Revision Plan for SIGCOMM'24 Paper #293

Dear Prof. Praveen Kumar,

Thank you for shepherding our paper (*#293 PPT: A Pragmatic Transport for Datacenters*). We carefully went through the reviews and outlined our revision plan.

Please let us know if you have any feedback. Thank you for your time and help to improve the quality of our paper!

Best Regards,
Authors of SIGCOMM'24 Paper #293

I Top-level Concerns:

Concern 1: *Disconnect between the proposed goals vs the evaluation. The paper pitches itself to deliver comparable performance to proactive transports while being readily deployable. But the evaluations do not necessarily back this up.*

Response 1: Thanks for the reviewer's comments. The proactive transports proactively allocate the bottleneck link bandwidth and prioritize it to flows with fewer remaining bytes. Proactive transport typically displays the performance of the large and small flows, respectively. To support our claim of "comparable performance to proactive transports." First, we compare the overall performance of the different transports through the overall average FCT. Second, we show the average FCT and 99th percentile tail FCT for small flows to illustrate the ability of different transport to avoid small flow packet scheduling and queuing delays. Finally, we show the average FCT for large flows to display the gap between different schemes' ability to utilize the spare bandwidth gracefully. These are the performance metrics that proactive transports focus on and work with. Therefore, we use them as the primary performance metrics for simulation and testbed and conclude they are comparable to performance in proactive transport. To show more details, we have shown the lines of code required to be modified to deploy PPT and Homa-Linux, respectively (Tables 3 and 4), and the CPU usage of PPT and DCTCP under different loads (Figure 21) in the appendix. We believe that the above performance metrics support our proposed goals well.

Concern 2: *There's lacking discussion on other related work such as PCC Proteus.*

Response 2: Thanks for the reviewer's comments. PCC Proteus divides flows into primary and scavenger flows according to different application requirements and proactively reduces scavenger flow bandwidth to minimize the impact on the primary flow, thus improving the overall user experience. By contrast, PPT is insensitive to the application type. In PPT, all flows equally utilize the bandwidth. PPT's proposed purpose of buffer-aware flow scheduling is to ensure that small flows bypass queued packets of most large flows in the network to optimize the FCT. Therefore, PPT and PCC Proteus are two lines of work due to different

flow behaviors and optimization goals. We will add a discussion of PCC Proteus and other related work in Appendix B.

Concern 3: *The paper needs to discuss more insights on how the system is able to deliver the performance benefits.*
Response 3: Thanks for the reviewer's comments. In the overall average performance analysis in Section 6.2, we mentioned that the PPT benefits from gracefully utilizing the spare bandwidth without causing bandwidth waste or sending opportunistic packets too aggressively. To further explain the insights, we add a remark at the end of Appendix C.1 to analyze how the PPT achieves its superior performance in more detail.

Concern 4: *Other claims such as "integration with other transports is easy" need to be further justified.*
Response 4: Thanks for the reviewer's comments. We implemented a delay-based transport conceptually equivalent to Swift in the ns3 simulator. It doesn't include the component in Swift that handles endpoint congestion because (1) PPT doesn't focus on endpoint congestion, and (2) the ns3 simulator hardly simulates endpoint congestion as well as it does. We compared this variant with the original delay-based transport and plotted the result in Figure 26. We find that when incorporating PPT's design with the original delay-based transport, the overall average FCT, the average//tail FCT of small flows, and the average FCT of large flows can be reduced by 16.7%, 56.5%/72.1%, and 11%, respectively. To show more details, we plan to add a more detailed description of this variant in the corresponding section of Appendix C.3. We also plan to add a remark paragraph at the end of the chapter to discuss how delay-based transport can benefit from the design of the PPT in further depth.

Concern 5: *Missing details on evaluation setup and parameters.*
Response 5: Thanks for the reviewer's comments. We record the maximum value of the window from the beginning of the flow until now as MW (Maximum Window). Section 2.3 first shows how much DCTCP can benefit from padding the spare bandwidth to MW and plots the results in Figure 2. Moreover, we show that padding the spare bandwidth to MW is just suitable without wasting bandwidth or causing congestion throughout the experiment, and we plot the results of our experiment in Figure 3. Another question is about specific TCP send buffer thresholds. To reproduce the behavior of the actual deployed application as much as possible, we selected specific application traces for the Memcached application and the web server application, respectively. The flow size distribution in specific application traces is concentrated in particular intervals, e.g., there is no flow in the ETC trace of the paper[8] that exceeds 100KB. Under such traces, we set different thresholds depending on the traces to verify the effectiveness of the buffer-aware flow identification. In the large-scale simulation and testbed, the TCP send buffer threshold is set to the fixed value of 100KB, with outstanding results. To evaluate if this can benefit PPT's performance, we construct a PPT variant that turns off this approach and considers all flows non-identified. We plot the results in Figure 20, showing that this improves the small flow performance as expected. In summary, the TCP send buffer threshold does not assume that the application type and flow size distribution are known a priori and effectively improves small flow performance.

II Other Concerns

1 Reviewer #293A

A.1: Reviewer Comment: *The number of used queues could be clarified: p. 6 mentions two such queues (a high-priority and a low-priority queue with a different λ. But then it mentions that when queueing the opportunistic (presumably low-priority) packets, this may hurt normal (presumably high-priority) packets, suggesting they share the queue. Then, p. 8 mentions 8 priorities, such that "switches can use strict priority to dequeue packets", implicitly pointing to 8 different queues. Then, p.9 mentions 2 queues again.*

Response A.1: Thanks for the reviewer's comments. Actually, PPT uses eight strict priorities in the switch. To prevent HCP traffic harmed by LCP, priorities 0$\sim$3 are used to transmit HCP packets and priorities 4$\sim$7 are used to transmit LCP packets. Therefore, the *high-priority* and *low-priority* mentioned in p.6 as well as p.9 refer to priorities 0$\sim$3 and priorities 4$\sim$7 in the switch, respectively. To avoid ambiguity for the reader, we will include footnotes in Section 3.2 for explanation.

A.2: Reviewer Comment: *In addition, PPT seems to couple each LCP flow with an HCP flow, by defining the window size of the LCP flow using the window size of the associated HCP flow. But what if there are no HCP flows now? Are LCP flows stuck? Or more generally, what if the numbers of flows are not equal? Or what if they are equal but the flows are destined to different destinations? How does this coupling help?*

Response A.2: Thanks for the reviewer's comments. Although very sorry, the reviewer seems to have misinterpreted PPT's strategy. At the beginning of Section 3, we mentioned that "LCP sends opportunistic packets from the tail end". Therefore, both HCP and LCP send data in same flow. However, many of the review's comments were based on LCP and HCP transmitting data from different flows, which made them difficult for us to answer.

2 Reviewer #293B

B.1: Reviewer Comment: *During LCP loop initialization for case 1, it is unclear how PPT computes BDP in the first RTT. As you mention, DCTCP is still probing for available bandwidth.*

Response B.1: Thanks for the reviewer's comments. Since the NIC rate can be read via system call and the datacenter network topology is relatively fixed, we assume that the RTT is known a priori. By NIC rate times RTT, we assume that the host knows the BDP by advance.

B.2: Reviewer Comment: *I wonder if tracking α_{min} can lead to underutilization due to network dynamics. Suppose we have a long flow which experiences transient congestion at the beginning (say due to an incast); the trans ient congestion causes α_{min} to be $>$ 0.5 for the flow. After the transient congestion subsides, the large flow is unable to utilize any spare bandwidth arising from Case 2.*

Response B.2: Thanks for the reviewer's comments. During the process of congestion subsiding, the switch queue length is reducing, the percentage of ECN marked packets for this long flow is decreasing, as well as the value of α. As the congestion subsides completely, the proportion of ECN-marked packets should be 0. PPT initializes LCP for the long flow whenever α takes the minimum value. At this point, α must be less than 0.5, which makes this large flow utilize the spare bandwidth.

B.3: Reviewer Comment: *It is unclear how PPT deals with fairness as flows arrive and leave. Suppose two large flows sharing a common bottleneck link arrive one after the other. The latter flow might see much lower W_{max} (50%?) than the earlier flow. In every LCP loop initialization (case 2), the latter flow would*

compute a lower value of initial congestion window and send less packets via LCP compared to the earlier flow.

Response B.3: Thanks for the reviewer's comments.

B.4: Reviewer Comment: *As the paper targets the issue of underutilizing high datacenter bandwidth, I wonder if some of the simulations could have been done at higher line rates (400+ Gbps) to demonstrate the problem more clearly.*

Response B.4: Thanks for the reviewer's comments. We will add a new simulation experiment with topology consists of 144 servers, 9 leaf switches and 4 spine switches, with the host and core links operated at 400 and 1000Gbps, respectively. We will plot the results in the appendix.

B.5: Reviewer Comment: *What are the overheads of PPT?*

Response B.5: Thanks for the reviewer's comments. Actually, we measure the kernel space CPU overhead of PPT and DCTCP in the our testbed. We plot the results in Figure 21. For more details can read Appendix C.1.

3 Reviewer #293C

C.1: Reviewer Comment: *It would be particularly useful to have an intuitive and straightforward example scenario where LCP can discover spare bandwidth other than during the slow start phase.*

Response C.1: Thanks for the reviewer's comments. Indeed, in section 2.3 we mention that *DCTCP marks ECN at the switch for arriving packets if queue occupancy exceeds a threshold K, and the sender cuts the window based on the fraction of ECN marked ACKs. So, when there are multiple concurrent flows, DCTCP may mark ECN for packets from many flows, which may cut windows simultaneously, thus causing a sudden drain on the switch buffer and leaving bandwidth underutilized.* To show this point, We run ns-3 simulations with two senders and one receiver sharing the bottleneck. We sample the bottleneck link utilization every 100us for 10ms when DCTCP enters a steady state. We plot the results in Figure 1 and show that leaving nearly half of the bandwidth under-utilized.

C.2: Reviewer Comment: *Figure 13 doesn't add up by itself. RC3 has the highest average FCT for small and large flows. However, its overall average is not the highest. Given that large and small flows account for 13% and 87% respectively, the overall average should be around 0.13 * 39.63 + 0.87 * 0.77 = 5.8? Or am I missing something?*

Response C.2: Thanks for the reviewer's comments. After careful checking of our data processing code we found that under Datamining workloads, RC3 would result in few flows not completing due to a large number of packet losses. Our code incorrectly used the program runtime as the flow completion time when calculating the overall average FCT, resulting in the actual calculated result being larger. We will fix the bug and update the correct result.

C.3: Reviewer Comment: *Page 12, can you give an intuition why limiting RC3's low priority queue doesn't help? It kind of contradicts the argument that aggressive low priority packets hinder high-priority packets being a problem if the high-priority queue has enough buffers?*

Response C.3: Thanks for the reviewer's comments. Although limiting the RC3 low-priority queue can reduce the LCP from sending too many opportunity packets somehow, the following reasons still lead to low performance. First, during the slow-start phase of the flow, limiting the buffer LCP could use will leave a large amount of bandwidth underutilized. Second, RC3 keeps the LCP open instead of intermittent

initialization as PPT, which causes more severe congestion during traffic bursts. Last but not least, RC3 lack of using in-network priority causes small flow packets are queued after LCP packets, increasing small flow latency. We will add the above analyses to our paper.

4 Reviewer #293D

D.1: Reviewer Comment: *I'm not sure if I fully understand some of the experimental results. To give an example, why is Homa's performance so poor for the testbed incast experiment? You mention that PPT is effective because it can recover quickly but why can't Homa?*

Response D.1: Thanks for the reviewer's comments. In fact, as we mentioned in Section 6.1.2, Homa-Linux sends BDP-sized unscheduled packets for each arriving flow, and excessive opportunity packets cause heavier congestion in incast scenarios. Furthermore, Homa-Linux is connectionless, so packet loss recovery is achieved through retransmission after timeout. Large packet loss in Incast scenarios further impairs its performance. We will add more detailed analyses in the paper

5 Reviewer #293E

E.1: Reviewer Comment: *In production systems, priority queues are scarce resource. Using only two HW queues for one application will be a luxury. How does PTP perform with 2 prio queues (one high, one low) compared to existing production solutions like Swift, HPCC, DCTCP?*

Response E.1: Thanks for the reviewer's comments. In fact, we construct a stricter variant of PPT in our paper that assigns all packets the same priority. We plot the results in Figure 19. We find that even under these conditions, PPT still achieves the optimal overall average FCT among NDP, Homa, Aeolus, DCTCP, and RC3.

E.2: Reviewer Comment: *Even the 'large-scale' simulations cover only 144 servers w/ 40 100G as core link speeds. This resembles DC topology of more a decade ago. With the trend of AI clusters growing to have 10s of Ks of GPUs, I believe at least 2K or more endpoints must be simulated to see the real impact on shallow switch.*

Response E.2 Thanks for the reviewer's comments. We will add a large-scale simulation experiment with 2k nodes to the appendix.

References

[1] V. Dukic, S. A. Jyothi, B. Karlas, M. Owaida, C. Zhang, and A. Singla, "Is advance knowledge of flow sizes a plausible assumption?" in *Proc. of USENIX NSDI*, 2019.

[2] A. Rashelbach, O. Rottenstreich, and M. Silberstein, "A computational approach to packet classification," in *Proc. of ACM SIGCOMM*, 2020.