Controlling the Delay of Small Flows in Datacenters

Damiano Carra
Computer Science Dept.
University of Verona
Verona, Italy
damiano.carra@univr.it

***Abstract*—As datacenters grow in size, the communication between servers has emerged as a major bottleneck. Studies have shown that datacenter workloads are highly variable in sizes, comprising of a mix of mice and elephant flows which, when coupled with hard to predict arrivals, make bandwidth provisioning and flow scheduling a challenging task. There has been a significant progress in excess bandwidth provisioning and flow scheduling algorithms, especially using hybrid electrical-optical networks, aimed at providing effective throughput to elephant flows while ignoring the delay of small flows. The latency of small flows, however, is an important performance metric and existing solutions for improving this metric for traditional packet switched as well as hybrid datacenter networks continue to be inadequate. We aim at looking at the design of a datacenter network which ensures adequate bandwidth provisioning and minimizes delay for mice flows effectively. In this paper, we propose a novel hybrid architecture along with simple flow routing schemes to achieve these goals. The design schemes proposed in this paper can be incorporated in the existing datacenter networks. We evaluate the performance of our scheme and compare it to other existing schemes through simulations.**

I. Introduction

Datacenter networking has attracted a lot of attention in recent years. It is necessary to eliminate bottlenecks to exploit the growing amount of computational resources in a datacenter. One of the most important bottlenecks is the communication among the machines. Recent advances in datacenter network architecture aimed at providing redundant bandwidth [1], [2] and reducing the cost of providing this excess bandwidth using optical links [3]–[5]. Several solutions exist to control the traffic routing through the network, ranging from hash-based flow forwarding (ECMP) [6] to solutions that manage large flows well to improve performance like Hedera [7]. These solutions are enabled by Software Defined Networking (SDN) and the OpenFlow [8] standard. The hybrid networks, on the other hand, use similar version of hot-spot scheduling which involves polling the traffic matrix periodically and setting up high bandwidth paths for large flows in circuit switched networks.

The traffic in a datacenter is highly variable and composed of many small flows and few large flows [9], [10]. Large flows usually are generated when Virtual Machines are instantiated or moved across the datacenter, or when data to be processed are stored, or when the output of the processing need to be sent to other machines (e.g., the results of the Map tasks that are sent to the Reduce tasks). In all these cases, the delay does not play an important role for large flows but their throughput performance is important. On the other hand, small flows may be associated with RPC calls, high frequency trading, interactive applications etc., and therefore, need low latencies.

A common aspect shared by all the infrastructure designs is that they are focused on provisioning large bandwidths at low OPEX and CAPEX and ignore designs which could help reduce delay of delay sensitive flows. The flow routing schemes generally route largest flows most efficiently and consequently do not guarantee small latencies for small flows. Even with the excess bandwidth and efficient routing of large flows, small flows can still experience large latencies due to the following reasons: 1) presence of a long tailed flow at the ToR switch with the small flow, 2) routing delays associated with hierarchical topology of electrical networks. Recently, a few works attempted to consider the latencies for small flows for packet switched datacenter networks as a main performance index [11]–[15] but do not solve the problem completely.

We consider the problem of datacenter network design from a fresh perspective, driven by the following requirements: small flows will have special treatment in order to control their delay, and the main performance index for large flows is the throughput. We propose a general two-tier topology with one flat tier dedicated to small flows, while the other tier carries the large flows. While the topology of our network appears similar to [5], the architecture is completely different and provisioned for completely different goals. Instead of focusing on routing large flows efficiently, we focus on augmenting the existing datacenter network to support small delays for small flows.

There are different issues that need to be considered in the design, such as the identification of small and large flows, the bandwidth dimensioning of the two tiers, and the routing, to cite a few. In this paper we identify and discuss these problems, and propose a set of solutions. Further, we evaluate our design with a packet level simulator and compare it with different topologies – packet switched topologies, as well as other hybrid topologies. We show through simulations that our architecture is able to decrease the delay of small flows, without affecting the overall throughput of large flows while using traffic patterns as shown in actual datacenter measurement studies [9], [10].

The rest of the paper is organized as follows. In Sect. II

we provide the necessary background and discuss the related works. In Sect. III we present our design of the datacenter network architecture, and discuss the main issues related to this architecture. We evaluate with simulations our solution and present the results in Sect. IV. In Sect. V we discuss different aspects of our design, and we conclude in Sect VI.

II. Background and Related Work

In a datacenter, servers are grouped in racks with a switch on top, usually called Top-of-the-rack switch (ToR). The switch collects the traffic coming from and directed to the servers. Each rack contains typically 20-80 servers. The networking infrastructure is responsible for interconnecting the ToR switches.

The main challenge is to eliminate potential bandwidth bottlenecks among ToR switches. For instance, in a tree based architecture, the switch at the root of the tree would need ports with an extremely high bandwidth, which is impractical. For this reason, many alternative architectures, inspired by literature on the design of switching matrices, have been proposed [16]. In these architectures, the network interconnecting the ToR switches is composed by commodity switches, with standard links (e.g., 1 and 10 Gigabit Ethernet), arranged in a way to provide multiple paths. VL2 [1] is inspired by a Clos topology and [17] uses a fat-tree topology to provide full bisection bandwidth among the ToR switches.

The major drawback of the above architectures is the cost associated to the networking infrastructure. Since it is unlikely that all servers needs to communicate with all the other servers, it is common practice to oversubscribe the network, i.e., to provide a fraction of the bisection bandwidth. Another approach that aims at decreasing the cost, yet providing full bisection bandwidth to ToR switches, is to build hybrid solutions [3], [4]. Along with a packet switched topology, the infrastructure includes a (optical) circuit switched topology, where the circuits carry the large flows – recall that few flows bear most of the traffic of the datacenter. All the above architectures use variants of hot spot flow scheduling algorithms where a central scheduler routes flows to best possible links based on traffic matrix. More recently, a flat optical architecture Mordia [5] for hybrid networks has been proposed which uses microsecond order optical switching to provide a flat ring topology to large flows.

Some recent works started to investigate solutions that are able to improve another important performance index: the delay of delay sensitive flows in packet switched datacenter networks. The delay usually arises because of the following factors: 1) packet losses and retransmissions, 2) lack of priorities for different traffic classes, and 3) uneven load balancing.

DCTCP [11], and HULL [15] have been proposed to control the congestion in the network and therefore limit the delay of the packets. D^3 [14] proposes a protocol for deadline-sensitive application, thus allowing application to specify a maximum delay. DeTail [13] is a multi-path congestion mechanism scheme that aims at reducing the tail completion times of flows. In [18] a distributed preemptive scheduling scheme for small flows. Finally, MPTCP [12] balances TCP flows across multiple paths, so that to avoid hot spots and consequently congestion. All the above mentioned works share a common aspect: they require changes in the datacenter protocol stack, either in the end hosts, or in the switches. In our work, instead, we propose a solution that can be implemented on commodity hardware[1], on top of the existing datacenter networks. Our solution does not need any modification of the protocol stack or the applications running on the existing datacenters. The 2-tier architecture we propose has a focus on the delay of latency sensitive flows. With respect to the solutions based on hybrid packet and circuit switched topology, our solution has different objectives, namely the delay, therefore is structural different from those hybrid architectures, such as Hedera [7]. Hedera aims to solve the load balancing problem by periodically rerouting the heavy flows, but the solution cannot address delays of small flows. In summary, in our solution we propose a way to control the delay of small flows without modifying the hardware or the software of the machines or switches in the datacenter.

III. Design

We propose a 2-tiered architecture, where each tier carries different types of traffic – mice and elephants (see Fig. 1). The first tier is specifically designed to minimize the delay of mice. From a logical point of view, we have a fully connected topology: each Top-of-the-Rack (ToR) switch has a direct link to all the other ToRs, and therefore the number of hops is minimized. The second tier is dedicated to elephants: this tier can be built either using multi-hop paths with highly variable delays (packet switched topology with oversubscription), or a circuit switched topology with relative long switching times (e.g., order of ms, such as MEMS-based optical switches).

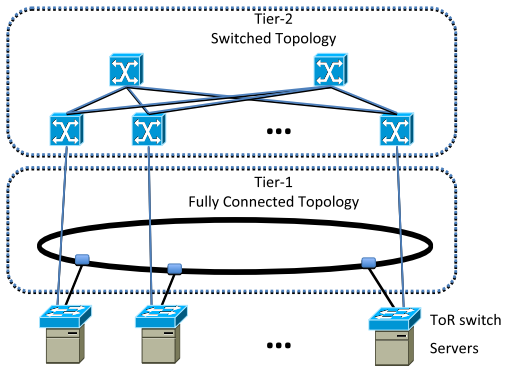


Fig. 1. 2-tier architecture: small flows are routed using a fully connected topology, large flows are offloaded to a switched topology.

Our solution poses several challenges. From the traffic management point of view, we need to be able to identify mice and elephants, so that the traffic can be properly routed. From the topological point of view, we need to identify a suitable solution for building a fully connected topology in the first tier

[1]The only requirement is the compliance with a SDN implementation, e.g., OpenFlow.

(the figure shows a possible implementation which we discuss in greater detail in § III-A).

A. Fully Connected Topology

Datacenters are extremely heterogeneous in the size: from small-scale datacenter with few hundreds of machines, to mid-scale with few thousands of machines, up to big datacenters with tens of thousands or hundreds of thousands of machines. The solution we propose targets mid-scale datacenters, or container-size datacenter. Considering that each ToR switch connects approximately 40 machines, a mid-scale datacenter with 5000-10000 machines will have 125-250 ToR switches. Today's high-end switches have up to 256 ports equipped with 10 Gb Ethernet, therefore the fully connected topology can be obtained with a single high-end switch[2].

The adoption of a single high-end switch has a main limitation: the maximum connection speed from one ToR switch to all the other ToR switches is bounded by the speed of the port, i.e., 10 Gb. In the following, we describe an alternative solution based on optical networks.

We consider a wavelength-division multiplexed (WDM) ring: within such a ring, it is possible to create different "channels". A channel may be an optical wavelength (i.e., λ) or, with the use of TDM, a fraction of a λ. In a fully connected topology, one would need $O(N^2)$ channels, where N is the number of ToR switches, which would result in a high fragmentation of the resources. Instead, we propose a different approach. We make use of N channels: in each channel, a single ToR switch can transmit, while all the other ToR switches listen. We call such channels "Single Source Broadcast Channels" (SSBCs). Since in each SSBC there is a single transmitting ToR, the channel is collision-free. Each ToR listens to all the SSBCs (except the one it uses to transmit), since it may receive traffic from any ToR (and hence from any SSBC). The selection of the destination (among the different listeners of a channel) is done at the protocol level (destination address in the packet header).

The source ToR switch can use the channel for unicast, multicast and broadcast communications, therefore this solution is extremely flexible. Note that, since ToR switches are spatially close one another, one may implement the ring inside a single switch, and each optical module would be a port of such a switch.

While designed with different objectives in mind (as explained in Sect. II), the Mordia optical switch architecture [5] also adopts a similar optical ring-based topology. However, Mordia employs wavelength-selective switches (WSSs) to selectively filter individual wavelengths to a single receiver per node, effectively realizing a circuit switch rather than a broadcast fabric.

Overall, the proposed design allows for a flexible assignment of the resources: a channel can use a configurable fraction of λ, and, if the number of resources in a single ring is not sufficient, it is possible to add additional rings.

[2]Such a switch does not need to support OpenFlow, since it is used to connect the ToRs, and the routing is done automatically.

B. Routing Flows

Given the 2-tier topology as shown in Fig. 1, the first issue is to identify the mouse and the elephant flows. There are several approaches to do this which have been well studied in literature: 1) Applications identify the flows as mice or elephant, 2) Maintaining per flow statistics at switches, 3) Random sampling of flows, 4) End to End host identification. All the approaches have their own advantages and benefits [19]. It is possible to adopt all of the above mentioned solutions in our architecture.

We would like to keep the design principle as simple as possible and use the existing functionalities available in common commercial products. For this purpose we use a simple scheme for elephant detection directly at the ToR switches, without any information coming from the servers or applications. The detection is simply based on packet (alternatively: byte) counting: given a threshold T_s, all the flows with more than T_s packets are considered elephants [3] (we discuss how to determine T_s at the end of this section).

The routing between the first and the second tier is extremely simple. The default route for a new flow is the fully connected topology. If the number of packets of a given flow hits the threshold T_s, then the flow is routed through tier 2. The flat ring architecture of first tier offers several advantages for routing the mice flow. Each switch has a view of the congestion that the outgoing flows will face. This allows to use several priority based or preemptive single path scheduling mechanisms within the first tier which are the motivation behind existing literature for e.g. [14], [18]. The routing within second tier will follow the paths determined by the specific technology (e.g., ECMP).

These simple rules can be implemented using the SDN framework, e.g. OpenFlow. Using OpenFlow, it is possible to set the threshold T_s in every ToR switch. When a flow sends more than T_s packets, the ToR switch asks to the OpenFlow Control Center the new route. Since the Control Center has a global view of the elephants currently routed through the switched topology, it will be able to compute the route of the new flow, trying to maximize the overall throughput. The new route is communicated to the ToR switch (and the other switches involved in the path), so that the switches can start forwarding the packets accordingly. With this solution, the OpenFlow Control Center is contacted only for the elephants, which are the 10-20% of the total flows, with a great reduction of the Control Center burden with respect to a solution without default routes.

The threshold T_s can be easily computed by the OpenFlow Control Center. Periodically, e.g., daily, the Control Center collects the statistics on the flows stored by the ToR switches, and, considering the available resources in the fully connected topology, calculates T_s such that the average load on such a topology would be minimal (e.g. 20-30%, in order to contain the delays).

[3]We admit this involves maintaining table containing information for each flow at a switch and could be buffer intensive.

IV. Evaluation

In this section, we show the results obtained with a simulator of datacenters, which has been derived from the one developed in [12].

A. Settings

The simulator is composed by AIMD (Additive Increase Multiplicative Decrease) sources, and switches, where each interface is implemented as a FIFO (First In First Out) queue.

We have implemented three datacenter architectures. The first is the VL2 architecture [1], which is a Clos topology; by setting the speed of the switch interfaces, it is possible to obtain either full bisection bandwidth, or an oversubscribed topology. The flow routing is implemented using the ECMP algorithm. The second architecture is our proposed 2-tier topology, which we call mDelay. In this case, the flows are routed initially using the fully connected topology; if the flow is bigger than T_s packets, then, starting from packet T_s+1, the flow is routed through the other tier (switched topology), for which we use VL2 architecture and ECMP routing. The third architecture is a 2-tier architecture with optical circuits used for carrying elephant flows: cThrough [4]. With cThrough, a central controller regularly computes the maximum weighted matching among ToR switches (the weights are the traffic to be transferred between any pairs of ToRs); the output is used to configure the circuits, while the rest of the traffic goes through a switched topology (implemented as VL2 topology).

Each server in the topology has 10 active flows, which is the average number of flows as reported in [4]. The size of the flows is decided when the flow is created, using the statistics provided in [4]. The average packet size is 1 KByte. When a flow is successfully transferred, a new flow is created, therefore the number of active flows remains constant: this has been done to let system work in the heavy load regime, where the delay and the throughput are more sensible to the design choices. When the flow is created, we have to assign the destination. This choice will determine the traffic pattern among different ToR switches. We have implemented three traffic patterns: uniform, diagonal with a light tail, and diagonal with a heavy tail. With a uniform distribution, the destination is chosen uniformly among all the possible destinations. With a diagonal distribution, the destination is chosen preferably from a subset of destinations (note that all the other destinations can still be chosen, but with low probability). In case of light tail, this subset is limited, while with heavy tail, this subset is broader.

In our simulations, given a traffic pattern, we compare the performance of the three architectures described above (VL2, mDelay and cThrough). In order to do so, we need to ensure that the comparison is fair: we assume that fairness is defined in terms of the amount of available resources, i.e., bandwidth[4]. In particular, we make sure that the total bandwidth from a ToR switch to all the other ToR switches (considering all the available links, i.e., the ones to the aggregate switches in the VL2 topology, and the optical paths in the 2-tier topologies – mDelay and cThrough) remains constant across the different topologies.

As performance index, we focus on flow delay, node throughput and flow completion time. In our preliminary evaluation, we have used the following parameters: 1280 servers divided into 64 ToR switches; no oversubscription (i.e., the output bandwidth of the switches is 20 times the server bandwidth); in case of mDelay topology, the output bandwidth of the switches is decreases by 10%, which is given to the fully connected topology, and the threshold for the flow routing is 35 packets; in case of cThrough topology, the output bandwidth of the switches is decreases by 50%, which is given to the circuit switched tier; given a specific configuration, we performed multiple runs in order to obtain statistical confidence, and we compute the 95% confidence interval.

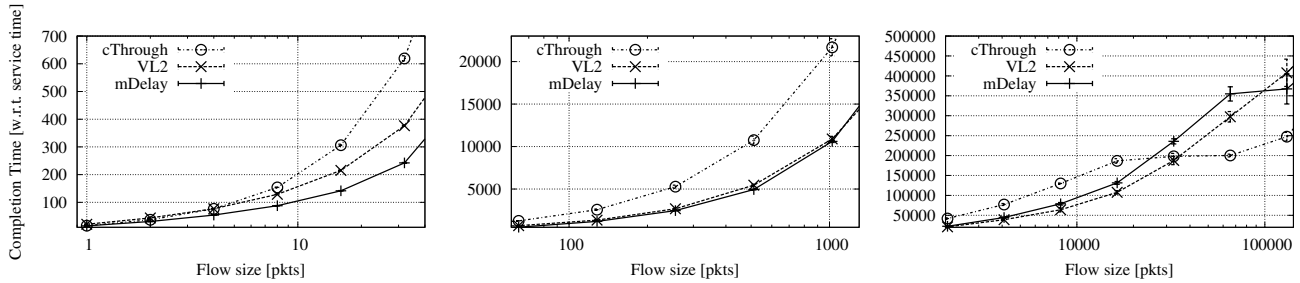
B. Results

In Table I we show the average values of the flow delay and node throughput in case of diagonal traffic with heavy tails (similar considerations can be done with other traffic patterns). The throughput is expressed as a percentage of the server bandwidth. The delay is expressed using as unit the service time (average packet size divided the node bandwidth). Rather than the absolute values, these results are interesting when we compare the different architectures. We immediately observe that the node throughput with cThrough is decreased (with respect to the other topologies). The reason is simple: the cThrough architecture dedicates a portion of the available ToR switch bandwidth (the optical path) to a single path (another ToR, the one for which there is more traffic to send). This means that the portion of bandwidth in the switched network is decreased, and congestion is more likely to occur, and overall the throughput decreases. Note that the traffic pattern (diagonal) represents the best possible situation for cThrough; with a uniform traffic pattern, the results are even worse. Note also that, in the original cThrough paper, the optical bandwidth is added, and comparison is not done with a constant switch bandwidth as we do. In case of VL2 or mDelay architectures, the throughput is equivalent, while the flow delay is significantly decreased (-17%). Rather than aggregate values, it is interesting to analyze the performances for different classes of flows (mice, elephants).

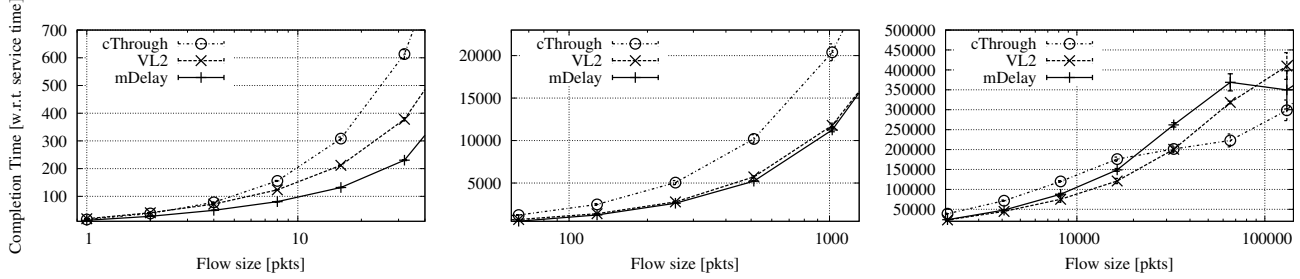
TABLE I
Average Flow Delay and Node Throughput for different topologies (traffic pattern: Diagonal Heavy Tails; 95% confidence interval)

	Flow Delay (w.r.t. service time)	Node Throughput (%)
VL2	21.2 ± 0.2	44.9 ± 1.9
mDelay	17.6 ± 0.1	44.1 ± 2.9
cThrough	18.2 ± 0.1	40.0 ± 1.5

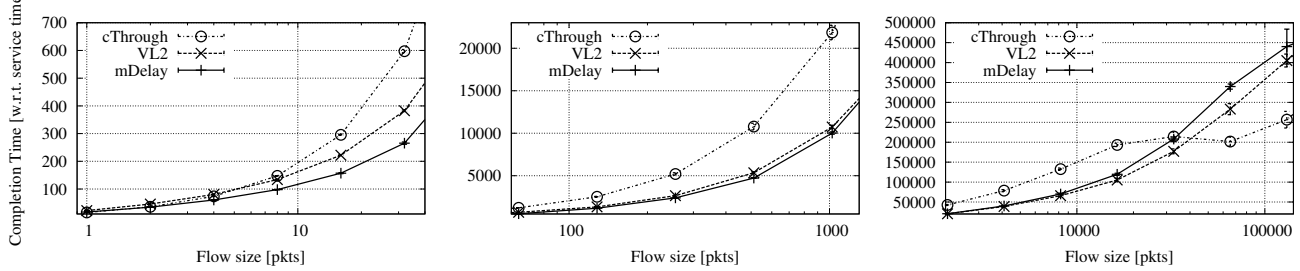
[4]Alternative definitions would take into account other aspects, such as the cost of the architectures, in terms of money necessary to build it, or energy consumption; here we focus on the bandwidth only.



(a) Diagonal Heavy Tails



(b) Diagonal Light Tails



(c) Uniform

Fig. 2. Flow completion time for different topologies and traffic patterns (95% confidence interval).

To this aim, we divide the flows in different bins, according to their sizes: bin i contains the flows with a number of packets between 2^{i-1} and 2^i. As a performance metric, we show the flow completion time, since it captures in a single metric the delay and the throughput – if the delay decreases, the probability for an AIMD source to increment the throughput becomes higher. Figure 2 shows the flow completion time for small, medium and large flows, with different traffic patterns (diagonal with heavy and light tails, and uniform).

The mDelay architecture is able to provide a smaller delay for small and medium flows, while only very large flows are affected (with flows larger than 100 MB, the confidence intervals overlap). It should be noted that the average node throughput remains the same for the VL2 and mDelay architectures, i.e., the data transferred on average is the same.

Figure 2 indicates that the cThrough architecture seems to provide a better throughput for the large flows (flows larger than 50 MB): it is important to note that the figure provide the results for the flows that have successfully transfer the data. The overall throughput is in any case less than the one obtained by the mDelay architecture (see Table I), therefore the actual gain in the average completion time is due to the fact that there are less flows overall that have been transferred using the cThrough architecture.

V. Discussion

The proposed architecture introduces a set of issues, which are in part shared by similar 2-tier architectures. In this section we summarize the main ones that we are studying in detail (not reported here for space constraints).

Scheduling at first tier: The existing solutions for deadline-aware scheduling of flows or scheduling based on priority to mice flows become complex due to a multipaths and tree architecture in standard datacenter designs. Further, the application of such approaches has been limited to hybrid networks due to large switching times of MEMS based optical switches. Our solution which uses a flat topology at the first tier opens up several interesting directions for scheduling policies for latency sensitive flows. As mentioned before, this is due to the fact that each switch has a view of congestion in the paths and the statistics of outgoing flows.

Adaptive threshold: In our solution, we use a threshold T_s for identifying the elephants. In § III-B we suggest that the OpenFlow Control Center periodically collect statistics on the

flow to decide how to set T_s. In order to maximize the use of the resources, it would be interesting to investigate the effect of an adaptive T_s, where the adaptation may be based on the instantaneous load of the fully connected topology.

Dimensioning: How many resources (in terms of bandwidth) should be assigned to the first and the second tier? For the second tier, the main drive could be, as usually done, the bisection bandwidth, or a fraction of it. Instead, for the first tier the dimensioning is still an open problem. Overdimensioning could help in maintaining under control the delay, but it can be a waste or resources, while underdimensioning may result into too small threshold T_s.

Analytical model: The above mentioned problems (scheduling, computation of T_s, dimensioning) can be studied through an analytical model, which should be focused on the main performance index used to drive the design of our solution: the delay of the flows. While the analysis of the throughput is usually more simple to address, the analysis of the delay is particularly challenging.

Alternative approach: Instead of having two separate tiers, an alternative solution would be to have a single packet switched network (e.g., VL2), and control the mice and the elephants with priority queues. The first T_s packets of a flow are enqueued in the high priority queue, while the remaining packets are enqueued in a low priority queue. While this solution would allow for a better exploitation of the resources, the multi-hop path would increase the probability to increase the delay. As a future work we plan to investigate this solution, also with the help of analytical tools.

VI. Conclusion

We have proposed a new hybrid datacenter architecture with emphasis on reducing the delay of small flows. The research community has treated datacenter infrastructure with the aim of excessive bandwidth provisioning cheaply. On the other hand, the existing solutions for reducing latencies of delay sensitive flows ignore the developments in hybrid networks. Our research is a step in direction to bridge this divide. The motivation for our design comes from new proposals of optical network design which allow small flows to be routed through circuit switched networks. We show through preliminary simulations that our design works well and we identify several research directions associated with the problem and our design. As a future step, we plan to explore the several open research directions that we described.

Acknowledgment

The authors would like to thank Gil Zussman, Varun Gupta and Howard Wang, from Columbia University, for the many useful discussions and feedback on the paper.

References

[1] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *Proc. ACM SIGCOMM'09*, 2009.

[2] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, 2009.

[3] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM'10*, 2010.

[4] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, "c-Through: part-time optics in data centers," in *Proc. ACM SIGCOMM'10*, 2010.

[5] N. Farrington, G. Porter, Y. Fainman, G. Papen, and A. Vahdat, "Hunting mice with microsecond circuit switches," in *Proc. ACM HotNets'12*, 2012.

[6] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," *RFC 2992 (Informational)*, 2000.

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *Proc. USENIX NSDI'10*, 2010.

[8] N. McKeown, T. Anderson, H. Balakrishan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, 2008.

[9] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proc. ACM IMC'09*, 2009.

[10] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM IMC'10*, 2010.

[11] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *ACM SIGCOMM'10*, 2010.

[12] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 266–277, 2011.

[13] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 139–150, 2012.

[14] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 50–61, 2011.

[15] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *Proc. USENIX NSDI'12*, 2012.

[16] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: a scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 2009.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[18] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, 2012.

[19] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *Proc. IEEE INFOCOM'11*, 2011.