TLB: Traffic-aware Load Balancing with Adaptive Granularity in Data Center Networks

Jinbin Hu
Central South University
Changsha, China
jinbinhu@csu.edu.cn

Jiawei Huang
Central South University
Changsha, China
jiaweihuang@csu.edu.cn

Wenjun Lv
Central South University
Changsha, China
wenjunlv@csu.edu.cn

Weihe Li
Central South University
Changsha, China
weiheli@csu.edu.cn

Jianxin Wang
Central South University
Changsha, China
jxwang@csu.edu.cn

Tian He
University of Minnesota
Minneapolis, MN, USA
tianhe@umn.edu

ABSTRACT

Modern datacenter topologies typically are multi-rooted trees consisting of multiple paths between any given pair of hosts. Recent load balancing designs focus on making full use of available parallel paths to provide high bisection bandwidth. However, they are agnostic to the mixed traffic generated by diverse applications in data centers and respectively use the same granularity in rerouting flows regardless of the flow type. Therefore, the short flows suffer the long-tailed queueing delay and reordering problems, while the throughputs of long flows are also degraded dramatically due to low link utilization and packet reordering under the non-adaptive granularity. To solve these problems, we design a traffic-aware load balancing (TLB) scheme to adopt different rerouting granularities for two kinds of flows. Specifically, TLB adaptively adjusts the switching granularity of long flows according to the load strength of short ones. Under the heavy load of short flows, the long flows use large switching granularity to help short ones obtain more opportunities in choosing short queues to complete quickly. When the load strength of short flows is low, the long flows switch paths more flexibly with small switching granularity to achieve high throughput. TLB is deployed at the switch, without any modifications on the end-hosts. The experimental results of NS2 simulations and Mininet implementation show that TLB significantly reduces the average flow completion time (AFCT) of short flows by ~15%-40% over the state-of-the-art load balancing schemes and achieves the high throughput for long flows.

CCS CONCEPTS

• **Networks → Network architectures**; **Routing protocols**; **Data center networks**; *Data path algorithms*;

KEYWORDS

Data center, TCP, load balancing, multipath

ACM Reference Format:
Jinbin Hu, Jiawei Huang, Wenjun Lv, Weihe Li, Jianxin Wang, and Tian He. 2019. TLB: Traffic-aware Load Balancing with Adaptive Granularity in Data Center Networks. In *48th International Conference on Parallel Processing (ICPP 2019), August 5–8, 2019, Kyoto, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3337821.3337866

1 INTRODUCTION

With the increasing traffic demands of latency-sensitive and throughput-oriented applications, modern data centers deploy the multi-rooted tree networks such as Fat-tree and Clos to provide high bisection bandwidth via multiple paths between any given pair of hosts [1], [2], [3], [4]. To achieve better application performance, how to efficiently balance traffic across available multiple paths becomes a crucially important issue in large-scale data center networks (DCNs).

ECMP (Equal Cost Multipath) [5] has been chosen as the $de facto$ load balancing scheme to randomly map each flow to one of the paths by using flow hashing. ECMP is very simple but suffers from well-known performance problems such as hash collisions and the inability to reroute flow adaptively. Recently, a lot of better load balancing designs have emerged in DCN. Random Packet Spraying (RPS) [6], DRILL [3] and Hermes [2] split flows at the granularity of packet and choose the next hop for each packet to leverage multiple paths. Presto [7] picks paths for fixed-sized chunks of data (i.e., 64KB) to achieve high throughput and reduce packet reordering. CONGA [1] and LetFlow [8] adopt flowlet switching to provide a finer granularity without causing much packet reordering.

Unfortunately, each of existing load balancing designs respectively uses the same granularity for different flows regardless of the flow type. They are agnostic to the traffic feature that the mixed short and long flows are generated by large-scale applications such as web search, social networking and retailing system in data centers [9], [10], [11]. Many works [11], [12], [13] show that the datacenter traffic is heavy-tailed distribution, that is, about 90% of delay-sensitive short flows with only about 10% of data are coexisting with around 10% of throughput-sensitive long flows providing around 90% of data.

When the short and long flows are rerouted at the same granularity, the short flows easily experience long-tailed queueing delay since they are difficult to seize the less-congested paths under the overwhelming data of long flows, resulting in large flow completion time (FCT). Moreover, under the non-adaptive granularity, the long flows suffer the throughput loss due to the low link utilization or out-of-order problem when the network traffic changes dynamically [14], [15].

Therefore, splitting heterogeneous traffic across multiple paths to achieve low latency for short flows and high throughput for long flows is important. In this paper, we present a new load balancing scheme TLB, which differentiates the switching granularity for different types of flows. When the traffic load of short flows is high, the long flows are rerouted at the large granularity to leave more less-congested paths for short flows. In contrast, under the low load of short flows, the long flows are switched at the small granularity to improve the link utilization. The short flows pick paths on packet-level to flexibly seize the fast paths. TLB successfully avoids the long-tailed queueing delay for short flows, improves the link utilization for long flows and reduces the packet reordering for both of them.

In summary, our major contributions are:

- We conduct an extensive simulation-based study to explore the issues of mixed heterogeneous flows rerouted at the same and non-adaptive granularity in typical load balancing designs: the short ones experience long-tailed queueing delay and out-of-order problem, and the long ones suffer the low link utilization and packet reordering.
- We propose a load balancing scheme TLB, which flexibly reroutes short and long flows at different granularities by perceiving the traffic load. Specifically, TLB adaptively adjusts the switching granularity for long flows according to the load strength of short flows and picks paths for short flows at packet-level to achieve low queueing delay for short flows and high link utilization for long ones.
- By using both NS2 simulations and Mininet implementation, we demonstrate that TLB performs remarkably better than the state-of-the-art load balancing schemes. Especially, TLB greatly reduces the AFCT by ∼15%-40% for short flows under heavy workload. Meanwhile, TLB yields up to ∼22% and ∼35% throughput improvement for long flows over Presto and LetFlow, respectively.

The rest of the paper is organized as follows. In Section 2 and 3, we describe our design motivation and overview, respectively. In Section 4, we give the model analysis of TLB. We discuss the implementation in Section 5. In Section 6 and 7, we show the test results of NS2 simulation and Mininet experiment, respectively. We present the related works in Section 8 and conclude the paper in Section 9.

2 DESIGN MOTIVATION

To motivate our design, we illustrate the typical load balancing schemes and investigate the impact of the switching granularity on short and long flows.

2.1 Load Balancing with Same Granularity

The existing load balancing designs typically use the following three switching granularities. In the flow-level schemes, each flow is transferred on one path without flexible switching. The flowlet-level schemes reroute flows only when the flowlets emerge. The packet-level schemes choose path for each packet to make full use of multiple paths, but easily cause serious packet reordering under the topology asymmetry.

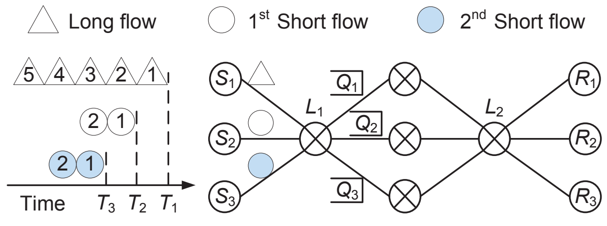


Figure 1: Leaf-spine topology

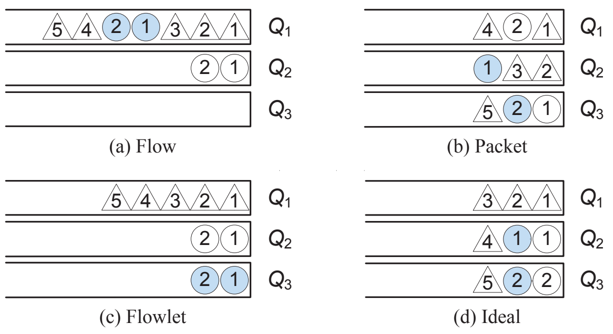
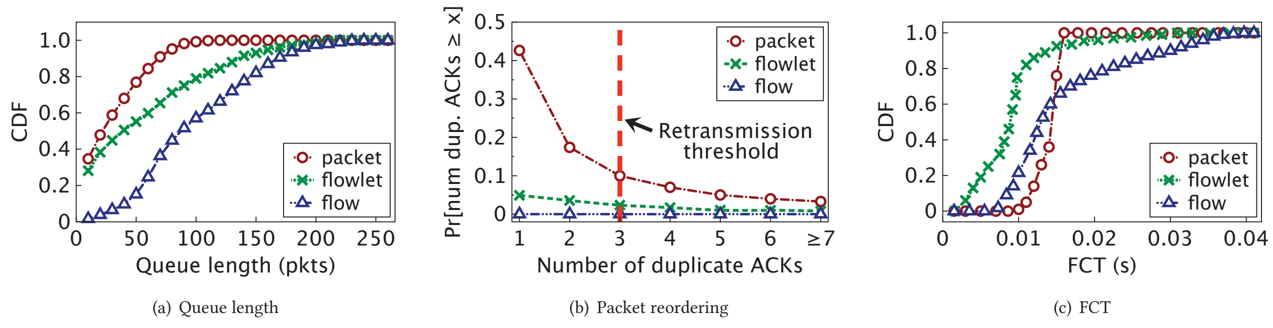


Figure 2: Queueing under different granularities

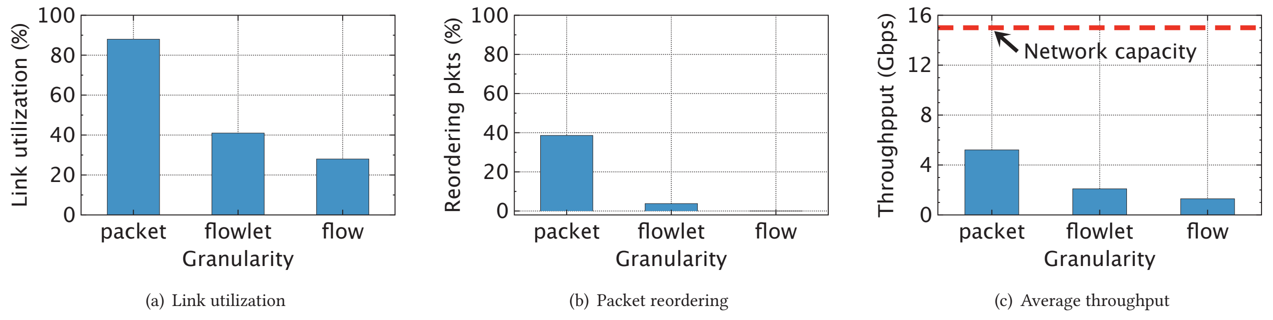
We use a simple example to illustrate the issues of typical load balancing schemes. Fig. 1 shows that 3 senders (S_1, S_2, S_3) and 3 receivers (R_1, R_2, R_3) connect to the corresponding leaf switches. There are 3 queues (Q_1, Q_2, Q_3) on the corresponding output ports of the leaf switch L_1. S_1 continuously sends a long flow to R_1 at time T_1, while S_2 and S_3 respectively send a short flow to R_2 and R_3 at time T_2 and T_3. We use four different granularities to reroute flows at the leaf switch L_1.

Fig. 2 shows that the packets of short and long flows are queued based on the flow, packet, flowlet and ideal switching granularity, respectively. Fig. 2 (a) shows that, under the flow-level switching, one short flow is queued behind the long flow in Q_1 though Q_3 is empty, resulting in the large queueing delay and low link utilization. The packet-based switching is shown in Fig. 2 (b). Although all packets of short and long flows are evenly spread among multiple paths, there exist reordering packets and large queueing delay experienced by short flows. Fig. 2 (c) shows the case of flowlet-based switching. Three flows are always transmitted in their respective queues due to insufficient inactivity gap. The inflexible switching unavoidably leads to the low link utilization. The ideal case is shown in Fig. 2 (d). At the beginning, the switching granularity of the long flow and the short ones are 3 and 1, respectively. The long flow is rerouted at its 4^{th} packet to leave the less-congested paths for the short flows, which flexibly pick the short queues of Q_2 and Q_3 per packet to avoid the large queueing delay. When the short flows are finished, the long flow decreases the switching granularity to packet to improve its throughput.



(a) Queue length (b) Packet reordering (c) FCT

Figure 3: The impact on short flows



(a) Link utilization (b) Packet reordering (c) Average throughput

Figure 4: The impact on long flows

2.2 Impact of Switching Granularity

We conduct NS2 simulation test to analyze the impact of switching granularity on the short and long flows. We use a leaf-spine topology with 15 equal-cost paths between host pairs. The bottleneck bandwidth is 1Gbps and the round-trip propagation delay is 100μs. The switch buffer size is 256 packets. Each sender sends a DCTCP [11] flow to a receiver via the leaf and spine switches. In our test, the mixture of 100 short flows with random size of less than 100KB and 5 long flows larger than 10MB are generated in heavy-tailed distribution. The flowlet timeout is 150μs [2].

We test the impact of switching granularity on short flows. We compare the cumulative density function (CDF) of queueing length experienced by each packet of short flows under three granularities. As shown in Fig. 3 (a), with the increasing of switching granularity, the queue length becomes larger. Moreover, due to the growing queue length caused by long flows, the difference of queue lengths under flow-level switching is large. Fig. 3 (b) shows the ratio of TCP duplicate ACKs to demonstrate the packet reordering. Compared with the flow-level and flowlet-level switching, the number of duplicate ACKs is quite large under the packet-level switching, causing the spurious packet loss. When the number of duplicate ACKs reaches the retransmission threshold, the TCP sender cuts the congestion window. Fig. 3 (c) shows the CDF of flow completion time. We observe that the tailed delay increases with the increasing of switching granularity due to the larger difference of queue lengths. Moreover, though obtaining the smallest queue length, the packet-level switching scheme still does not achieve the best performance of FCT because of its packet reordering problem.

Next, we examine the impact on long flows. Fig. 4 (a) shows that, since the long flows provide the overwhelming amount of data in data centers, the load balancing with large granularity (i.e., flow-level) causes low link utilization. As shown in Fig. 4 (b), though splitting the traffic of long flows in a more balanced way, the flow switching with smaller granularity introduces more out-of-order packets. Fig. 4 (c) shows that, due to the dilemma between the link utilization and packet reordering, the long flows only obtain the average throughput of less than 35% of the network capacity.

2.3 Summary

Our analysis of the impact of load balancing with same switching granularity on the network performance leads us to conclude that (i) the short flows experience the large queueing delay as the granularity increases and have more reordering packets as the granularity decreases, (ii) the long flows suffer the throughput degradation due to the non-adaptive granularity in rerouting flows. These conclusions motivate us to design and implement a traffic-aware load balancing scheme with adaptive granularity to achieve good performance for both short and long flows.

3 DESIGN OVERVIEW

In this section, we present an overview of TLB. The key point of TLB is to adaptively adjust the switching granularity of long flows and flexibly pick path for each packet of short flows to alleviate the large queueing delay and packet reordering. Specifically, on the one hand, the long flows are switched dynamically with adaptive granularity based on the load strength of short flows, leaving more less-congested paths for short flows. On the other hand, each packet of the short flows is routed on the shortest queue to avoid being blocked by long flows.

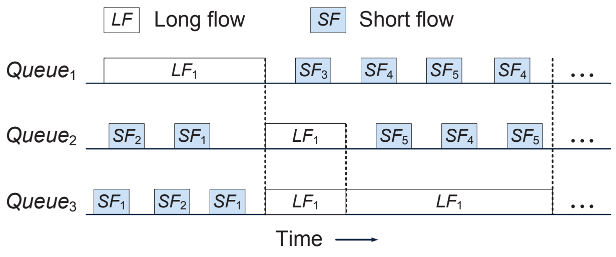


Figure 5: Queueing process

Fig. 5 shows the queueing process with adaptive switching granularity for short and long flows. The long flow is dynamically rerouted across multiple paths to achieve high throughput and simultaneously reduce the queueing delay for short flows. At the beginning, the long flow occupies the $Queue_1$ with large switching granularity, leaving non-congested $Queue_2$ and $Queue_3$ for the short flows with the packet-level switching granularity. As soon as the arrival rate of short flows decreases, the long flow is rerouted to $Queue_2$ and $Queue_3$ with smaller granularity to flexibly make use of multiple paths. In contrast, under the high load of short flows, the switching granularity of the long flow adaptively increases after being rerouted to $Queue_3$. Therefore, the short flows are able to efficiently and quickly transmit each packet on the paths unused by the long flow. The architecture of TLB consists of two modules, which are shown in Fig. 6.

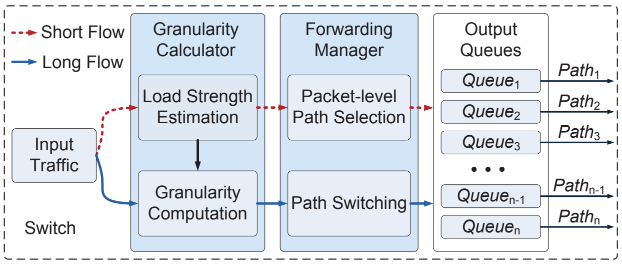


Figure 6: TLB architecture

1) **Granularity Calculator:** At the switch, the granularity calculator consists of two parts: load strength estimation and granularity computation. Firstly, the load strength of short flows is estimated according to their arrival rates, and then the switching granularity of long flows is calculated to guarantee the low queueing delay for short flows.

Specifically, when the traffic load of the short flows is heavy, the switching granularity of the long flows should be larger to ensure that the short flows have more opportunities to use enough non-congested queues to avoid large queueing delay. Conversely, with the decrease of the load strength of short flows, the switching granularity of long flows should be smaller to switch flexibly among the multiple paths to improve link utilization. At each time interval t (500μs by default [1]), the granularity computation module updates the switching granularity periodically. The most important work is adjusting the switching granularity of long flows to achieve the good tradeoff between the small flow completion time of short flows and high throughput of long flows.

2) **Forwarding Manager:**The forwarding manager is responsible for switching the short and long flows with different granularities and selecting the forwarding path according to the real-time queue lengths of the output ports.

For a long flow, when a new packet arrives at the switch, TLB reroutes the packet to the shortest queue only if the current queue length of the long flow reaches the switching threshold calculated by the granularity computation module. Otherwise, TLB forwards the new arriving packet to the same queue as the last arrival packet in the same flow. For short flows, TLB reroutes each arriving packet to the output port with shortest queue length to reduce queueing delay caused by the long flows.

4 ADAPTIVE SWITCHING GRANULARITY

In this section, we firstly use the queueing model to theoretically analyze how to calculate the optimal switching granularity to achieve low delay and good throughput for the short and long flows, respectively. Then we evaluate the accuracy of model analysis by comparing the numerical and simulation results.

4.1 Model Analysis

The goal of this paper is to develop a simple load balancing scheme that meets the requirements of both short and long flows. The primary goal of short flows is reducing their average flow completion time to meet their deadlines [16], [17], while the long flows require large throughputs. In our design, the switching granularity is elaborately adjusted to firstly meet the delay requirements (i.e., deadline) of short flows and then leave as much resources as possible for long flows to improve their throughputs. To achieve the design goal, we use the queueing model to calculate the optimal switching granularity as follows.

Let n denote the number of all equal-cost paths, which includes n_S and n_L paths allocated by LTB for m_S short flows and m_L long flows, respectively. We use C and RTT to denote the bottleneck link capacity and the round-trip propagation delay, respectively. The long flows continuously send packets with the maximum window size W_L limited by the receiver buffer (64KB by default in Linux) [18] after quickly entering the congestion avoidance phase. We use q_{th} to denote the switching threshold of queue length for rerouting the long flows. The value of q_{th} is updated periodically for each time interval t. Then according to the amount of data sent by long flows within the time interval t, we have the following equivalent relationship as

$$q_{th} \cdot n_L + t \cdot C \cdot n_L = m_L \cdot W_L \cdot \frac{t}{RTT}, \tag{1}$$

Given the total number of paths n=n_L+n_S, from the Equation (1), we obtain the number of paths n_S allocated for short flows as

$$n_S = n - \frac{m_L \cdot W_L \cdot \frac{t}{RTT}}{q_{th} + t \cdot C}. \tag{2}$$

In data centers, about 90% of TCP flows are less than 100KB [12], [18], [19], [20]. These short flows mostly finish in the slow-start phase [18], [21], in which if each flow firstly sends out 2 packets, then 4, 8, 16, etc. Thus, the number of RTT rounds r to finish transfer of a short flow with size X bytes is calculated as

$$r = \lfloor \log_2 \frac{X}{MSS} \rfloor + 1, \tag{3}$$

where $\lfloor x \rfloor$ is the largest integer smaller than x and MSS is the size of a TCP segment.

We use the M/G/1-FCFS (first come first serve) queueing model [18], [20], [22] to analyze the average waiting time in each queue. The expected queueing delay for each round of short flows is $E[W]$, which equals to the average waiting time in each queue because the short flows are transferred per packet across the multiple paths. The mean flow completion time of short flows FCT_s mainly includes the queueing and transmission delay, that is

$$FCT_S = E[W] \cdot r + \frac{X}{C}, \tag{4}$$

For a M/G/1-FCFS queue, by using the famous Pollaczek-Khintchine formula [23], the expected queueing delay $E[W]$ is calculated as

$$E[W] = \frac{1 + C_v^2}{2} \cdot \frac{\rho}{1 - \rho} \cdot E[S], \tag{5}$$

where ρ is the load strength, $E[S]$ is the service time of a single packet with its value as $\frac{1}{C}$, and C_v^2 is the coefficient of variation of the service time distribution. Since the service rate of each output port at switches is a constant C, the value of C_v^2 is 0. Then, Equation (5) can be simplified as

$$E[W] = \frac{\rho}{2(1 - \rho)} \cdot \frac{1}{C}. \tag{6}$$

With the average packet arrival rate λ and the service rate C, we get the load strength as $\rho = \frac{\lambda}{C}$.

Since m_S short flows, in which the average size of each flow is X bytes, are transmitted over n_S paths with an average completion time of FCT_s, the arrival rate λ of the short flows is calculated as

$$\lambda = \frac{m_S \cdot X}{FCT_S \cdot n_S}. \tag{7}$$

Thus, according to the former Equation (2), (4), (6) and (7), we have the equation about the mean FCT of short flows FCT_S as

$$FCT_S = \frac{m_S \cdot X \cdot \frac{r}{C}}{2[FCT_S \cdot (n - \frac{m_L \cdot W_L \cdot \frac{t}{RTT}}{q_{th} + t \cdot C}) \cdot C - m_S \cdot X]} + \frac{X}{C}. \tag{8}$$

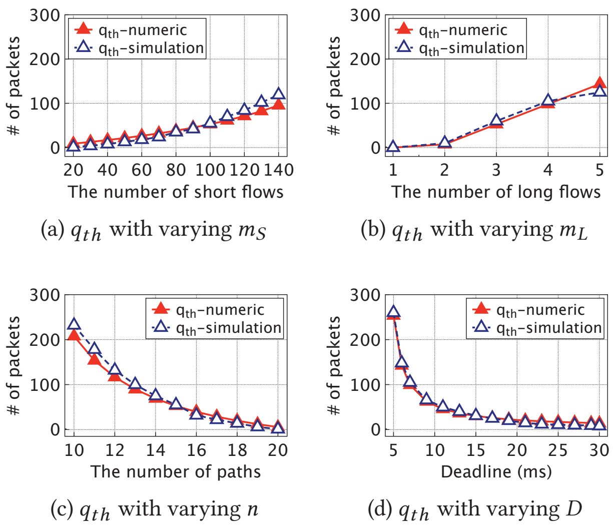
Given that the deadline of each short flow is randomly distributed between $[D_{min}, D_{max}]$, we set the associated default deadline D to the value of 25th percentile in CDF of the statistical flow deadlines to calculate the switching granularity for long flows. The reason why the deadline is set to the 25th percentile and the corresponding experiments are introduced in Section 6.3. To guarantee that the short flows complete within their associated deadline D, the mean flow completion time for short flows FCT_s should satisfy $FCT_S \leq D$. Finally, the switching threshold q_{th} of queue length for rerouting the long flows is adjusted according to the load strength of short flows to meet their deadlines. Specifically, q_{th} should satisfy the following expression

$$q_{th} \geq \frac{m_L \cdot W_L \cdot \frac{t}{RTT}}{n - \frac{r \cdot \frac{X}{C} + 2(D - \frac{X}{C}) \cdot X}{2(D - \frac{X}{C}) \cdot D \cdot C} \cdot m_S} - t \cdot C. \tag{9}$$

To leave as much resources as possible for long flows to improve their throughputs, the long flows are rerouted at the minimum value of q_{th} satisfying Equation (9) in our design. Specifically, when a new packet of a long flow arrives at the switch, TLB makes forwarding decision based on the real-time queue length of the current output port of the flow. Once the queue length reaches the minimum value of q_{th} (i.e., the optimal switching granularity), TLB selects the shortest queue to reroute the arriving packet. Otherwise, TLB forwards the arriving packet to the same path as the last arrival packet in the same flow without switching.

4.2 Model Verification

We evaluate the correctness of the theoretical analysis by NS2 simulations. We use the leaf-spine topology with 15 equal-cost paths between any pair of source and destination hosts. The bottleneck link bandwidth and switch buffer size are 1Gbps and 512 packets, respectively. We use DCTCP [11] as the underlying transport protocol. By default, 3 long flows (>10MB) and 100 short flows (<100KB) are generated with heavy-tailed distribution. The average size of short flows is 70KB in this simulation. The deadline of each short flow is randomly distributed between [5ms, 25ms] as illustrated in [16]. We set the default deadline D as the value of 25th percentile deadline (i.e., 10ms), which is also used in the following simulations. In this test, we measure the minimum value of switching threshold q_{th} for rerouting the long flows under the condition that no short flows miss their deadlines in the time interval t of 500μs, which is the general inactivity gap between two bursts of packets in short flows [1].



(a) q_{th} with varying m_S (b) q_{th} with varying m_L

(c) q_{th} with varying n (d) q_{th} with varying D

Figure 7: Numeric and simulation comparison

Fig. 7 shows the comparison of numerical and simulation results. The switching threshold q_{th} of queue length for rerouting the long flows increases as the number of short flows increases as shown in Fig. 7 (a). With the increasing number of long flows, q_{th} also increases as shown in Fig. 7 (b), because the long flows need larger switching granularity under the increasing workload to meet the delay requirements of short flows. As shown in Fig. 7 (c) and (d), q_{th} decreases with the increasing of total number of path and deadline, respectively. In brief, the switching granularities of long flows in numerical analysis closely follow the corresponding simulation results.

5 IMPLEMENTATION

We implement TLB with less than 300 lines of code changes at the switch and make no modifications at the end-hosts. The flow switching of TLB should meet the deadlines of short flows and meanwhile improve the throughputs of long flows. There are three key points in our implementation.

The first key consideration of TLB is how to work well in the case of lacking deadline information of short flows. Although most Online Data Intensive (OLDI) datacenter applications such as web search, social networking, advertisement and retail have strict deadline constraints (i.e., 300ms latency) [16], [20], [24], some datacenter applications such as HTTP chunked-transfer and database access generate short flows without deadlines [20]. Moreover, various applications may have different deadline requirements of short flows. For these applications, TLB needs to make short flows complete quickly even without the deadline information in advance. We use an alternative method which deduces the specified flow deadline from the statistics of network traffic. Specifically, we set the deadline to the value of 25^{th} percentile in CDF of the statistical flow deadlines. The corresponding experiment results in Section 6.3 show that TLB works well in dark.

Since the switches have no priori knowledge of the flow size, TLB needs to identify short and long flows when making load balancing decisions. Without flow size information, TLB distinguishes the short and long flows according to the number of bytes sent of each flow. At the beginning, all flows are treated as short flows. Once the number of received bytes belonging to one flow exceeds the threshold (i.e., 100KB) [13], [18], [21], [25], the flow is considered as a long one. Even if the long flows are mistaken for short ones at the beginning, the negative impact is very small due to few number of long flows and the small threshold. The experimental results in Section 6 show that TLB achieves good performance under these situations.

The last key point is how to accurately measure the current number of active short and long flows at the switches. TLB uses the TCP handshake messages, SYN and FIN, to count the number of flows, which is increased or decreased by one when switch receives a SYN or FIN message, respectively. However, some flows may be closed without FIN message or become idle without any packets to transfer. To avoid unnecessary waste of bandwidth due to the lost FIN and the idle connection, TLB samples the flows periodically at the switch. If no packet is received during the sampling interval, the corresponding flow record is removed from the flow table. The sampling interval is set to 500μs as same as the updated interval of switching granularity of long flows [1].

6 SIMULATION EVALUATION

In this section, we firstly examine the basic performance of TLB. Then we conduct the large-scale simulations to evaluate TLB with ECMP, RPS, Presto and LetFlow in two typical datacenter applications. Finally, we evaluate the performance of TLB in the deadline-agnostic case.

6.1 Basic Performances Test

We conduct NS2 simulations to evaluate the effectiveness of TLB. The simulation settings are as same as that in Section 4.2. There are 15 parallel paths with 3 long flows (>10MB) and 100 short flows (<100KB) generated in heavy-tailed distribution. The deadlines of short flows are randomly distributed between [5ms, 25ms]. The switching granularity of long flows is updated every 500μs [1].

1) **Performance of Short Flows:** In our design of TLB, the short flows are forwarded per packet to the shortest queue in order to avoid being blocked on the congested paths with long flows. Under this situation, since the short flows are well balanced, the out-of-order event happens infrequently due to the similar queueing delay between the shortest queues.

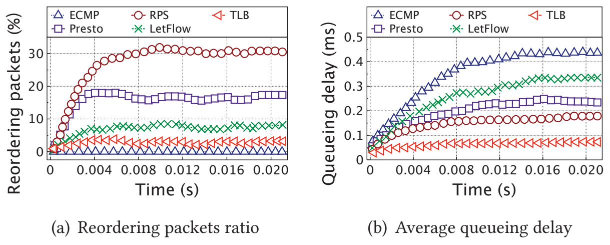


Figure 8: Performance of short flows

We measure the instantaneous reordering and queueing delay of short flows as shown in Fig. 8. The real-time reordering ratios of short flows in different schemes are compared in Fig. 8 (a). TLB reduces the number of out-of-order packets significantly compared with RPS and Presto, because the short and long flows are not mixed simultaneously on the same path. Fig. 8 (b) compares the average queueing delay of short flows. Since rerouting the short flows per packet to the shortest queue without queueing buildup, TLB achieves the lowest queueing delay all the time.

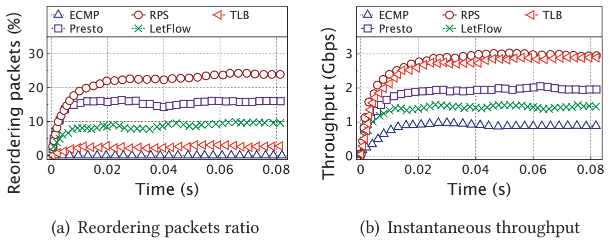
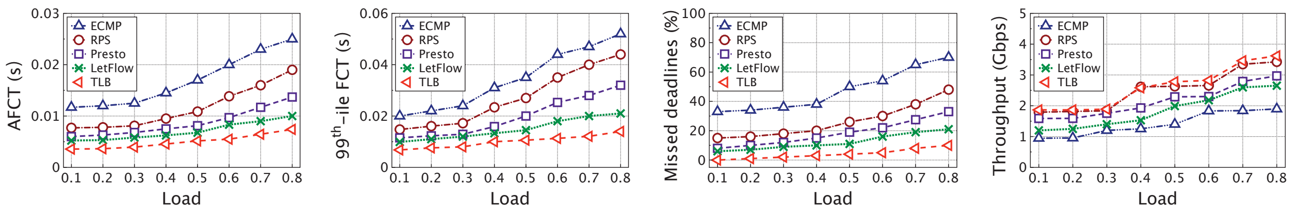
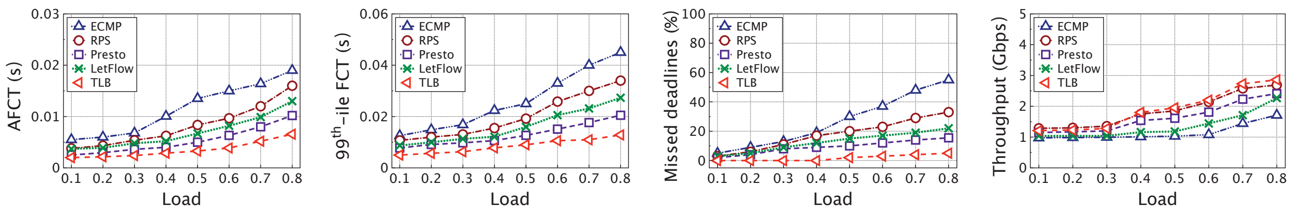


Figure 9: Performance of long flows



(a) AFCT of short flows (b) 99^{th} percentile FCT of short flows (c) Missed deadlines of short flows (d) Throughput of long flows

Figure 10: Web search application



(a) AFCT of short flows (b) 99^{th} percentile FCT of short flows (c) Missed deadlines of short flows (d) Throughput of long flows

Figure 11: Data mining application

2) **Performance of Long Flows:** For long flows, instead of being switched at every moment, they are adaptively rerouted among the parallel paths. The gain of long flows comes from the good tradeoff between the out-of-order and low link utilization. From Fig. 9 (a), we observe that the reordering packets of long flows are reduced by TLB, because the long flows are not affected by the short ones in the current queues. As shown in Fig. 9 (b), the instantaneous throughput of TLB is larger than ECMP, Presto and LetFlow because the rerouting granularity of long flows swiftly changes with the varying strength load of short flows. The long flows can make full use of multiple paths in most of the time since the short flows only provide less than 10% of the data. This means TLB achieves good performances of the short and long flows simultaneously by preventing them from being mixed together.

6.2 Large-scale Test

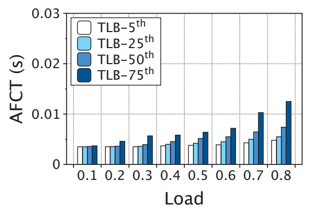
We conduct the large-scale simulation tests to evaluate TLB's performance in the typical applications in data centers. We choose the web search and data mining applications [2], [18], [20], [21]. In these workloads, the distributions of flow size are heavy-tailed, that is, more than 90% bytes are provided by around 10% flows. For the web search workload, about 30% flows are larger than 1MB, while in data mining scenario there are around less than 5% flows larger than 35MB [18], [21].

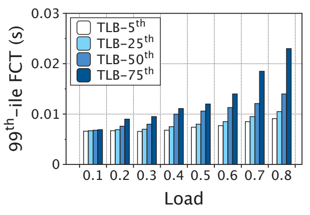
We use the common leaf-spine topology with 8 ToR and 8 core switches [2]. The whole network has 256 hosts connected by 1Gbps links. The round-trip propagation delay is 100μs and the switch buffer size is set to 256 packets. The traffic is generated by randomly starting flows via a Poisson process between random pair of hosts. The deadlines of short flows are randomly distributed between [5ms, 25ms]. The switching granularity of long flows is updated every 500μs. We vary the overall workload from 0.1 to 0.8 to thoroughly evaluate TLB's performance.

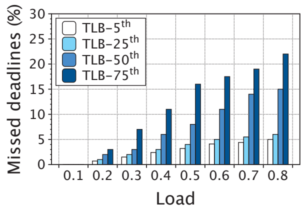
Fig. 10 (a), (b) and Fig. 11 (a), (b) show the average and 99^{th} percentile FCT of short flows under the web search and data mining workloads, respectively. We observe that TLB improves AFCT and tail FCT significantly compared with the other four schemes, especially in high workloads. Specifically, in the web search scenario, TLB reduces AFCT by ~68%, ~55%, ~45% and ~25% at 0.8 workload over ECMP, RPS, Presto and LetFlow, respectively. Fig. 10 (c) and Fig. 11 (c) show the deadline miss ratio of short flows. TLB ensures that more than 90% short flows meet their deadlines.

These test results demonstrate the advantage of TLB with adaptive switching granularity for different kinds of flows. Under the same switching granularity, when the workload becomes high, more mixed flows are queued in the same output ports at the switches, with the result that more short flows hit the long tail and experience packet reordering. Especially, since most short flows experience large queueing delay under large rerouting granularity (e.g., ECMP) or have lots of reordering packets under small switching granularity (e.g., RPS), the delay performance is degraded. For LetFlow, the performance is better in the high load because more flowlet gaps occur for switching under busty and congestion scenario, while the performance is not good under the low workload due to less opportunities to reroute flow. Compared with the other schemes, TLB is able to get enough gain by adaptively adjusting the switching granularity of long flows according to the arrival rate of short flows. Therefore, TLB alleviates the impact of queueing delay and out-of-order problem because of its adaptive rerouting granularity of short and long flows.

Moreover, we find that short flows in the web search workload have larger FCT than those in the data mining workload. The reason


(a) AFCT of short flows


(b) 99th percentile FCT of short flows


(c) Missed deadlines of short flows

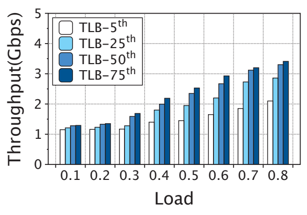

(d) Throughput of long flows

Figure 12: Performances of deadline-agnostic TLB

is that there are more medium flows with the size from 100KB to 1MB and more long flows in the web search workload, leading to larger queue length and more packet reordering. While in the data mining workload, the size between large number of short flows and a few long flows has obvious boundary, thus resulting in less out-of-order packets. For LetFlow scheme, however, the performance in the data mining scenario is worse than that of in the web search because there are fewer flowlet gaps to change path.

We also test the throughputs of long flows. As shown in Fig. 10 (d) and Fig. 11 (d), the long flows in the schemes with large rerouting granularity suffer the greater throughput degradation. Since TLB flexibly adjusts the size of switching granularity for long flows based on the traffic strength of short flows, it makes good use of multiple paths and reduces packet reordering, thus achieving the high throughputs of long flows.

6.3 Deadline-agnostic Case

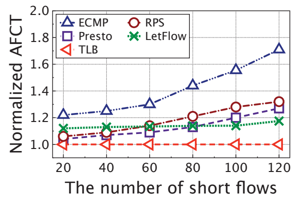
In some cases, it is hard to get the deadline information of short flows in advance. Moreover, since some latency-sensitive applications have various delay requirements, the deadlines of short flows may be different. For these scenarios without prior deadline knowledge or with different deadlines, we use an alternative method to protect short flows. Specifically, we set a fixed deadline for all short flows according to the statistics of deadline distribution. In this test, the accurate deadline of each short flow is randomly distributed between [5ms, 25ms] as illustrated in [16]. We specify the 5th, 25th, 50th and 75th percentile deadlines with the specified values of 5ms, 10ms, 15ms and 20ms, respectively [16]. The simulation topology and settings are as same as that in the large-scale simulations in section 6.2. We measure the AFCT, 99th percentile FCT and deadline miss ratio of short flows, and the throughputs of long flows under the web search application scenario.

Fig. 12 shows the performances of deadline-agnostic TLB. As shown in Fig. 12 (a) and (b), the AFCT and 99th percentile FCT become larger with the increasing workload. Moreover, the cases of TLB-5th and TLB-25th obtain the small FCT compared with the other cases. Fig.12 (c) shows the deadline miss ratios of TLB-5th and TLB-25th are much lower than the laxer deadline settings. In Fig. 12 (d), the cases of TLB-25th, TLB-50th and TLB-75th obtain the almost same throughputs of long flows, which are much higher than that of TLB-5th case. Therefore, we set the deadline of all short flows as the 25th percentile of statistical deadlines, which achieves the best performance in the deadline-agnostic TLB.

7 TESTBED EVALUATION

In this section, we use a realistic testbed to evaluate the applicability and effectiveness of TLB. The testbed is based on Mininet, which is a network emulation system with high fidelity on Linux kernel [18], [21], [26], [27]. The implementation results in [11], [28] show that Mininet's behavior is similar to the real hardware elements [26]. We implement TLB with P4 program [29] to specify how a switch processes packets.

In this test, we implement the packet processing pipeline of TLB with P4$_{16}$ 1.0. We use Mininet 2.3.0 to create the leaf-spine topology with 10 equal-cost paths between the leaf and spine switches. We respectively set the link bandwidth to 20Mbps and delay to 1ms as recommended in [18], [21]. BMv2 is installed as the software programmable switch with the buffer size of 256 packets. The default numbers of short flows less than 100KB and long flows larger than 5MB are 100 and 4, respectively. The overall traffic obeys the heavy-tailed distribution in web search workload as illustrated in [13], [18], [21] and the deadlines of short flows are randomly distributed between [2s, 6s]. We still use the 25th percentile deadline (i.e., 3s) to calculate the switching threshold of queue length for rerouting long flows. The updating interval of the switching granularity of long flows and the flowlet timeout are set to 15ms.


(a) AFCT of short flows

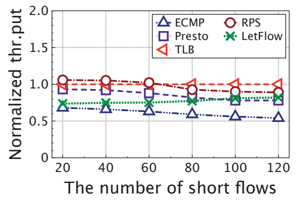
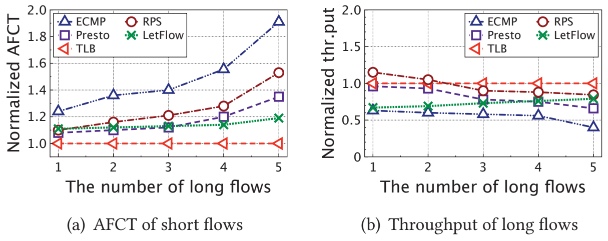

(b) Throughput of long flows

Figure 13: Varying the number of short flows

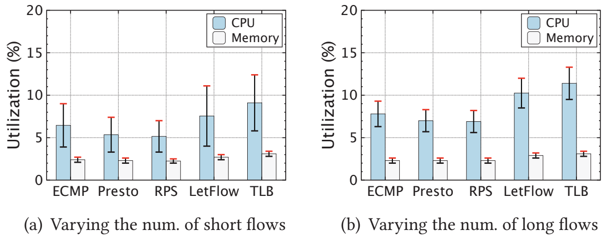
We normalize the results of ECMP, RPS, Presto and LetFlow to that of TLB. Fig. 13 (a) shows that TLB reduces the AFCT of short flows by ~18%-40%, ~6-24%, ~5-21%, ~10-15% with the increasing number of short flows over ECMP, RPS, Presto and LetFlow, respectively. As shown in Fig. 13 (b), TLB improves the average throughput of long flows by ~45%-80%, ~5-22%, ~20-35% over ECMP, Presto and LetFlow, respectively. With more long flows, Fig. 14 (a) and (b) show that TLB achieves better performance compared with the other schemes by adaptively adjusting the switching granularity



(a) AFCT of short flows (b) Throughput of long flows

Figure 14: Varying the number of long flows

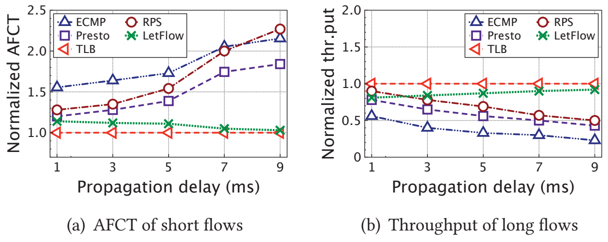
of long flows. Due to the inability to flexibly reroute flows, ECMP and LetFlow suffer from the long-tailed delay and low link utilization. For RPS and Presto, the packet reordering degrades their performances especially under the heavy workload.



(a) Varying the num. of short flows (b) Varying the num. of long flows

Figure 15: Overhead of the leaf switch

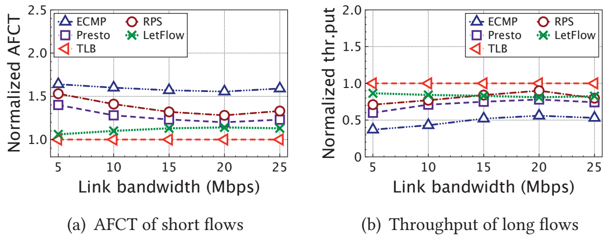
To evaluate the system overhead of TLB under the above two scenarios, we measure the maximum, minimum and average CPU and memory utilization ratio at the leaf switch as shown in Fig. 15 (a) and (b), respectively. For ECMP, RPS and Presto, due to their simple operations at switches, the CPU utilization is very low. Overall, since the calculation overhead of switching granularities only generates a tiny fraction of CPU load, TLB does not incur excessive CPU overhead and brings about negligible memory utilization compared with other schemes.



(a) AFCT of short flows (b) Throughput of long flows

Figure 16: Varying delay in asymmetric scenario

We further explore the performance of TLB in the asymmetric scenarios. We respectively vary the propagation delay and bandwidth of 2 randomly selected leaf-to-spine links [2] to create the topology asymmetry. In Fig. 16 and Fig. 17, we observe that under the greater the delay or bandwidth difference, TLB achieves more performance improvement compared with the other schemes except



(a) AFCT of short flows (b) Throughput of long flows

Figure 17: Varying bandwidth in asymmetric scenario

LetFlow. Under ECMP, the short and long flows experience large tailed delay and throughput degradation once being hashed to the bad paths. For Presto and RPS, the flows suffer the packet reordering and perform badly due to the delay or bandwidth diversity under the asymmetric scenarios. Since using flowlet switching, LetFlow is resilient to network asymmetry and performs well especially under the high degree of asymmetry. Overall, TLB outperforms all other schemes because it is able to perceive traffic and flexibly reroutes short and long flows with different granularities.

8 RELATED WORKS

The representative of the flow-based load balancing mechanisms is ECMP, which forwards each flow by using flow hash. To avoid hash collision in ECMP, Hedera [28] dynamically schedules flows by a central controller to alleviate hotspots. MicroTE [30] schedules flows by leveraging the partial predictability of traffic matrix through OpenFlow switches. FlowBender [31] reroutes the flow when the congestion or link failure is detected. The authors in [32] propose a congestion-aware algorithm to assign flows to the available paths online. Some other flow-based multi-path transmission schemes are proposed to schedule heterogeneous flows in different ways. Karuna [33] balances the interests of mix-flows with and without deadlines by using rate-control and priority-based flow scheduling. RepFlow [18] simply replicates each short flow to reap multi-path diversity in data centers. Freeway [34] adaptively partitions the transmission paths into low latency paths and high throughput paths respectively for the short and long flows. In brief, the above flow-based scheduling schemes have no packet reordering, but easily lead to the long-tailed latency or low-utilization problems under the highly dynamic traffic since the long flows are not able to switch flexibly among multiple paths.

As a flowlet-based load balancing scheme, CONGA [1] uses in-network congestion feedback to estimate load and then switches flowlets with the global congestion information to achieve good performance in data centers. LetFlow [8] randomly reroutes flowlets to naturally balance the traffic among parallel paths since the flowlet size changes automatically based on the traffic conditions on their paths. Flowtune [35] allocates an optimal transmission rate for each flowlet by using a centralized allocator. Clove [36] picks paths for flowlets in a weighted round-robin manner at the end-hosts. However, since the flowlet-based schemes only perform rerouting when the flowlets emerge, if the flowlet timeout value is large, they can not always make full use of multiple paths due to rare flowlet opportunities [2].

Presto [7] performs load balancing at the same granularity of fixed-size flowcell (i.e., 64KB) for both short and long flows, which are rerouted in a congestion-oblivious way. Presto needs to use the TCP offload functionality at the receiver to reassemble the out-of-order flowcells to prevent the reordered packets from being pushed up the networking stack.

RPS [6], a per-packet load balancing scheme, randomly sprays all packets to multiple paths to achieve high network utilization. MMPTCP [19] randomly spreads the packets of short flows to reduce the FCT and transmits the long flows by MPTCP [37] to improve their throughputs. DRILL [3] switches each packet quickly and flexibly based on the local queue information. However, the above packet-based load balancing schemes potentially incur reordering especially under the asymmetric topology. Hermes [2] reroutes flows only when the size sent exceeds a given threshold and cautiously makes rerouting decisions only when it will be benefit. Compared with Hermes, instead of using ECMP, TLB flexibly routes short flows to all non-congested paths at packet granularity. Thus, the short flows avoid being blocked by the long ones and almost have no packet reordering.

In contrast with the above load balancing schemes with same switching granularity on different types of flows, our solution TLB works through a new perspective: TLB adopts different granularities to switch short and long flows rather than all flows pick paths with same granularity. The switching granularity of long flows is adaptively changed based on the traffic load of short flows. Thus, TLB successfully addresses the long-tailed queueing delay, low network utilization and packet reordering problems to guarantee both the low latency of short flows and high throughput of long flows.

9 CONCLUSION

We proposed a novel traffic-aware adaptive granularity load balancing design TLB that reduces flow completion time for short flows and simultaneously improves throughputs for long flows. Specifically, TLB adaptively adjusts the rerouting granularity of long flows according to the load strength of short ones. Therefore, the short flows have more opportunities to make full use of the less-congested paths unused by long flows. The long flows are also able to flexibly change the switching granularity to improve their throughputs. The results of Mininet testbed and large-scale NS2 simulations demonstrate that TLB significantly reduces the AFCT of short flows by up to around ~40%, ~24%, ~21% and ~15% under high workload over ECMP, RPS, Presto and LetFlow, respectively. The throughput of long flows is remarkably improved by up to ~80% compared with flow-based load balancing schemes.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61872387, 61572530, 61872403), CERNET Innovation Project (Grant No. NGII20170107).

REFERENCES

[1] M. Alizadeh, T. Edsall, S. Dharmapurikar, et al. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In Proc. of ACM SIGCOMM.
[2] H. Zhang, J. Zhang, W. Bai, et al. 2017. Resilient datacenter load balancing in the wild. In Proc. of ACM SIGCOMM.
[3] S. Ghorbani, Z. Yang, P. Godfrey, et al. 2017. DRILL: Micro load balancing for low-latency data center networks. In Proc. of ACM SIGCOMM.
[4] G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, et al. 2017. Aphid: Hierarchical task placement to enable a tapered fat tree topology for lower power and cost in hpc networks. In Proc. of IEEE/ACM CCGrid.
[5] C. Hopps. 2000. Analysis of an equal-cost multi-path algorithm. RFC 2992, Internet Engineering Task Force.
[6] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella. 2013. On the impact of packet spraying in data center networks. In Proc. of IEEE INFOCOM.
[7] K. He, E. Rozner, K. Agarwal, et al. 2015. Presto: Edge-based load balancing for fast datacenter networks. In Proc. of ACM SIGCOMM.
[8] E. Vanini, R. Pan, M. Alizadeh, et al. 2017. Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching. In Proc. of USENIX NSDI.
[9] A. Putnam, A. M. Caulfield, E. S. Chung, et al. A reconfigurable fabric for accelerating large-scale datacenter services. ACM SIGARCH Computer Architecture News 42, 3(2014), 13-24.
[10] L. Zhou, C. Chou, L. N. Bhuyan, et al. 2018. Joint Server and Network Energy Saving in Data Centers for Latency-Sensitive Applications. In Proc. of IEEE IPDPS.
[11] M. Alizadeh, A. Greenberg, D. A. Maltz, et al. 2010. Data center tcp (dctcp). In Proc. of ACM SIGCOMM.
[12] A. Munir, I. A. Qazi, Z. A. Uzmi, et al. 2013. Minimizing flow completion times in data centers. In Proc. of IEEE INFOCOM.
[13] T. Benson, A. Akella, and D. Maltz. 2010. Network traffic characteristics of data centers in the wild. In Proc. of ACM IMC.
[14] L. Suresh, M. Canini, S. Schmid, and A. Feldmann. 2015. C3: Cutting tail latency in cloud data stores via adaptive replica selection. In Proc. of USENIX NSDI.
[15] S. Bak, H. Menon, S. White, et al. 2018. Multi-level load balancing with an integrated runtime approach. In Proc. of IEEE/ACM CCGrid.
[16] B. Vamanan, J. Hasan, and T. N. Vijaykumar. 2012. Deadline-aware datacenter tcp (d2tcp). In Proc. of ACM SIGCOMM.
[17] K. Zheng and X. Wang. 2017. Dynamic control of flow completion time for power efficiency of data center networks. In Proc. of IEEE ICDCS.
[18] H. Xu and B. Li. 2014. RepFlow: Minimizing flow completion times with replicated flows in data centers. In Proc. of IEEE INFOCOM.
[19] M. Kheirkhah, I. Wakeman, G. Parisis. 2016. MMPTCP: A multipath transport protocol for data centers. In Proc. of IEEE INFOCOM.
[20] L. Chen, K. Chen, W. Bai, and M. Alizadeh. 2016. Scheduling mix-flows in commodity datacenters with karuna. In Proc. of ACM SIGCOMM.
[21] J. Hu, J. Huang, J. Lv, et al. 2018. CAPS: Coding-based adaptive packet spraying to reduce flow completion time in data center. In Proc. of IEEE INFOCOM.
[22] M. Alizadeh, S. Yang, M. Sharif, et al. 2013. pFabric: Minimal near-optimal datacenter transport. In Proc. of ACM SIGCOMM.
[23] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris. 2008. Fundamentals of queueing theory. Wiley-Interscience.
[24] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron. 2011. Better never than late: meeting deadlines in datacenter networks. In Proc. of ACM SIGCOMM.
[25] W. Bai, L. Chen, K. Chen, et al. 2015. Information-Agnostic Flow Scheduling for Commodity Data Centers. In Proc. of USENIX NSDI.
[26] N Handigol, B. Heller, V. Jeyakumar, et al. 2012. Reproducible network experiments using container-based emulation. In Proc. of ACM CoNEXT.
[27] A. Khurshid, X. Zou, W. Zhou, et al. 2013. Veriflow: Verifying network-wide invariants in real time. In Proc. of USENIX NSDI.
[28] M. Al-Fares, S. Radhakrishnan, B. Raghavan, et al. 2010. Hedera: Dynamic flow scheduling for data center networks. In Proc. of USENIX NSDI.
[29] P. Bosshart, D. Daly, G. Gibb, et al. 2014. P4: Programming protocol-independent packet processors. In Proc. of ACM SIGCOMM Computer Communication Review.
[30] T. Benson, A. Anand, A. Akella, and M. Zhang. 2011. MicroTE: Fine grained traffic engineering for data centers. In Proc. of ACM CoNEXT.
[31] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene. 2014. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In Proc. of ACM CoNEXT.
[32] M. Shafiee and J. Ghaderi. 2016. A simple congestion-aware algorithm for load balancing in datacenter networks. In Proc. of IEEE INFOCOM.
[33] L. Chen, K. Chen, W. Bai, et al. 2016. Scheduling mix-flows in commodity datacenters with karuna. In Proc. of ACM SIGCOMM.
[34] W. Wang, Y. Sun, K. Zheng, et al. 2014. Freeway: Adaptively isolating the elephant and mice flows on different transmission paths. In Proc. of IEEE ICNP.
[35] J. Perry, H. Balakrishnan, and D. Shah. 2017. Flowtune: Flowlet Control for Datacenter Networks. In Proc. of USENIX NSDI.
[36] N. Katta, A. Ghag, M. Hira, et al. 2017. Clove: Congestion-Aware Load Balancing at the Virtual Edge. In Proc. of ACM CoNEXT.
[37] C. Raiciu, S. Barre, C. Pluntke, et al. 2011. Improving datacenter performance and robustness with multipath TCP. In Proc. of ACM SIGCOMM.
[38] M. Mizenmacher. The Power of Two Choices in Randomized Load Balancing. IEEE Tansactions on Parallel and Distributed Systems 12, 10(2001),1094-1104 .