as retransmission timeout, slow start, fast retransmission and fast recovery, are the same as TCP.

Steps of processes are shown in Figure 1, and Figure 2.

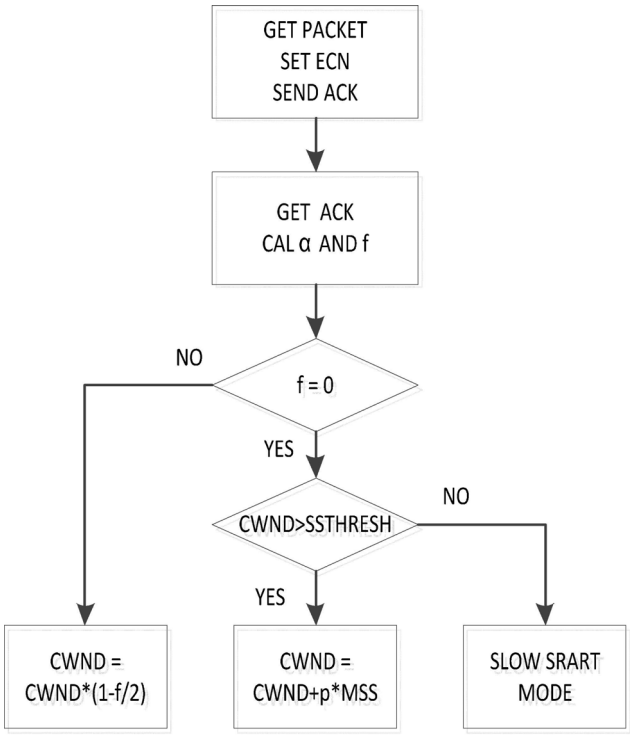

Figure 1. The step of processes for an end in DBTCP

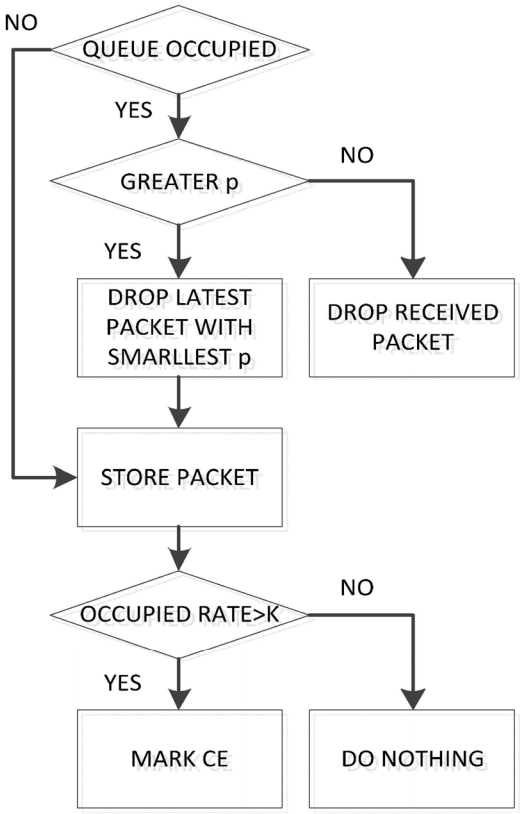

Figure 2. The step of processes for a switch in DBTCP

4 Simulation and Evaluation of DBTCP

In this section, we simulate and evaluate the proposed DBTCP via NS-2, and compare DBTCP with other transmission mechanisms used currently in DCNs.

4.1 Design Details of DBTCP

Above all, we verify that DBTCP can adjust the congestion window according to the priority of a flow, to preempt bandwidth for the flow with higher priority. Meanwhile we observe the impact of the flow's deadline and size on its priority.

We simulate 30 concurrent flows, which are divided into 3 groups of 10 flows each. The deadline and size of the flows of each group are shown in Table 2.

Table 2. The deadline and size of the flows of each group.

Group	Deadline (s)	Size (MB)
1	0.9	120
2	0.9	150
3	0.6	150

We can tell from it that the deadline of group 1 is 0.9s and the size of the flow is 120MB; the deadline of group 2 is 0.9s and the size of the flow is 150MB; the deadline of group 3 is 0.6s and the size of the flow is 150MB.In each of the 3 groups, we randomly select a flow's change of the congestion window in the simulation to show in Figure 3. Note that *f1* belongs to the first group, and *f2* belongs to the second group, and *f3* belongs to the third group.

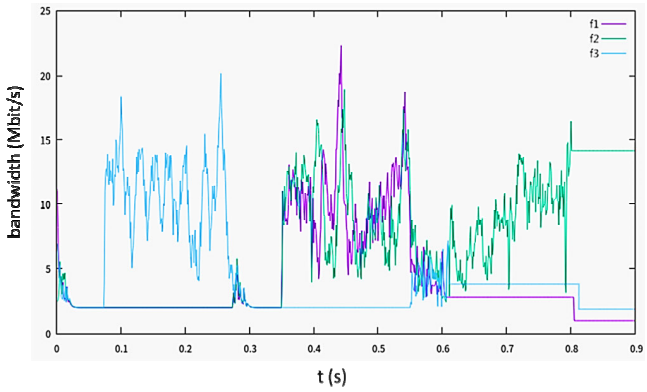

Figure 3. The congestion window of DBTCP.

As can be seen from Figure 3, although f3's size is equal to or greater than f1's size and f2's size, it is assigned bandwidth preferentially, ensuring that f3 is first completed within its deadline, whose deadline is less than f1's deadline and f2's deadline. Besides, f1 and f2 have the same deadline, but f1's size is less than f2's size, so f1 has a slightly bigger congestion window. Therefore, we can know that DBTCP is able to determine the priority of each flow according to its deadline and size. In addition, compared with flow's size, the deadline of a flow has a greater impact on its priority.

Then, in order to intuitively compare the performance of DBTCP and the current mainstream transmission mechanisms in DCNs, we compare the throughput of DBTCP, D^{2}TCP and DCTCP.

We simulate 40 concurrent flows, which are divided into 5 groups and each group has 8 flows. The transmission mechanism, deadline and size of these flows of each group are indicated in Table 3.

Table 3. The scheme, deadline and size of the flows of each group

Group	Scheme	Deadline (s)	Size (MB)
1	D^{2}TCP	0.9	120
2	D^{2}TCP	0.9	150
3	DBTCP	0.9	120
4	DBTCP	0.9	150
5	DCTCP		120

The first group obeys D^{2}TCP whose deadline is 0.9s and flow is 120MB; the second group has a bigger size. While the third group is DBTCP with a 0.9s deadline and 120MB flow; the fourth group also uses DBTCP which has a larger flow of 150MB. The fifth group applies DCTCP. We select randomly one flow in each group, plotting their throughput in Figure 4, and each flow's identifier is the same as the identifier of the corresponding group.

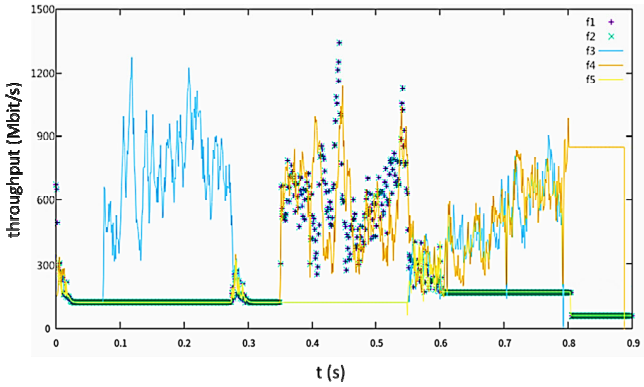

Figure 4. The congestion throughput of DBTCP

As shown in Figure 4, *f1* and *f2* has basically same throughput. Although *f1* and *f2* have high throughput between 0.35s and 0.55s, their throughput are low at other times. Besides, both *f1* and *f2* miss their deadline. In contrast, *f3* and *f4* maintain a high throughput for most of the time, and both of them complete within their deadline. Because *f3*'size is less than *f4*'size, *f3* has higher throughput on the whole compared with *f4*. Obviously, the throughput of *f5* is always low.

Thus we can see that D^{2}TCP can achieve high performance, however because there is no difference in treatment between the different streams when adjusting the window, its performance cannot be stabilized under the high load; in comparison, DBTCP can keep good performance to avoid missing flows' deadline as far as possible due to the consideration of both deadline and flow size; and DCTCP has poor performance in the simulation, without considering the deadline or size of flows.

4.2 Deadline Missing Rate

We construct a three-level tree topology, which is shown in Figure 5, in order to evaluating and comparing the deadline missing rate of DBTCP and other transmission mechanisms used in DCNs. In this way, we observe whether DBTCP can make more flows for backup complete within their deadline.

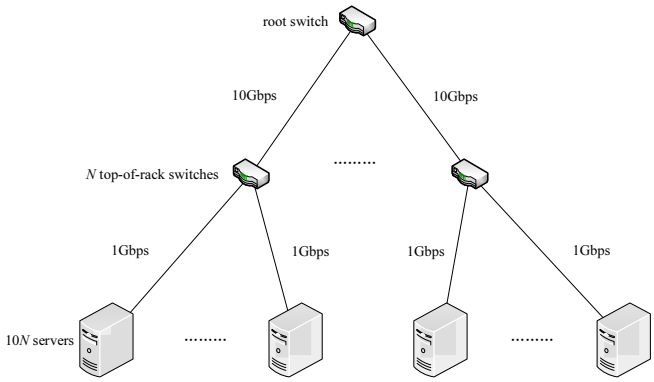

Figure 5. The topology for simulation

There are N racks in the topology. Then through the 1 Gbps links, each of them is connected to a top-of-rack switch. In addition, there is a total of $10N$ servers, because each of rack includes 10 servers. And these top-of-rack switches are connected to a root switch, by the 10 Gbps links.

We assume that all of these $10N$ servers are the nodes which are about to be damaged unfortunately by a disaster detected. As a result, through the converging via top-of-rack switches, these backup data have to be ultimately transmitted from servers to the root switch. During this process, high burst and throughput will be generated by a large number of aggregated data. Because the number of severs is far more than the number of top-of-rack switches, and there are many top-of-rack switches connected to only one root switch. Through this method, we can simulate the scene of the back-up for disaster recovery in DCNs.

Then, we set switches' threshold $K = 30$, and servers' fan-in degree is 20. The RTO and RTT of each flow is 10ms and 100 us respectively. Furthermore, we set $t = 500$ms with $t_{max} = 1$s, and $s = 500$KB on average, with $s_{max} = 1$MB.

When the number of severs backed up is 20, 30, 40, 50 and 60 respectively, which means that N correspondingly takes 2, 3, 4, 5 or 6, we calculate the percentage of the flows that miss the deadline.

We compare DBTCP with the other transmission mechanisms in DCNs, such as D^{2}TCP, DCTCP and TCP, under the same conditions. Besides, we alter s to 800 KB or modify t to 200ms, but other parameters are constant. The simulation and comparison results are shown in Figure 6.

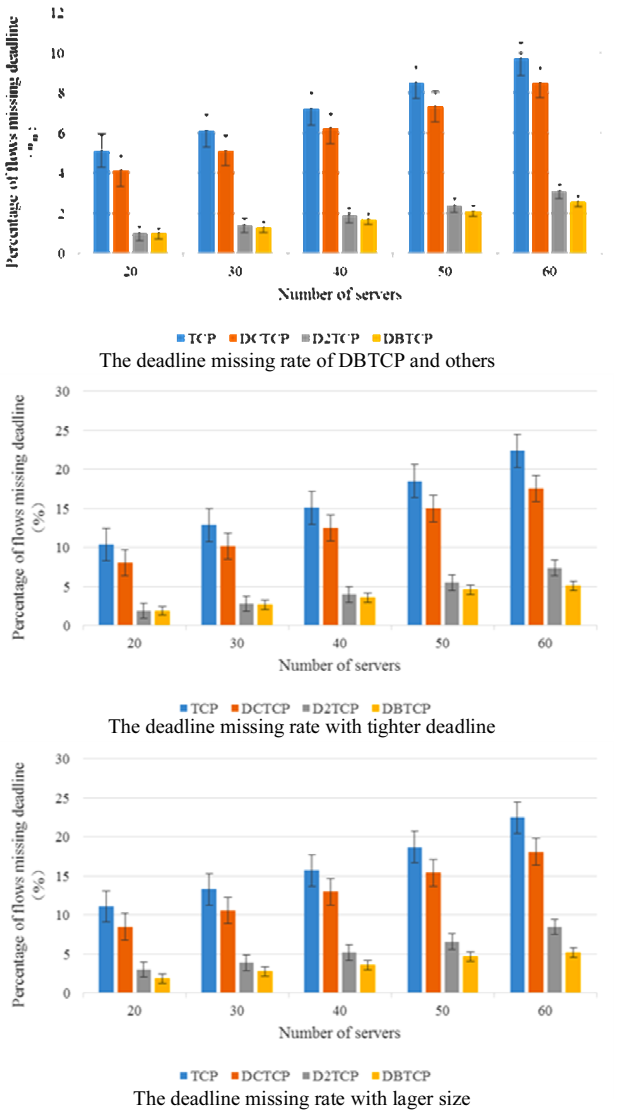

Figure 6. The deadline missing rate of DBTCP and others

According to these test results, compared with DCTCP and TCP, both DBTCP and D^{2}TCP are able to make more flows complete within their deadline. However, D^{2}TCP's performance may degrade under quite high load, while DBTCP is better capable of adapting to this scene, to maintain great performance.

The reason of these test results is that, DBTCP considers deadline as well as the size of flows, and deadline plays a major role. While DCTCP ignores the deadline, and D^{2}TCP does not take the flow into consideration, which makes performance fail under heavy load.

5 Conclusion

Disaster backup in DCNs promises great performance using DBTCP by monitoring deadlines and packet size at the same time with low latency, high throughput, burst robustness, tight deadline, massive loads and easy deployment, where the mass treasure data in DCNs nodes can be transferred at the most

degree and meet the tight deadline after the alarm for a potential nature extreme phenomena is sound.

The two principal design ideas for the current transmission mechanisms in DCNs are respectively called Earliest Deadline First (EDF) and Shortest Job First (SJF). In the way of adjusting the transmission rate, existing transmission mechanisms in DCNs are classified by their manner of regulating the transmission rate, on the basic conception of which we compare the advantages and disadvantages of these transmission mechanisms.

In summary, the major characteristic of disaster backup is the tight deadline to send mass data in a burst manner. The features of DCNs should also be considered, such as the equal ability to back up data at each nodes. In any cases, the mechanism must avoid significant changes of every part like hosts and switches in order to implement incremental deployment. Beyond that, it should be able to coexist with the legacy transmission mechanisms in DCNs, and require only less changes to hosts and switches, so that it can be easy to deploy incrementally.

Taken all that features into consideration, we propose DBTCP where the priority of a flow is determined according to its deadline and size, with the deadline a greater impact on the priority. ECN is used as DCTCP to indicate the degree of congestion, as an auxiliary means of adjusting congestion window besides its priority. Besides, the switch decides whether to drop and send packets accordingly. The other mechanisms are the same as TCP.

Through the simulation and evaluation, it is confirmed that DBTCP outperforms previous presented mechanisms, like DCTCP, D^2TCP. In typical scenarios, DBTCP can reduce missing deadline rate and improve both throughout and robustness when flows burst, keeping the outstanding easy deployment feature of former mechanisms at the same time.

Furthermore, considering that the features and requirements of OLDI applications in DCNs are similar to those of the backup for disaster recovery, we will continue to study in depth whether DBTCP can be applied to OLDI applications in the future. And as DBTCP is easy implemented, we should also consider the safety when TCP and DBTCP cooperate in the current network avoiding DBTCP leading to the collapse of the existing network by taking absolute advantage of bandwidth.

References

[1] D. Li, G. -H. Chen, F. -Y. Ren, C. -L. Jiang, M. -W. Xu, Data Center Network Research Progress and Trends, Chinese *Journal of Computers*, Vol. 25, No. 7, pp. 87-89, July, 2014.

[2] W. -F. Xia, P. Zhao, Y. -G. Wen, H. -Y. Xie, A Survey on Data Center Networking (DCN): Infrastructure and Operations, *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 1, pp. 640-656, January, 2017.

[3] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, S. Sengupta, VL2: A Scalable and Flexible Data Center Network, *ACM SIGCOMM Conference*, Barcelona, Spain, 2009, pp. 95-104.

[4] M. Al-Fares, A. Loukissas, A. Vahdat. A Scalable, Commodity Data Center Network Architecture, *ACM SIGCOMM Conference*, Seattle, WA, USA, 2008, pp. 63-74.

[5] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, M. Handley, Improving Datacenter Performance and Robustness with Multipath TCP, *ACM SIGCOMM Conference*, Toronto, Ontario, Canada, 2011, pp. 266-277.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, Hedera: Dynamic Flow Scheduling for Data Center Networks, *USENIX Symposium on Networked System Design and Implementation (NSDI)*, San Jose, CA, 2010, p. 19.

[7] M. Alizadehzy, A. Greenbergy, D. A. Maltzy, J. Padhyey, P. Pately, B. Prabhakarz, S. Senguptay, M. Sridharan, Data Center TCP (DCTCP), *ACM SIGCOMM Conference*, New Delhi, India, 2010, pp. 63-74.

[8] B. Vamanan, J. Hasan, T. N. Vijaykumar, Deadline-Aware Data-center TCP (D2TCP), *ACM SIGCOMM Conference*, Helsinki, Finland, 2012, pp. 115-126.

[9] C.-Y. Hong, M. Caesar, P. B. Godfrey, Finishing Flows Quickly with Preemptive Scheduling, *ACM SIGCOMM Conference*, Helsinki, Finland, 2012, pp. 127-138.

[10] C. Wilson, H. Ballani, T. Karagiannis, A. Rowstron, Better Never than Late: Meeting Deadlines in Datacenter Networks, *ACM SIGCOMM Conference*, Toronto, Canada, 2011, pp. 50-61.

[11] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, B. Khan, *Flow Completion Times in Data Centers, International Conference on Computer Communications*, Turin, Italy, 2013, pp. 2157-2165.

[12] H. Zhang, X.-G. Shi, X. Yin, F.-Y. Ren, Z.-Y. Wang, More Load, More Differentiation- A Design Principle for Deadline-Aware Congestion Control, *International Conference on Computer Communications*, Hong Kong, China, 2015, pp. 127-135.

[13] K. K. Ramakrishnan, S. Floyd, D. L. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, RFC 3168, 2001.

[14] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, S. Shenker, pFabric: Minimal Near-Optimal Datacenter Transport, *ACM SIGCOMM Conference*, Hong Kong, China, 2013, pp. 435-446.

[15] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, M. Yasuda, Less is More: Trading a Little Bandwidth for Ultra-Low Latency in the Data Center, *USENIX Symposium on Networked System Design and Implementation (NSDI)*, San Jose, CA, 2012, pp. 253-266.

[16] Defense Advanced Research Projects Agency Information Processing Techniques Office, *Transmission Control Protocol*, RFC 793, September, 1981.

[17] M. Allman, V. Paxson, E. Blanton, *TCP Congestion Control*, RFC 5681, September, 2009.

Biographies

Chen-Yue Wang, born in yantai, shandong province in 1993, is now a postgraduate student at Beijing jiaotong university. She is mainly engaged in the research of the theory and technology of the new generation of info rmation network.

Zhan Shi, born in Beijing in 1992, is graduated from Beijing Jiaotong University. His main research direction is the new generation of information network.

Wei Su, born in 1978 in hebei province, Ph.D., is professor at Beijing Jiaotong University, whose main research direction is he new generation of key theory and technology of information network.

Qi-Li Wen, born in qinghai province in 1995 and earned her bachelor of Communication Engineering degree in 2017. Now she is a postgraduate student at Beijing Jiaotong University. Her main research direction is the new generation of information network.

Hua-Chun Zhou, Ph.D., is professor at Beijing Jiaotong University and vice President of electronic information engineering college, whose main research direction is the mobile Internet, network and information security, etc.