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 Revision Guidance Response:

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. Firstly, our proposed goal is to explore a new transport that can achieve comparable performance to proactive transports while being readily deployable. To this point, we start with DCTCP and use LCP to utilize the available bandwidth gracefully. Moreover, we complement its design with buffer-aware flow scheduling to optimize the small flow's performance. Our analysis in Section 2.3 carefully explains how PPT's design matches the proposed goals. We then count the FCT of flows with different sizes in large-scale simulation experiments and testbed to compare the performance of PPT with other transports. PPT achieves lower overall average FCT than the proactive transports across all scenarios and achieves better small flow performance in some cases. Finally, we proved its readily deployable nature by implementing PPT based on Linux-kernel and showing its low CPU overhead. The above experiments and results can back up our proposed goals.

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

Response 2: Thanks for the reviewer's comments. We will add a discussion of PCC Proteus and other related work in Appendix B. However, note that PCC Proteus and PPT are two lines of work, so we have yet to discuss it. 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 requirements and treats all flows equally. PPT uses a dual-loop rate control design to utilize the available bandwidth gracefully and further complements its design with a buffer-aware flow scheduling to optimize small flow's performance.

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. We will provide a more detailed discussion of how the system can deliver the performance benefits. PPT uses LCP to gracefully utilize the spare bandwidth left by DCTCP in the slow start and queue buildup phases. For the queue buildup phase, which the reviewer focused on, we will add a more detailed analysis and show the potential for improvement in Section 2.3. Furthermore, we complement buffer-aware flow scheduling to optimize the performance of small flows. We will also add a discussion of how PPT get benefits from buffer-aware flow scheduling in Appendix.

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. Actually, we have not claimed that "integration with other transports is easy." We claim that the design of PPT can be integrated with delay-based transport. To this point, we implemented a delay-based transport conceptually equivalent to Swift in the ns3 simulator. 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 and discuss how delay-based transport can benefit from the design of the PPT in Appendix C.3.

Concern 5: *Missing details on evaluation setup and parameters.*

Response 5: Thanks for the reviewer's comments. We will complete the missing details about the experimental setup and parameters in the form of tables or footnotes.

II Reviewers Response

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. To avoid ambiguity, we will include footnotes in Section 3.2 for explanation. 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$7 are used to transmit LCP packets. Therefore, the *high-priority* and *low-priority* mentioned in p.6 and p.9 refer to priorities 0$\sim$3 and 4$\sim$7 in the switch, respectively.

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. At the beginning of Section 3, we mentioned that "LCP sends opportunistic packets from the tail end." Therefore, HCP and LCP send packets from the begin-

ning and end of the same flow, respectively. So, there is no scenario in which there is no HCP flow. Since HCP and LCP come from the same flow, they must have the same number and destination.

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 $BDP = rtt * rate_{nic}$. While the $rate_{nic}$ can be read via system call and the datacenter network topology is relatively fixed, we assume the rtt is known a priori. So we can calculate the BDP in 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. Actually, it is very rarely in our experiments that α is greater than 0.5, as it requires the number of ECN marked packets to be more than half of the number of all packets over many sequential RTTs. While α greater than 0.5 represents the link experienced severe congestion, there is no spare bandwidth for LCP.

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. It is true that we have not taken into account the fairness issue in this scenario, but it can be solved by a simple design. We will discuss this in Appendix A.

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 consisting of 144 servers, 9 leaf switches, and 4 spine switches, with the host and core links operated at 100 and 400Gbps, 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. We measure the kernel space CPU overhead of PPT and DCTCP in our testbed. We plot the results in Figure 21. For more details, please 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, showing that nearly half of the bandwidth is underutilized.

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 we checked the experimental data we found that the results were error due to our input errors. We will correct the error and double-check the rest of data.

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. We will add intuitive reasons to Section 6.2.

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. Homa's poor performance in the testbed is due to the inefficient implementation of the Homa-Linux network stack. We have explained this in the *remark* paragraph in Section 6.1.1.

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 our paper, we have constructed the variant of PPT that assigns all packets with two priority queues. We plot the results in Figure 19. 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. Our current topology is commonly used in academia. Simulation experiments containing 2k nodes require significant time consumption, which we will implement in future work.