SprayPass: Multi-path Credit-scheduled Congestion Control for High-speed Data Centers

Shan Huang, Dezun Dong, Zejia Zhou, Xiangke Liao
dept. of Computer Science, National University of Defense Technology
Changsha, China
{huangshang12, dong, zhouzejia, lxk}@nudt.edu.cn

***Abstract*—Nowadays, data centers deploy topologies with high link-speed and abundant equivalent paths, providing good service for global users. However, network congestion inevitably occurs when traffic bursts. In a high-speed environment, reactive congestion control has poor performance managing congestion because of the long responding time. ExpressPass, as an outstanding proactive congestion control scheme, has the merits of fast convergence, low latency, and low buffer occupancy which give it high applicability in current data centers. However, ExpressPass operates without the consideration of utilizing plentiful parallel paths, this affects its performance even with the deployment of ECMP.**

To optimize the end-to-end latency and fully utilize the network resources, this paper proposes SprayPass, a multi-path credit-scheduled proactive congestion control scheme. Based on the credit scheduling mechanism of ExpressPass, SprayPass leverages credit spraying to uniformly balance the traffic on multiple paths and employs sequence-free feedback control to remedy credit out-of-order. Besides, we develop path probing for SprayPass to handle path failures. Compared to ExpressPass, SprayPass reduces average flow completion times to 0.53x in realistic workload, greatly improves the throughput to 1.28x with permutation traffic, and achieves much better fairness in asymmetry network. Moreover, SprayPass maintains the ExpressPass's advantages of bounded queue length and fast convergence.

***Index Terms*—data center networks, congestion control, multi-path, credit**

I. Introduction

Data center inner traffic is heavy and bursty, which is mainly composed of by the short latency-sensitive messages mixed with some large flows [1]. However, current data center switches have very shallow buffers [2], and this typical traffic is likely to cause network congestion when the buffers are overfilled. Once the congestion occurs, data queuing and packet loss will severely affect the flow completion time of small flows and hurt the throughput of large flows, further impact the user's experience. Lots of approaches have been proposed to manage congestion, mainly classified as reactive congestion control and proactive congestion control [3].

Reactive schemes [4]–[7] are designed to control the congestion after it occurs, and they often need at least one Round-Trip-Time (RTT) reacting to congestion. However, link speed inside the data center network is rapidly increasing, from 10 Gbps to 100 Gbps or even faster with low end-to-end base latency (10-100 μs) [8], [9], more flows will be completed within one RTT. Reactive schemes are hard to react to congestion in time in such a high-speed environment, and thus packet loss becomes commonly. To prevent retransmission delay, they resort to lossless flow control to achieve link-layer lossless [10], [11]. However, those lossless flow control mechanisms can result in the problems of tree saturation, deadlock, and Head-of-Line [12]–[14], which sharply degrade the network performance. Consequently, reactive schemes are becoming less efficient in current high-speed data centers. Proactive congestion control schemes are characterized by their high throughput, low latency, and fast convergence, etc., which make them highly applicable in current high-speed data centers [15]. pHost [16] leverages end-to-end token to schedule the packet transmission, which helps it achieve fast convergence and end-to-end lossless. Homa [17] improves the performance of small message transmission in low-latency data center environments and also improves the performance on fair sharing. FastPass [18] is a centralized congestion control protocol, it achieves near-zero queue and high throughput. NDP [19] achieves low-latency and high throughput in different traffic conditions.

In modern data centers that deploy multi-path topologies to provide high bandwidth [20], [21], we still face a great challenge of designing a congestion control achieving high throughput, low latency, and fast convergence. FastPass employs a centralized arbiter to decide sending time and transmission path for each packet, this generates computation overhead and communication overhead which prolong the transmission delay. NDP cannot prevent packet loss in fabric, and it leverages some techniques which are hard to implement in commodity switches. pHost and Homa assume that congestion occurs primarily at host downlinks instead of the core of the network, and packets are discarded in network when congestion occurs. As an outstanding representative, ExpressPass can maintain bounded-queue and achieve lossless without requiring lossless link layers [22], and it can be deployed in current data centers without modifying network hardware. However, ExpressPass is subjected to limited path diversity, merely one single path transport, and cannot efficiently utilize the high bandwidth. Therefore, to make full use of the existing network with high bandwidth, a multi-path low latency proactive congestion control scheme is required.

In this paper, we propose SprayPass, the first multi-path credit reservation protocol. Based on the principle of Equal Cost Multiple Path (ECMP), SprayPass develops a credit spraying mechanism. When enables ECMP in the network, the switches first hash the packet header, and then route the packet according to the hashed value. With the credit spraying, receivers randomly set path-id on the credits head, and the switches will indirectly spray the credits on different paths randomly. In turn, when a credit arrives, the sender will set the same path-id with the credit on the data packet head. By doing this, each data packet will travel through the network on the same path as the corresponding credit.

Intuitively, credit spraying is practical and easy to implement, but actually, it is not trivial to realize. In the credit-scheduling mechanism of ExpressPass, each credit carries a continuous increment sequence number. The consecutiveness of the credit sequence number reflects the credit loss rate, and this is the key to credit feedback control. However, credit spraying may lead to the out-of-order of credits, and it is hard to solve by simply re-ordering (more details in §III). To solve this problem, we develop a sequence-free feedback control mechanism, which is out-of-order tolerable.

Moreover, for a multi-path protocol, there are two significant challenges need to be addressed, i.e. 1) data out-of-order and 2) network asymmetry. SprayPass maintains short bounded queue by credit-scheduling and keeps near-equal queue length among the equivalent paths by uniform credit spraying. This helps achieve near-zero out-of-order degrees in varied circumstances. Given that, we employ a simple reordering mechanism to eliminate the slight disordering problem. Besides, we propose a path probing mechanism to handle network asymmetry caused by path failures.

We develop SprayPass based on the objectives of high applicability in current data centers, i.e. less modification on hardware and visible performance improvement. The key techniques of this SprayPass include (i) credit spraying to break the limitation of ExpressPass to gain maximum network utility, (ii) sequence-free feedback control to avoid the credit out-of-order problem, and (iii) path probing to handle network asymmetry. Our experimental results show SprayPass greatly improves the performance of ExpressPass without hurting its superiorities.

The rest of this paper is organized as follows. §II introduces the key insights and observations that motivate this work. §III detailedly describes the design of SprayPass. §IV shows the experimental results and analyzes the performance of SprayPass. §V discusses future works and limitations of SprayPass. §VI introduces some related works. Finally, §VII concludes the whole paper.

II. Motivation

As a proactive congestion control scheme, ExpressPass is characterized by fast convergence, low end-to-end latency, and bounded queue, which make it high applicability in today's high-speed data centers [22]. ExpressPass leverages a credit-scheduling mechanism to manage traffic. The sender first sends a credit_request to the receiver, and then the receiver continuously sends credits to the sender after receiving the credit_request. Each switch port keeps a credit queue, and

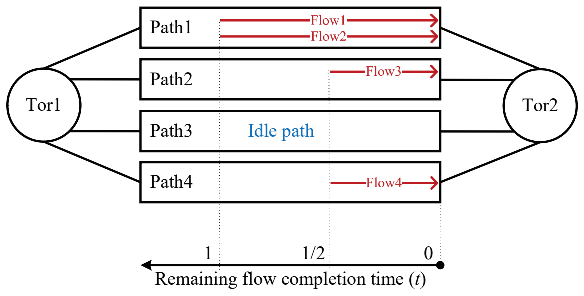


Fig. 1. Traffic by single path transport with ECMP

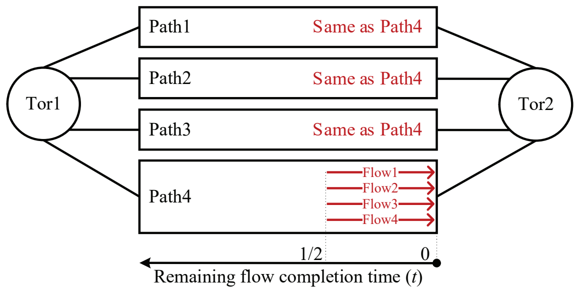


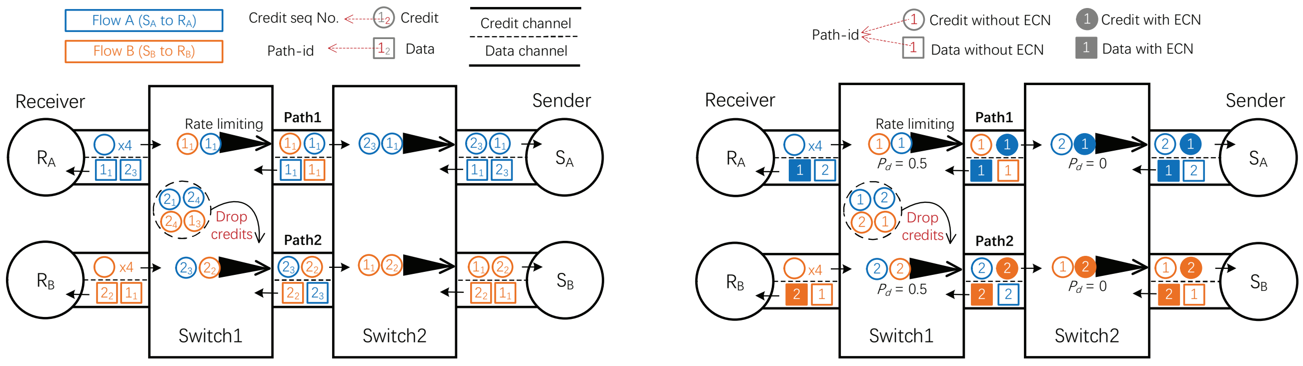
Fig. 2. Uniformly balanced traffic by multi-path transport

credits pass the queue at the limited rate of 5% link speed[1]. In the network, over-speed credits are discarded by the switch and the remaining credits across the network arriving at the sender. When a sender receives a credit, it correspondingly sends a data packet to the receiver. In this way, ExpressPass can guarantee the data not being congested in the network. Besides, by using the efficient credit-scheduled mechanism, ExpressPass can achieve high utilization on a single link.

Current data centers host varies applications, e.g. web search, cache follower, web server, and data mining [23]–[27], inter-rack communications occupy most of the data center traffic [28]. As data centers widely deploy the topologies with multiple parallel paths to achieve high bisection bandwidth, the inter-rack data flows usually have multiple equal-cost paths to travel through. Trying to efficiently use the multiple paths in current data center networks, ExpressPass cooperates with ECMP to balance the bursty traffic. ECMP operates at the granularity of flow, and this will decrease network utilization and increase the flow completion times under a realistic workload which is mixed by mice and elephant flows. Prior work [29] found ECMP will reduce throughput by 40%, because of the ECMP hash collision.

Fig. 1 shows a usual case in real data centers, assuming there are four equal-length flows transmitted through four equal paths. In this case, it would be great if the flows are balanced into four different paths. However, after using ECMP, Flow1 and Flow2 are concurrently routed into Path, so the Flow Completion Time (FCT) of them is prolonged. Besides, we can observe that no flow is transmitted on Path3, to

[1]Each credit is a minimum Ethernet Frame of 84 Bytes, and the corresponding data packet is a maximum size Ethernet frame of 1538 Bytes. To limit the credit rate to $84B/(1538B + 84B) \approx 5\%$ can precisely full use of the link bandwidth



(a) Overview of ExpressPass with credit spraying (b) Overview of SprayPass

Fig. 3. Evolution from credit spraying to SprayPass

balance the Flow1 or Flow2 into it can significantly reduce the overall FCT. In theory, network performance can be obviously improved by uniformly balancing the flows into multiple equivalent paths. As shown in Fig. 2, four flows are equally routed into four paths. Compared to Fig. 1 shows, the average flow completion time of these flows is $(0.5t \cdot 4)/4 = 0.5t$, much better than $(0.5t \cdot 2 + t \cdot 2)/4 = 0.75t$. In this case, credit spraying efficiently improves the performance of ExpressPass.

With the comprehension and insight of the credit-scheduling mechanism, we find that routing credits appropriately is the key to improving the load balancing ability of ExpressPass. ExpressPass employs the symmetric hashing algorithm of ECMP so that data packets are transmitted in the network via the symmetry path of their corresponding credits. For less modification of switches, we focus on host-based routing approaches. Per-flow ECMP decides path based on the hashed value of the flow information, and it has been implemented in the commodity switches [22]. From the above analysis, we can balance credits on idle paths by regulating their ECMP hashed value, thereby guiding data packets to pass through the network faster.

In this paper, we develop SprayPass, a multi-path credit-scheduled congestion control scheme, which greatly improves the ExpressPass without hurting its performance.

III. Design of SprayPass

ExpressPass uses symmetric hashing with deterministic ECMP forwarding to ensure flow level path symmetry, which has been supported by the commodity switches [22]. Similarly, per-packet ECMP can guarantee the path symmetry of data packet and credit packet in the granularity of packet, this gives us inspiration.

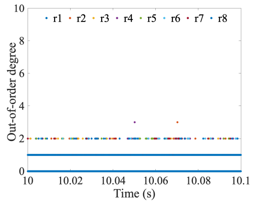
A. Credit Spraying

For simplifying the protocol design, we first raised a question: *can we directly expand the ExpressPass's single-path credit reservation mechanism to multiple paths?* To answer this, we develop credit spraying, which leverages per-packet ECMP to achieve packet-level load balancing. Fig. 3(a) shows the operation process of credit spraying. Receivers randomly set path-id on each credit's header, and the ECMP-enabled switches route the credits according to their path-id. For instance, there are two equal paths in the network, and receivers randomly set the path-id of 1 and 2 on credits. In consequence, credits are equably sprayed into two equal paths, and data packets are uniformly balanced into these paths. Besides, credit spraying has many merits as following.

Uniformly balancing traffic with bounded queue build-up. Based on the operating of per-packet ECMP and symmetric hashing, each data packet travels through the track of corresponding credit. Therefore, data packets are balanced uniformly by randomly spraying credits to all available equal paths, and this greatly improves the utilization compared to ExpressPass. Besides, overspeed credits are discarded in credit queues, this ensures that data injected by the sender within link capacity. Theoretically speaking, no packet will be queued in the network, but there is a small delay between the passing time of credits and data at the same link. This is mainly caused by the propagation delay of links and processing delay of hosts and switches, and they are varied between different flows. Fortunately, the delay difference is bounded as the path length and host processing time are all bounded [22], so that the queue length is bounded.

Keeping low data out-of-order degree. Out-of-order is mainly caused by the delay differences between different paths, including queuing delay, propagation delay, and end-host processing delay. Intuitively, randomly spraying credits packet-by-packet may cause high data out-of-order degree, for the flows are fine-grained broken up and balanced into multiple delay-varied paths. However, in SprayPass, propagation delay and end-host delay are equal for the same flow. Ideally, zero queuing can avoid out-of-order, because packets pass the network without any queuing delay. Given that, credit spraying helps keep a low out-of-order degree, for two reasons: 1) Data queues are bounded to a low length by credit scheduling mechanism, which limits the maximum out-of-order degree; 2) Moreover, queue length in the equal paths are almost the same by using randomly spraying, and this helps keep a low delay difference between different paths.

We build a topology with 4 equal paths, 8 senders and 8 receivers, and each sender sends data to a receiver in a one-one correspondence manner. By doing that, the topology has an over-subscription ratio of 2:1 when all the senders


(a) Data out-of-order degree

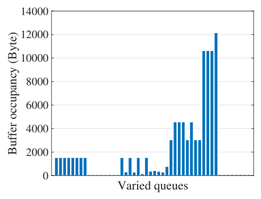
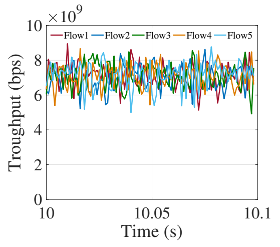

(b) Max buffer occupancy

Fig. 4. Data out-of-order degree and max buffer occupancy in in a over-subscripted circumstance


(a) Credit spraying with sequence-based feedback control

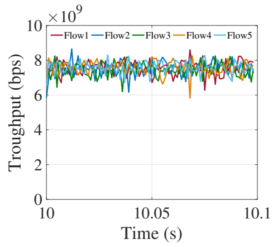

(b) Credit spraying with sequence-free feedback control

Fig. 5. Throughput of the servers when using credit spraying

are operating. Fig. 4(a) shows the data out-of-order degree at receivers, Fig. 4(b) shows the max queue length of the switch ports and host NICs. The buffer occupancy maintains low in a heavy over-subscripted circumstance, the max buffer occupancy among all the queues is only 12144 Bytes (8 data packets). The max out-of-order degree is 3, much less than the max queue length. We use a simple mechanism in SprayPass to reorder the out-of-order packets at the receiver.

Detecting network asymmetry at receiver. In data center networks, links and switches occasionally fail [30]. As the sender has no idea when and where the failure happens, the failed paths are usually bypassed by routing protocols. What has value is that each arrived data packet carries a path-id, and receivers are able to probe the path state by monitoring the arriving speed of relevant path-id-carried data. Given that, we develop an easy-implement mechanism to handle network asymmetry in §III-C.

However, through analyzing and simulating, we found a significant problem of simply enabling credit spraying in ExpressPass. ExpressPass employs a feedback control algorithm based on the credit loss rate, which is calculated by the consecutiveness of the credit sequence number. The feedback control is important for reducing credit waste, so it is necessary for SprayPass. However, the credit packets would be out-of-order[2] when they are sprayed into multiple paths. As Fig. 3(a) shows, the credits are sent by receivers in sequence, but the data packets arrive at the receiver in an out-of-order manner in return. Consequently, the credit feedback control mechanism is less efficient with credit spraying.

[2]Each data packet carries two sequence numbers, i.e. a credit sequence number (CSN) and a data sequence number (DSN). The CSN is used for credit feedback control, and the DSN is set by the higher layer protocol. Here we mean the disordering of CSN

To verify the problem of sequence-based feedback control, we build a topology with 4 spine switches and 2 leaf switches and 10 servers (5 under each leaf), where 4 equivalent paths exist. In the evaluation, each sender continuously sends data to a receiver in a one-one correspondence manner. We use different configurations on the servers to reveal this problem. Enabling ExpressPass with credit spraying brings a large amount of out-of-order credits (14971 in 100 ms), and leads to throughput fluctuation as shown in Fig. 5(a). For comparison, we replace the origin feedback control mechanism with our proposed sequence-free feedback control (see §III-B for detail). As Fig. 5(b) shows, the throughput stability is improved, and the average throughput is increased by 6%. Theoretically speaking, when the transmission delays difference among varied paths increases, the credits out-of-order degree can be upgraded. Therefore, the performance could be downgraded more by employing sequence-based feedback control in real data centers.

To address the problem of credit out-of-order, we first attempt to use a reordering approach to manage the disordered credits. However, as some credits are discarded in the network, reordering becomes extremely hard. Given that, we develop SprayPass, employing a sequence-free feedback control which has no requirements on the credits sequence.

B. *Sequence-free Feedback Control*

Fig. 3(b) shows the operation process of SprayPass. Receivers randomly set a path-id on each credit, and the switches route the credits into different paths based on the path-id. When passing the credit queue, the credits are marked with ECN label according to the probability of P_d, which is decided by the credit dropping rate. Upon receiving credits, senders copy their ECN information to corresponding data packets and send the packets to their destinations. When data packets arrive, receivers record their ECN information and periodically adjust the credit transmitting rate in line with the received ECN proportion.

1) Credit queue marks ECN label on credits: Each port of the switches employs a data queue and a credit queue, and they are physically isolated. In SprayPass, the transmission rate of credit queue is limited to 5%, which ensures the reversed data packets travel through the upriver data queue without congestion. In credit queue, the credits injection rate can be faster than the transmission rate, so they will be discarded. To record the credit loss rate, each port maintains a value of P_d, which is calculated as follows:

$$P_d = \#cdt\_drop / \#cdt\_sum \tag{1}$$

Where $\#cdt\_drop$ means the number of dropped credits, and $\#cdt\_sum$ refers to the sum of the injected credits. When a credit leaves credit queue, the port marks ECN label on it with the probability of P_d.

To reflect near real-time credit loss rate, P_d is periodically updated. As the credit injection rate is dynamically changing, SprayPass refreshes P_d upon each N_{cdt} injected credits instead of a fixed time period. However, setting the value of N_{cdt}

TABLE I
CASES WITH 1/2 CREDIT LOSS RATE

11000	11001	10100	10101	10010	10011
terrible	*good*	*terrible*	*terrible*	*terrible*	*good*
01100	01101	01010	01011	00110	00111
good	*terrible*	*terrible*	*terrible*	*good*	*terrible*

involves a trade-off. Intuitively, setting N_{cdt} to 1 can reflect the credit loss rate: If the last credit is discarded, the current credit is marked; otherwise, the current credit is directly sent. However, this incurs problems. We assume five credits arrive at a credit queue, two of the first four credits are discarded and two of them are normally transmitted. Only the transmitted credits can be marked with ECN. There are totally 12 cases. If the ECN-marking probability of the last four credits can reflect the loss rate of the first four credits, we label the case with *good*. Otherwise, we label the case with *terrible*. As TABLE I shows, 0 means discarded, 1 means normally transmitted. Only a third is *good*, this impacts the accuracy of the feedback control algorithm. We find the *good* proportion is increased to two thirds when set N_{cdt} to 2.

In theory, the accuracy could be improved with the increase of N_{cdt}, but a too big value can also affect the real-time performance of loss rate reflection. As the credit sending rate is dynamically adjusted by receivers, the credit loss rate may burst in a small period. Besides, the period of feedback control at the receiver is an RTT, and the P_d updating period must no longer than it. In this paper, we set $N_{cdt} = 4$.

2) Sender reacts to credits: Each sender maintains a data buffer and a credit buffer. On receiving a credit, the sender first extracts a data packet from the data buffer if it exists in data buffer. If there is no corresponding data to the arrived credit in the data buffer, the credit will be stored in the credit buffer waiting for data from the higher layer. When no data arrives in a small time period, the sender will send a credit_stop packet to the receiver and clear the related credits in credit buffer. Otherwise, if data packets from a new flow arrive, the sender will send a credit_request to the receiver.

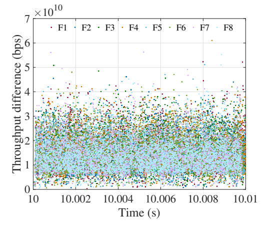
The scheduled data packet is the foremost one of the relevant flow, this ensures data packets are transmitted in order. Then, the sender checks whether the credit's ECN bit field is set to 1, and marks the data head the same way as this credit. Finally, the processed data packet is sent to the receiver. Note credits only schedule the data from its relevant data flow.

3) Receiver adjusts the credit sending rate: When receives a credit_request, the receiver begins to send credits. Each credit carries a randomly path-id and initially sent without ECN. Meanwhile, the receiver starts to perform the sequence-free feedback control algorithm and update the credit rate periodically, as Algorithm 1 shows. The update period is set to an RTT by default.

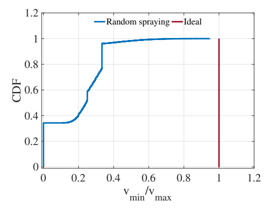
If ecn_ratio is less than the aim_ratio, the receiver considers the network is under-utilized and enters the increasing phase. The ecn_ratio is the proportion of received ECN-marked packets ($\#ecn\_pcks$) in the sum received packets

Algorithm 1 sequence-free feedback control algorithm

```
$\omega \leftarrow \omega_{init}$
$cur\_rate \leftarrow max\_rate \cdot \omega_{init}$
**repeat**
    per update period (RTT by default)
    $ecn\_ratio \leftarrow \#ecn\_pcks/\#sum\_pcks$
    **if** $ecn\_ratio \leq aim\_ratio$ **then**
        $(increasing\ phase)$
        **if** precious phase was increasing phase **then**
            $\omega = (\omega + \omega_{max})/2 \quad (\omega_{max} = 0.5)$
        **end if**
        $cur\_rate = (1-\omega) \cdot cur\_rate + \omega \cdot max\_rate \cdot (1 + aim\_ratio)$
    **else**
        $(decreasing\ phase)$
        $cur\_rate = cur\_rate \cdot (1 - ecn\_ratio) \cdot (1 + aim\_ratio)$
        $\omega = max(\omega/2, \omega_{min}) \quad (\omega_{min} = 0.01)$
    **end if**
**until** End of Flow
```



(a) Maximum speed difference



(b) Minimum v_{min}/v_{max} value

Fig. 6. Overall speed difference and v_{min}/v_{max} value between varied paths in symmetric network

($\#sum\_pcks$) in last RTT, aim_ratio means the tolerated ecn_ratio in order to fully utilizing the link resources. Moreover, if the last period was in the increasing phase, the receiver will aggressively improve the credit rate by increasing the aggressiveness factor of ω. Otherwise, if the ecn_ratio exceeds the aim_ratio, the network is considered to be congested and the credit rate is decreased. When receives a credit_stop, the receiver stops transmitting credits of the relevant flow.

C. Path probing

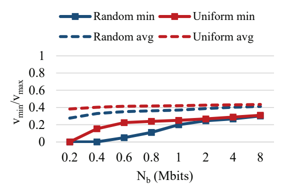
We develop a path probing mechanism for SprayPass, which makes path failure avoiding decision by receivers.

Each receiver keeps a speed board for the available paths, and it records the arriving speed of data packets. Under normal circumstances, multiple paths have the same link speed, so the data packets arriving from different paths have equal speed. When failure occurs, data arriving speed from the failed path is sharply degraded and soon being recorded in the speed board. Receivers monitor the speed board in real-time. Once detecting data speed on a path is decreased below a threshold v_{min}, the receiver directly excludes the homologous path-id for credit spraying. The value of v_{min} is calculated as follows:

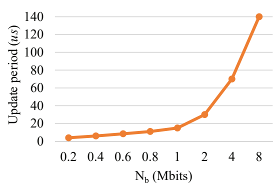
$$v_{min} = \theta \cdot v_{max} \qquad (0 < \theta < 1) \tag{2}$$

Where v_{max} refers to the maximum speed of all the paths, and θ is a proportion factor that decides the maximum allowed speed difference between the failed path and normal path.

However, because of the randomness of setting path-id, dropping credit, and marking credit, the speed between different paths wildly fluctuates even in asymmetric networks.



(a) Minimum v_{min}/v_{max} value



(b) Average update period

Fig. 7. Speed difference between varied paths and speed board updating period with different N_b value

It is hard to accurately find out the failed path. We make an experiment to evaluate the speed fluctuation, the data arriving speed is averaged upon every received 0.1 Mbits data. We use an indicator of v_{min}/v_{max} to reflect the speed difference between the fastest path and the slowest path. As Fig. 6(a) shows, the speed difference is large and unstable, it is dense in the range of 10 Gbps to 20 Gbps. The ideal value of v_{min}/v_{max} is 1, which every path has the same data arriving speed. However, Fig. 6(b) shows that the actual situation is much terrible than the ideal state. The minimum value of v_{min}/v_{max} is 0, which means the minimum data arriving speed on one path is 0 over a period. This hinders us to design path probing mechanism.

To mitigate this problem, we improve the credit spraying mechanism, setting path-id in a uniformly random way. Receivers use a path table to store the remaining valid path-ids and uniformly set path-id on credits round-by-round. When sends a credit, the receiver randomly picks up a path-id from the table to set on the credit and deletes it in the path table. When the path table is emptied, receiver refreshes it to restore all the valid paths. By doing this, credits uniformly travel all valid paths in every round, thereby mitigating the speed fluctuation between different paths.

The speed board is updated periodically to track real-time data speed. In the beginning, we set the update period as an RTT, but it causes problems. The data arriving speed is slow at the very beginning of the flow, thus small amounts of data can arrive at the receiver. By that, the difference in speed between different paths is extremely large, even some paths hold the speed of zero. Therefore, we improve the initial idea, updating the speed board upon N_b bits of arriving data.

However, to decide the value of N_b involves a complex trade-off between the tolerability of speed fluctuation, and reacting time to path failures. We make some experiments for setting a better value of them and verifying that the uniform credit spraying has better performance in speed fluctuation. If no credit is discarded in different paths, data arriving speed can be near-equal. Since current data center topologies have sufficient bisection bandwidth, credits can pass the network with few losses. Therefore, to simulate a worse-than-real case, we employ an experiment with the oversubscription ratio of 2:1 (8 flows in 4 equivalent paths).

Fig. 7(a) reveals the maximum value of speed difference between diverse paths, with varied N_b from $2 \cdot 10^5$ to $8 \cdot 10^6$. The smaller the v_{min}/v_{max} is, the bigger the speed difference is. As is shown, when $N_b = 2 \cdot 10^5$, both uniform spraying and random spraying lead to the minimum value of $v_{min}/v_{max} = 0$. With the increase of N_b, the minimum value of v_{min}/v_{max} when using uniform spraying is much smaller than using random spraying, especially when $N_b < 10^6$. This proves uniform spraying has much better performance than random spraying. Although the speed difference is decreasing with the increase of N_b, such a big value of N_b may prolong the update period and further affects the path failure reacting time, as Fig. 7(b) shows. By uniform spraying, The value of $v_{min}/v_{max} = 0$ changes little when $N_b > 6 \cdot 10^6$, but the reacting time grows linearly. Therefore, to gain the best balance between fast reacting to path failure and small inter-path speed fluctuation, we set $N_b = 6 \cdot 10^5$.

As misdiagnosing path failure could lead to severe performance reduction, we set $\theta = 0.1$, which is effective in the networks with the oversubscription ratio less than 2:1.

TABLE II
CONFIGURATIONS OF SPRAYPASS

	Sequence-free feedback control	Path probing
SprayPass-1		
SprayPass-2	✓	
SprayPass	✓	✓

IV. EVALUATION

We use OMNeT++ to testify the performance of SprayPass with varied objectives, including throughput, convergence, flow completion time, buffer occupancy, and handling asymmetry. To be concentrated, we mainly compare SprayPass to ExpressPass. In order to verify the effectiveness of sequence-free feedback control and path probing, varied configurations of SprayPass are used as TABLE II shows. Note, to simplify the configurations, uniform spraying is bound with path probing.

A. Throughput of Permutation Traffic

We use a leaf-spine topology with 4 spine switches, 24 leaf switches, and 240 servers, where each leaf connects to 10 servers. The link between server and leaf is 40 Gbps and the link between leaf and spine is 100 Gbps, which forms a full-bisection network. The basic RTT passing the spine is 32 μs. We use a permutation traffic [21], [31], where all the servers act as senders and send data to one of the other servers across different leaf switches, and each server only acts as a receiver one time. In total there are 240 connections. For ExpressPass, we enable ECMP to balance the traffic.

Fig. 8 shows the goodput of each server. SprayPass has much better overall performance than ExpressPass, as many flows are hashed into the same paths by ECMP routing. Random spraying may lead to an instantaneous burst in one path and then activate the feedback control to downgrade the credit sending rate. SprayPass achieves better throughput than SprayPass-2 because credits are uniformly sprayed into the equivalent paths by uniform spraying. The average throughput

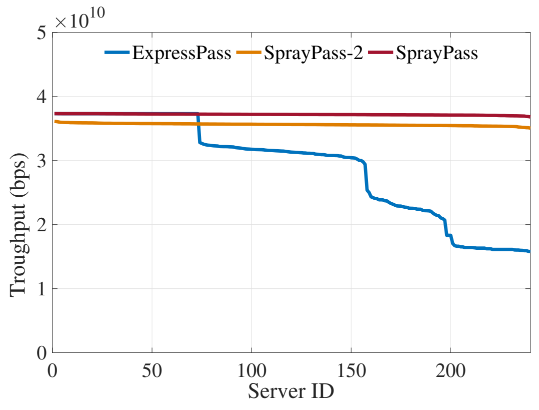
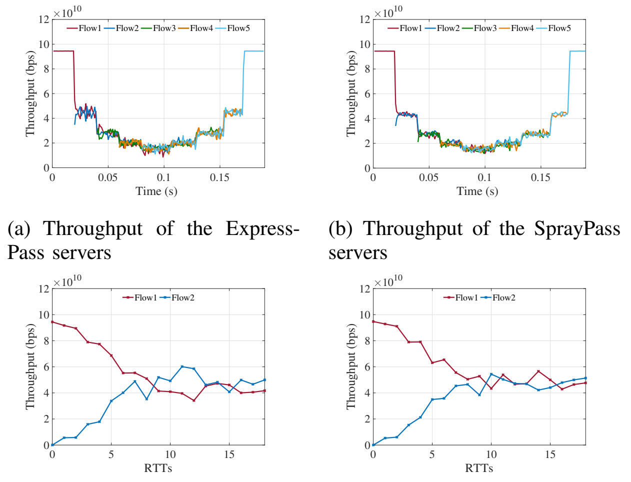


Fig. 8. Overall throughput of the servers with permutation traffic



(a) Throughput of the ExpressPass servers (b) Throughput of the SprayPass servers

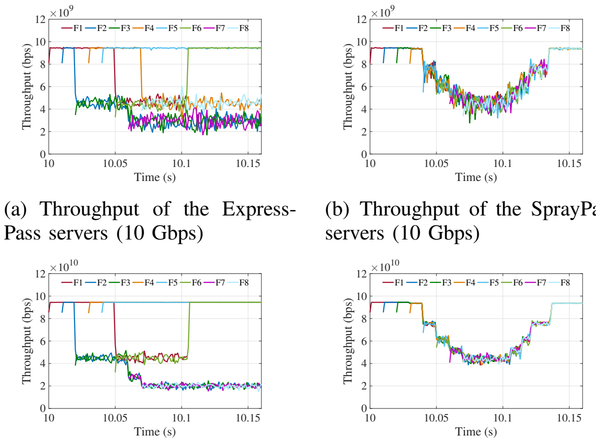
(c) Converge time of ExpressPass (d) Converge time of ExpressPass

Fig. 9. Single path convergence compared to ExpressPass @ 100G

of SprayPass is 27.9% better than ExpressPass, and SprayPass-2 is 22.5% better than ExpressPass. Besides, SprayPass and SprayPass-2 achieve better fairness than ExpressPass across the servers, the minimum throughput among the servers is 36.8 Gbps with SprayPass and 35.1 Gbps with SprayPass-2 while 15.8 Gbps with ExpressPass.

B. Convergence in Varied Conditions

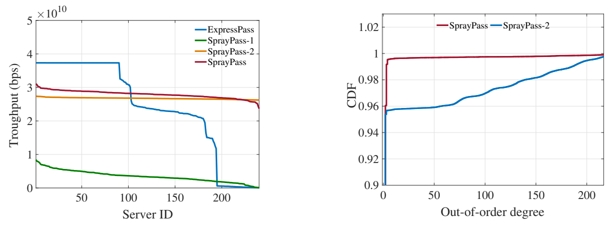
The convergence of SprayPass on a single path is theoretically the same as ExpressPass, we run a simple test with five flows in a dumbbell topology to verify it. The link speed is set to 100 Gbps, throughput is averaged and recorded every 1 ms. As Fig. 9(a) and Fig. 9(b) shows, the maximum throughput of SprayPass and ExpressPass are around 95% as the other 5% is used for credit transmission. The stability of SprayPass is a little bit better than ExpressPass, mainly because the randomness of ECN marking can help achieve better fairness than credit dropping. Besides, Fig. 9(c) and Fig. 9(c) shows SprayPass and ExpressPass have the same converge time of 7 RTTs. This proves that sequence-free feedback control has no impact on the convergence of ExpressPass on a single path.



(a) Throughput of the ExpressPass servers (10 Gbps) (b) Throughput of the SprayPass servers (10 Gbps)

(c) Throughput of the ExpressPass servers (100 Gbps) (d) Throughput of the SprayPass servers (100 Gbps)

Fig. 10. Multi-path convergence compared to ExpressPass



(a) Throughput of the servers (b) Overall out-of-order degree

Fig. 11. Overall throughput and out-of-order degree in asymmetric network

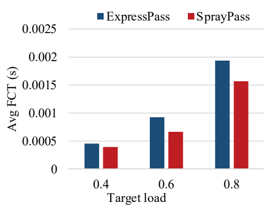
To further study the convergence of SprayPass, we propose a concept of multi-path convergence. Multi-path convergence indicates the speed a multi-path transport converges to fair and the stability when it is converged (with multiple paths network). We build a topology with 4 spine switches and 2 leaf switches and 16 servers (8 under each leaf), where 4 equivalent paths exist. The servers under one leaf switch act as senders and the servers under another leaf switch act as receivers, each sender sends flow to a receiver in a one-one correspondence manner. The link speed is set to 10 Gbps and 100 Gbps in two experiments, throughput is averaged every 1 ms.

As shown in Fig. 10(a) and Fig. 10(b), the server throughput converges to fairness within 1 ms by SprayPass, while it cannot converges to fairness by ExpressPass. Besides, the throughput fluctuation of SprayPass is no more severe than ExpressPass, even if the flows are split into multiple paths. Since more credits and data packets can be transmitted within an RTT in 100 Gbps network, the credit dropping probability and ECN-marking probability are much more uniform. Therefore, the throughput stability in 100 Gbps network is better than it in 10 Gbps network, as Fig. 10(c) and Fig. 10(d) shows.

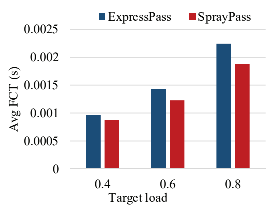
C. Reacting to Path Failure

We ran a permutation experiment in the same topology as IV-A, and reduce the speed of the links connect to one spine switch to 1 Gbps.

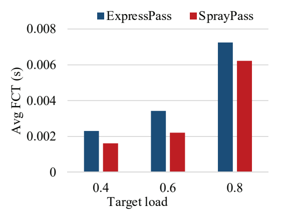
Fig. 11(a) shows that some of the ExpressPass flow is defected by the path failure and their throughput are downgraded



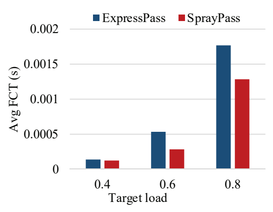
(a) Average FCT of Cache Follower



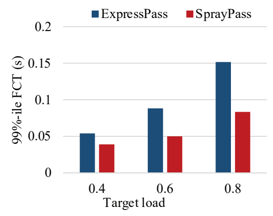
(b) Average FCT of Web Search



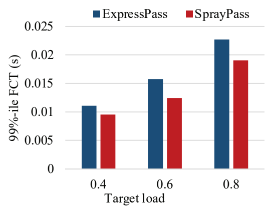
(c) Average FCT of Data Mining



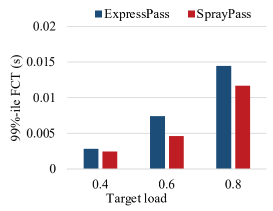
(d) Average FCT of Web Server



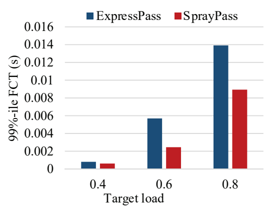
(e) 99%-ile FCT of Cache Follower



(f) 99%-ile FCT of Web Search

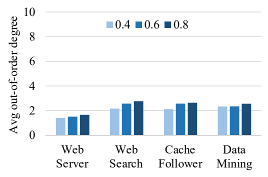


(g) 99%-ile FCT of Data Mining

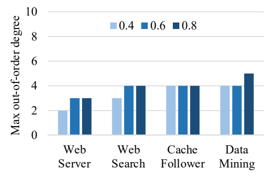


(h) 99%-ile FCT of Web Server

Fig. 12. Average FCT and 99%-ile FCT compared to ExpressPass under realistic workload



(a) Average out-of-order degree



(b) Maximum out-of-order degree

Fig. 13. Out-of-order degree among all the servers under realistic workload

to below 1 Gbps. As some credits and data are transmitted through the failed path, the overall throughput of SprayPass-1 is extremely downgraded by the out-of-order credits. The funny thing is, the overall throughput of the naive SprayPass and SprayPass is almost the same. This is mainly because the data packets arrived from the failed path is much slower than it from normal paths, so the ecn_ratio is impacted by the failed path to a small degree. Fig. 11(b) shows the overall out-of-order degree among all the servers, and the 99th percentile out-of-order degree of SprayPass-2 is 185, while it of SprayPass is 3. The out-of-order degree of the SprayPass is fast increasing at the beginning because some packets travel through the failed path. Thanks to the path probing, SprayPass detects the path failure and quickly transfers the flows into the other three normal paths, this helps it keep a low out-of-order degree. However, SprayPass-2 keeps high out-of-order degree, because the packets which travel through the failure path are much slower than the packets across the normal paths.

D. Performance under Realistic Workloads

We use the same topology in IV-A and employ the same realistic workload as ExpressPass [22], including Cache Follower, Web Search, Data Mining, and Web Server. The source and destination of each flow are randomly picked from the servers. We mainly evaluate the buffer occupancy, out-of-order degree and flow completion times of SparyPass, and we simulate three target loads 0.4, 0.6, and 0.8 for ToR up-links.

As Fig. 12 shows, for all kinds of traffic, SprayPass gains better performance than ExpressPass. As we can see, SprayPass reduces FCT more at the target load of 0.6. With 0.8 target load, some flows are simultaneously transmitted to the same receiver, and this leads to an end-point oversubscription. This prolongs FCT and cannot be mitigated by SprayPass. With 0.4 target load, fewer flows can be concurrently routed into the same path, so that the performance of ExpressPass is less impacted than with 0.6 target load. Besides, SprayPass has much performance improvement with web server traffic and data miming traffic. Particularly, in web server traffic with 0.6 target load, SprayPass achieves 0.531x of average FCT and 0.431x 99^{th} tail FCT compared to ExpressPass.

Fig. 13 shows the data out-of-order degree of SprayPass under realistic workloads. The overall out-of-order degree maintains a low level, and the maximum degree is 6. The degree is higher in the traffic modes with long average flow sizes, and it in web server keeps lowest.

TABLE III shows the average and maximum queue occupancy of ExpressPass and SprayPass under realistic workloads. SprayPass inherits low queue occupancy from ExpressPass, and it even achieves lower occupancy than ExpressPass under a light target load. Both the queue size of SprayPass and ExpressPass is neither related to the traffic mode nor the target load, their queue bound is decided by the topology.

V. Discussion and Future Work

A. Improving credit scheduling mechanism

First RTT transmission: The sender first generates and sends a credit_request to the receiver, and the receiver sends credits to the sender in return after receives a credit_request. This costs an RTT. For the elephant flows, an RTT counts for little impact on their flow completion times. But for the mice flows, this largely increases their transmission latency. To eliminate the first RTT problem, SMSRP [32] and Aelous [33] are proposed. In these approaches, packets in first RTT

TABLE III
AVERAGE/MAXIMUM QUEUE OCCUPANCY

		Average Queue (KB)		Max Queue (KB)	
Traffic Mode	*Target Load*	Express Pass	Spray Pass	Express Pass	Spray Pass
Cache Follower	0.4	1.11	1.03	39.6	25.5
	0.6	1.33	0.96	49.0	36.2
	0.8	1.31	1.22	46.5	42.9
Web Search	0.4	0.95	0.87	62.3	57.8
	0.6	1.31	1.21	73.6	56.3
	0.8	1.12	1.15	65.4	54.7
Data Mining	0.4	1.61	1.24	80.5	66.9
	0.6	1.53	1.49	76.0	71.4
	0.8	1.69	1.55	81.2	75.5
Web Server	0.4	0.92	0.71	36.4	36.7
	0.6	1.11	0.93	37.4	38.2
	0.8	0.83	1.04	34.3	33.2

are directly sent by the sender without any grant (credit) from the receiver, we call them speculative packets. However, this may lead to data queuing and even packet loss. To protect the transmission of normal packets and prevent extra retransmission delay, speculative packets are set with a low priority and they are discarded by the switches when congestion occurs. By doing this, the flow completion times of mice flows can be largely reduced. SprayPass is orthogonal to these approaches, and we believe the performance of SprayPass can be improved by cooperating with them.

Credit efficiency: In SprayPass, each credit schedules a data packet, assuming the packets are the maximum Ethernet frame size. However, the size of data packets is varied: the extremely small flows (smaller than a maximum Ethernet frame size); the tail packet of each flow. We can mitigate this problem by each credit schedule a fixed amount of data, instead of absolutely a packet. This requires deep consideration, and we are going to improve credit efficiency in the future.

B. Limitations of credit feedback control

Our proposed sequence-free feedback control algorithm is based on the algorithm of ExpressPass feedback control [22], so it has the same limitations. Short flows cause credit waste. There is a time interval between the end of a flow and the receiver stops sending credits, it is composed of the time used for judging whether the flow is completed and the transmission time of credit_stop. For the mice flows, the time interval is relatively large, and the throughput of elephant flows can be severely affected when lots of mice flow concurrently exist. If the end of a flow can be reliably estimated in advance, this problem can be addressed. Besides, we assume all hosts have the same link capacity, but different host link speed may lead to unfairness.

VI. RELATED WORKS

A large body of works has been proposed for data center congestion control, they are mainly classified as reactive schemes and proactive schemes. Many of them are multi-path transports.

Multi-path reactive congestion control. MPTCP [31] and DeTail [5] are multi-path congestion control for TCP traffic. MPTCP modifies TCP to enable multi-path transmission, it splits the original TCP flow into several sub-flows and distributes packets among these sub-flows according to their congestion states. Thereby MPTCP adds additional states proportional to the number of sub-flows and explores a large re-ordering buffer at the transport layer to handle out-of-order packets. In DeTail, switches dynamically allocate the shortest and less congestion path for packets, thus, achieving effective load balancing. But it relies much on switches, which makes it hard to deploy. Besides, MP-RDMA is a multi-path transport for managing RDMA traffic. MP-RDMA leverages a multi-path ACK-clocking and out-of-order aware path selection to choose the best network paths and distribute packets among them in a congestion-aware manner. However, both DeTail and MP-RDMA require employing PFC to achieve link-layer lossless, which may lead to deadlock and congestion tree.

Multi-path proactive congestion control. Fastpass [18] is a centralized multi-path proactive congestion control mechanism for data centers. Fastpass precisely specifies the transmission time and transmission path for each packet through a centralized scheduler. As Fastpass needs to make accurate scheduling decisions for all packets, it will introduce a lot of additional communication and computational overhead. To reduce these expenses, Flowtune [34] was proposed. Flowtune is also a centralized proactive mechanism, but unlike Fastpass, it manages congestion and balances traffic at the granularity of flowlet [35], which has high flexibility. Since a flowlet contains multiple packets, Flowtune can effectively reduce the overhead of Fastpass. However, the centralized schemes are hard to implement and the communication over-head between central controller and end-hosts is difficult to eliminate. NDP [19] uses packet spraying and pull packets to achieve high throughput and low latency, but it cannot prevent packet loss and is hard to deploy.

VII. CONCLUSION

In current high-speed data centers, ExpressPass cannot fully utilize the multiple equivalent links even with the cooperation of ECMP. To improve its performance, we propose SprayPass. SprayPass leverages credit spraying to uniformly balance the bursty traffic, uses sequence-free feedback control to address the problem of credit out-of-order, and employs path probing to handle network asymmetry.

In total, our evaluations show that SprayPass is consistent with ExpressPass in terms of queue occupancy, zero data loss, and fast convergence on a single link. Moreover, compare to ExpressPass, SprayPass greatly improves network utilization in permutation traffic, reduces flow completion times under realistic workload, and achieves fast converge to fair and high throughput with multiple paths. Besides, SprayPass can fast react to network asymmetry when the path failure occurs.

References

[1] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements &amp; analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 202–208.

[2] W. Bai, K. Chen, S. Hu, K. Tan, and Y. Xiong, "Congestion control for high-speed extremely shallow-buffered datacenter networks," in *Proceedings of the First Asia-Pacific Workshop on Networking*. ACM, 2017, pp. 29–35.

[3] S. Huang, D. Dong, and W. Bai, "Congestion control in high-speed lossless data center networks: A survey," *Future Generation Computer Systems*, vol. 89, pp. 360–374, 2018.

[4] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *ACM SIGCOMM computer communication review*, vol. 40, no. 4. ACM, 2010, pp. 63–74.

[5] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "Detail: reducing the flow completion time tail in datacenter networks," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 139–150, 2012.

[6] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ecn in multi-service multi-queue data centers." in *NSDI*, 2016, pp. 537–549.

[7] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale rdma deployments," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 523–536.

[8] J. D'ambrosia, "40 gigabit ethernet and 100 gigabit ethernet: The development of a flexible architecture [commentary]," *IEEE Communications Magazine*, vol. 47, no. 3, 2009.

[9] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, 2015.

[10] H. Barrass *et al.*, "Proposal for priority based flow control," *vol*, vol. 2, pp. 1–9, 2008.

[11] G. F. Pfister, "An introduction to the infiniband architecture," *High Performance Mass Storage and Parallel I/O*, vol. 42, pp. 617–632, 2001.

[12] G. F. Pfister and V. A. Norton, ""hot spot" contention and combining in multistage interconnection networks," *IEEE Transactions on Computers*, vol. 100, no. 10, pp. 943–948, 1985.

[13] D. R. Pannell, "Network switch with head of line input buffer queue clearing," Jan. 21 2003, uS Patent 6,510,138.

[14] D. Lee, S. J. Golestani, and M. J. Karol, "Prevention of deadlocks and livelocks in lossless, backpressured packet networks," Feb. 22 2005, uS Patent 6,859,435.

[15] L. Jose, L. Yan, M. Alizadeh, G. Varghese, N. McKeown, and S. Katti, "High speed networks need proactive congestion control," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. ACM, 2015, p. 14.

[16] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, "phost: Distributed near-optimal datacenter transport over commodity network fabric," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 2015, p. 1.

[17] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 221–235.

[18] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 307–318, 2015.

[19] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 29–42.

[20] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese *et al.*, "Conga: Distributed congestion-aware load balancing for datacenters," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 503–514.

[21] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and T. Moscibroda, "Multi-path transport for {RDMA} in datacenters," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 357–371.

[22] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 239–252.

[23] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[24] D. Abts and J. Kim, "High performance datacenter networks: Architectures, algorithms, and opportunities," *Synthesis Lectures on Computer Architecture*, vol. 6, no. 1, pp. 1–115, 2011.

[25] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[26] C. Evangelinos and C. Hill, "Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2," *ratio*, vol. 2, no. 2.40, pp. 2–34, 2008.

[27] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[28] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 123–137.

[29] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "Rdma over commodity ethernet at scale," in *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 2016, pp. 202–215.

[30] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 350–361, 2011.

[31] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. Citeseer, 2011, pp. 266–277.

[32] N. Jiang, L. Dennison, and W. J. Dally, "Network endpoint congestion control for fine-grained communication," in *High Performance Computing, Networking, Storage and Analysis, 2015 SC-International Conference for*. IEEE, 2015, pp. 1–12.

[33] S. Hu, W. Bai, B. Qiao, K. Chen, and K. Tan, "Augmenting proactive congestion control with aeolus," in *Proceedings of the 2nd Asia-Pacific Workshop on Networking*. ACM, 2018, pp. 22–28.

[34] J. Perry, H. Balakrishnan, and D. Shah, "Flowtune: Flowlet control for datacenter networks." in *NSDI*, 2017, pp. 421–435.

[35] S. Sinha, S. Kandula, and D. Katabi, "Harnessing tcp's burstiness with flowlet switching," in *Proc. 3rd ACM Workshop on Hot Topics in Networks (Hotnets-III)*, 2004.