Concise Paper: Freeway: Adaptively Isolating the Elephant and Mice Flows on Different Transmission Paths

Wei Wang*‡,Yi Sun*†, Kai Zheng§, Mohamed Ali Kaafar¶, Dan Li‖, Zhongcheng Li*
*Institute of Computing Technology, CAS,†State Key Laboratory of Networking and Switching Technology,
‡University of Chinese Academy of Science,§IBM China Research Lab,¶NICTA, Australia,‖Tsinghua University

***Abstract*—The network resource competition of today' datacenters is extremely intense between long-lived elephant flows and latency-sensitive mice flows. Achieving both goals of high throughput and low latency respectively for the two types of flows requires compromise, which recent research has not successfully solved mainly due to the transfer of elephant and mice flows on shared links without any differentiation. However, current datacenters usually adopt clos-based topology, e.g. Fat-tree/VL2, so there exist multiple shortest paths between any pair of source and destination. In this paper, we leverage on this observation to propose a flow scheduling scheme, Freeway, to adaptively partition the transmission paths into low latency paths and high throughput paths respectively for the two types of flows. An algorithm is proposed to dynamically adjust the number of the two types of paths according to the real-time traffic. And based on these separated transmission paths, we propose different flow-type-specific scheduling and forwarding methods to make full utilization of the bandwidth. Our simulation results show that Freeway significantly reduces the delay of mice flow by 85.8% and achieves 9.2% higher throughput compared with Hedera.**

I. Introduction

Datacenters are widely deployed as large-scale computing and storage facilities. More and more Internet services have migrated to datacenters such as web, video, online social network, etc. Datacenter network (DCN) interconnects thousands of servers to provide high performance communication for various applications, which has been a hot-spot in recent research proposals.

Traffic in datacenters mainly consists of two types of flows [7]: bulk data transfer (also called elephant flows, generated by applications such as data backup and virtual machine migration), and short-lived data exchange (also called mice flows, generated by applications such web search and Map-Reduce). Elephant flows require large bandwidth in order to achieve an expected high throughput without a strict completion deadline. As shown by Benson et al in [5], the elephant flows contribute to approximately 80% of the whole traffic in datacenter networks. On the other hand, short-lived mice flows are often very sensitive to latency and have to obey to a deadline constraint. The latency requirements of these applications have a significant impact on the users' perceived quality, which requires mice flows to be granted with higher priorities than the long-lived elephant flows.

Resource competition and conflicts are inevitable when simultaneously transmitting the two different types of flows through the same path in DCNs. Specifically, high throughput requires more packets queuing in the buffer, while low latency requires ultra short queue to reduce queuing latency. So far, scheduling algorithms design in DCN traded the latency constraint of mice flows off the need for high throughput of elephant flows or vice versa and considered this as a necessary devil. Hedera [2], MicroTE [6] and HULL [4] propose a few solutions where they transfer elephant flows and mice flows through shared paths, making it challenging and complex to find the optimal tradeoff between throughput and latency. Even, exploring the tradeoff between throughput and latency on the same path is not the best way to solve the problem.

In addition, DCN exhibits a different topology from that of typical enterprise networks. Generally, there are multiple shortest paths between any source and destination pair. For instance, in a K-ary Fat-tree network [1], there are $k^2/4$ shortest paths between each pair of servers. Recent datacenter design relies on a large number of commodity switches so that the topology often takes the form of multi-rooted tree to achieve horizontal scaling of hosts but decreasing aggregate bandwidth moving up the hierarchy as noted by [2]. Therefore, with the horizontal scaling consideration of datacenter topology, there exist increasing number of shortest paths.

With such a large number of available paths to transmit the traffic, in this paper we propose a new scheme named Freeway to isolate and distinguish different transmission paths for elephant flows from those for mice flows by dynamically partitioning the paths into low latency paths and high throughput paths. Specifically, elephant flows are transferred through the paths in which the corresponding links' buffers accommodate more queuing packets to achieve high throughput. Mice flows are transferred through the paths in which the corresponding buffers keep an ultra-low queue to achieve low latency. Freeway efficiently utilises these isolated multipath resources in DCN to achieve both low delay and high throughput. The main contribution of Freeway consists of two parts:

Firstly, we propose a dynamic path partition algorithm for elephant and mice flows. Our goal is to provide differentiated transmission services through adaptive path isolation which meets the demand of real-time traffic. The low latency paths maintains a low queuing time requested by mice flow, and high throughput paths builds up longer queues to accommodate more packets to meet the demand of throughput hungry flows. Path isolation in Freeway fundamentally decouples the two conflicting goals, which enables us to independently optimise the latency and throughput.

Secondly, based on the isolated paths, we propose to use different scheduling and forwarding methods for the two types of flows in both control plane and data plane. In Freeway, elephant flows are scheduled in controller from a global view of the network to get the optimal paths, while mice flows are scheduled in local switch and forwarded to the least congested output link on low latency paths.

The organisation of this paper is as follows: we present our design of Freeway in Section II, describing the path partition

978-1-4799-6204-4/14 $31.00 © 2014 IEEE
DOI 10.1109/ICNP.2014.59

algorithm and the scheduling algorithm. The performance of Freeway is evaluated in Section III. We present the related work in Section IV and conclude our work in Section VI.

II. Design of Freeway

A. Overview

We present an overview of Freeway. The first key point of Freeway is to dynamically isolate transmission paths of mice and elephant flows, which aims to guarantee not only low latency for mice flows but also a high throughput for elephant flows. As illustrated in Fig. 1, Freeway transfers mice flows through low latency paths with low queuing time, while simultaneously transferring elephant flows through high throughput paths with high buffering. Secondly, based on the isolated paths, Freeway uses different scheduling and forwarding strategies for the two types of flows respectively.

Nowadays, most commodity switches in datacenter have large buffer size to increase the throughput. However, queuing time in the buffers increase the transmission latency in DCN. In order to reduce the queuing latency while keeping high throughput, Freeway dynamically partitions the paths to *low latency path* and *high throughput path*, the definitions of which are given as follows:

Low latency path consists of low-latency oriented links (LOL). LOL maintains an ultra-low utilisation in its input/output/shared buffer to reduce packet buffering delay. Only mice flows can be transferred on these paths.

High throughput path consists of high-throughput oriented link (HOL). HOL maintains long queues in buffer to provide high throughput, so as to fully explore the potential bandwidth resource. Only elephant flows can be transferred through high throughput paths.

Clos-based network, such as Fat-tree, VL2 and Spine-Leaf [10] is widely deployed in recent data centers. In these networks, there are a large number of equal length paths between each pair of servers. Freeway utilises this topological characteristics of Clos-based networks and partitions the multiple paths into low latency paths and high throughput paths. The number of paths for each type is dynamically adjusted according to traffic volume of elephant and mice flows. A dynamic path partition algorithm running in the controller is proposed by considering a global view of the network.

Recent work including [2] and [6] often utilise centralised controllers for flow scheduling. However, with the explosion of traffic volume in DCN, the controller can be a bottleneck caused by large amounts of flow setup, and may prove difficult to scale. At the same time, the size of mice flows is usually of several hundreds KB. A large number of mice flow entries implies then a waste of controller's resources. In addition, the long setup time on the controller is not acceptable for the mice flows. Elephant flows on the other hand, not having a strict deadline, might be more suitable to be scheduled at the controller level.

Based on the isolated paths, Freeway utilises the centralised scheduling for every elephant flow in the controller, aiming to calculate the optimal scheduling paths. For mice flows, controller does not calculate the scheduling paths, and instead, installs the ToR (Top of Rack) flow entries in the switches. In the data plane, switch locally forwards the mice flow to the destination ToR through the least congested output link of the corresponding low latency paths.

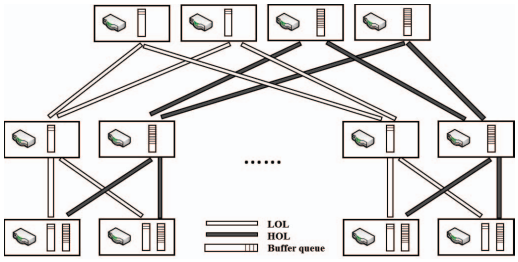


Fig. 1: The general architecture of Freeway

B. Dynamic Path Partition Algorithm

In this subsection, we present the path partition algorithm. Worth mentioning is that, although we illustrate our algorithm on a Fat-tree topology, the scheme actually can be used in any other Clos-based network (i.e. Spine-Leaf and VL2 [10]) with minor changes.

The objective of our algorithm is to priorly provide enough low latency paths for each pair of servers, and then to fully utilise the residual paths for high throughput flows. To achieve this goal, the algorithm needs to meet two constraints:

- ***Constraint 1***: there are at least one low latency path and one high throughput path between each pair of servers.
- ***Constraint 2***: based on real-time mice flow traffic, low latency path is dynamically added or deleted.

Fat-tree has three layers (edge ToR, aggregate and core), and the pod is a block consisting of some edge switches and aggregate switches [1]. For a k-ary Fat-tree topology, there are $k^2/4$ paths between each inter-pod ToR pair.[1] In addition, Fat-tree has a good property that given a core switch, there is one and only one path between a pair of ToRs. If core switches are numbered from 1 to $k^2/4$, the paths between each ToR pair can be labeled by the core switch number. We can sort the paths of each ToR pair separately by their core switch number in ascending order. For example as shown in Fig. 2, the core switches are numbered from 1 to 16 in a $k = 8$ Fat-tree, and we can get the sorted paths in the order from path 1 to path 16 between ToR A and ToR B. The sorted paths are very important to our path partition algorithm when adding or deleting the low latency paths.

In order to meet the first constraint, we define that for every pair of ToRs, the first path in its sorted paths is categorised as low latency paths. Thus, every edge switch in a pod connects the first aggregate switch by LOL, which means that the paths from 2 to $k/2$ could only transfer mice flows. Therefore, we make the first $k/2$ paths categorised as the low latency path. Similarly, the last $k/2$ paths are categorised as high throughput

[1] In Freeway, we mainly focus on the inter-pod traffic. And the ToR pair in the paper is equivalent to inter-pod ToR pair

paths. For example as shown in the Fig. 2, between ToR A and ToR B, path 1 to path 4 are fixed as low latency paths, while the path 13 to path 16 are fixed as high throughput paths. This could guarantee that for inter-pod traffic, there will be at least $k/2$ paths to transfer the elephant flows and mice flows. By doing this, it provides a bound for Constraint 2, for example, if there are large amounts of mice flows between a certain pair of ToRs, at most the first $(k^2/4 - k/2)$ in the sorted paths can be dynamically set as low latency paths and on the contrary, there are at most $(k^2/4 - k/2)$ high throughput paths from path $k/2 + 1$ to path $k^2/4$.

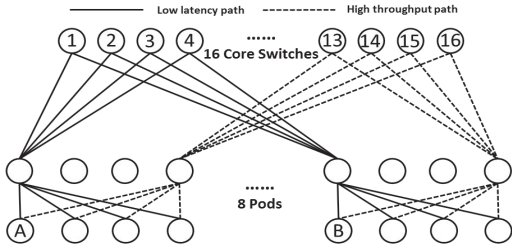


Fig. 2: Fixed low latency paths and high throughput paths in a Fat-tree for k=8

For each ToR pair, besides the $k/2$ fixed low latency paths and $k/2$ high throughput paths, the residual $k^2/4 - k$ paths can be dynamically set as any one type. To initialise the network, we set the residual $k^2/4 - k$ paths as high throughput paths. With the varying volume of ToR-ToR mice traffic, the low latency path is dynamically added or deleted. Next, we describe how to add and delete low latency paths with minimum change on links' type to meet the Constraint 2. Note that adding(deleting) a low latency path is not actually to add(delete) a link, but to change a HOL(LOL) to LOL(HOL) in the corresponding path.

The controller has a global view on the network-wide link utilisation in an SDN-based network. For each ToR pair, we calculate links' utilisation of all its low latency paths. If the average links' buffer utilisation of one path exceeds the threshold of LOL, we assume that this path is congested; while if the average buffer occupancy is less than the threshold, we assume this path is idle. Thus, for a ToR pair, if more than half of the low latency paths are congested, a new low latency path should be added. On the other hand, if more than half of the low latency paths are idle, a low latency path should be removed.

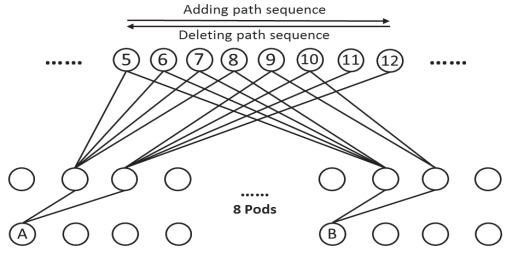


Fig. 3: Dynamic addition of a low latency path for the traffic from ToR A to ToR B in a Fat-tree for k=8

In order to adjust the least number of links when adding or deleting a path required by Constraint 2, we propose a greedy method to decide which path should be added or deleted. When adding a low latency path, we greedily select the path just after the last low latency path in the sorted paths, and when deleting a path we operate in the reverse way. As illustrated in Fig. 3, the low latency paths between ToR A and ToR B are added in the order from path 5 to path 12. If all ToR pairs add new path in this order, all LOLs will concentrate in certain aggregate switches and core switches. As shown in Fig. 3, the LOLs concentrate on the leftmost aggregate switches and core switches. One benefit of this simple method is to make the LOLs shared by more low latency paths of different ToR pairs, so that minimum links types are changed when adding paths. Secondly, by increasing the low latency paths of inter-pod ToR pair, the paths for inner-pod ToR pair are automatically added.

In addition, we assign a counter for each link, which is maintained in the controller. The counter is initialised to be 0, which shows the link is an HOL. If the counter is bigger than 0, the link is a LOL and its value means how many low latency paths share this link. Thus, when adding a new low latency path, counter of each link in the path is added. If the counter of a link becomes 1 from 0, we need to change its type from HOL to LOL. When deleting the path, the last low latency path in the sorted path will be deleted. Each link's counter in the deleted path is decreased by 1. When a link's counter becomes 0 from 1, we need to change the link to HOL. Otherwise, the links type cannot be changed, because it is used by other low latency paths.

C. Flow Scheduling and Forwarding

Based on the path isolation described above, we present the flow scheduling and forwarding in Freeway in both control plane and data plane. Considering the different throughput and latency requirements of elephant and mice flows, we propose to separately use different scheduling and forwarding strategies for the two types of flows.

Elephant flows usually transfer several MB to GB data bytes, which accounts for more than 80% of total traffic in DCN. However, the elephant flows, usually generated by the applications like virtual machine migration and data backup service, do not have a strict completion deadline. Thus, elephant flows can be scheduled in centralised controller by taking optimisation from a global view. The centralised scheduling algorithm for elephant flow aims to explore bandwidth resource of high throughput paths.

An important issue is how to identify an elephant flow from a mice flow. Hedera defines an elephant flow if the flow occupies at least 10% of NIC's bandwidth, while DevoFlow defines one as an elephant flow that has transferred at least a threshold number of 1-10MB bytes. However, a common weakness shared by the above definitions is that the first few MB bytes of the elephant flows have been transferred as a mice flow, before it can be identified as a large flow by the network. In Freeway, before a host sends a large flow (e.g. more than 10MB), it should send a "handshake" packet to the controller to claim the flow as an elephant flow, as A1 shown in Fig. 4. The "handshake" packet contains the basic information of the flow including flow size, expected sending rate, etc. The "handshake" latency is usually 1-2 RTTs, which is acceptable

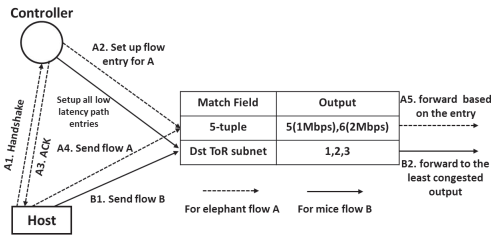


Fig. 4: Scheduling and forwarding processes

compared to the elephant flow's total transmission time.

Once the controller receives the "handshake" packet, it begins to calculate the optimal paths for the flow. Based on the ToR pair obtained from the "handshake" packet, the controller can select the available high throughput paths for the flow. Splitting the flow evenly across the multiple paths while not exceeding the capacity of each link along the paths is called multi-commodity flow (MCF) problem, which is NP-complete. The problem now can be formalised as a linear programming model, as shown in Eq. 1

$$\begin{aligned} Minimize: \quad & Max(\mu(e)) \\ Subject\ to: \quad & \sum_{\forall f_K \in subflow} f_K = f \\ & \mu(e) < Cap(e), e \in path\ of \forall f_k \end{aligned} \tag{1}$$

Where, $\mu(e)$ is the utilisation of link e. $Cap(e)$ is the capacity of link e. The objective is to minimise the maximum link utilisation among the available high throughput paths. There are some heuristic algorithms to solve this problem, such as Bin-Pack [12], Global First Fit and Simulated Annealing [2]. Here, we simply use Bin-Pack algorithm to calculate the optimal paths for the elephant flow.

The output of the elephant flow scheduling algorithm is a set of transmission paths and corresponding sending rate on each path. Then the controller will install a flow entry on each switch along the corresponding paths (A2 in Fig. 4), and send an acknowledgement to inform the host to start the flow (A3, A4 in Fig. 4)). In the data plane, the switch uses exact match by 5-tuple (src IP, src Port, dst IP, dst Port, protocol) in the flow table for each incoming elephant flow, and forwards the flow across multiple output links based on flow entry at corresponding rate (A5 in Fig. 4)). Mice flows, on the other hand, are usually generated by delay sensitive applications such as Map-Reduce and web search. Because mice flows usually last for a short period of time, the latency yield by invoking the controller for scheduling is not acceptable. Therefore, in Freeway mice flows are locally scheduled in switch.

Unlike sending a "handshake" packet before sending an elephant flow, the mice flows are sent out directly by the host (B1 in Fig. 4)). As described previously, the path partition algorithm has provided enough low latency paths for each ToR pair based on mice flow traffic. Instead of calculating the paths for each specific mice flow, the controller just set up the flow entries for low latency paths of each ToR pair. So in the data plane, if the flow's destination address matches the subnet address of a ToR, the switch will forward the flow to the least congested output link among all available low latency output links (B2 in Fig. 4)). Locally scheduling and forwarding mice flows reduces then the latency, which makes the flow complete by a fixed deadline.

Finally, re-scheduling is required when adding or removing low latency paths: (a) When adding a low latency path, there may be some links changing from HOLs to LOLs. In this case, the controller will re-schedule those elephant flows, which are paused by the link type switch, to a new high throughput path. (b) When removing a low latency path, there may be some links changing from LOLs to HOLs. In this case, we defer the time of the path type switch until all the mice flows on the path complete delivery. Thereby, some extra latency is required for deleting a low latency path, but the latency is limited due to the short lifetime of the mice flows.

III. Performance Evaluation

In this section, we present the simulated evaluation of Freeway. Firstly, we describe the simulation environment including simulator, topology and workloads. Secondly, we show the simulation results and analysis.

A. Simulation methodology

Simulator: to evaluate the performance of Freeway working on large-scale network and workloads, we develop a high-speed packet level datacenter network simulator with a centralised controller based on htsim [11]. The simulator is event-based and allows large numbers of concurrent flows and arbitrary network topologies. We also implement the TCP stack including congestion control and fast recovery.

Topology. We conduct our simulation on a 8-ary Fat-tree [1] topology, which is widely deployed in recent commercial datacenter networks. Specifically, the network has 16 core switches, 8 pods, and 4 edge switches as well as 4 aggregate switches in each pod. The network interconnects 320 servers, in which every 10 servers access to an edge switch. For each pair of servers, there are 16 shortest paths. And the controller directly connects the switches.

Workload. The workload is generated using SWIM project [8]. The original traces of the SWIM project span over 1.5 months (from Facebook). In the workload, there are 13630 mice flows, accounting for more than 85% of the total flows, and 2333 elephant flows. Mice flows account for 11% of the total data bytes, while elephant flows consisting of nearly 89% of all total size. We suppose that all flows are TCP flows. Note that our workload pattern is consistent with the traffic pattern described in [5].

B. Freeway performance

In this subsection, we show the performance of Freeway compared to three schemes: ECMP, Hedera and per-packet load balance (PerPktLB). In Hedera, we use Global First Fit as the schedule algorithm and set the schedule interval to be 1 second which is same with Freeway. In PerPktLB, it is supposed that every packet is scheduled to the least congested

path by the centralised controller. And we ignore the packet re-order problem which is out of the scope in this paper.

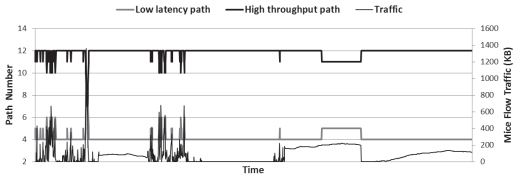


Fig. 5: The number of low latency path and high throughput path varying with mice flow traffic for a ToR pair

Firstly, we evaluate the performance of dynamic path partition algorithm. As illustrated in Fig. 5, we examine the varying number of low latency paths and high throughput paths for a pair of ToR as time goes on, comparing with the mice flow traffic. At first, we can see that in most of time, the number of low latency paths stays at 4, which is the initialized lower bound by path partition algorithm. Periodically, low latency paths increases to 5 or at most 6, but it will rapidly go down to the lower bound 4. This result also shows that the nature of mice flow traffic is burst and short-lived, which could also be proved by the line of mice flow traffic in the figure. Then, the number of low latency path tightly fits the variation of mice flow traffic, which validates the accuracy and promptness of the path partition algorithm proposed.

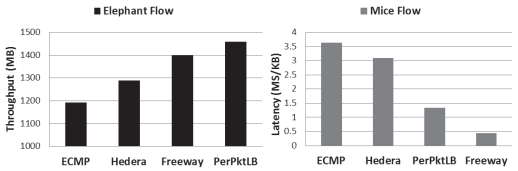
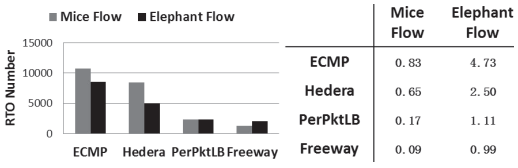


Fig. 6: (a) throughput (b) latency

Fig. 6 presents elephant flow's throughput and mice flow's latency in Freeway compared to the other 3 schemes. As shown in Fig. 6(a), Freeway improves throughput by up to 8.7% compared to Hedera, and it achieves almost the same throughput as PerPktLB. PerPktLB achieves largest throughput at about 1457MB, while ECMP just transfers less than 1200MB. From Fig. 6(b), it shows that for mice flow, Freeway significantly reduces latency by 87.8%, 85.8% and 66.9% respectively compared to ECMP, Hedera and PerPktLB. Because Hedera uses ECMP to split mice flow traffic across multiple paths, its latency is near to ECMP.



	Mice Flow	Elephant Flow
ECMP	0.83	4.73
Hedera	0.65	2.50
PerPktLB	0.17	1.11
Freeway	0.09	0.99

Fig. 7: (a) total RTO (b) average RTO per flow

Fig. 7(a) demonstrates the retransmission timeout (RTO). The mice flow RTO number of Freeway is obviously reduced by 88.1%, 84.9% and 44.3% respectively compared to the other three schemes. And the elephant flow RTO number is reduced by 75.6%, 58.8% and 9.9%. Specifically in 7(b), it presents the average RTO number per flow. It shows an obvious reduction that the RTO number per flow in Freeway is 0.09 and 0.99 respectively for mice flow and elephant flow. So in Freeway, ultra less RTO happens in each flow than the other 3 schemes, which will make a contribution to complete the flow as soon as possible within the deadline.

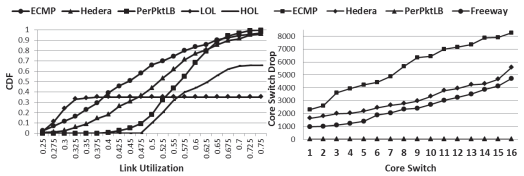


Fig. 8: (a) link utilisation (b) switch drops

To evaluate the scheduling algorithm in Freeway, we show the CDF of the link utilisation for all links, as presented in Fig. 8(a). As illustrated in Fig. 8(a), ECMP has almost the same proportion in each link utilisation interval from 0.25 to 0.65, which means that the traffic is not balanced caused by hash collision of elephant flows. In Hedera, only about 20% of link utilisation is less than 0.425 or more than 0.6, and more than 60% of link utilisation distribute between 0.425 and 0.6. This proves that the traffic is balanced among the links in Hedera. In PerPktLB, approximately more than 70% of link utilisation distributes between 0.5 and 0.65, better than Hedera. In Freeway, because a link could be LOL or HOL with time varying, we respectively examine their utilisation during the experiment. The link utilisation of LOL mainly distributes between 0.2 and 0.325, and this low utilisation is own to its low packet buffering. For the HOL, about 60% of its utilisation distributes between 0.5 and 0.65, which is caused by high buffering to achieve high throughput. No matter what type the link is, the traffic of mice flow and elephant flow are balanced in respective type of link.

At last, we examine switch drops, as illustrated in Fig. 8(b). We focuses on core switch drops, because huge traffic is aggregated in core layer and the Incast problem often happens in core switches [5]. In Fig. 8(b), we sort the core switches by their drop number in increasing order. The core switches in PerPktLB have no packet drops, which also proves that the network resource is enough in most datacenters even in some extreme situation. Therefore the drops in the other 3 schemes indicates that the core switches experience momentary bursts or long-lived elephant flows collision which is avoidable if the flow is appropriately scheduled. As it shows, Freeway reduces the drops by at least 42.7% compared to ECMP, and also outperforms the performance of Hedera.

To conclude, ECMP performs poor because the hash-based static mapping of flows to paths does not account for either account for either current network utilisation of flow size. Hedera performs not well in the latency of mice flows because it makes no discrimination between elephant and mice flow. PerPktLB achieves a good throughput and a low latency, however these results do not consider the re-ordering issue in PerPktLB. In practical deployments, packets re-ordering issues brought by PerPktLB may lead to significant TCP performance degradation. Freeway finds a better trade-off by adaptively partitioning the paths for the two types of flows, and achieves both low latency and high throughput.

IV. Related Work

There have been many related works to improve datacenter network' performance considering the horizontal scaling topologies. Hedera [2] and MicroTE [6] propose centralized scheduler to schedule the flow using global information. Hedera presents a scalable dynamic flow scheduling system that adaptively schedules elephant flows through multipath [2], but it can't guarantee the flow completion time (FCT) of latency-sensitive mice flows. MicroTE utilizes a centralized routing component to compute network paths by leveraging the short term and partial predictability of the traffic matrix [6]. But the common weakness of centralized control is the scalability especially in large-scale datacenter networks. DeveFlow [9] improves the scalability through devolving control to local switch by rule cloning and local actions. However, all of these scheduling approaches have a common point that elephant and mice flows are mixed on the shared paths. In the shared paths, competition of high throughput and low latency causes the unavoidable conflict of queuing control. So it is hard for recent proposals to achieve both goals of the two types of flows.

Some other works focus on congestion control on the transmission path. These works try to reduce latency for mice flow by sacrificing a little bandwidth. DCTCP [3] employs an adaptive reaction to ECN so as to limit the buffer occupancy of elephant flows to avoid congestion. Based on DCTCP, HULL [4] proposes to achieve ultra-low latency by sacrificing about 10% of bandwidth for reducing packet queuing delay down to zero. Similarly, D2TCP [14] uses ECN to allow latency-sensitive flow with strict deadline to obtain more bandwidth. However, considering the horizontal scaling of DCNs, there are plentiful shortest paths between any pair of servers. So among the available paths, it's better to adaptively make part of paths keep ultra-low queue, while other paths are allowed to build up high queues for high throughput.

V. Discussion

Monitoring: Freeway utilizes the global view of SDN-based controller to measure the network-wide link utilization. Several techniques could reduce this overhead. Openflow 1.3 adds the support of per-flow meters, which helps to get statistics of flow and switch port. There are many recent works focusing on the monitoring and measurement in SDN networks to reduce controller's overhead and increase scalability, such as OpenTM [13], FlowSense [15], and OpenSketch [16]. Cooperating with these methods, Freeway could achieve more scalable control plane and data plane.

Elephant Flow Detection: the methods proposed by Hedera and DevoFlow suffers the detection delay. Freeway makes a map between application and flow type previously in deployment environment. We can add a hook function to send the "handshake" packets for elephant flow before TCP three-way handshake, which is transparent to users. More accurate elephant flow detection method will be studied in future work.

VI. Conclusion

In this paper, we present Freeway to isolate the transmission paths for elephant and mice flow by dynamically partitioning the paths to high throughput paths and low latency paths. Through path isolation, Freeway fundamentally decouples the tradeoff between throughput and latency, which enables the network independently improve these two goals. We propose a path partition algorithm which dynamically adjusts number of low latency path for each ToR pair according to the mice flow traffic. Leveraging on the isolated paths, Freeway uses different scheduling algorithms for both types of flows. By taking a global view of the network, a controller optimally schedules the elephant flows to make full use of the bandwidth resources. Each switch locally schedules the mice flows, which reduces the flow setup time and increases the scalability of the controller. From our evaluation, Freeway outperforms the state-of-the-art ECMP and recent proposals from both the throughput and the latency perspectives.

VII. Acknowledgement

This work is partially supported by the National Key Basic Research Program of China (973 program) under Grant 2012CB315802 and 2014CB347800, and the National Natural Science Foundation of China under Grant 61379133 and 61170291.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM 2008*, pages 63–74, 2008.

[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI 2010*, volume 10, pages 19–19, 2010.

[3] M. Alizadeh, A. Greenberg, and e. a. Maltz. Data center tcp (dctcp). In *SIGCOMM 2011*, volume 41, pages 63–74, 2011.

[4] M. Alizadeh, A. Kabbani, and e. a. Edsall. Less is more: trading a little bandwidth for ultra-low latency in the data center. In *NSDI 2012*, pages 19–19, 2012.

[5] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *IMC 2010*, pages 267–280, 2010.

[6] T. Benson, A. Anand, A. Akella, and M. Zhang. Microte: fine grained traffic engineering for data centers. In *CoNEXT 2011*, page 8, 2011.

[7] Y. Cao, M. Xu, X. Fu, and E. Dong. Explicit multipath congestion control for data center networks. In *CoNEXT 2013*, pages 73–84, 2013.

[8] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz. The case for evaluating mapreduce performance using workload suites. In *MASCOTS 2011*, pages 390–399, 2011.

[9] A. R. Curtis, J. C. Mogul, and e. a. Tourrilhes. Devoflow: scaling flow management for high-performance networks. In *SIGCOMM 2011*, pages 254–265, 2011.

[10] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. In *SIGCOMM 2009*, pages 51–62, 2009.

[11] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath tcp. In *SIGCOMM 2011*, pages 266–277, 2011.

[12] S. Sen, D. Shue, S. Ihm, and M. J. Freedman. Scalable, optimal flow routing in datacenters via local link balancing. In *CoNEXT 2013*, pages 151–162, 2013.

[13] A. Tootoonchian, M. Ghobadi, and Y. Ganjali. Opentm: traffic matrix estimator for openflow networks. In *PAM 2010*, pages 201–210, 2010.

[14] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). In *SIGCOMM 2012*, pages 115–126, 2012.

[15] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, and G. Jiang. Flowsense: monitoring network utilization with zero measurement. In *PAM 2013*, pages 31–41, 2013.

[16] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with opensketch. In *NSDI 2013*, pages 29–42, 2013.