QoS-aware Multipathing in Datacenters Using Effective Bandwidth Estimation and SDN

Runxin Wang
VMware Inc., Ireland
Email: runxinw@vmware.com

Simone Mangiante
EMC^2, Ireland
Email: simone.mangiante@emc.com

Alan Davy, Lei Shi, Brendan Jennings
Waterford Institute of Technology, Ireland
Email: {adavy, lshi, bjennings}@tssg.org

***Abstract*—Datacenter networks are commonly structured with hierarchical topologies in which multipaths are provided to create redundant paths between end-to-end servers. Without a dedicated flow control mechanism, different sizes of traffic flows are statically allocated to links without sensing the current link utilization, which can result in transient network congestion that impacts on the latency experienced by users of the hosted applications. Existing works aim to realize load balancing by efficiently scheduling flows based on their sizes and link utilization. However, many applications' performance is bounded with specific QoS targets, so load balancing may not directly address their QoS requirements. In this paper, we present a latency-aware flow scheduling system that schedules flows based on a tenant application's QoS target and empirical estimations of the effective bandwidth required to meet these QoS targets. The approach seeks to ensure that, following a network-aware placement of VMs hosting application components, the latency associated with communications between these VMs is controlled even as the inter-VM traffic patterns change, or as the background traffic associated with other applications changes. We describe a prototype implementation of the approach that extends our MAPLE network-aware VM placement system. Our experiments in an emulated datacenter indicate that our scheduling approach leads to fewer QoS violations compared to ECMP.**

I. Introduction

Recent studies indicate that the network is often the bottleneck in the performance of applications hosted in datacenters. Delivering performance guarantees to datacenter-hosted applications is challenging [1], so many cloud providers have resorted to the practice of over-provisioning bandwidth resources in order to prevent violation of QoS guarantees for cloud tenants. Several network-aware resource allocation and VM placement techniques [2]–[4] have been proposed in order to deliver guaranteed performance to datacenter-hosted applications. These proposals aim to allocate network and computing resources to the application VMs to guarantee the satisfaction of their performance targets, while optimizing utilization of compute, storage and network resources. However, given the nature of networking demands that can vary dramatically over time, traffic loads can become unbalanced in a datacenter network, causing performance degradation of applications associated with the congested links while other links having unused bandwidth are not efficiently utilized due to static provisioning schemes.

There is a gap between existing network-aware VM placement techniques and flow scheduling algorithms. Most network-aware VM placement techniques are designed assuming that network resources can be abstracted as a hose model [1], in which an application specifies per-VM bandwidth requirements for all of components deployed in individual VMs. As long as the network requirements of VMs do not exceed the capacity specified, adequate resources can be secured and effectively allocated. Given the multipath technologies commonly used in nowadays datacenters [5], this assumption relies on the presence of a flow scheduler to ensure that every link has equal networking conditions, thus the hose model may not be aware if VMs use the congested links over time. ECMP [6] is the common protocol used in datacenter networks to map flows into multi-equal paths; existing studies [7] have shown its deficiency in the presence of larger flows, which often leads to link congestion onset. On the other hand, most proposed flow scheduling techniques ignore the individual QoS required by each VM—they only ensure that, overall, the flows are evenly allocated to the given links, but they do not directly address the network resources needed by each VM. The aforementioned issue can be usually found in the resource allocation approaches that are based on end host measurement and end host congestion feedback mechanisms [2], [4], [8], as measuring at the end host level cannot accurately estimate the conditions of the associated up links.

To address the issue, we leverage the Software Defined Networking (SDN) paradigm [9] to develop a flow scheduler, termed the MAPLE-Scheduler, which monitors the network changes in switches and dynamically reschedules flows to meet the QoS requirements, while balancing the link utilization across links. This QoS-aware flow scheduler is designed for incorporation with the MAPLE system, a network-aware VM placement system described in [8], [10]. Whilst MAPLE succeeds in finding appropriate, network-aware placements of an application's VMs at the time of initial placement, it does not address the impact of subsequent changes in networking conditions. The addition of the MAPLE-Scheduler seeks to address this shortcoming by harnessing SDN to adjust flow routing in order to avoid QoS violations due to increased latency caused by datacenter link congestion. Crucially, the MAPLE-Scheduler utilizes an empirical effective bandwidth estimation technique [11] to assess whether flows need to be re-routed in order to meet QoS targets. We carried out experiments on a testbed using 4 servers to emulate a multipath datacenter with up to 210 VM nodes. We show that the MAPLE-Scheduler can ensure the QoS targets are met for the VM requests in the presence of cross-competing traffic, which is not the case for an approach based on the use of ECMP.

The paper is structured as follows. In §II we review some related work on flow scheduling techniques in datacenters.

978-3-901882-85-2

The concept of effective bandwidth applied to manage QoS requirement is introduced in §III. In §IV and §V the MAPLE-Scheduler is described and its flow scheduling algorithm is specified. In §VI, MAPLE-Scheduler is evaluated based on a testbed constructed using 4 physical servers. Finally, §VII provides concluding remarks.

II. Related Work

In recent years SDN, albeit a relatively new paradigm, has been deeply investigated and already deployed in datacenters [12] and wide area networks [13]. However there are still some research challenges to be addressed. Several proposals target innovation in datacenter network design, QoS improvement and traffic engineering by leveraging SDN concepts.

In [14] a deep analysis of multipath routing outlines the weakness of ECMP and encourages more advanced packet-based routing mechanisms in datacenter networks. In the literature there are many proposals implementing such advanced mechanisms; we provide here an overview of the most significant proposals for such advanced mechanisms that embody SDN approaches using both centralized and distributed architectures.

MicroTE [15] proposes a fine-grained traffic engineering system using OpenFlow [16] and a centralized network controller, but it requires minor modifications to end hosts. Hedera [7] implements an adaptive flow scheduling system for multi-stage commodity switching fabric and unmodified hosts, leveraging OpenFlow and a centralized controller. In contrast, the focus of our work is to develop an alternative solution to static load-balancing methods (e.g., ECMP) for multipath tree topologies towards an efficient usage of network resources, concentrating on QoS target satisfaction. Other approaches try to improve QoS in OpenFlow networks by extending the OpenFlow protocol to support rate limiters, packet schedulers and more efficient actions. Particularly interesting is how DevoFlow [17] achieves a scalable flow management: it keeps flows in the data plane as much as possible to avoid overheads in the controller, but it maintains enough visibility over network flows to provide effective aggregated flow statistics. It uses a trigger-based flow statistic gathering mechanism which inspired us to design an efficient link monitoring component. Presto [18] pushes the load-balancing functionality to the edge nodes, requiring modifications to edge switches (implemented using Open vSwitch [19]).

Many researchers have been exploring distributed solutions to QoS issues in networks. We highlight DIFANE [20], one of the first contributions to this area, and the more recent Conga [21], which proposes a congestion-aware load balancing mechanism for datacenters using custom ASICs inside networking fabric. It is optimized for a 2-tier Clos topology (also called Leaf-Spine) typically deployed in enterprise datacenters. Unlike our approach operating at flow level, it uses a novel mechanism to convey path-wise congestion metrics between pairs of leaf switches operating on flowlets to achieve a higher granularity of control. Recently the flow balancing technique in [22] operates on newly introduced coflows.

Our proposed solution addresses the problem of meeting QoS targets in a multipath datacenter controlled by a logically centralized SDN controller. This work is an enhancement of our previous work MAPLE [8], [10], which is a network-aware VM ensemble placement system that uses empirical estimations of the effective bandwidth required between servers to ensure that QoS violations are within targets specified in the SLAs for tenant applications.

III. Effective Bandwidth in multipath topologies

Effective bandwidth (EB) is the minimum amount of bandwidth required by a traffic source (for example, a VM hosting an application component) to maintain a specified QoS target. In a communications network, the EB of a traffic source depends not only on the traffic generation behavior of that source, but also on the whole set of systems, including link capacities, background traffic characteristics and the desired QoS target [23]. EB can be analytically estimated through the application of large deviation theory [24]. In this work, we apply an empirical approach based on the analysis on traffic traces collected at real time [11]. At predefined intervals, we estimate the EB for a given delay-based QoS target for the aggregated traffic collected on devices.

The previous work MAPLE [8] measures EB for links by relying on dedicated modules installed on end servers to monitor and regulate their traffic. To cope with multipath topologies, EB estimation needs to be revisited in order to take into account that it is practically and computationally hard to measure and run the EB estimation algorithm at every single link in the network. We logically divide the network in two parts—the edge and the core—and accordingly use two different approaches to conduct EB estimation. At the edge of the network, where top-of-rack switches (ToR) are, we apply the precise EB estimation method for every uplink port connecting the ToR switch to the core. Since the ToR switch aggregates traffic coming from the host and the VMs in the same rack, it is practically feasible to make the link selection (flow scheduling) decision informed by an accurate estimation of the EB supported by the ToR switch.

For the switching layers forming the core of the datacenter network, we employ the approach of Effective Bandwidth Coefficient (EBC) [25] to estimate the EB for the observed links. EBC captures the relation between estimated EB and its corresponding mean throughput of a link regarding a specified QoS target. Let $R_{\text{eff},i}$ represent the EB of a given link i, mean_i is the current mean throughput of the link, the corresponding EBC can be calculated as:

$$\text{EBC}_i = \frac{R_{\text{eff},i}}{\text{mean}_i} \tag{1}$$

The EBC can be calculated offline from collected packet traces and then used to estimate the EB required by a given link for a given QoS target (see [25] for a full description of this process), as long as the throughput of the given link is known.

IV. MAPLE-Scheduler

The MAPLE-Scheduler is a flow scheduling system that monitors flows' QoS performance, and dynamically reschedules flows, aiming to maintain their QoS needs. We assume that ECMP is used as the default function to map new flows to equal paths, whilst the MAPLE-Scheduler monitors the network status and provides flow rescheduling when it is

needed. This means that after the initial VM placement, ECMP will manage routing new flows from source VMs to destination VMs, when the flows first arrive to the switches and no associated flow rules can be applied. If existing flow rules can be applied, the switch will apply them to route the flows. Meanwhile, the MAPLE-Scheduler monitors the QoS changing over time and reschedules some flows to mitigate QoS degradation. This rescheduling essentially means that some higher bandwidth flows that are contributing to congestion on a link are rescheduled to another link with higher residual bandwidth. This rescheduling is controlled such that flows are re-routed through links with sufficient residual bandwidth to meet their EB requirements and in a manner that ensures that there is sufficient capacity on the original link to meet the EB requirements of those flows that were not reschedule.The key components of MAPLE-Scheduler are:

- A centralized controller built on SDN technologies (for our prototype the Ryu controller is used);
- A flow scheduling algorithm making schedule decisions based on a combination of different measurement mechanisms at the edge and at the core of the network, both relying on EB estimations;
- Distributed monitoring agents residing on servers and edge switches (which do not require any modifications to hardware or protocols in the network devices).

The MAPLE-Scheduler supports fat tree like topologies, which are typically used in datacenter networks [5]. In theory, it can be extended to support any kind of multipath network topology, as long as the requirements for network to be composed of OpenFlow enabled switches and for traffic being captured and analyzed are met. The MAPLE-Scheduler leverages the separation of control plane and data plane, as stated by the SDN paradigm: switches only forward packets according to the rules installed by the logically centralized controller. The MAPLE-Scheduler is always aware of the network topology and of the resources consumed by each running flow, by using its distributed monitoring module.

The MAPLE-Scheduler achieves end-to-end dynamic QoS-aware switching by performing a two stage flow scheduling, as shown in Figure 1. This figure abstracts datacenter network into two layers: the edge layer and core layer. It schedules flows based on the availability of sufficient residual bandwidth, calculated based on estimations of EB. At the edge layer, effective bandwidths are estimated based on captured traces, in which case the residual bandwidth is calculated using Eq. 2. For other switches, the MAPLE-Scheduler first queries the flow statistics on switches using the OpenFlow library, and obtains the EBC from the MAPLE system. Once all the required values are available, the residual bandwidth is calculated using Eq. 3:

$$\text{residual\_BW}_{\text{edge}} = \text{link\_capacity} - \text{effective\_BW} \quad (2)$$

$$\text{residual\_BW}_{\text{core}} = \text{link\_capacity} - \hat{\text{EBC}} \times \text{mean} \quad (3)$$

The different approaches for the edge and the core mean that we need only deploy EB agents on edge switches to capture traffic in order to calculate the EBs. For the other switches that are not associated with EB agents, the EBs are estimated

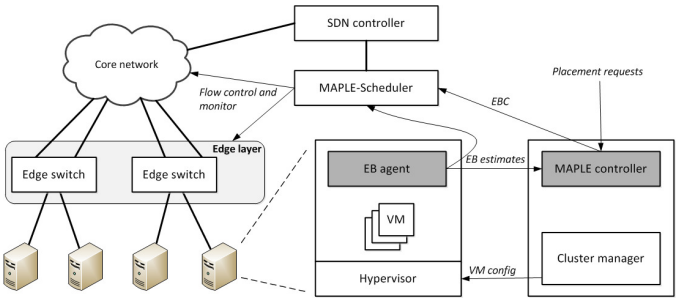


Fig. 1: The logical architecture of MAPLE-Scheduler and MAPLE system.

by using Eq. 1, where the EBC is learned from previous traces, and the mean throughput can be queried from the flow statistics available to SDN controller. We deploy EB agents on Edge switches to calculate the corresponding effective bandwidths in order to obtain more accurate estimations, as they provide first accessing points to the network, which are required to be managed more accurately. A similar design can be found in the Conga architecture [21]. The module performing EBC estimation is deployed on the MAPLE controller, which implements the techniques described in [25]. It communicates with switches to collect the related datasets and then perform the related training described in [25] to obtain EBC regarding different QoS and applications.

The flow statistics of edge switches are learned by the distributed monitoring agents who capture traffic packets and perform the corresponding analysis. For the other network information, MAPLE-Scheduler builds the related information by using the OpenFlow protocol. Generally there are two types of methods, namely *pull based* and *push based*, for SDN controllers to gather flow statistics from the OpenFlow switches. In the proposal of DevoFlow [17], the technical comparison of these two approaches is discussed. In this work we implemented a trigger based push method for gathering flow statistics, with the aim to reduce the work loads and enable the controller to be timely responsive to state changes in switches. Basically, trigger based method means that the OpenFlow switches update statistics to the central controller only if the switches determined it is necessary to do so. The related function is implemented on the monitoring agents, which monitor if any large flows appear on the monitored switches and if any QoS violations occur. The monitoring agents will notify the controller to consider rescheduling flows if it observes QoS violation. In other words MAPLE-Scheduler will be only activated for scheduling large flows when QoS violations occur.

V. MAPLE Flow Scheduling

The flow scheduling algorithm is designed to work on fat tree topologies, but it is topology agnostic as long as the given topology has a hierarchical tree like structure.

The MAPLE-Scheduler works in parallel with ECMP; it only reschedules large flows with the aim to maintain QoS targets. It reschedules a flow based on two conditions: 1) if it will not exceed the selected link's residual bandwidth, based

Algorithm 1 MAPLE Flow Scheduling Algorithm

Input: $Links$, $Capacity$, α, β
Output: updated links

1: **sort** $\uparrow Links$ based on the remaining bandwidths
2: $i = 0$,
3: $j = Links.length$
4: **for** i in $[0, Links.length)$ **do**
5: $ALink = Links[i]$
6: $BLink = Links[j]$
7: **if** $ALink.remainBW > \beta * Capacity$ **then**
8: return
9: **end if**
10: **for** $flow$ in $ALink.flows$ **do**
11: **if** $flow.rate < \alpha * capacity$ **then**
12: continue
13: **end if**
14: $x = ALink.remainBW - flow.rate$
15: $y = BLink.remainBW + flow.rate$
16: **if** $x < y$ **then**
17: move flow to BLink
18: BLink.remainBW = y
19: break
20: **end if**
21: **end for**
22: $j = j - 1$
23: **end for**

on the EB estimations; and 2) that max-min fairness is met—max-min fairness is applied to achieve fair loads of traffic distributed across a multipath network. By meeting these two conditions, this algorithm in turns provides two advantages over ECMP alone: better QoS and better load balancing. When these two conditions conflict with each other, we choose to meet the QoS requirement over load balancing. Note that, given all the feasible flow schedules, we consider a flow schedule achieve max-min fairness by maximizing the residual bandwidth of the link that has minimum residual bandwidth. Note that the MAPLE-Scheduler algorithm only considers moving large flows, where we apply the definition of large flow from Hedera [7], i.e. a flow is large flow if it consumes more than $\alpha = 10\%$ of its hosting link's capacity.

Algorithm 1 presents the pseudo code of the MAPLE flow scheduling algorithm. The input is a list of links on the switch providing equal paths to flows, with the assumption that all links have the same capacity. Algorithm 1 loops through the flows on each link to check if it is suitable to move a given flow from the current link to its equal links.

Line 1 sorts all the links in an increasing order, thus subsequently the links with less residual bandwidth (denoted as remainBW) will be processed first. Line 2 sets $i = 0$ to select flows to be rescheduled from the link with smallest residual bandwidth at first, while in line 3 setting j to $Links.length$, attempting to find a feasible place to move in the selected flows into the link that has largest residual bandwidth. $LinkA$ represents the link that has flows to be moved out, and $LinkB$ represents the link attempted to be moved in flows.

Line 7 makes a condition to stop the algorithm earlier, if the given link has remaining bandwidth greater than $\beta\%$ of the link capacity (we use $\beta = 30$ in our experiments). The algorithm can return at this point as all the rest links will only have greater (or at least equal) remaining bandwidths. This is an optional condition that operators can change based on their need, our justification is that it is barely necessary to reschedule any flows for a link that still has $\beta\%$ capacity left, considering that the possibility of this link to be filled up immediately is low enough (e.g., a large flow generally consumes 10% capacity, and only 20% flows are large flows).

Line 14–16 present the condition that a flow should be moved from its current link to an equal link. This condition require a flow schedule should meet the max-min fairness. Line 18 updates a link's residual bandwidth, once it is determined to host a new flow. Thus in a later iteration, it will not be incorrectly added in another flow. However, we do not update the order of the sorted flow list, the iteration will just need to continue on until all the observed flows are revisited.

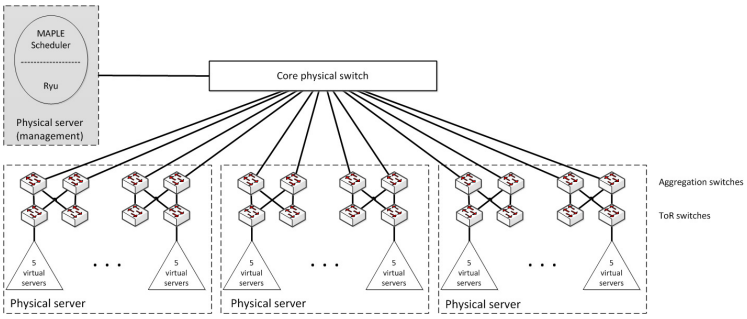


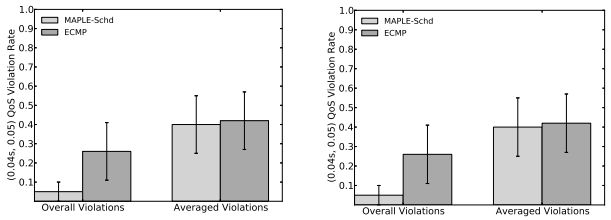
Fig. 2: The emulated datacenter has a single root switch connecting 4 physical servers. One server is used as the management server; the rest servers hosting 210 VMs that are connected by 2 levels of vSwitches.

VI. Evaluation

A. Experimental Setup

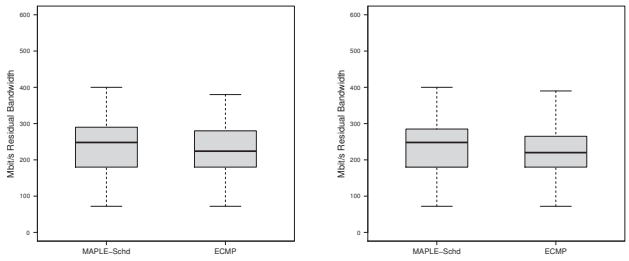
We emulated a fat tree datacenter network using 4 physical servers, as shown in Fig. 2. One server is reserved as the management server where the SDN controller and MAPLE-Scheduler are deployed and connected to a separate management subnet, while each of the rest servers virtualizes 2 pods in a fat tree topology. Switches within the pods are Open vSwitch instances with 1 Gbps upstream and downstream capacity. Each Virtual Switch (vSwitch) is connected to our MAPLE scheduler that is implemented based on the Ryu framework. Each server runs Ubuntu 12.04 as the host operating system, with the `libvirt` library being used to manage up to 70 guest VMs deployed on each physical server. Among these VMs, we group 5 VMs together to form a virtual server (vServer), wherein all VMs are directly connected to a vSwitch, to emulate the scenarios that 5 VMs share one bottleneck link within a physical server.

To emulate bulk data transfers within the network, we used the following data applications: 1) a Hadoop application that runs word count on distributed documents; 2) a data serving application that combines YCSB (data client) and Cassandra (data server), a benchmark recommended in cloudsuite [26]; finally, 3) a custom data backup application, created using the SCP utility, which simulates the scenario where a data file is



(a) (b)

Fig. 3: Given the QoS target (0.02s, 0.1) and (0.04s, 0.05), MAPLE-Scheduler generally meets the QoS targets, although averaged violation rates represent that there are some links still suffered QoS violations. On contrast, ECMP cannot meet the QoS target at all.



(a) (b)

Fig. 4: MAPLE-Scheduler achieves to meet the QoS target as discussed previously, while also delivers slightly better throughput comparing to ECMP.

copied to a number of remote nodes for backup purposes. The applications are placed into the testbed with their corresponding VMs by the MAPLE VM placement algorithm, according to some pre-defined VM placement requests. To simulate large flows that effectively change the QoS performance, we run iperf on a dedicated VM to some randomly selected VMs (this generates iperf sessions lasting 10 seconds), since there are not many flows having size much larger than the others given that all applications have similar configurations.

B. The Analysis of QoS Violations

We analyze QoS violations for the flows generated over the emulated datacenter (EDC). The QoS violation rate is calculated based on individual server's egress links. Namely, the violation rate is the proportion of a server's outgoing packets that experience delays longer than that specified in the QoS target. We employ two metrics: 1) the *overall violation rate* as defined in Eq. 4, which measures the overall performance; and 2) the *averaged violation rate* as defined in Eq. 5, which measures how strong the violations are on affected links.

$$\text{overall\_violation\_rate} = \frac{\text{sum(QoS\_violation\_rates)}}{|\text{all\_links}|} \quad (4)$$

$$\text{averaged\_violation\_rate} = \frac{\text{sum(QoS\_violation\_rates)}}{|\text{links\_with\_violations}|} \quad (5)$$

Fig. 3a depicts the QoS violation results incurred respectively by MAPLE-Schduler and ECMP for the QoS target (0.02s, 0.1). Initially, VMs are placed in the EDC based on the QoS requirement and MAPLE could achieve the QoS guarantee. However, once we turned on the cross traffic generation program, MAPLE using ECMP suffered increasing packet delays, as shown in Fig. 3a, leading to an overall violation rate up to $\approx 35\%$, exceeding the requirement that only 10% can be delayed more than 0.02s. On the contrary, MAPLE-Scheduler once detected the QoS degradations, it managed to reschedule the flows to keep overall violation rates within the targets. The difference is also evident in the averaged violation rates, where MAPLE using ECMP shows *slightly* higher violation rates on the affected links.

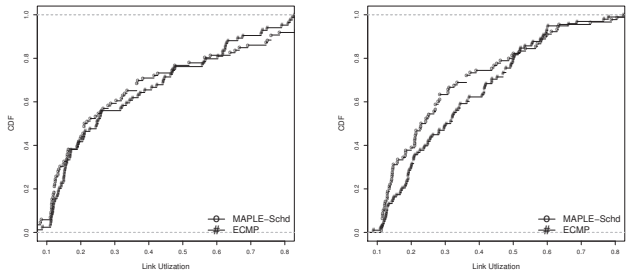
Fig. 3b presents the second set of experiments comparing MAPLE-Scheduler and ECMP based on a new QoS target (0.04s, 0.05). This set of experiments show similar results, MAPLE-Scheduler is able to keep the overall violation rates within required range, where ECMP overall has approximately 30% traffic suffered violations. Regarding the links that do suffer violations, their averaged rates come close, yet MAPLE-Scheduler still has *slightly* smaller violation rates. Comparing to the previous Fig. 3a, all the involved techniques received relatively less violation samples, though which is mainly due to the relaxed delay requirement.

C. Analysis of Throughput Performance and Link Utilization

To evaluate the efficiency of MAPLE-Scheduler's flow scheduling, we analyze the link utilization and the throughput performance of the servers based on the servers' egress traffic. Note that MAPLE-Scheduler reschedules flows in order to maintain the delay QoS targets, delivering better throughput or better link utilization is not the main goal.

Fig. 4a presents the throughput performance for the test cases related to first QoS target (0.02s, 0.1). In general, MAPLE-Scheduler shows significantly better throughput performance comparing to ECMP. Though in terms of the minimum throughput, there is no difference, they all down to 80 Mbps, while the median throughput for MAPLE-Scheduler is approximately 260 Mbps and for ECMP is approximately 240 Mbps. The CDF of mean link utilization levels averaged across the duration of an experimental run, presented in Fig. 5a, indicates that ECMP also shows similar link utilizations, yet MAPLE-Scheduler shows slightly lighter loads on its links.

Fig. 4b presents the throughput performance for the test cases related to first QoS target (0.04s, 0.05). Similar to previous experiments, MAPLE using MAPLE-Scheduler shows significantly better throughput performance. The minimum throughput is consistent, in that for all cases they all go down to approximately 80 Mbps. In Fig. 4b, the difference in median throughput is larger, MAPLE-Scheduler produced approximately 260 Mbps as the median, while MAPLE using ECMP produced approximately 220 Mbps. The CDF of mean link utilization levels, averaged across the duration of an experimental run, presented in Fig. 5b, shows that a relatively bigger gap in the utilization rates up to 40%, regarding MAPLE-Scheduler, there is about 70% of links with utilization rates smaller than 40%, while regarding ECMP, there is about 59% of links having utilization rates smaller than 40%.



(a) (b)

Fig. 5: In the test cases based on QoS target (0.02s, 0.1), two comparing methods show similar link utilization; while in the test cases based on QoS target (0.04s, 0.05), the gap between two lines on the range of link utilization between 0.2 to 0.4 that indicates MAPLE-Scheduler achieve lighter loads across its links, comparing to ECMP.

VII. Conclusion

This proposed work addresses the issue that the QoS and network aware VM placement system such as MAPLE has no control to the VMs' flows after initial placement of those VMs. Common load balancing mechanism (e.g., ECMP) that performs static mapping of flows to multi equal paths does not consider link utilization status and QoS requirements, where large flows can collide on same paths, making network links congested and not suitable for latency sensitive flows. In order to take into account of QoS targets, we designed MAPLE-Scheduler built on SDN controller, which can effectively schedule flows to enforce QoS guarantee in multipathing datacenters. In the experiments based on our SDN testbed consisting of 6 pods of multipaths, we show that, when large flows arrive into the networks, MAPLE using ECMP cannot maintain the delay requirement, while MAPLE with MAPLE-Scheduler can effectively reschedule some appropriate flows to keep the delay performance within the QoS targets.

References

[1] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 44–48, 2012.

[2] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: incorporating time-varying network reservations in data centers," in *Proc. ACM SIGCOMM*, 2012, pp. 199–210.

[3] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "Elasticswitch: practical work-conserving bandwidth guarantees for cloud computing," in *Proc. ACM SIGCOMM*, 2013, pp. 351–362.

[4] K. LaCurts, D. Shuo, A. Goyal, and H. Balakrishnan, "Choreo: Network-aware task placement for cloud applications," in *Proc. ACM IMC*, 2013, pp. 191–204.

[5] N. Bitar, S. Gringeri, and T. Xia, "Technologies and protocols for data center and cloud networking," *IEEE Comm. Mag.*, vol. 51, no. 9, pp. 24–31, 2013.

[6] C. Hopps, "Analysis of an equal-cost multi-path algorithm," Internet Requests for Comments, RFC 2992, November 2000.

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *Proc. NSDI*, 2010, pp. 11–19.

[8] R. Wang, J. Araujo Wickboldt, R. Pereira Esteves, L. Shi, B. Jennings, and L. Zambenedetti Granville, "Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in atacenters," *IEEE Transactions on Network and Service Management*, 2016.

[9] D. Kreutz, F. M. V. Ramos, P. J. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: {A} Comprehensive Survey," *Proceedings of the {IEEE}*, vol. 103, no. 1, pp. 14–76, 2015.

[10] R. Wang, R. Esteves, L. Shi, J. Wickboldt, B. Jennings, and L. Granville, "Network-aware placement of virtual machine ensembles using effective bandwidth estimation," in *Proc. IFIP/IEEE CNSM*, 2014, pp. 100–108.

[11] A. Davy, D. Botvich, and B. Jennings, "Revenue optimized iptv admission control using empirical effective bandwidth estimation," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 599–611, 2008.

[12] S. Jain, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, A. Vahdat, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, and J. Zhou, "B4," *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13*, p. 3, 2013.

[13] C. Hong, S. Kandula, and R. Mahajan, "Achieving high utilization with software-driven WAN," *Sigcomm 2014*, pp. 15–26, 2013.

[14] N. Chrysos, M. Gusat, F. Neeser, C. Minkenberg, W. Denzel, and C. Basso, "High Performance Multipath Routing for Datacenters," in *Proceedings of IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*, 2014, pp. 70–75.

[15] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: fine grained traffic engineering for data centers," *ACM CoNEXT*, 2011.

[16] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[17] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM*, 2011, pp. 254–265.

[18] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based Load Balancing for Fast Datacenter Networks," *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*, pp. 465–478, 2015.

[19] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," in *Proc. ACM HotNets Workshop*, 2009.

[20] M. Yu, J. Rexford, M. Freedman, and J, "Scalable flow-based networking with DIFANE," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, 2010, pp. 351–362.

[21] M. Alizadeh, N. Yadav, G. Varghese, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, and R. Pan, "Conga," *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*, pp. 503–514, 2014.

[22] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with Varys," *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*, pp. 443–454, 2014.

[23] F. Kelly, "Notes on effective bandwidths," *Stochastic Networks: Theory and Applications*, vol. 4, pp. 141–168, 1996.

[24] C. Courcoubetis, V. A. Siris, and G. D. Stamoulis, "Application and evaluation of large deviation techniques for traffic engineering in broadband networks," in *Proc. ACM SIGMETRICS*, 1998, pp. 212–221.

[25] A. Davy, D. Botvich, and B. Jennings, "On the use of accounting data for qos-aware ip network planning," in *Managing Traffic Performance in Converged Networks*, 2007, ch. 33, pp. 348–360.

[26] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *Proc. ACM ASPLOS*, 2012, pp. 37–48.