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. In fact, the proactive transports proactively allocate the bottleneck link bandwidth and prioritise it to flows with fewer remaining bytes. As such, proactive transport typically displays the performance of the large and small flows respectively. To justify our claim of " comparable performance to proactive transports". First, we show the overall average FCT to represent the performance of the transport over all flows in the network. 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 in their ability to gracefully utilize the spare bandwidth. These are the performance metrics that proactive transports focus on and work with. Therefore we use them as the main performance metrics for simulation and testbed and conclude that comparable performance to proactive transports.

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 by different application requirements and proactively reduces scavenger flow bandwidth to reduce impact on the primary flow, thus improving overall user experience. By contrast, PPT is application type insensitive. In PPT, all flows equally utilize the bandwidth. PPT's proposed purpose of buffer-aware flow scheduling is to ensures that small flows can 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 behaviours as well as 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 being able to gracefully utilize the spare bandwidth without causing bandwidth waste or sending opportunistic packets too aggressively. To further explain the insights, we plan to add a remark at the end of Appendix C.1 to analyse in more detail how the PPT achieves its superior performance.

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. In fact, we implemented a delay-based transport which is 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 (2) the ns3 simulator hardly simulates endpoint congestion as well as it actually does. We compared this variant with those of the original delay-based transport and plot 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. Actually, we record the maximum value of the window from the beginning of the flow until now as MW (Maximum Window). In Section 2.3, we first show how much DCTCP can benefit from padding the spare bandwidth to MW and plot the results in Figure 2. Moreover, we show that padding the spare bandwidth to MW is just suitable without wasting bandwidth or causing congestion through experiment, and we plot the results of our experiment in Figure 3. Another question is about specific TCP send buffer thresholds. In fact, to reproduce the behaviour of the actual deployed application as much as possible, we selected specific application trace for the Memcached application and the web server application respectively. The flow size distribution in specific application traces is concentrated in specific intervals, e.g., there is no flow in the ETC trace of the paper[8] that exceeds 100KB. Under such traces, to verify the effectiveness of the buffer-aware flow identification, we set different thresholds depending on the traces. While 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 and show that this indeed improves the small flow performance as expected. In summary, TCP send buffer threshold does not assume that the application type and flow size distribution is known a priori, and effectively improves small flow performance.

II Other Concerns

1: Reviewer Comment: *Although 25 Gbps line-rates do a better job of demonstrating the feasibility of QCLIMB, this is still an order of magnitude less than today's common line-rates. As such, it's still not clear that QCLIMB would be immediately deployable. However, this was not a requirement asked in the revision, and this work is still interesting even if more work is needed to reach 200 Gbps and faster line-rates.*

Response 2: We agree with the reviewer that 25Gbps is still an order of magnitude less than today's 100Gbps+ line-rates. However, 100G+ testbeds are unavailable in our lab because of limited hardware resources. In fact, our QCLIMB may still work in higher-speed networks. The reason is that the dominant overhead of an end-to-end QCLIMB flow is the end-host processing delay. Such processing delay is independent of the line-rates and is mainly caused by kernel network stack processing. In our evaluation, we observe that a 100KB flow requires roughly 200μs for the end-host processing on sender- and receiver-sides. By contrast, QCLIMB's model inference takes only 3μs, which is negligible as compared to such processing delay. One can further reduce this inference latency to 1μs with advanced FPGA hardware [1] or may even reduce it to tens of nanoseconds by carefully pipelining RF's decision tree on hardware [2]. We plan to add these text in our revision to discuss the deployabablity of QCLIMB in higher-speed networks.

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.