MEET: Rack-Level Pooling Based Load Balancing in Datacenter Networks

Jiaqing Dong, Lijuan Tan, Chen Tian, Yuhang Zhou, Yi Wang, Wanchun Dou, and Guihai Chen

Abstract—Datacenter networks enable multiple paths between hosts to provide large bisection bandwidth. It requires load balancers to cope with network uncertainties such as traffic dynamics and topology asymmetry. Existing edge-based load balancing schemes are usually faced with the problem of limited network visibility. This article proposes MEET, a rack-level pooling based load-balancer deployed at the edge that can handle the aforementioned uncertainties. MEET utilizes both passive information as well as active probing to comprehensively sense the network conditions with relatively low cost. MEET dynamically reroutes flows effectively based on the visibility of the network condition. MEET has been tested with extensive flow-level simulations against state-of-the-art load balancers. It outperforms Hermes by up to 10% in the experiments, and outperforms others solutions such as DRILL by up to 50%. MEET requires no modifications to the switches and is feasible to deploy at the edge.

Index Terms—Load balancing, datacenter network

1 INTRODUCTION

DATACENTER networks setup multiple paths between pairs of hosts and balance the traffic among these paths to provide large bisection bandwidth and satisfy the increasing traffic demands of applications such as big-data analytics, web services, and cloud storage. Load balancing schemes play an important role in datacenter networks to reasonably distribute traffic among available paths in response to network uncertainties such as link failure and asymmetry.

However, most datacenters still use Equal-Cost Multi-Path (ECMP) as the default load balancer [24], which randomly selects paths based on the 5-tuple hash value of a flow without considering path conditions. ECMP is widely adopted for its simplicity, but it causes problems such as hash collision and bandwidth waste. To address the shortcomings of ECMP, a lot of alternative load balancing schemes have been proposed [1], [13], [22], [28], [38], [44]. Most of the switch-based solutions such as LetFlow [43], DRILL [16], and CONGA [2], have deployment limitations as they require specific switch chips or modified switches, while existing edge-based solutions [22], [28], [30] lacks network awareness to make accurate rerouting decisions.

In order to support the large-scale expansion of applications and data, two opposite trends of work are ongoing at the same time. One is to do more things at the switch. Manufacturers such as Cisco [11], Arista [4] and Barefoot [37] are now manufacturing programmable switches supporting languages like P4 [9]. The other is to move unnecessary or complex functions from core network to the edge. For instance, NetFATE [34] introduces an open framework to enable network functions at the edge. [12] presents a container-based platform that can bring network functions to the network edge. Clean [26] proposes to control the switch queue length at the access point of the network. The core network switch retains only the basic forwarding functions, with the complex network functions being moved to the edge of the network. Essentially, datacenter network will become simple network controlled by intelligent edges, with switches implementing simple static routing strategies. This work presents a load balancing scheme that follows the second trend. In terms of load balancing, the rapid development of smart or advanced network interface cards(smart NICs) at the edge enables end-hosts to participate in the routing construction and load balancing decisions in datacenter networks.

Edge-based load balancing schemes are usually faced with the problem of limited network visibility compared with switch-based solutions, where switches can quickly and timely obtain the global path congestion information and make a better routing decision afterwards. For example, CLOVE [30] and Hermes [45] are the two representative edge-based load balancing solutions. CLOVE is with limited capability to detect network congestion and update the path congestion signals slowly. Hermes can respond to congestion in a shorter time through proactive congestion detection strategies and cautious

- *Jiaqing Dong is with the State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China. E-mail: jiaqing.dong@outlook.com.*
- *Lijuan Tan, Chen Tian, Yuhang Zhou, Wanchun Dou, and Guihai Chen are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China. E-mail: {854038422, 1223870886}@qq.com, {tianchen, douwc, gchen}@nju.edu.cn.*
- *Yi Wang is with the Sustech Institute of Future Networks, Southern University of Science and Technology, Shenzhen 518055, China, and also with the Pengcheng Lab, Shenzhen 518066, China. E-mail: wy@ieee.org.*

Manuscript received 1 Feb. 2021; revised 13 Mar. 2022; accepted 21 Mar. 2022. Date of publication 25 Mar. 2022; date of current version 11 July 2022. This work was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101390001, in part the National Natural Science Foundation of China under Grants 62072228 and 61971382, in part by the Fundamental Research Funds for the Central Universities, in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization, and in part by Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program.
(Corresponding author: Chen Tian.)
Recommended for acceptance by S. Chen.
Digital Object Identifier no. 10.1109/TPDS.2022.3162297

TABLE 1
Comparison of Existing Load Balancing Schemes and MEET

Scheme	Minimum routing unit	Deployment location	Congestion-aware	Advanced hardware
ECMP [24]	flow	switch	✗	✗
Hedera [1]	flow	edge switche	✔	✗
MPTCP [39]	flow	host	✔	✗
FlowBender [28]	flow	host	✔	✗
CLOVE [30]	flowlet	software edge switch	✔	✗
CONGA [2]	flowlet	edge switch	✔	✔
HULA [31]	flowlet	edge switch	✔	✔
DRB [10]	packet	host	✗	✗
Presto [22]	flowcell	edge switch	✗	✗
LetFlow [43]	flowlet	switch	✗	✔
DRILL [16]	packet	switch	✔	✔
Hermes [45]	packet	host	✔	✗
MEET (Ours)	flowlet	host	✔	✗

rerouting. However, as each host in Hermes only receives partial congestion information from the arriving packets, routing decisions based on the limited local path congestion information cannot be accurate. Both of the solutions suffer drawbacks caused by limited network visibility.

Given the above observation, we ask the following question: *can we design an edge-based load balancing solution that can obtain global path information and gracefully balance the traffic in response to the network uncertainties?* In this paper, we propose MEET to answer this question affirmatively.

MEET is a load balancing solution that is deployed at the edge, and obtains global path congestion information through in-rack cooperation. The core idea of MEET is to obtain a better network visibility through converging path congestion information received by all the hosts under the same rack together with a proactive detection mechanism. MEET derives the path congestion status from round-trip-time (RTT) of packets. In MEET, hosts under the same rack send the received path congestion information to a central host under the current rack, so that the central host will have congestion information of all the paths related to that rack. In addition, the central host actively probes the network condition as a complement under the guidance of the power of two choices technique [36], which can effectively increase the sensing scope at minimal probing cost. The central host will broadcast the accumulated path condition to other hosts under the same rack, which then make rerouting decisions accordingly. As discussed in Section 5, MEET is limited by the inherent shortcomings of edge-based load balancing schemes and cannot handle tiny flows and bursts in time. We leave the improvement of MEET for tiny flows and bursts as a future work.

MEET is a pure edge-based load balancing solution, which is feasible to deploy and requires no switch modifications. The edge-based distributed approach that MEET adopts makes it have good scalability without compromising the global view of the network. It has been tested extensively in large-scale simulations with realistic web-search [3] and data-mining [18] workloads. The experiment results demonstrate that MEET handles network congestion and uncertainties well for most popular workloads in both symmetric and asymmetric topology.

2 BACKGROUND AND CHALLENGES

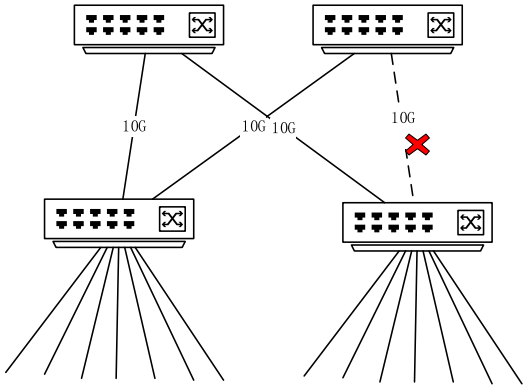
This section introduces the background about the trend of edge-based solutions for datacenter network. Then we describe the challenges we meet in designing an edge-based load balancer for datacenter network.

2.1 Marginal Deployment

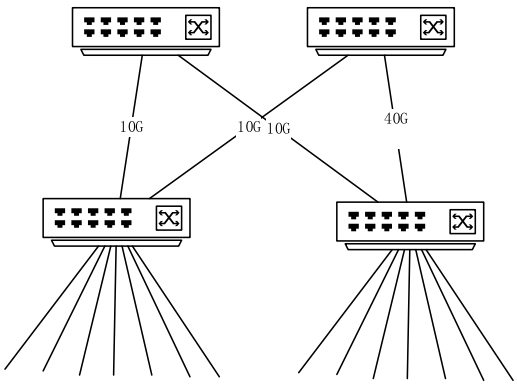
Driven by business and applications, a new wave of network demand is coming toward IT infrastructure technology. Through research on enterprises and cloud service providers, it is not difficult to know that the network construction of next-generation datacenter should cater to the following trends: modularization, standardization and simplification. It means that the complex network functions will be stripped from the core network and transferred to the edge, which is conducive to enhancing the scalability of the network. At the same time, there is a question, whether the edge has the ability to handle the complex network function? The rapid development of smart NICs provides an affirmative answer. With the help of smart NICs, edges will be able to participate in handling complex network functions and realize many advantages of software-defined networking (SDN) and network function virtualization (NFV). Removing network load balancing and other low-level functions from the server CPU will ensure maximum processing power for applications. At the same time, the smart NICs can also provide distributed computing resources, so that users can develop their own software or provide access services, thereby accelerating specific applications. Smart NICs also ensure that the edge has the ability to participate in datacenter load balancing decisions. However, Table 1 shows that most of existing load balancing schemes were deployed on switches. Even if some of them were deployed on the end, they are not all congestion-aware.

2.2 Complex Network Status

Traffic Dynamics. Traffic of datacenter network is constantly changing [6], [8], [10], [17], [33], [46]. The occurrence of congestion is uncertain, in the datacenter. Load balancing schemes in datacenter network should be able to cope with the dynamic traffic. CLOVE passively deals with congestion, and adjusts the path weights only when congestion has occurred. The consequence of passive response is that the congestion cannot be mitigated in a timely manner. It may happen that the congestion has been over before the



(a) Link failure.



(b) Different link capacity.

Fig. 1. Two asymmetric topologies.

countermeasures are taken, and it cannot respond to the rapid and dynamic traffic in datacenter network in time.

Asymmetry. Asymmetry is not uncommon in datacenter network, where multiple reasons such as as link failure, switch malfunction and heterogeneous devices (different link bandwidth and different number of ports) can lead to network asymmetry [17], [19]. This causes the bandwidth of multiple paths between pairs of hosts to be different. Fig. 1 shows two typical asymmetric topologies. The asymmetry in Fig. 1a is caused by link failure, while asymmetry in Fig. 1b is resulted from the heterogeneous link capacity. A load balancer should deal with these asymmetry gracefully or severe congestion will occur.

In a network with the symmetric topology, it is enough for a load balancing scheme to evenly distribute traffic among all available paths to achieve good results. For instance, Presto [22] divide flows into datacells with the same size (64KB) which called flowcell [22], and then evenly distribute the flowcells to different paths by polling. However, it is not the case in a network with the asymmetric topology in that the bandwidth among different paths may be different, even uniform distribution will cause congestion and waste of bandwidth. In an asymmetric network, there are problems such as slow response and slow convergence. Letflow [43] splits flows into flowlets [29] with different unit sizes. However, the scheme cannot respond to path dynamics timely as it does not adopt a proactive pairing strategy, and passively adjusts only after congestion occurs.

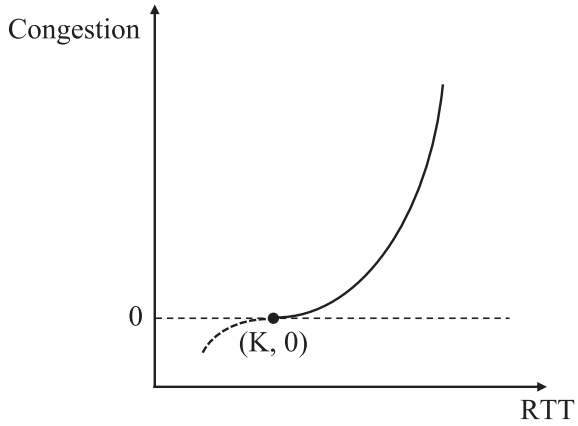


Fig. 2. The cubic curve of congestion value.

2.3 Limited Visibility

Comprehensive visibility of network conditions is important for load balancers to make decisions. However, edges of datacenter networks can only get relatively limited information about the network condition. Round-Trip Time (RTT) and Explicit Congesion Notification (ECN) are two of the most essential signals about network congestion status that edges can obtain. RTT is the length of time it takes for a signal to be sent plus the time it takes for an acknowledgement of that signal to be received. Both propagation time and queueing delay of the paths between the two communication endpoints are included in RTT, which can directly reflect the path congestion. Intuitively, if most of the data packets passing through a path have a large RTT, that path is likely under congestion. ECN enables end-to-end congestion notification between two endpoints over TCP/IP based networks. When ECN is successfully negotiated, an ECN-enabled switch will set a mark in the IP header instead of dropping a packet in order to signal impending congestion. ECN indicates the congestion of data packets queued in the switch buffer. However, ECN will suffer from errors due to limited sample size and other reasons.

In contrast, load balancers deployed on the switch is able to obtain the information of the data packets passing through the switch globally. For example, the swith-based load balancing solution, CONGA [2], is able to reroute flows to paths with smallest congestion level derived from measured rate of each path. However, for edge-based load balancing schemes, decisions based on limited information will introduce errors.

3 System Design

This section introduces the detailed design of Meet. Limited visibility of the network condition together with network uncertainties in datacenter network restricts the effectiveness of edge-based load balancing schemes. In order to overcome the limitations, Meet adopts a rack-level pooling based solution to monitor the network status in real time and reroute flows accordingly.

3.1 Congestion Awareness From RTT

RTT of a packet is affected by several factors, such as network congestion, number of hops to the destination, and network stack delay of end hosts. It can be regarded as a

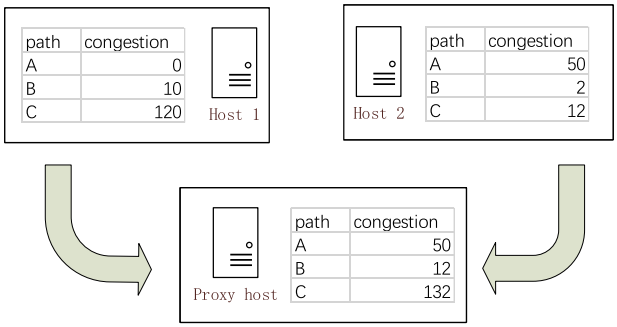


Fig. 3. Examples of congestion value aggregation.

semaphore which intuitively expresses the path congestion. However, the RTT of a single packet cannot express the path congestion deterministically. The intuition is that if the RTTs of most packets on a path is larger than normal, the corresponding path is likely to be congested. This observation inspires us to take advantage of accumulated RTTs of packets to derive the network congestion status.

We define the congestion value C derived from the RTT of one packet as follows:

$$C = (t - K)^3, \tag{1}$$

where t is the measured RTT of the packet, and K is a threshold close to the base RTT of the path.

The figure of this equation is a cubic curve as shown in Fig. 2. The philosophy behind this equation is that if the measured RTT is near the threshold K, then the path can be considered as non-congested and the calculated congestion value will be very small. When the path encounters congestion, the measured RTT will grow and the congestion value will grow fast correspondingly.

Algorithm 1. RTT Converted to Congestion Value

Input: RTT value of each arriving packet
Output: Congestion value
1: **if** $RTT < K$ **then**
2: congestion $= 0$
3: **else**
4: congestion $= (RTT - K)^3$
5: **end if**

The detailed algorithm is listed as Algorithm 1. If the measured RTT is close to K, it indicates that the path is slightly congested and thus the voting weight will be small. When the measured RTT grows far larger than K, it indicates that the path is congested, and the voting weight should be large.

3.2 Rack-Level Pooling of Network Visibility

After converting RTTs of incoming data packets into the form of congestion value, the congestion value of each path will be gathered. The accumulated congestion value will be normalized and used as a metric to represent the congestion level of the corresponding path. The philosophy of this accumulative method is that the determination of a path's congestion is equivalent to a democratic voting, where flows vote with RTTs of data packets. 007 [5] shares a similar but different idea. Its main idea is that failed links are travelled by flows with different paths containing that link. Weight of a flow is evenly distributed to the links of that flow, so that failed links are likely to have higher aggregated weight(votes). 007 allocates weight to each link of a path to identify and locate failed links, while MEET allocate congestion value as weight to packets to rank paths with different congestion level. Packets experiencing longer RTT are given higher weight to vote the path with larger congestion value. More data packets with larger RTTs results in larger cumulative value, indicating a more congested path.

Algorithm 2. Aggregating Congestion Values in a Rack

Input: Congestion value per packet
Output: Gongestion value of each path
1: **for** each $packet$ on each $path$ **do**
2: Get C(congestion) by Algorithm 1
3: $C_{total}[path] += congestion$
4: $P_{total}[path] += 1$
5: **end for**
6: **for** each $path$ **do**
7: $C[path] = C_{total}[path]/P_{total}[path]$
8: **end for**

The method of accumulating congestion values requires a sufficient amount of data so that such congestion values are more accurate. It is not enough to only depend on the data volume of one host. MEET requires hosts under a same rack to collaborate with each other in order to achieve more comprehensive data for a better visibility of paths related to hosts from that rack. Specifically, one host is selected as a proxy under each rack, while all the other hosts transfer the accumulated congestion value to that proxy host. In this way, the proxy host under each rack will have the aggregated congestion value of all paths related to the traffic from that rack. Consequently, the determination of the path congestion status will be more accurate with more comprehensive data. In order to reduce the additional bandwidth overhead for transmitting data to the proxy host, MEET let each host accumulate the current congestion value first, and then transmit the cumulative value to the proxy, which will do a global accumulation. In this way, additional bandwidth consumption for data aggregation can be reduced.

Fig. 3 illustrates an example of congestion value aggregation. Host1 and Host2 locate in the same rack and record the congestion value of three paths A, B and C respectively. At this time, the records are separated among different hosts(i.e., Host1 & Host2). In order to make the network more visible, both Host1 and Host2 will send the congestion information to the proxy host. Then the proxy host accumulates the collected congestion values. The proxy host will get the congestion status of all paths through normalizing the accumulated congestion value as in Line 7, Algorithm 2. A path with larger normalized congestion value is with a higher probability of path congestion.

Congestion Awareness Aging/Updating. Algorithm 1 is executed periodically at the interval of T_{update} at each host in order to keep a fresh view of the network. Algorithm 2 adopts the form of sliding window. Only the latest congestion data from one host will be aggregated, and out-dated data will be cleared. The proxy host also broadcasts the

latest network view to other hosts at the interval of T_{update}. We suggest to set T_{update} to around tens of cross-rack RTT.

It's worth noting that intra-rack information exchange introduces an additional latency, denoted as $T_{latency}$. We define the total delay of the calculated network status as T_{delay}. The calculated congestion status can be regarded as a snapshot of the network T_{delay} ago. Here we have $T_{delay} = T_{update} + T_{latency}$. As $T_{latency}$ is relatively small compared to T_{update}, its impact on the total delay is insignificant. In addition, the total delay T_{delay} can be adjusted through changing the value of T_{update}.

3.3 Active Congestion Detection

Through accumulated congestion value, the proxy host is able to have more clear sight of all available paths. Nevertheless, it is still possible that the proxy host has biased perception of congestion status of paths or is blind to some network failures. On the one hand, when the load of network is low, the amount of RTT data is small, which will introduce inaccurate congestion calculation. On the other hand, passive congestion awareness cannot handle asymmetry caused by link failures, in which case no data packets pass through the failed path so that the end host has no way to know whether the link has failed. To overcome this problem,Meet adopts an active congestion detection mechanism. Regularly and proactively sending probe packets to all paths can considerably improve the network visibility. However, probing packets to all paths in the network will introduce considerable bandwidth overhead. In order to maximize the detection range at the minimum cost, Meet solves this problem under the guidance of the well-known power-of-two-choices technique [36]. Each host periodically probes the network according to their destination racks. Specifically, each host selects three paths for every destination rack of its current flows to send probe packets: the path with the smallest aggregated congestion value, the best path detected last time, and a randomly selected path. If there are uncovered links, the third path will choose paths with uncovered links with higher priority. The host then selects the path with the best probing result for its flows. In this way, Meet is able to effectively handle asymmetry and gets a better scope of sensing at minimal probing cost.

3.4 Routing

Meet adopts flowlet as the smallest granularity for routing. Flowlets [41], [43] are bursts of packets of the same flow that are sufficiently apart in time so that they can be sent on different paths. When the gap between two packets in the flow exceeds threshold, two flowlets will be generated. The gap between data packets reflects the congestion of the network, and allows the flow to be distributed on each path of the network with a reasonable granularity. In addition, flowlet-based rerouting can solve the problem of out-of-order packets at the receiver side.

Idealy, TCP sends packets smoothly and the packets are sent at the same time interval before. However, in real-world implementation, TCP tends to send packets in one or a few clustered bursts. This is because TCP sends packets together in a window unit, and all packets in that window will be sent together when receiving ACK. This will create gaps between bursts and flowlets will be generated. When a new flowlet is generated, Meet selects a new path for that flow based on current perception.

Meet adopts XPath [25] for explicit routing. XPath is readily-deployable at commodity switches and enables edge hosts in Meet to explicitly choose a path between hosts with a path ID [25]. Implementation details of XPath is out of scope for this paper.

3.5 Parameter Settings

There are some important parameters in Meet, which play an important role in the system. In this section, we discuss the impacts of these parameters on the routing performance and how we choose the parameter values. The objective is to find a balance between system performance and resource consumption.

First, we consider the parameter K used for converting RTT to congestion value. The parameter K is used as a threshold to judge whether a path is congested and how congested that path is. If the parameter K is set too large, then the algorithm will not be able to distinguish congestion. In the meanwhile, if K is too small, all packets will be regarded as suffered congestion. In Meet, we suggest to set K to $20us$ plus base RTT so that lightly loaded paths will be regarded as non-congested ones.

Second, we consider the parameter T_{update} and T_{probe}. The parameter T_{update} is the interval that each host sends cumulative congestion values to the proxy host and the interval that the proxy host broadcasts the aggregated path status to each host. The parameter T_{probe} controls how often the probe module of Meet sends probing packets. It is supposed to be equal to T_{update} so that the probing results can be synchronized with the aggregated congestion value. The setting of the T_{update} parameter requires a trade-off between bandwidth consumption and timeliness. If the T_{update} is small, the path congestion status will be sent to the agent host at a short interval and the local path knowledge of the agent host will be updated in a timely manner, which is conducive to making routing decisions that requires fresh path congestion status. However, frequent update of path congestion status will introduce a lot of bandwidth consumption. We suggest T_{update} be set to tens of the cross-rack RTT, which brings acceptable freshness and additional in-rack bandwidth overhead. Taking a 2-tier clos-based topology with 8 spine switches and 8 leaf switches as an example. In this topology, each host has congestion value of 64 paths to send to the proxy host. The congestion value of each path is composed with 8 bytes(a float number and an integer). If the host send the congestion value of all 64 paths every 100us, the additional in-rack bandwidth overhead for that host will be around $40Mbps$, which is negligible in that in-rack link bandwidth is usually around tens of Gbps.

Third, the flowlet timeout $T_{flowlet}$. $T_{flowlet}$ is an important parameter used as an interval threshold for judging new flowlets between bursts. If this parameter is set very small, a flow will likely to be broken into more small flowlets, and the load balancer will change the path of that flow more frequently, resulting in out of order packets. If it is too large, the load balancer will not be able to capture the interval between bursts and eventually flowlet will be the same as a

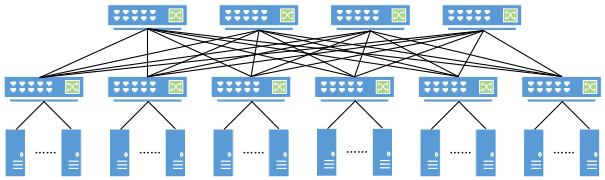


Fig. 4. Clos topology.

flow. In MEET, the parameter $T_{flowlet}$ is set according to the recommendations in [43].

4 EVALUATION

This section evaluates MEET's performance against other schemes with large-scale ns-3 simulations.

Topology. In the simulation, MEET adopts a 2-tier clos-based topology, which has been widely used in modern datacenter networks. Fig. 4 illustrates a typical network with clos-based topology. The topology we use in the simulation has 8 spine switches connected to 8 leaf switches. Each of the leaf switches is connected to 16 hosts. All hosts and switches are connected with 10Gbps links. Given that, the leaf-level switch has a 2:1 oversubscription and there are eight disjoint paths between each pair of hosts from different racks. For each flow, the traffic starts from the host and arrives at the leaf switch first. Then the leaf switch routes packets of the flow to specific spine switches according to load balancing strategies. Paths from the spine switch to the destination host are deterministic for each flow in this topology.

Parameters. The basic RTT in the experiment is $40us$, and we set the parameter $K = 60us$. Considering probing and updating frequency, we decide to update the view of the network around every 10 RTTs. We set $T_{update} = T_{probe} = 500us$. The parameter $T_{flowlet}$ is set to $500us$ according to the recommendation.

Workloads. The simulation adopts three widely-used realistic workload patterns observed in modern datacenters. The first one, websearch, is captured from production clusters for web search services [3]. The second one, datamining, comes from a large cluster for data mining services [18]. To better understand the performance of MEET under burst scenario, we adopt the third workload, which we name as enterprise. The enterprise workload is from a typical large enterprise datacenter [2]. For the enterprise workload, as 50% of the flows are smaller than 75 bytes, 90% of the flows are smaller than 350 bytes and 99% of the flows are smaller than 10K bytes, bursts are more common than the other two workloads under the same network load.

Fig. 5 illustrates the flow size distributions of these workloads. As we can see from the figure, all these three workloads are heavy-tailed. Most of the flows are small while a small fraction of the flows account for most of the traffic. During the simulation, we generate Poisson-distributed flows according to the CDF distribution of each workload. Senders and receivers are randomly selected from different racks to ensure that all traffic passes through the core network.

Transport. DCTCP is used as the default transport layer protocol in the evaluation. We implement DCTCP on top of ns-3's TCP New Reno, with pameters set as suggested in [3].

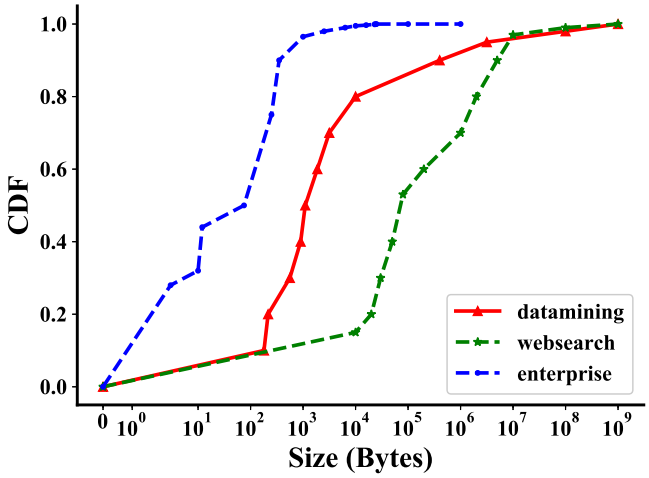


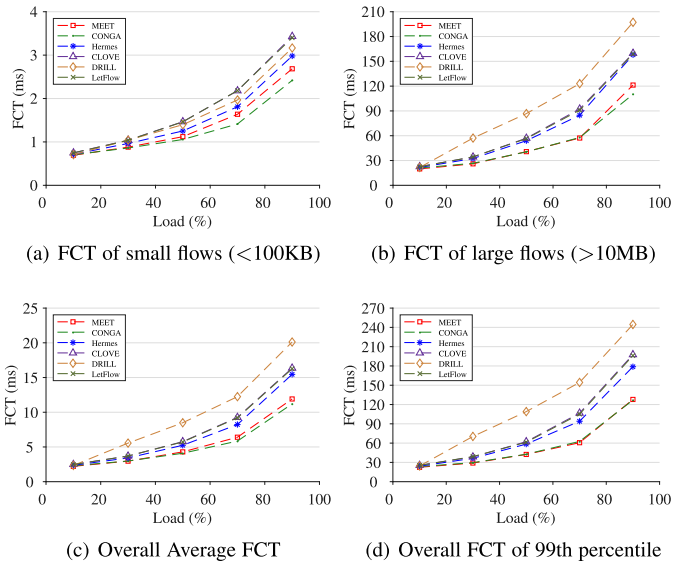
Fig. 5. Traffic patterns used for evaluation.

Schemes Compared. In the simulation, we compare MEET against five other load balancing schemes: Hermes [45], CLOVE [30], LetFlow [43], DRILL [16], and CONGA [2]. Among these schemes, Hermes and CLOVE are edge-based solutions, while others are deployed on the switch. We use the open-source code from [45] for these load balancers. The parameters of these schemes are set as suggested in corresponding papers.

Metrics. Flow completion time (FCT) is used as the primary performance metric. Specifically, to better access the performance of each solution, we use small flow (<100KB) FCT, large flow (>10MB) FCT, overall average FCT and 99th percentile FCT under different network pressure as indicators to evaluate the performance of MEET. For the enterprise workload, as 99% of the flows are smaller than 10K bytes, we use 200 bytes flow (<200B) FCT, 800 bytes flow (<800B) FCT, 3200 bytes flow (<3.2KB) FCT, 12800 bytes flow (<12.8KB) FCT alternatively. The results are calculated by averaging the results of 10 rounds of experiments.

Summary of Results. For the websearch and datamining workloads, MEET works well on both symmetric and asymmetric topologies. Specifically, for the symmetric topology, MEET outperforms Hermes in terms of small flow (<100KB), large flow (>10MB) FCT, overall average FCT and tail flow FCT. As an edge-based scheme, MEET has limited visibility to the network and responds to congestion more slowly compared to CONGA, which is a switch-based solution and can monitor the flow in real time and respond quickly. However, it demonstrates a comparable performance as CONGA in symmetric topology. For the asymmetric topology, MEET performs as good as and sometimes even better than Hermes, considering that it requires a single node to collect less information from the network. MEET performs better than CONGA in that MEET can achieve a better view of the network due to its active detection mechanism.

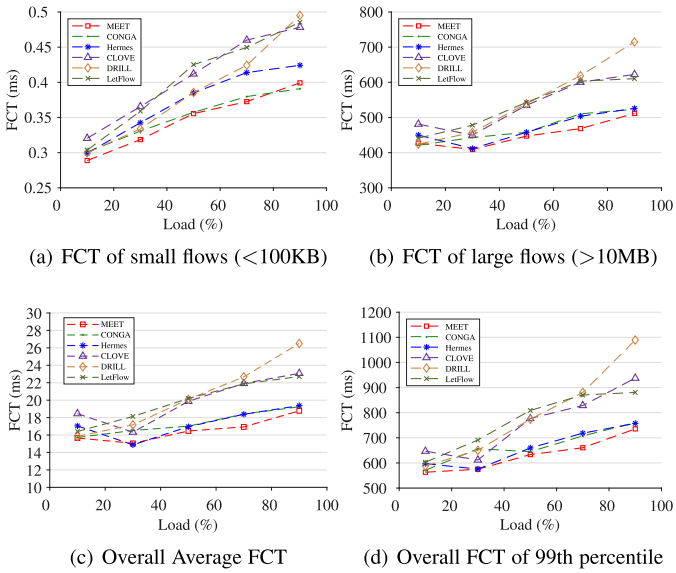
From the aspect of traffic pattern, MEET performs good in both of the representative workloads (websearch and datamining) collected from real-world datacenters. The datamining workload represents the work mode where more short flows exist in the network, while the websearch workload corresponds to the situation where long flows account



(a) FCT of small flows (<100KB) (b) FCT of large flows (>10MB)

(c) Overall Average FCT (d) Overall FCT of 99th percentile

Fig. 6. FCT of websearch workload in symmetric topology.



(a) FCT of small flows (<100KB) (b) FCT of large flows (>10MB)

(c) Overall Average FCT (d) Overall FCT of 99th percentile

Fig. 7. FCT of datamining workload in symmetric topology.

for the majority. The results illustrate that MEET can cope with various workload patterns in the network.

In terms of the enterprise workload, where tiny flows dominate the network and bursts are common, the results are different. In this scenario, MEET has similar performance with Hermes. The switch-based solution, CONGA, outperforms all others in both symmetric and asymmetric topologies.

4.1 Symmetric Topology

Websearch Workload. We first compare MEET with other schemes using websearch workload in symmetric topology. Fig. 6 shows the results.

For flows smaller than 100KB, Fig. 6a shows that MEET outperforms Hermes by around 10%. In this experiment, CONGA is about 13% better than MEET.

For flows larger than 10MB, the results in Fig. 6b illustrate that MEET performs as good as CONGA, which requires switch modifications, and outperforms 25% better than Hermes, and achieves 35% $\sim$ 50% better performance than other solutions.

From Fig. 6c we can see that in this experiment setting, the average FCT of MEET is always within 9% of CONGA. MEET outperforms Hermes by up to 23%, and is 25% $\sim$ 50% better than the other solutions.

Fig. 6d shows the results of 99th-percentile FCT. The result demonstrates that MEET reduces the FCT by around 22% compared to Hermes and is slightly better than CONGA by around 1%. Compared against other schemes, MEET achieves a lower FCT by around 23% $\sim$ 49%.

Datamining Workload. Fig. 7 shows the experiment results of datamining workload under symmetric topology.

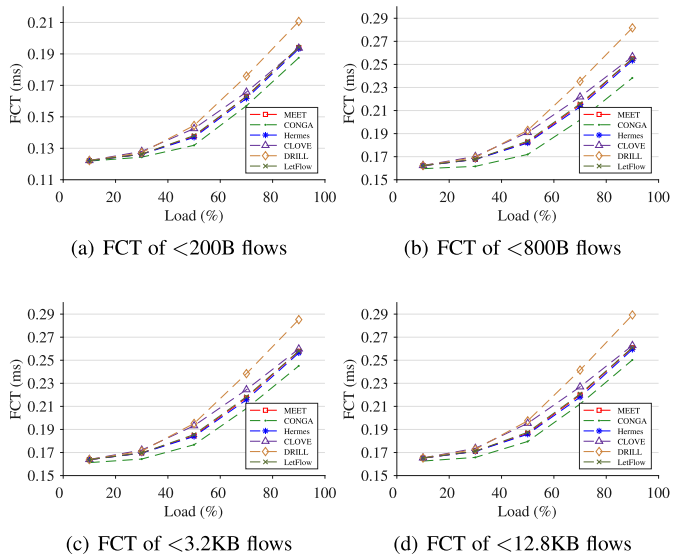
Fig. 7a illustrates that for small flows in this workload, MEET is only slightly worse than CONGA when the network load is very high (90%). It outperforms CONGA by 1% $\sim$ 8% in other cases and is better than all other solutions. Compared to Hermes, MEET achieves up to 9% smaller FCT.

The results of the large flow, 99th percentile and overall average experiments are similar. For these experiments, Hermes achieves comparable performance as CONGA. MEET performs better than Hermes by around 1% $\sim$ 8%.

Enterprise Workload. Large amount of very small flows dominates the network in the enterprise workload. Fig. 8 illustrates the experiment results. These figures shows that MEET does not perform well in this scenario, where bursts are common. Both MEET and Hermes perform worse than CONGA by around 6%.

Analysis. The congestion information aggregation mechanism is simpler than the congestion detection in Hermes and enables MEET to have a better visibility into the network. As a result, MEET outperforms Hermes for both workloads in the symmetric topology in the evaluation.

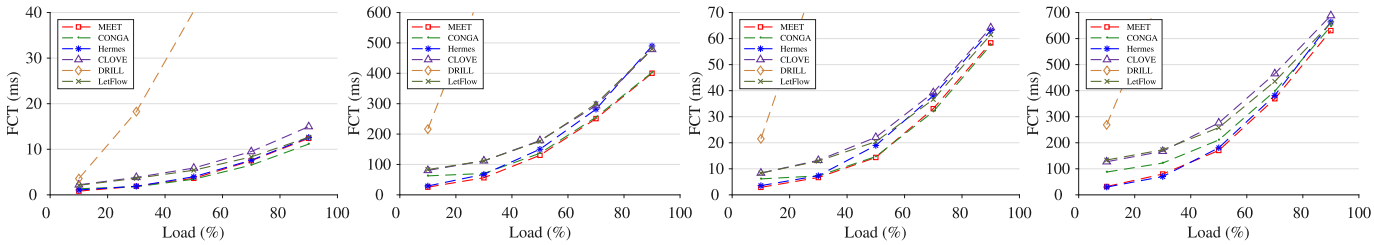
CONGA detects the network status by monitoring packets at the switch in real time, which is a big advantage over MEET. Nevertheless, the performance of MEET is almost the same as CONGA for the websearch workload. On the one hand, the majority of websearch traffic is large flow, which has a long transmission time and leaves enough response



(a) FCT of <200B flows (b) FCT of <800B flows

(c) FCT of <3.2KB flows (d) FCT of <12.8KB flows

Fig. 8. FCT of enterprise workload in symmetric topology.



(a) Small Flows (<100KB) avg. FCT (b) Large Flows (>10MB) avg (c) Overall Average FCT (d) Overall 99-th percentile

Fig. 9. FCT statistics for the websearch workload with the asymmetric topology.

time for the load balancer to take effect and select better paths for the flow. On the other hand, MEET has a proactive detection module, which actively gathers the path status and learns the path information in advance, rather than waiting for the congestion to occur and passively respond.

For the congestion-agnostic schemes, the FCT is very high. DRILL does not perform well in that it makes routing decisions based on the buffer queue length of the local switch, which only represents the congestion status of current node and cannot reflect the overall congestion in a clos-based topology where flows have to pass through several switches. CLOVE adjusts routing decisions based on ECN feedbacks, which only represents the queuing status of one hop in the path and is not enough to reveal the congestion level of the path. Letflow is not aware of congestion and routes packets over randomly selected switch ports.

However, for the workload where tiny flows dominate the network and bursts are common, MEET does not perform well. This is because even though MEET achieves better view of the network through rack-level pooling and active detection, it cannot respond to the very fast burst of tiny flows. From the figure we can see that the average FCT is smaller than 300us, which is even smaller than the probing and updating interval.

4.2 Asymmetric Topology

Then we compare MEET with the other schemes under asymmetric topology. The asymmetric topology is created upon the symmetric topology by reducing the capacity of randomly selected leaf-spine links from 10Gbps to 1Gbps. The randomly selected 'failure' links account for 20% $\sim$ 30% of all leaf-spine links. The random capacity reduction simulates the uncertainty of failure in real datacenter network.

Websearch Workload. We first compare MEET with the other schemes with websearch workload. Fig. 9 shows the evaluation results.

Fig. 9a shows that for flows smaller than 100KB, MEET has very close performance with Hermes. CONGA outperforms MEET by around 9%.

In terms of flows larger than 10MB, the result tells that MEET is better than Hermes. When the network load is around 10%, MEET and Hermes have similar performance and is better than CONGA. As the network load rises, the performance of CONGA gets better. MEET has very close performance with CONGA when the network load is around 90%. The results of the overall average experiment is similar to results of the large flow experiment.

Fig. 9d shows the result of the 99th-percentile experiment. We can see that MEET always works better than CONGA at this percentile. In terms of Hermes, MEET is better than Hermes when network load is above 40%.

Datamining Workload. Fig. 10 shows the results of the experiment where we evaluate MEET with datamining workload under asymmetric topology. In summary, MEET and Hermes both perform better than all the other solutions in this scenario. Furthermore, MEET works better than Hermes in all four experiments.

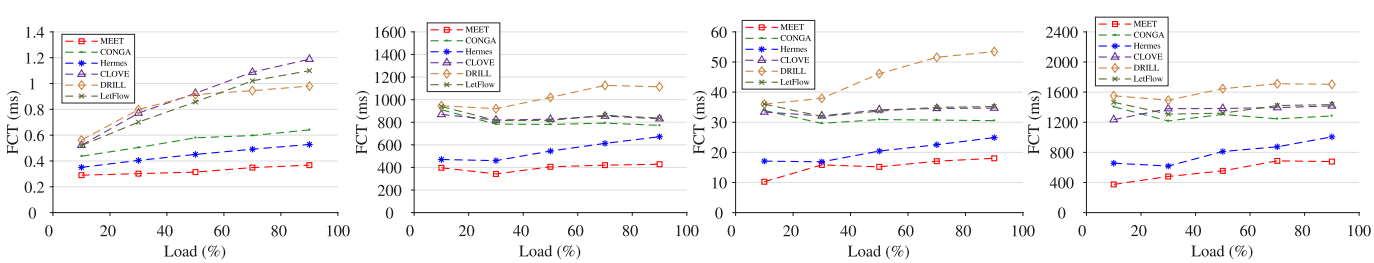
Fig. 10a shows that for flows smaller than 100KB, MEET outperforms Hermes by up to 28%. MEET achieves a 50% better performance than CONGA and outperforms other solutions by 50% $\sim$ 70%.

The other three figures illustrate similar results. MEET outperforms Herms by around 20% and is far better than the other solutions.

Enterprise Workload. Fig. 11 shows the results of enterprise workload under asymmetric topology. Similar as in the symmetric topology, MEET does not perform well in this scenario. CONGA performs best in this case, and outperforms MEET by around 25%.

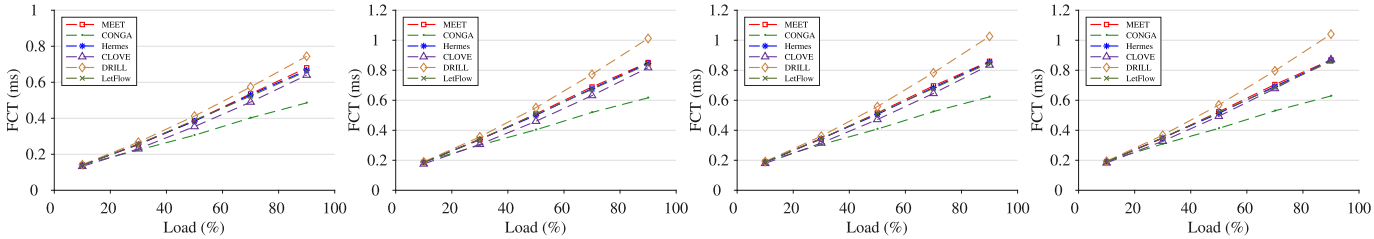
Analysis. From the evaluation results, we can see that for the asymmetric topology, MEET is better than or at least as good as Hermes, irrespective of workload patterns.

MEET adopts a simpler but effective approach to proactively monitor the network congestion. The mechanism



(a) Small Flows (<100KB) avg. FCT (b) Large Flows (>10MB) avg. FCT (c) Overall Average FCT (d) Overall 99-th percentile

Fig. 10. FCT statistics for the datamining workload with the asymmetric topology.



(a) FCT of <200B flows (b) FCT of <800B flows (c) FCT of <3200B flows (d) FCT of <12800B flows

Fig. 11. FCT statistics for the enterprise workload with the asymmetric topology.

Hermes uses to perceive the network congestion is relatively complicated. It involves ECN, RTT and other transport layer semaphores such as timeout retransmission events to determine congestion and path failures. However in MEET, only the RTT of data packets is used perceive the network congestion. The RTTs are conveted to 'congestion value' and then aggregated to get a better view of the network. Rerouting decisions are made based on the aggregated congestion value. Although the mechanism in MEET is simple, it provides a good view of the network and enables MEET to perform even slightly better than Hermes in asymmetric topology. The algorithm converting RTT to congestion value considers both link failure and capacity variation. In terms of asymmetry caused by link failure, failed links will be considered as congested as probing packets passing through the failed links will not get any response back. For asymmetry caused by different link capacities, links with smaller capacity are more likely to be the bottlenecks and get congested. Probing or data packets passing through links with small capacity is more likely to suffer congestion and hence result in larger RTTs. Consequently, MEET copes well with asymmetry caused by various link capacity.

However, MEET cannot perform as well as CONGA for the workload in which bursts caused by large amount of concurrent tiny flows are common. As it cannot respond to the very fast bursts in time due to its delayed accumulating, probing and updating at the edge.

5 DISCUSSIONS

Congestion Control. Congestion control and load balancing are fundamentally different. Load balancers aim at reducing the chance of congestion by spreading traffic accross paths, while congestion control [3], [14], [15], [21], [42] is to help flows converge to fairness when congestion happens. Load balancers cannot completely eliminating congestion. And when congestion happens, congestion control steps in to alleviate congestion and lead to bandwidth allocation converge to fairness share. The proposed edge-based load balancer is orthogonal with these congestion control protocols and can be combined with them.

Single Point Failure. MEET aggregates congestion value from nodes in a rack to the proxy host to get the congestion awareness. This methodology requires that the links between the proxy host and the leaf switch experience no failure, which is not always the fact. Once the other hosts cannot pass the congestion value to the proxy, MEET loses the capability of congestion aggregation, path detection and rerouting. Consens protocols like PAXOS [32] can be a possible solution.

In-Rack Overhead. In order to get a global visibility of the available paths, the proxy in MEET aggregates information (congestion value) from other nodes in the same rack, which introduces additional in-rack bandwidth overhead. More frequent aggregation leads to more instantaneous view of the network status thus is ble to make rerouting decisions timely. There is a trade-off between bandwidth consumption and network visibility. How to intelligently adjust the aggregation frequency to get better view of the network and consumes less bandwidth in MEET remains as a future work.

Tiny Flow and Bursts. Edge-based deployment is an important trend for complex network functions. MEET is readily-deployable at the edge, and requires no modifications to the switches. This advantage comes at the cost that edge-based load balancing schemes cannot detect the network status in real time and respond to congestion as quickly as in-network techniques.

The evaluation results demonstrate that MEET works better on traffic with large flows as majority. Even though equipped with a proactive detection module, MEET is still limited by the inherent disadvantage of edge-based load balancing schemes that it cannot detect the network status in realtime and respond to bursts and congestion timely, which is a must for tiny flows. Fortunately, 20 % of the flows in the network accounts for 80% of the network traffic in datacenter, indicating that the majority of datacenter traffic are large flows. We leave the improvement for MEET to cope with tiny flows and bursts as a future work.

6 RELATED WORK

This section briefly decribes the related works. Several centralized load balancing solutions [7], [20], [23], [27], [38] have proposed, among which Hedera [1] is a representative. Hedera is a passive solution designed to balance heavy traffic in datacenter network. Large flows are detected and allocated to selected paths in advance. However, it cannot cope with small flows in time and the response is slow. AggreFlow [20] is a centralized flow scheduling scheme designed for power-efficient data center networks with OpenFlow [35] controller. It can dynamically schedule flows to achieve high power efficiency and load balancing with improved QoS. This paper mainly focus on distributed load balancing solutions, which can be categorized as congestion-aware and congestion-agnostic.

Congestion-Aware Distributed Solution. CONGA [2] is a distributed load balancing scheme which works on a 2-layer leaf-spine topology. It splits the flow into flowlets and uses the flowlet as the minimum routing unit. CONGA is

deployed on switches to obtain global congestion information between leaf switches and requires specific hardware support.

CLOVE [30] is implemented in software switches. It uses standard ECMP and uses flowlets as the routing unit. Traceroute mechanism is adopted to explore various paths in the network. In order to reduce the cost of exploring paths, CLOVE only re-explores and updates paths every hundreds of milliseconds. As a result, the response to network conditions in asymmetric topology is slow and FCT of both large flow and small flow is not as good as MEET.

Hermes [45] is an edge-based load balancing scheme with global congestion awareness. It uses ECN and RTT to estimate link congestion and relies on timeout and retransmission to detect switch or link failures. However, as each node in Hermes detects three paths to the destination separately, it has insufficient network visibility.

FlowBender [28] is a distributed load balancing solution. It is deployed at the edge and detects congestion based on the ECN feedbacks of the packets. FlowBender modifies the packet header of a flow so that the flow will be routed to another path by the switch with hash-based routing mechanisms. Although FlowBender can change the path of a flow when congestion occurs, it cannot decide which path the flow will be rerouted to. It's possible that a flow will be routed to a more congested path by FlowBender.

HULA [31] is a load balancing scheme that uses programmable switches to achieve global congestion awareness. In a large-scale topology, recording path information requires a lot of memory. To reduce memory overhead, HULA only stores the best next hop. Leaf switches send low-cost probes and broadcast congestion information to other switches to achieve global congestion awareness. HULA requires specific hardware support.

Congestion Agnostic Distributed Solutions. Presto [22] can be deployed at the edge by modifying the system modules. It relies on the edge to transform flows into a large number of near uniformly sized small flowcells and proactively spreads those flowcells over the network in a balanced fashion. Presto is designed for symmetric topology and performs poorly in an asymmetric topology.

LetFlow [43] is a solution deployed on switches that requires specific hardware support. The size of flowlet in LetFlow is positively correlated to the flow rate. Paths with higher speed will have longer flowlet, which results in smaller probability of path resouting. Eventually, from a probabilistic perspective, more packets will be transmitted through faster paths. LetFlow is a passively scheme and cannot respond to congestion in time.

DRILL [16] is a scheme deployed on the switch, using the buffer queue length of the switch to achieve random routing. DRILL is also inspired by the "power of two choices" paradigm. When a packet arrives at the switch, it randomly selects two ports and compares the queue length of the forwarding port recorded last time, and selects the port with the smallest queue length for forwarding. DRILL only selects routes based on local information.

LocalFlow [40] is a switch-based load balancing scheme which ignores the global network congestion and reroutes packets based on local loads. Based on its observation that large flows in the network cannot effectively use multiple paths, LocalFlow proposes to split the flow on the switch and then forward the subflows to different paths. LocalFlow performs good on a symmetric topology. However, it does not perform well for asymmetric topologies.

7 Conclusion and Future Work

In this paper, we propose MEET, a rack-level pooling based load balancing solution deployed at the edge. MEET aggregates congestion knowledge learned by all nodes under the same rack thus to achieve a better visibility of the network with comparatively low cost. MEET also adopts a proactive detection module to further improve the timeliness of congestion sensing. Experiments demonstrate that MEET handles network congestion and uncertainties well for most popular workloads in both symmetric and asymmetric topology. MEET outperforms Hermes by up to 10% in all the experiments, with relatively simpler mechanism. It achieves comparable performance with CONGA in most of the experiments and even outperforms CONGA for the data-mining workload in both symmetric and asymmetric topology. It's worth noting that MEET requires no modifications to the switches, while the switch-based CONGA requires switch modifications.

As an edge-based load balancer, MEET can be implemented in smart NIC in the future to further improve the performance and reduce the CPU overhead. Limited by the inherent disadvantage of edge-based load balancing schemes, currently MEET cannot handle bursts caused by large amount of tiny flows well. We leave this as an important future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedback.

References

[1] M. Al-Fares *et al.*, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. USENIX Conf. Netw. Syst. Des. Implementation*, 2010, pp. 89–92.
[2] M. Alizadeh *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 503–514.
[3] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 63–74.
[4] Arista, "Arista 7170 series," 2020. [Online]. Available: https://www.arista.com/en/products/7170-series
[5] B. Arzani *et al.* "007: Democratically finding the cause of packet drops," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, 2018, pp. 419–435. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/arzani
[6] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.
[7] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th Conf. Emerg. Netw. Experiments Technol.*, 2011, pp. 1–12.
[8] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving failures in bandwidth-constrained datacenters," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 431–442, 2012.
[9] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[10] J. Cao *et al.*, "Per-packet load-balanced, low-latency routing for clos-based data center networks," in *Proc. 9th ACM Conf. Emerg. Netw. Experiments Technol.*, 2013, pp. 49–60.
[11] Cisco, "Cisco Nexus 34180YC and 3464C programmable switches data sheet," 2020. [Online]. Available: https://tinyurl.com/cicsoprog
[12] R. Cziva and D. P. Pezaros, "Container network functions: Bringing NFV to the network edge," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 24–31, Jun. 2017.
[13] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proc. IEEE INFOCOM*, 2013, pp. 2130–2138.
[14] J. Dong *et al.*, "Uranus: Congestion-proportionality among slices based on weighted virtual congestion control," *Comput. Netw.*, vol. 152, pp. 154–166, 2019.
[15] Y. Gao, Y. Yang, T. Chen, J. Zheng, B. Mao, and G. Chen, "DCQCN+: Taming large-scale incast congestion in RDMA over ethernet networks," in *Proc. IEEE 26th Int. Conf. Netw. Protocols*, 2018, pp. 110–120.
[16] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro load balancing for low-latency data center networks," in *Proc. Conf. ACM Special Int. Group Data Commun.*, 2017, pp. 225–238.
[17] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 350–361.
[18] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 51–62.
[19] C. Guo *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proc. ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 139–152.
[20] Z. Guo *et al.*, "AggreFlow: Achieving power efficiency, load balancing, and quality of service in data center networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 17–33, Feb. 2021.
[21] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.
[22] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 465–478, 2015.
[23] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 15–26.
[24] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," *RFC*, vol. 2992, pp. 1–8, 2000.
[25] S. Hu *et al.*, "Explicit path control in commodity data centers: Design and applications," in *Proc. 12th USENIX Symp. Netw. Syst. Des. Implementation*, 2015, pp. 15–28.
[26] X. Huang *et al.*, "Clean: Minimize switch queue length via transparent ECN-proxy in campus networks," in *Proc. IEEE/ACM 29th Int. Symp. Qual. Serv.*, 2021, pp. 1–6.
[27] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 3–14.
[28] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "FlowBender: Flow-level adaptive routing for improved latency and throughput in datacenter networks," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Experiments Technol.*, 2014, pp. 149–160.
[29] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 51–62, 2007.
[30] N. Katta, M. Hira, A. Ghag, C. Kim, I. Keslassy, and J. Rexford, "CLOVE: How I learned to stop worrying about the core and love the edge," in *Proc. 15th ACM Workshop Hot Top. Netw.*, 2016, pp. 155–161.
[31] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proc. Symp. SDN Res.*, 2016, pp. 1–12.
[32] L. Lamport *et al.*, "Paxos made simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 18–25, 2001.
[33] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 399–412.
[34] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene, "An open framework to enable NetFATE (network functions at the edge)," in *Proc. 1st IEEE Conf. Netw. Softwarization*, 2015, pp. 1–6.
[35] N. McKeown *et al.* "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[36] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, Oct. 2001.
[37] B. Networks, "Tofino ASIC," 2020. [Online]. Available: https://tinyurl.com/baretafino
[38] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 307–318.
[39] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 266–277.
[40] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *Proc. 9th ACM Conf. Emerg. Netw. Experiments Technol.*, 2013, pp. 151–162.
[41] S. Sinha, S. Kandula, and D. Katabi, "Harnessing TCP's burstiness with flowlet switching," in *Proc. 3rd ACM Workshop Hot Top. Netw.*, 2004, pp. 67–72.
[42] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protoc. Comput. Commun.*, 2012, pp. 115–126.
[43] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, "Let it flow: Resilient asymmetric load balancing with flowlet switching," in *Proc. 14th USENIX Symp. Netw. Syst. Des. Implementation*, 2017, pp. 407–420.
[44] P. Wang, H. Xu, Z. Niu, D. Han, and Y. Xiong, "Expeditus: Congestion-aware load balancing in clos data center networks," in *Proc. 7th ACM Symp. Cloud Comput.*, 2016, pp. 442–455.
[45] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proc. Conf. ACM Special Int. Group Data Commun.*, 2017, pp. 253–266.
[46] J. Zhou *et al.*, "WCMP: Weighted cost multipathing for improved fairness in data centers," in *Proc. 9th Eur. Conf. Comput. Syst.*, 2014, pp. 1–14.

Jiaqing Dong received the BS degree in computer science and technology from Peking University, Beijing, China, in July 2013, and the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in July 2020. He is currently an assistant researcher with the State Key Laboratory of Media Convergence and Communication, Communication University of China. From December 2020 to July 2021, he worked as a visiting scholar with the Department of Computer Science and Technology, Nanjing University, China. His research interests include data center networks and distributed systems.

Lijuan Tan received the BEng degree from the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. She is currently working toward the master's degree at Nanjing University, Nanjing, China. Her research interests include load balancing in data center networks.

Chen Tian received the BS, MS, and PhD degrees from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2000, 2003, and 2008, respectively. He is a professor with the State Key Laboratory for Novel Software Technology, Nanjing University, China. He was previously an associate professor with the School of Electronics Information and Communications, Huazhong University of Science and Technology, China. From 2012 to 2013, he was a postdoctoral researcher with the Department of Computer Science, Yale University. His research interests include data center networks, network function virtualization, distributed systems, Internet streaming, and urban computing.

Yuhang Zhou received the BS degree in computer science and technology from the Department of Computer Science and Technology, Nanjing University, Nanjing, China. He is currently working toward the master's degree at Nanjing University, Nanjing, China. His research interests include data center networks and distributed systems.

Yi Wang received the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in July 2013. He is a research associate professor with the Sustech Institute of Future Networks, Southern University of Science and Technology. Before joined Sustech, he worked as a senior researcher with the Huawei Future Network Theory Lab, Hong Kong. His research interests include future network architectures, information centric networking, software-defined networks, and the design and implementation of high-performance network devices.

Wanchun Dou received the PhD degree in mechanical and electronic engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2001. He is currently a full professor of the State Key Laboratory for Novel Software Technology, Nanjing University. From April 2005 to June 2005 and from November 2008 to February 2009, he respectively visited the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, as a visiting scholar. Up to now, he has chaired three National Natural Science Foundation of China projects and published more than 60 research papers in international journals and international conferences. His research interests include workflow, cloud computing, and service computing.

Guihai Chen received the BS degree in computer software from Nanjing University, Nanjing, China, in 1984, the ME degree in computer applications from Southeast University, Nanjing, China, in 1987, and the PhD degree in computer science from the University of Hong Kong, Hong Kong, in 1997. He is a distinguished professor of Nanjing University. He had been invited as a visiting professor by Kyushu Institute of Technology in Japan, University of Queensland in Australia and Wayne State University in USA. He has a wide range of research interests with focus on parallel computing, wireless networks, data centers, peer-to-peer computing, high-performance computer architecture, and data engineering. He has published more than 350 peer-reviewed papers, and more than 200 of them are in well-archived international journals such as *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE/ACM Transactions on Networking* and *ACM Transactions on Sensor Networks*, and also in well-known conference proceedings such as HPCA, MOBIHOC, INFOCOM, ICNP, ICDCS, CoNext, and AAAI. He has won nine paper awards including ICNP 2015 Best Paper Award and DASFAA 2017 Best Paper Award.