RPO: Receiver-driven Transport Protocol Using Opportunistic Transmission in Data Center

Jinbin Hu*, Jiawei Huang*, Zhaoyi Li*, Yijun Li*, Wenchao Jiang§, Kai Chen†, Jianxin Wang*, Tian He‡
*Central South University, Changsha, China. Email: jinbinhu@126.com,{jiaweihuang,lizhaoyi,yijunli,jxwang}@csu.edu.cn
§Singapore University of Technology and Design, Singapore. Email: wenchao_jiang@sutd.edu.sg
†Hong Kong University of Science and Technology, Hong Kong, China. Email: kaichen@cse.ust.hk
‡University of Minnesota, Minneapolis, MN, USA. Email: tianhe@umn.edu

***Abstract*—Modern datacenter applications bring fundamental challenges to transport protocols as they simultaneously require low latency and high throughput. Recent receiver-driven transport protocols transmit only one data packet once receiving each grant or credit packet from the receiver to achieve ultra-low queueing delay and zero packet loss. However, the round-trip time variation and the highly dynamic background traffic significantly deteriorate the performance of receiver-driven transport protocols, resulting in under-utilized bandwidth. This paper designs a simple yet effective solution called RPO that retains the advantages of receiver-driven transmission while efficiently utilizing the available bandwidth. Specifically, RPO rationally uses low-priority opportunistic packets to ensure high network utilization without increasing the queueing delay of high-priority normal packets. In addition, since RPO only uses Explicit Congestion Notification (ECN) marking function and priority queues, RPO is ready to deploy on switches. We implement RPO in Linux hosts with DPDK. Our small-scale testbed experiments and large-scale simulations show that RPO significantly improves the network utilization by up to 35% under high workload over the state-of-the-art receiver-driven transmission schemes, without introducing additional queueing delay.**

***Index Terms*—Data center, Receiver-driven, Utilization**

I. Introduction

Diverse cloud-based applications such as web search, social networking, data mining and on-line retails coexist in data centers [1]–[6]. Recently, to achieve ultra-low queueing delay and zero packet loss, receiver-driven transport protocols such as Homa [7], NDP [8], ExpressPass [9], pHost [10] and AMRT [11] are proposed to transmit new data packets only in response to grant packets from the receiver. Therefore, these proactive congestion control solutions effectively avoid queue buildup and packet loss, significantly improving the latency performance of delay-sensitive applications.

However, the variations of round-trip times (RTTs) and the highly dynamic background traffic significantly deteriorate the performance of receiver-driven transport protocols with the following two fundamental problems. (1) When multiple receiver-driven flows with different RTTs share the bottleneck link, limited by the conservativeness of the receiver-driven transmission mechanism, even if the flows with enlarged RTT release the bandwidth, the other flows are unable to seize the available bandwidth, further reducing the link utilization. (2) Due to the dynamic ON/OFF nature of data center traffic [14]–[17], the receiver-driven transport protocols cannot fill up the available bandwidth when the background flows are in their OFF periods, leading to low link utilization.

978-1-6654-4131-5/21/$31.00 ©2021 IEEE

To improve link utilization, the recent proposed overcommitment mechanism in Homa [7] allows multiple senders to concurrently transmit data to a single receiver. However, such overcommitment does not directly address the above problems because that, when some flows experience OFF periods or enlarged RTTs, the other flows cannot proactively seize the free bandwidth. What's worse, as we show in our evaluation (Section VIII-B), the overcommitment solution potentially introduces queueing delay under highly dynamic traffic scenarios, causing degradation of delay performance.

In this paper, we propose a simple yet effective approach called RPO that retains the advantages of receiver-driven transmission while efficiently obtaining high link utilization. Specifically, RPO utilizes opportunistic packets to fill up the spare bandwidth left by the receiver-driven transmission. The opportunistic packets are data packet deliberately marked as low priority in the transmission buffer. They are triggered simultaneously with normal data packets, i.e., the high priority packets, by the same native grants in the receiver-driven transmission. At the switch, the high-priority normal packets and the low-priority opportunistic packets are fed into respective priority queues. Due to the low priority, the opportunistic packets will not block the transmission of normal packets, but only seize the spare bandwidth left by the normal packets.

In addition, RPO controls the queueing delay and traffic overhead of opportunistic packets under fully utilized link. It is achieved through the Explicit Congestion Notification (ECN) marking function commonly available in commodity switches. Specifically, when the queueing length of opportunistic packets in the low priority queue at switch is larger than a threshold, the ECN flags of the normal packets are set within the switch. A grant packet with the ECN-Echo flag set will no longer trigger the opportunistic packet to avoid queue buildup and unnecessary traffic overhead. With all these innovations, RPO simultaneously achieves high link utilization and low latency performance.

In summary, our major contributions are:

- We conduct an extensive testbed-based study to analyze two key issues that lead to low link utilization under receiver-driven transmission: 1) when multiple flows with

different RTTs share the bottleneck link, the spare bandwidth released by the flows with enlarged RTT cannot be utilized by the other flows, 2) the highly dynamic background traffic with ON/OFF nature leads to spare bandwidth on shared bottleneck. For example, as shown in Section VIII-A, pHost, ExpressPass, Homa, NDP and Aeolus (integrated with Homa by default) only obtain 56%, 63%, 65%, 67% and 69% link utilization under the dynamic data mining workload, respectively.

- We propose a receiver-driven transport protocol RPO, which uses low-priority opportunistic packets to fill up the spare bandwidth to improve link utilization without increasing the queueing delay for high-priority normal packets. RPO can be easily deployed on switches using the built-in ECN function and priority queues.
- By using both 10Gbps small-scale testbed experiments and 40Gbps large-scale simulations, we demonstrate that RPO remarkably outperforms the state-of-the-art receiver-driven transmission schemes. RPO yields up to 35%, 21%, 17%, 12% and 10% network utilization improvement over pHost, ExpressPass, Homa, NDP and Aeolus, respectively. Meanwhile, RPO does not increase the queueing delay of normal packets and reduces the average flow completion time (AFCT) by 23%-52% under heavy workload.

The rest of the paper is organized as following. In Section II and III, we respectively describe our design motivation and overview. In Section IV and V, we give the design details and model analysis of RPO, respectively. We discuss the implementation in Section VI. In Section VII and VIII, we show the testbed experimental and NS2 simulation results, respectively. In Section IX, we present the related work and then conclude the paper in Section X.

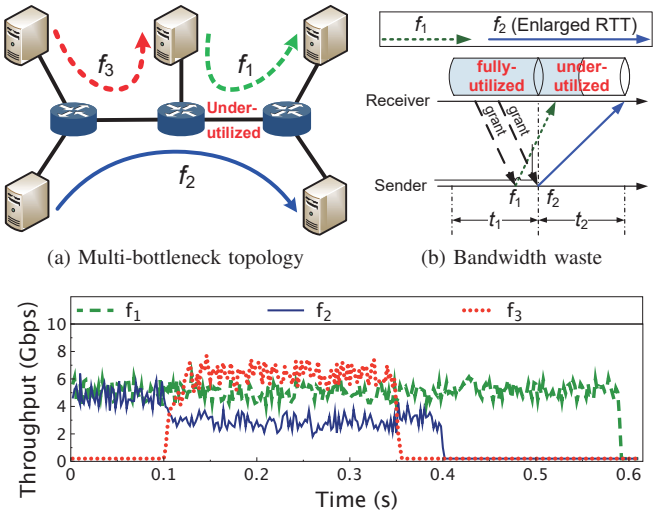
II. Background and Motivation

A. Receiver-driven transmission

In receiver-driven transmission, bottleneck link capacities are proactively allocated by the receivers, which use grant or credit packets to trigger new data packets. Under pHost [10], Homa [7] and NDP [8], the sender starts new flows at line rate. After that, whenever a data packet arrives at the receiver, a corresponding grant packet is delivered back to the sender to trigger one new data packet. Moreover, Homa [7] uses in-network priority queues [18] to ensure low latency for short messages. When the switch queue length exceeds eight packets, NDP trims the payloads of packets and forwards the packet headers with high priority. ExpressPass [9] allocates bandwidth by shaping the flow of credit packets at switches to effectively achieve zero data loss and low delay. To avoid serious network congestion due to blind transmission or extra waiting delay for credits in the first RTT, Aeolus [12], [13] is integrated into the above receiver-driven schemes to let new flows only utilize the spare bandwidth instead of aggressive transmission.

B. Impact of RTT Variation

In data centers, traffic changes so quickly that congestion can rapidly arise and dissipate, leading to great RTT variations during transmission [14], [20]. Meanwhile, modern data centers deploy the multi-rooted tree topology with multiple paths and bottlenecks [21], [22]. Consider a many-to-one scenario in which multiple flows from different paths with various congestion states and finally compete for a single bottleneck, the flows with increased RTT incur spare bandwidth of the bottleneck link. However, the other flows cannot grab the available bandwidth under the receiver-driven transmission.



(a) Multi-bottleneck topology (b) Bandwidth waste

(c) Throughput loss

Fig. 1: Under-utilized link due to RTT variation

We use a simple typical example to show the impact of RTT variation. Without loss of generality, in Fig. 1 (a) and (b), we demonstrate the impact on two pHost flows f_1 and f_2 sharing the 1st bottleneck link (from the right) with the capacity of 2 packets. Two packets from f_1 and f_2 can fully utilize the 1st bottleneck bandwidth in t_1 period, and each arrival packet generates a grant packet to drive a new data in the next time period t_2. Unfortunately, when f_3 starts at the high rate, half of the spare bandwidth at the 1st bottleneck is not utilized in t_2 period due to the enlarged RTT of f_2.

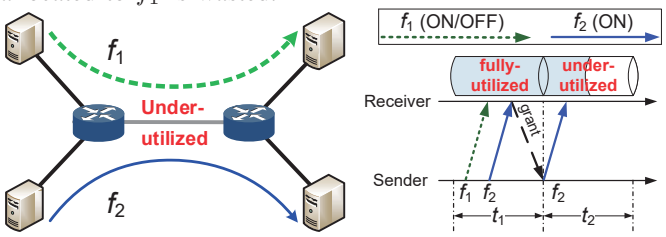
We further conduct a testbed experiment to investigate the impact of RTT variation, which is implemented by using Intel DPDK [19]. In the testbed, each server (Dell PRECISION TOWER 5820 desktop) is equipped with 10 cores Intel Xeon W-2255 CPU, 64GB memory, Intel 10GbE 2P X520 Network Interface Cards (NICs) and Ubuntu18.04 (Linux version 4.15.0-1090). Two servers with two Intel 10GbE 2P X520 NICs work as the four-port DPDK switches. We set the buffer size to 8 packets. The round-trip propagation time is 50μs. As shown in Fig. 1 (c), at the beginning, f_1 and f_2 share bottleneck link fairly without queue buildup until 0.1ms. During the periods [0.1s, 0.35s], f_2 experiences congestion due to by-passing flows in other bottlenecks, resulting in increased RTT. Then the link utilization of bottleneck shared by f_1 and f_2 is reduced by around 20%. The test results of

Homa and NDP under the same scenario are shown in Fig. 12 in Section VII.

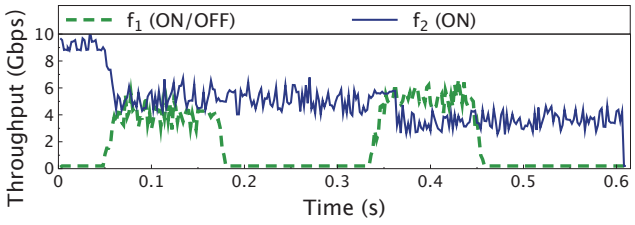
C. Impact of ON/OFF Traffic

Dynamic traffic will also cause throughput loss under the receiver-driven mechanism. In data centers, the highly dynamic traffic exhibits ON/OFF characteristic in nature [14], [15], [16]. For example, the update flows periodically copy fresh data to adjust the control state of workers [1]. According to the observation in [14], more than 50% flows come and go, lasting less than 0.1s. Unfortunately, under the receiver-driven transmission style, when some flows change from ON mode to OFF mode, the other flows sharing the same bottleneck link will not increase their sending rates. Thus, the spare bandwidth left by flows in OFF mode cannot be filled up by other flows in ON mode, resulting in under-utilized bandwidth.

We use another typical example to illustrate the impact of ON/OFF traffic. In Fig. 2 (a), two pHost flows $f_1$and f_2 share a bottleneck link with respective senders and receivers in the dumbbell topology. As shown in Fig. 2 (b), f_1 exhibits the ON/OFF pattern and f_2 is always in ON mode. In t_1 period, the bottleneck link with its capacity of 2 packets is fully utilized. However, when f_1 changes from ON mode to OFF mode in t_2 period, f_2 alone cannot make the bottleneck to be saturated, resulting in 50% reduction of bottleneck utilization, illustrated by the half-filled tube in Fig. 2 (b). In this scenario, ExpressPass also suffers from low link utilization because the credit packets to f_1's sender cannot trigger new packet when f_1 enters the OFF period. Therefore, the link bandwidth allocated to f_1 is wasted.



(a) Dumbbell topology (b) Bandwidth waste



(c) Throughput loss

Fig. 2: Under-utilized link due to ON/OFF traffic

We also conduct testbed experiment to analyze the under-utilized link problem due to ON/OFF traffic pattern. The experimental settings are the same as that in the RTT variation test. As shown in Fig. 2 (c), f_1 and f_2 start at 0s and 0.05s, respectively. The bottleneck utilization is greatly reduced when f_1 switches from ON mode to OFF mode at 0.18s and 0.46s, respectively. The test result demonstrates that the conservativeness of receiver-driven transport protocol easily leads to under-utilization under ON/OFF traffic pattern.

Moreover, we conduct large-scale NS2 simulation tests under different realistic workloads. The test results indicate that the receiver-driven transmission schemes suffer from reduced link utilization due to the impact of ON/OFF traffic and RTT variation. For example, under the data mining workload, pHost, ExpressPass, Homa, NDP and Aeolus only obtain 56%, 63%, 65%, 67% and 69% link utilization, respectively. The corresponding test results are shown in Section VIII-A.

D. Summary

The main challenge of proactive receiver-driven transmission is to achieve ultra-low latency and high link utilization simultaneously. To guarantee ultra-low queueing delay, the receiver-driven transport protocols enforce proactive congestion control, under which no new data packets are driven at the sender unless grant packets are received. Unfortunately, when the delay of some paths changes, the flow with larger RTT also easily leads to spare bandwidth at the bottleneck link. When some flows change from ON mode to OFF mode, the other flows will not grab the spare bandwidth. In both cases, the corresponding sender are not able to make full use of the bandwidth of bottleneck link due to inadequate grant packets from the receiver.

To improve bottleneck link utilization in receiver-driven transmission, an intuitive way is to trigger more data packets per grant. However, the highly dynamic nature of datacenter traffic makes it difficult to control the number of data packets driven by each grant. If the transmission is too conservative, the bottleneck link is still potentially under-utilized. While being too aggressive can avoid spare bandwidth, the queueing buildup violates the ultra-low latency property of proactive congestion control. These challenges motivate us to design and implement a new receiver-driven transport protocol to simultaneously provide high network utilization and low latency.

III. Design Overview

In this section, we present an overview of RPO. The key point of RPO is using low-priority opportunistic packets to utilize the spare bandwidth without increasing the queueing delay of high-priority normal packets. Specifically, the receiver sends grant packets back to the sender to drive normal packets and opportunistic packets simultaneously. The opportunistic packets in the low-priority queue do not increase the queue length of high-priority normal packets, but they can use the remaining available bandwidth of normal packets. The architecture of RPO is shown in Fig. 3.

(1) **Receiver-driven Opportunistic Transmission:** On the sender side, when a grant packet from the receiver arrives, a normal packet and an opportunistic packet are assigned with high and low priority, respectively. To control the length of low-priority queue, when the sender receives a grant packet with the ECN-Echo flag, it stops sending opportunistic packets until receiving the grant packets without ECN-marked. In the first RTT of whole transmission, each flow starts sending

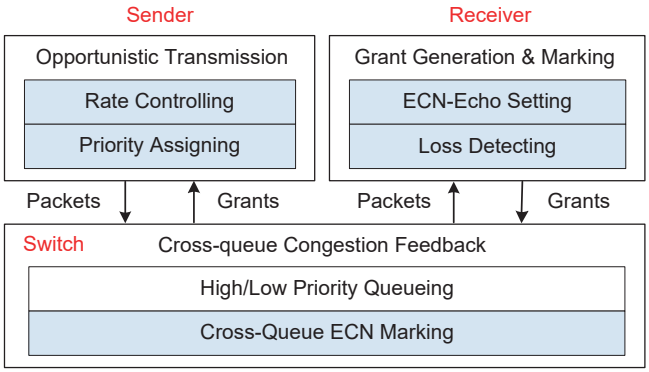


Fig. 3: Overview of RPO

a bandwidth-delay product (BDP) worth of normal packets ($BDPbytes$) to quickly saturate bandwidth.

(2) **Cross-queue Congestion Feedback:** At the switch, the normal and opportunistic packets enter the high and low priority queues, respectively. Furthermore, we propose a cross-queue marking scheme to quickly notify the congestion state of low-priority queue. Specifically, when the length of the low priority queue exceeds the ECN marking threshold k, the dequeued normal packets in the high priority queue are marked with ECN flag. RPO needs enough opportunistic packets to seize the available bandwidth while controlling the low-priority queue length to avoid long waiting time for opportunistic packets at the receiver. Therefore, the key challenge is to adjust the cross-queue marking threshold k to make full use of network bandwidth and limit the queueing delay of opportunistic packets.

(3) **Grant Generation and Marking:** On the receiver side, grant packets are generated according to the data arrival rate. Once a normal packet or an opportunistic packet arrives, the receiver generates the corresponding grant packet. Meanwhile, if an ECN-marked normal packet arrives, the receiver sets the ECN-Echo flag in the corresponding grant packet to notify congestion. In addition, RPO uses a fine-grained timer to detect packet loss at the receiver side.

IV. Design Details

A. Receiver-driven Opportunistic Transmission

At the sender, RPO leverages grant packets from the receiver to drive normal and opportunistic packets. As illustrated in Fig. 4, the high-priority normal packets are sent in order from the beginning of sending buffer. To minimize the overlap between the normal and opportunistic packets, RPO transmits the low-priority opportunistic packets in reverse order from the very last packet in the same sending buffer. When the maximum sequence number of transmitted normal packet is equal to the smallest sequence number of transmitted opportunistic packet, the sender will start to retransmit the opportunistic packets as the high-priority normal ones in the sending buffer. Therefore, even some opportunistic packets are backlogged in the low-priority queue of switch, they will not introduce additional waiting delay at the receiver.

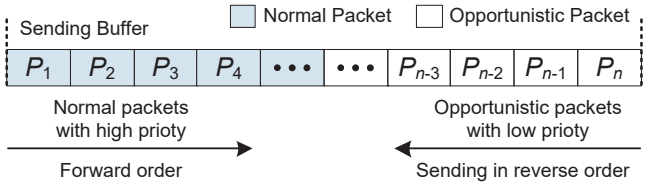


Fig. 4: Priority assignment and forward order for normal and opportunistic packets in the sending buffer

Moreover, RPO uses ECN marking as congestion sign of low-priority opportunistic packets. Specifically, if the grant packet is not ECN-marked, it indicates that the low-priority queue occupied by opportunistic packets is not congested. In this case, RPO can drive a normal packet and an opportunistic packet through a grant to rapidly make use of the available bandwidth. On the contrary, if an ECN-marked grant packet arrives at the sender, it means that the opportunistic packets are queued at the switch. If RPO continues to send opportunistic packets, it will cause queue backlog, resulting in larger queueing delay or even packet loss. Thus, once receiving ECN-marked grants, RPO stops transmitting opportunistic packets to control the number of inflight opportunistic packets.

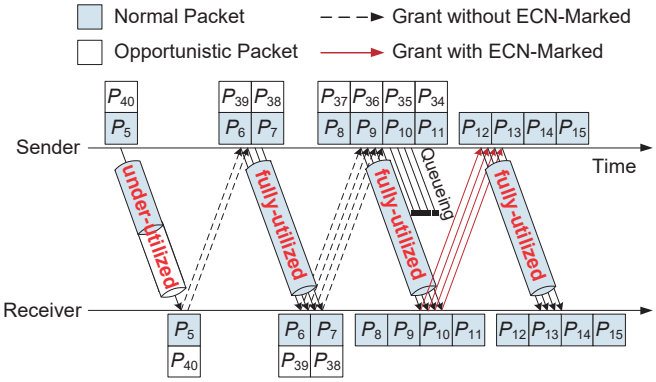


Fig. 5: Receiver-driven opportunistic transmission

To show how RPO transmits the normal and opportunistic packets, Fig. 5 illustrates a simple example with only one flow. We assume that the capacity of bottleneck link is 4 packets. At the beginning, a normal packet P_5 and an opportunistic packet P_{40} utilize only half of the bandwidth. When they arrive at the receiver, two corresponding grants are generated and sent back to the sender. Then each grant triggers a new normal packet and a new opportunistic packet. The bottleneck link is fully utilized by 4 packets, i.e., P_6, P_7, P_{38}, P_{39}, which fire the next four corresponding grants. These four new grants drive 4 normal packets ($P_8 \sim P_{11}$) filling the bottleneck pipe and 4 opportunistic packets ($P_{34} \sim P_{37}$) queued at the low-priority queue. When the length of low priority queue exceeds the ECN marking threshold, the normal packets are marked. The sender reacts by stopping sending the opportunistic packets until receiving the grant without ECN marking flag again. Therefore, after the four ECN-marked normal packets ($P_8 \sim P_{11}$) arrive at the receiver, four ECN-marked grants are generated to drive only four new normal packets ($P_{12} \sim P_{15}$) from the sender. In short, RPO adaptively adjusts the number

of opportunistic packets according to whether the arrival grants are ECN-marked or not to balance the high network utilization and low queueing delay.

Moreover, RPO starts sending a BDP bytes of normal packets to quickly saturate bandwidth at the beginning of transmission [23]. To guarantee the near-zero queueing delay of normal packets, like Aeolus [12], RPO drops the normal packets at switch when the queue length of high-priority queue exceeds a very small dropping threshold (i.e., 1 packet).

B. Cross-queue Congestion Feedback

At the switch, RPO employs a simple queue management scheme. As illustrated in Fig. 6, switches are mainly responsible for the cross-queue ECN marking. To ensure low latency in the receiver-driven transmission, the normal packets and opportunistic packets enter the high and low priority queues, respectively. In the priority queues, the normal packets with high priority are served before the opportunistic packets with low priority. As soon as the low-priority queue occupancy is greater than the ECN marking threshold k, the switch sets the Congestion Experienced (CE) codepoint of the dequeued normal packets. Otherwise, the normal packets are not marked. This scheme ensures that the receivers are quickly notified of the congestion state in low-priority queue. Furthermore, the cross-queue marking can be easily implemented using ECN, a built-in function of most commodity switches.

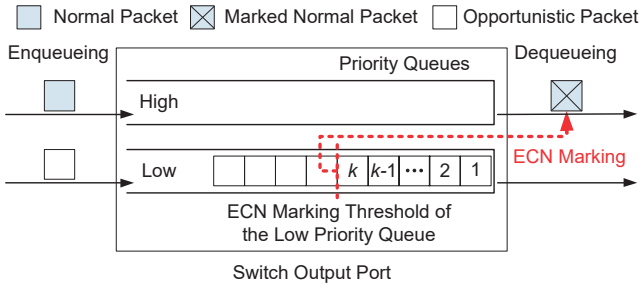


Fig. 6: Cross-queue ECN marking

It is important to set the appropriate cross-queue marking threshold k to make a tradeoff between network utilization and limited queueing delay of opportunistic packets. If the value of k is too small, the opportunistic packets may under-utilize the available bandwidth due to premature marking of normal packets to stop sending opportunistic packets. As a result, the opportunistic packets cannot fill up the free bandwidth during the receiver-driven transmission. On the contrary, a too large value of k may lead to large queueing delay and even packet loss. Therefore, we set the value of k as follows.

Let C and MSS denote the bottleneck link capacity and the TCP segment size, respectively. RPO measures the output rate C_d of normal packets and updates k in every time period t. To fully utilize the spare bandwidth with opportunistic packets, the following equation should be satisfied $k \geq \frac{(C-C_d)\times t}{MSS}$.

Finally, RPO sets k as the minimum value (i.e. $k = \frac{(C-C_d)\times t}{MSS}$) to reduce the queueing delay for opportunistic packets. The value of k is periodically updated according to the remaining bandwidth at the switch output port. To estimate the available bandwidth quickly and accurately, the updating period of t is set to the round trip time (i.e. 50μs) [24].

C. Grant Generation and Marking

At the receiver, once a normal packet or an opportunistic packet arrives, a corresponding grant is generated and sent back to the sender to trigger new packets. Consequently, the sender naturally knows how many packets can be sent without causing network congestion, rather than reactively reducing sending rate after congestion occurs. Once receiving a normal packet that has a marked CE codepoint, the receiver sets the ECN-Echo flag in the corresponding grant packets to notify the sender to stop sending opportunistic packets. Thus, by controlling data transmission rate, RPO is able to proactively control congestion and obtain ultra-low queueing delay.

RPO detects packet loss at the receiver side. In order to submit packets to the application layer in a timely manner, the receiver uses a fine-grained timer to detect lost packets. Specifically, after a grant has been sent, if the receiver has not received the corresponding packet after timeout (default $3\times RTT$ [4]), the grant will be retransmitted. For lost normal packets, RPO sender retransmits them with high priority. For lost opportunistic packets, if the maximum sequence number of transmitted normal packets is larger than the smallest sequence number of transmitted opportunistic packets, the lost packets are retransmitted quickly as high-priority normal packets, otherwise they are still retransmitted with low priority. In addition, since the opportunistic packets with large sequence number introduce out-of-order arrivals, RPO uses the re-sequencing buffer to absorb the disordered packets at the receiver [25].

V. Model Analysis

RPO uses low-priority opportunistic packets to seize the spare bandwidth without increasing additional queueing delay to the high-priority normal packets. Compared to the existing receiver-driven transmission protocols, RPO achieves both low latency and high link utilization. Here, we construct the theoretical model to analyze the performance gains of RPO in terms of link utilization and flow completion time.

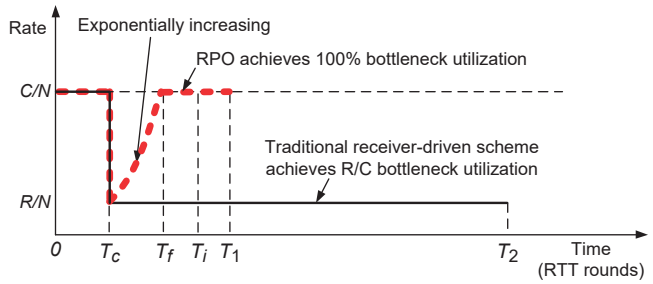


Fig. 7: RPO vs. Traditional receiver-driven scheme

Fig. 7 illustrates the advantage of RPO. Consider that N flows fairly sharing a bottleneck link with capacity of C. At time T_c, the available bandwidth of bottleneck link is reduced from C to R due to congestion. Then, the sending rate of each flow decreases from $\frac{C}{N}$ to $\frac{R}{N}$.

RPO leverages opportunistic packets to make full use of the free bandwidth. Specifically, one grant triggers a normal packet and an opportunistic packet if there is spare bandwidth. Similar to the slow-start process in the traditional TCP, the total number of normal and opportunistic packets increases exponentially until RPO achieves 100% bottleneck utilization at time T_f. The value of T_f in RTT rounds is calculated as

$$T_f = \lceil \log_2 \frac{C}{R} \rceil + T_c. \quad (1)$$

After T_f, RPO fully utilizes all of the available bandwidth. We assume that the flow size is S. Then we have

$$S = \frac{C}{N} \times T_c + \frac{R}{N} \times (2^{T_f - T_c} - 1) + \frac{C}{N} \times (T_1 - T_f). \quad (2)$$

Then, we obtain the flow completion time T_1 of RPO as

$$T_1 = T_f + \frac{N \times S - C \times T_c - R \times (2^{T_f - T_c} - 1)}{C}. \quad (3)$$

Under the existing receiver-driven transport protocols, the sending rate R cannot be increased due to the conservativeness of flow control mechanism. Thus, the flow is completed at time T_2, which is calculated as

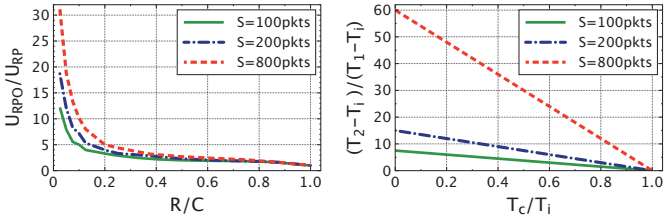
$$T_2 = \frac{S - \frac{C}{N} \times T_c}{\frac{R}{N}} + T_c. \quad (4)$$

Let U_{RPO} and U_{RP} denote the bottleneck utilization ratios of RPO and the traditional receiver-driven protocols, respectively. Then we get the utilization gain U_{gain} as

$$U_{gain} = \frac{T_2}{T_1} = \frac{\frac{S - \frac{C}{N} \times T_c}{\frac{R}{N}} + T_c}{T_f + \frac{N \times S - C \times T_c - R \times (2^{T_f - T_c} - 1)}{C}}. \quad (5)$$

Assuming that the flow does not experience congestion, the ideal flow completion time is $T_i = N \times \frac{S}{C}$. The gain of flow completion time $FCT_{gain} = \frac{T_2 - T_i}{T_1 - T_i}$, which is calculated as

$$FCT_{gain} = \frac{\frac{S - \frac{C}{N} \times T_c}{\frac{R}{N}} + T_c - N \times \frac{S}{C}}{T_f + \frac{N \times S - C \times T_c - R \times (2^{T_f - T_c} - 1)}{C} - N \times \frac{S}{C}}. \quad (6)$$



(a) Increasing $\frac{R}{C}$ (b) Increasing $\frac{T_c}{T_i}$

Fig. 8: Utilization and FCT gain with varying $\frac{R}{C}$ and $\frac{T_c}{T_i}$

Fig. 8 shows the theoretical gain of bottleneck utilization and FCT with different flow sizes. The link capacity C is 10Gbps and the round-trip time is 50μs. We set the number of flows N to 1 and the congestion time T_c to 0. In Fig. 8 (a), it is shown that when the ratio of $\frac{R}{C}$ increases, the gain of utilization decreases in RPO due to less spare bandwidth. In Fig. 8 (b), the sending rate R is set to 20 packets per round-trip time. The results show that the improvement of flow completion time increases with larger S under a certain $\frac{T_c}{T_i}$. Similarly to Fig. 8 (a), with the increasing value of $\frac{T_c}{T_i}$, the FCT gain of RPO decreases due to fewer opportunities to increase utilization.

VI. Implementation

We implement RPO by using Intel DPDK [19], which allows the network stack to bypass OS kernel and directly communicate with NIC to accelerate packet processing.

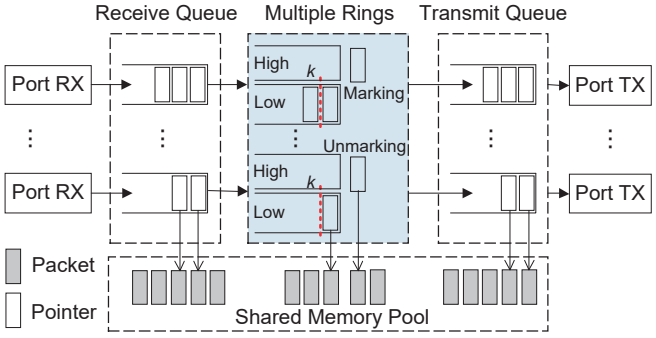


Fig. 9: RPO's DPDK Implementation at switch

Fig. 9 illustrates the architecture of RPO's DPDK implementation at switch, which consists of the receiving, marking and transmitting components. Specifically, when a packet arrives at the receiving ports, it is transmitted to the corresponding receive queue, which is allocated and set up by $rte\_eth\_rx\_queue\_setup()$ function. Each receive queue is served by an individual CPU to guarantee fast packet processing. Then the packet is enqueued into a logic queue, which is created by $rte\_ring\_create()$ function. In RPO, each port has a high and a low priority logical queues. When the low-priority queue length obtained by the $rte\_ring\_count()$ function exceeds the marking threshold k, the packet is marked to indicate under-utilization. Then the packet is delivered to the transmit queue of the corresponding output port by using $rte\_eth\_tx\_buffer()$ function. Note that the above DPDK processing only manipulates the packet pointer. Finally, the real packet described as rte_mbuf struct is driven from the shared memory pool.

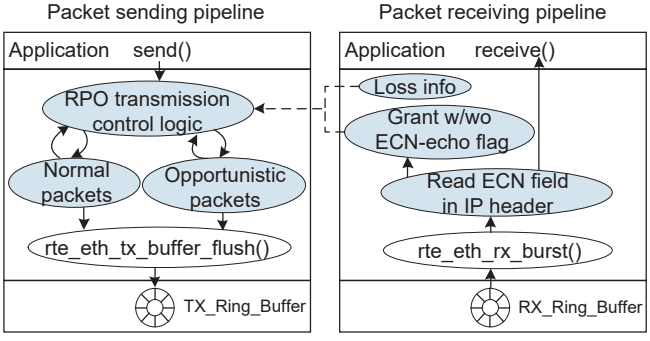


Fig. 10: RPO's DPDK Implementation at hosts

Fig. 10 shows the architecture of RPO's DPDK implementation at hosts, which consists of the packet sending and

receiving pipelines. At the sender, application starts data transmission by calling the $send()$ function. The RPO transmission control logic checks the ECN-Echo flag in each grant sent from the receiver, controls the triggering of normal and opportunity packets, and assigns high and low priority for the packets. Then the packets are sent out to the TX Ring buffer of NIC by calling $rte\_eth\_tx\_buffer\_flush()$ function.

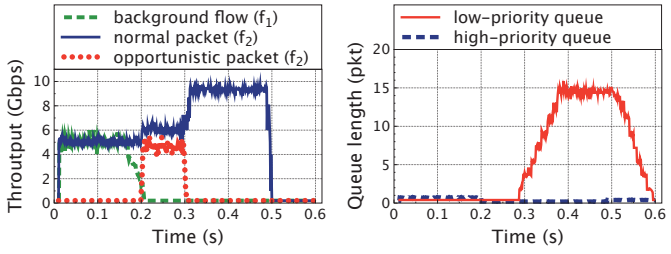
At the receiver, packets are retrieved from the RX Ring buffer of NIC by using $rte\_eth\_rx\_burst()$ function. For each arrival data packet, according to the ECN fields in the IP header, a corresponding grant with or without ECN-Echo flag is generated and sent back to the sender. The data packets are directly delivered to the application by calling a $receive()$ function. The packet loss information is fed back to the sender.

In addition, RPO can also be feasibly deployed on the P4 programmable switch since only few matching tables and static random access memory (SRAM) are required. Specifically, RPO only needs 4 match action tables including packet priority identifying, queue length reading, ECN marking and packet forwarding to implement per-packet processing. Each packet only adds 2-bit metadata for 1-bit priority marking and 1-bit ECN marking. RPO records the ECN marking threshold, the low-priority queue length, the number of packets forwarded in a certain period at the switch.

VII. Testbed Evaluation

In this section, we use a real 10Gbps testbed to evaluate the applicability and effectiveness of RPO. The details of testbed are the same as that in Section II-C. We compare RPO against pHost (in Section II), Homa and NDP. The protocols are implemented via Intel DPDK and all parameters are set to the default values in the related literatures [10], [7], [8].

We firstly evaluate the basic performance of RPO in a well-known dumbbell scenario with 2 senders and 2 receivers sharing a single 10Gbps bottleneck link, as shown in Fig. 2 (a). We set the buffer size to 32 packets. Throughput and queue length of switch buffer are shown in Fig. 11 (a) and (b), respectively.



(a) Normalized throughput (b) Queue length

Fig. 11: Normal and opportunistic packets transmitted in RPO

At the beginning, each of two flows starts with a BDP bytes (i.e. ∼80 packets) to compete for the bottleneck link. For flow f_2, 40 normal packets arrive at the receiver and 25 normal packets are dropped due to buffer overflow. In this case, the ECN marking threshold k of the low-priority queue is reduced to 0 due to full link utilization. Since all normal packets are ECN-marked, none of opportunistic packets are generated. When the background flow f_1 enters OFF state at 0.2s, since the link bandwidth becomes available, RPO increases the ECN marking threshold k to allow the sender to generate more opportunistic packets. From 0.2s, with the exponentially increasing total number of normal and opportunistic packets, f_2 quickly seizes all bottleneck bandwidth.

At 0.3s, once the number of normal packets increases to 80, which is enough to fill up the bottleneck bandwidth, 15 opportunistic packets are queued at the low-priority queues as shown in Fig. 11 (b). Then k is updated to 0 and the opportunistic packets are stopped. The low-priority queue's length maintains at 15 packets to avoid low link utilization. The result in Fig. 11 (b) shows that, under dynamic traffic, RPO is able to achieve low queueing delay for normal packets while using opportunistic packets to obtain high link utilization.

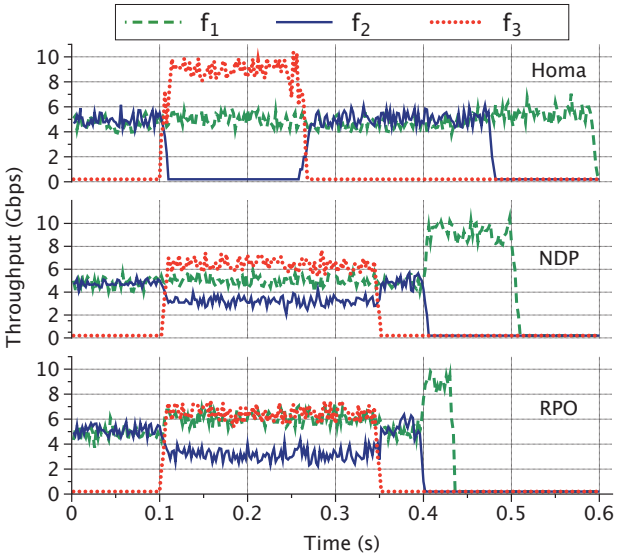


Fig. 12: Receiver-driven transmission under dynamic traffic

Next, we compare the RPO performance with the state-of-the-art receiver-driven transport protocols. We test three receiver-driven flows in the multi-bottleneck topology as shown in Fig. 1 (a). The experimental settings are as same as that in Section II-C. Flow f_1, f_2 and f_3 are sent to different receivers. All flows exhibit ON/OFF pattern in transmission.

Fig. 12 shows the test results of Homa, NDP and RPO. At the beginning, f_1 and f_2 achieve their fair bandwidth allocations of the 10Gbps bottleneck link. At 0.1s, a new flow f_3 starts transmission and competes with f_2. The round trip time of f_2 increases under multiple bottlenecks. Homa prioritizes the shortest flow f_3 to fully use the bottleneck link through the highest priority queue. Under NDP and RPO, f_1 and f_3 fairly share the bandwidth. The total link utilizations of f_1 and f_2 are around 50% and 80% in Homa and NDP, respectively. However, even under the impact of RTT variation in f_2, RPO is still able to fully utilize bottleneck bandwidth by using opportunistic packets from f_1.

In Homa, after f_2 finishes at 0.48s, f_1 only utilizes 50% of the bottleneck bandwidth because of its conservativeness in seizing free bandwidth. NDP saturates the bottleneck link after

0.4s. Since NDP trims the payloads of packets when the switch queue is filled to avoid packet loss, NDP fully uses bandwidth after f_2 stops. RPO uses opportunistic packets to make use of the spare bandwidth and achieve full link utilization. Overall, since always achieving high link utilization under ON/OFF traffic and RTT variation, RPO reduces the AFCT by 15%, 14% and 8% over pHost, Homa and NDP, respectively.

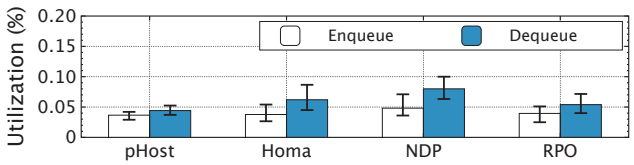


Fig. 13: CPU overhead

Finally, we evaluate the system overhead of RPO at switch. The link utilization estimation and packets marking function of RPO are implemented using DPDK, causing overhead in the enqueue and dequeue modules. In this test, we measure the average CPU utilization ratio at switches. As shown in Fig. 13, RPO consumes slightly more resources than pHost due to the cross queue marking. However, RPO achieves lower CPU utilization in both enqueue and dequeue processes compared with Homa and NDP due to the simpler marking and forwarding operations.

VIII. Simulation Evaluation

A. Performance under realistic workload

We conduct large-scale NS2 simulations to evaluate RPO performance in the typical datacenter applications. We choose four realistic workloads, namely web server (WSv), cache follower (CF), web search (WSc) and data mining (DM) [7], [10], [9]. The flow size distribution and average flow size for each workload are shown in Table 1.

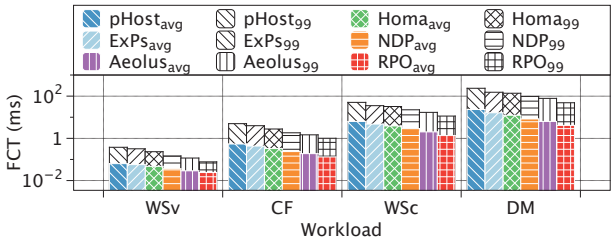
TABLE I: Flow size distribution of realistic workload.

	Web Server	Cache Follower	Web Search	Data Mining
0-100KB	81%	53%	62%	83%
100KB-1MB	19%	18%	18%	8%
>1MB	0	29%	20%	9%
Average flow size	64KB	701KB	1.6MB	7.41MB

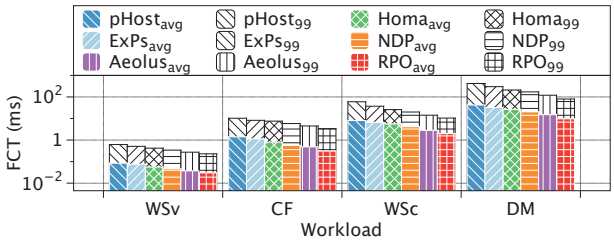
We use the common leaf-spine topology with 10 leaf and 8 core switches [16], [21]. Each leaf switch connects to 40 hosts and the whole network has 400 hosts connected by 40Gbps links. The round-trip propagation delay is 25μs and the switch buffer size is 256 packets. The traffic is generated by randomly starting flows via a Poisson process between random pair of hosts. We change the traffic load from 0.1 to 0.6 and compare RPO with pHost, Homa, NDP and Aeolus, which is integrated with the latest receiver-driven protocol Homa.

In Fig. 14, each bar indicates the average and 99^{th} percentile FCTs of all flows. RPO reduces AFCT across different workloads with varying load compared to the other schemes. Moreover, RPO obtains better improvement of AFCT under the workloads with larger average flow size, because RPO has

(a) 0.1 load



(b) 0.3 load



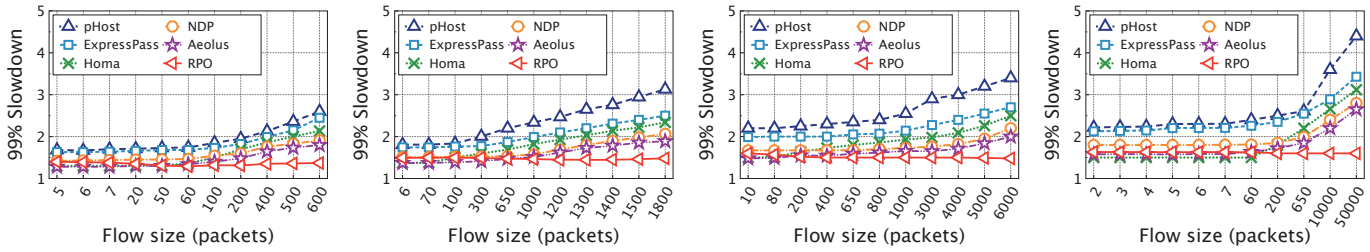
(c) 0.5 load

(d) 0.6 load

Fig. 14: AFCT and 99^{th} percentile FCT. WSv, CF, WSc and DM stand for Web Server, Cache Follower, Web Search and Data Mining, respectively.

more opportunities to increase link utilization. Specifically, in data mining, RPO reduces the AFCT by 52%, 42%, 38%, 27% and 23% under 0.6 load over pHost, ExpressPass, Homa, NDP and Aeolus, respectively. NDP performs better than pHost and Homa. The reason is that, once the switch queue length exceeds a small threshold, NDP trims payloads of packets to avoid packets loss and quickly recover the sending rate after congestion is alleviated. In addition, Homa obtains lower AFCT than pHost by employing priority queues to guarantee low latency of short flows. ExpressPass (ExPs) performs worse than Homa due to waiting for credits to start data transmission in the first RTT and bandwidth wastage when there is no data to send. With scheduled-packet-first mechanism, Aeolus guarantees low tail latency and quickly recovers the dropped unscheduled packets to reduce AFCT.

Fig. 14 shows the 99^{th} percentile FCT of all flows to



(a) Web Server (b) Cache Follower (c) Web Search (d) Data Mining

Fig. 15: The 99^{th} percentile slowdown

present the tail FCT. RPO performs well with varying load by using opportunistic packets to reduce FCT. Moreover, with the decrease of load, more free bandwidth appears, and RPO has more opportunities to speed up the flow transmission.

We also test the 99^{th} percentile slowdown as a function of flow size under 0.6 load. Fig. 15 shows the test results of slowdown, which is defined as the ratio of the observed FCT to the ideal FCT without any congestion. The X-axis are linear in the total number of flows (the first tick is 50% of all flows, then each tick is increased by 5% of all flows). The results show that RPO achieves lower slowdown than the other schemes in all workloads. Specifically, the slowdown improvement of RPO becomes higher with increasing flow size in the same workload. Across the different workloads, the slowdowns of all schemes become higher with increasing average flow size due to larger tail FCT of long flows. The reason is that larger flows under RPO have more opportunities to utilize free bandwidth, resulting in smaller slowdown.

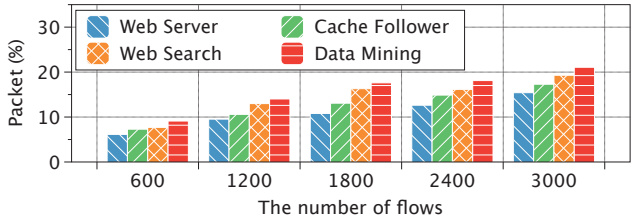


Fig. 16: The ratio of opportunistic packets in RPO

RPO utilizes the opportunistic packets to make full use of available bandwidth. However, since simultaneously transferring the normal and opportunistic packets, RPO potentially needs to reorder packets at the receiver side. The required reordering buffer size at the receiver is determined by two factors: the number of flows simultaneously arriving at the receiver and the number of buffered opportunistic packets per flow. In the following, we generate 600∼3000 flows with Poisson distributed inter arrival time. The maximum number of flows simultaneously arriving at one receiver is 100. Fig. 16 shows that, with larger number of flows, the ratio of opportunistic packets increases since RPO sends more opportunistic packets to improve link utilizations under the impacts of dynamic ON/OFF traffic and RTT variation.

As shown in Fig. 17, each bar indicates the results with different number of flows. RPO obtains higher bottleneck utilization than the other receiver-driven protocols by ratio-

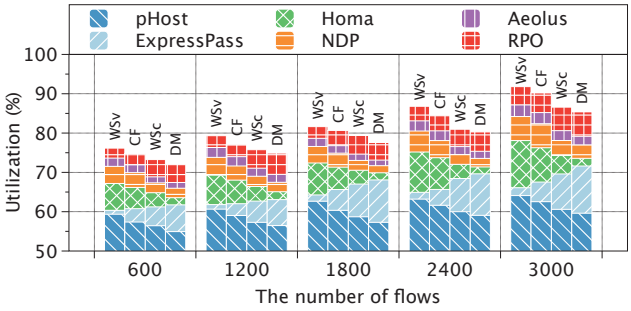


Fig. 17: Link utilization

nally using the low-priority opportunistic packets. Specifically, pHost, ExpressPass, Homa, NDP and Aeolus only obtain 56%, 63%, 65%, 67% and 69% link utilization in data mining workload with 1200 flows and RPO improves link utilization by 35%, 21%, 17%, 12% and 10% over pHost, ExpressPass, Homa, NDP and Aeolus, respectively. In addition, NDP also achieves good link utilization by pacing pull packets and trimming packet payload. Overall, the workloads with smaller average flow size have higher link utilization because the large flows have long FCTs and are more likely to experience throughput losses.

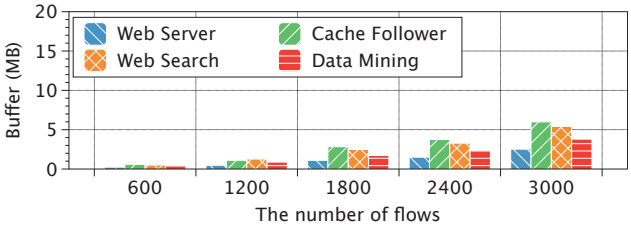


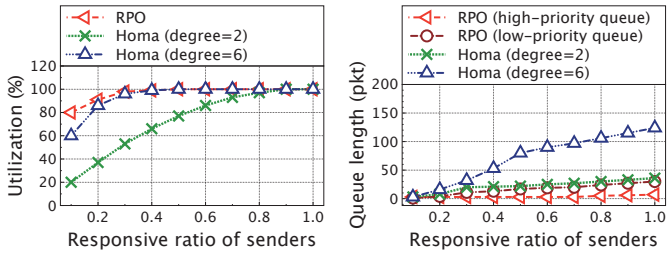
Fig. 18: Maximum required reordering buffer in RPO

Fig. 18 shows the maximum required buffer at a receiver in RPO. As the number of flows increases, more opportunistic packets are generated, resulting in larger reordering buffer space. Since cache follower and web search contain more long flows, these workloads have higher requirements of reordering buffer. Nonetheless, the buffer requirement is acceptable in all cases compared with the utilization improvements.

B. Performance in Many-to-many Communications

In this section, we conduct simulations in many-to-many scenario to compare RPO's opportunistic transmission method with Homa's overcommitment mechanism. We use leaf-spine topology with 3 leaf switches. The simulation settings are

same as in Section VIII-A. Each of the first two leaf switches connects with 10 senders, each of which establishes 2 connections with 2 receivers under the 3^{rd} leaf switch. We measure the maximum queue length and link utilization with gradually increased ratio of responsive senders from 0.1 to 1. We test Homa with overcommitment degree of 2 and 6. That is, each receiver concurrently sends multiple grants to 2 or 6 random senders. For RPO, each receiver sends grants to all senders. We repeat the test for 50 times to get the average results.



(a) Bottleneck utilization (b) Maximum queue length

Fig. 19: Varying the ratio of responsive senders

As shown in Fig. 19 (a), the link utilization increases with more responsive senders. Since the opportunity transmission quickly uses the free bandwidth, RPO sustains high link utilization with different ratios of responsive senders. For Homa, if the degree of overcommitment is 2, the link usage is degraded when some senders are unresponsive. By increasing the degree of overcommitment to 6, Homa improves the link utilization. However, this improvement comes at the cost of large queueing delay. In Fig. 19 (b), the maximum queue length increases with increasing degree of overcommitment in Homa. In RPO, the queue length of high-priority normal packets is always close to zero. Only the low-priority opportunistic packets have small queue buildups, which have no negative impact on normal packets.

In short, it is hard for Homa to obtain a good balance between low queueing delay and high link utilization by setting a proper degree of overcommitment under dynamic traffic. RPO is able to flexibly achieve ultra-low latency for normal packets and high utilization via opportunistic transmission.

IX. Related Works

At the sender side, DCTCP [1], TCN [26] and MQ-ECN [27] leverages ECN marking to control the queueing delay. D^{2}TCP [2] and D^3 [3] modulate the sending rate to minimize the deadline-missing ratio. MPTCP [28] transmits subflows on parallel paths to improve link utilization. pFabric [4] and PIAS [18] schedule packets based on the priorities at switches. HULL [29] uses phantom queues to deliver congestion signal. ECN$^\#$ [30] aggressively marks packets to avoid buffer overflow and conservatively marks packets in order not to adversely affect throughput. DCQCN [5], TIMELY [6], DX [31], Swift [32] and Tagger [33] control congestion well in lossless networks. Unfortunately, they still suffer from buffer overflow under highly bursty traffic.

As a rate control scheme, PDQ [34] calculates flow rate to enable flow preemption. Fastpass [35] uses a centralized arbiter to determine the transmission time for each packet. Karuna [36] focuses on scheduling a mix of flows with and without deadlines by controlling the flow rate. TFC [24] allocates the link bandwidth for each flow. However, the rate calculating and centralized scheduling in the above schemes may be slow for small and tiny flows at datacenter scale.

Recently proposed HPCC [37] precisely controls congestion by leveraging in-network telemetry (INT) to obtain accurate link load information to achieve near-zero queueing delay and high throughput. Although HPCC effectively reduces the flow completion time, it requires dedicated hardware support and introduces traffic overhead for congestion feedback.

Recent receiver-driven transmission aim for near-zero queueing delay. pHost [10] uses token packets to decouple scheduling from the network fabric. NDP [8] controls the incoming traffic at the receiver by using accurate pacing of pull packets. ExpressPass [9] allocates bandwidth by shaping the flow of credit packets at the switch. Homa [7] leverages priority queues and the receiver-driven SRPT policy to ensure low latency for short messages. Aeolus [12] is integrated into the receiver-driven schemes to eliminate extra delay in the first RTT. Though achieving low latency, these approaches experience low link utilization under highly dynamic traffic, resulting in suboptimal network performance.

In contrast with the above transmission schemes, our design RPO works through a different perspective: RPO uses low-priority opportunistic packets to fill spare bandwidth immediately without affecting the high-priority normal packets to simultaneously achieve low latency and high link utilization. Compared with RC3 [38], which reversely sends low-priority packets to fill the free bandwidth left by the default TCP packets, RPO uses explicit network feedback from ECN marking to control the queueing delay and traffic overhead of opportunistic packets at the end hosts, without centralized scheduler or complex rate allocation.

X. Conclusion

We propose a simple yet effective receiver-driven transport protocol RPO that uses low-priority opportunistic packets to utilize the spare bandwidth without introducing queueing delay of high-priority normal packets. Specifically, according to the low-priority queue length, RPO adaptively adjusts the number of opportunistic packets to ensure the high link utilization for the receiver-driven transmission across a wide variety of dynamic traffic workloads. The results of real testbed and large-scale simulations demonstrate that RPO significantly improves the link utilization by up to 35% over pHost, ExpressPass, Homa and NDP. RPO remarkably reduces the average flow completion time by up to 52% compared with these state-of-the-art receiver-driven transmission protocols.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (62132022, 61872387, 62102046), MOE Start-up Research Grant (SRG ISTD 2020 159), Hong Kong RGC GRF-16215119.

REFERENCES

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data Center TCP (DCTCP). In Proc. ACM SIGCOMM, 2010.

[2] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware Datacenter TCP (D2TCP). In Proc. ACM SIGCOMM, 2012.

[3] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better Never Than Late: Meeting Deadlines in Datacenter Networks. In Proc. ACM SIGCOMM, 2011.

[4] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pFabric: Minimal Near-optimal Datacenter Transport. In Proc. ACM SIGCOMM, 2013.

[5] Y. Zhu, H. Eran, D. Firestone, et al. Congestion Control for Large-scale RDMA Deployments. In Proc. ACM SIGCOMM, 2015.

[6] R. Mittal, V. T. Lam, N. Dukkipati, et al. Timely: Rtt-based Congestion Control for The Datacenter. In Proc. ACM SIGCOMM, 2015.

[7] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout. Homa: A receiver-driven Low-latency Transport Protocol Using Network Priorities. In Proc. ACM SIGCOMM, 2018.

[8] M. Handley, C. Raiciu, A. Agache, A.Voinescu, A. Moore, G. Antichi, and M. Wójcik. Re-architecting Datacenter Networks and Stacks for Low Latency and High Performance. In Proc. ACM SIGCOMM, 2017.

[9] I. Cho, K. Jang, and D. Han. Credit-scheduled Delay-bounded Congestion Control for Datacenters. In Proc. ACM SIGCOMM, 2017.

[10] P. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker. pHost: Distributed Near-optimal Datacenter Transport Over Commodity Network Fabric. In Proc. ACM CoNEXT 2015.

[11] J. Hu, J Huang, Z. Li, et al. AMRT: Anti-ECN Marking to Improve Utilization of Receiver-driven Transmission in Data Center. In Proc. ACM ICPP, 2020.

[12] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan, and Y. Wang. Aeolus: A Building Block for Proactive Transport in Datacenters. In Proc. ACM SIGCOMM, 2020.

[13] S. Hu, W. Bai, B. Qiao, K. Chen, and K. Tan. Augmenting Proactive Congestion Control with Aeolus. In Proc. ACM APNet 2018.

[14] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Data Center Traffic: Measurements & Analysis. In Proc. ACM IMC, 2009.

[15] T. Benson, A. Akella, and D. Maltz. Network Traffic Characteristics of Data Centers in The Wild. In Proc. IMC, 2010.

[16] J. Hu, J. Huang, W. Lv, Y. Zhou, J. Wang, and T. He. CAPS: Coding-based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center. In Proc. IEEE INFOCOM, 2018.

[17] J. Hu, J. Huang, W. Lv, Y. Zhou, J. Wang and T. He. CAPS: Coding-based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center. IEEE/ACM Transactions on Networking, 2019, 27(6): 2338-2353.

[18] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang. Information-agnostic Flow Scheduling for Commodity Data Centers. In Proc. USENIX NSDI, 2015.

[19] DPDK Plane Development Kit, Intel DPDK, 2019.

[20] Y. Cui, S. Xiao, X. Wang, Z. Yang, S. Yan, C. Zhu, X. Li, and N. Ge. Diamond: Nesting the Data Center Network with Wireless Rings in 3-d Space. IEEE/ACM Transactions On Networking, 2017, 26(1): 145-160.

[21] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury. Resilient Datacenter Load Balancing in The Wild. In Proc. ACM SIGCOMM, 2017.

[22] S. Ghorbani, Z. Yang, P. Godfrey, Y. Ganjali, and A. Firoozshahian. DRILL: Micro Load Balancing for Low-latency Data Center Networks. In Proc. ACM SIGCOMM, 2017.

[23] B. Stephens, A. L. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter. Practical DCB for Improved Data Center Networks. In Proc. IEEE INFOCOM, 2014.

[24] J. Zhang, F. Ren, R. Shu, and P. Cheng. TFC: Token Flow Control in Data Center Networks. In Proc. ACM EuroSys, 2016.

[25] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella. Presto: Edge-based Load Balancing for Fast Datacenter Networks. In Proc. ACM SIGCOMM, 2015.

[26] W. Bai, K. Chen, L. Chen, C. Kim, and H. Wu. Enabling ECN over Generic Packet Scheduling. In Proc. ACM CoNEXT, 2016.

[27] W. Bai, L. Chen, K. Chen, H. Wu. Enabling ECN in Multi-Service Multi-Queue Data Centers. In Proc. USENIX ATC, 2016.

[28] 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, 2011.

[29] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda. Less is More: Trading a Little Bandwidth for Ultra-low Latency in The Data Center. In Proc. USENIX NSDI, 2012.

[30] J. Zhang, W. Bai, K. Chen. Enabling ECN for datacenter networks with RTT variations. In Proc. ACM CoNEXT, 2019.

[31] C. Lee, C. Park, K. Jang, S. Moon, and D. Han. Accurate Latency-based Congestion Feedback for Datacenters. In Proc. USENIX ATC, 2015.

[32] G. Kumar, N. Dukkipati, K. Jang, et al. Swift: Delay is Simple and Effective for Congestion Control in The Datacenter. In Proc. ACM SIGCOMM, 2020.

[33] S. Hu, Y. Zhu, P. Cheng, et al. Tagger: Practical PFC deadlock prevention in data center networks. In Proc. ACM CoNEXT, 2017.

[34] C. Y. Hong, M. Caesar, and P. B. Godfrey. Finishing Flows Quickly with Preemptive Scheduling. In Proc. ACM SIGCOMM, 2012.

[35] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal. Fastpass: A Centralized "zero-queue" Datacenter Network. In Proc. ACM SIGCOMM, 2014.

[36] L. Chen, K. Chen, W. Bai, and M. Alizadeh. Scheduling Mix-flows in Commodity Datacenters with Karuna. In Proc. ACM SIGCOMM, 2016.

[37] Y. Li, R. Miao, H. H. Liu, et al. HPCC: High Precision Congestion Control. In Proc. ACM SIGCOMM, 2019.

[38] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Recursively Cautious Congestion Control. In Proc. USENIX NSDI, 2014.