Fair Queuing Aware Congestion Control

Maximilian Bachl

https://github.com/muxamilian/fair-queuing-aware-congestion-control

***Abstract*—Fair queuing is becoming increasingly prevalent in the internet and has been shown to improve performance in many circumstances. Performance could be improved even more if endpoints could detect the presence of fair queuing on a certain path and adjust their congestion control accordingly. If fair queuing is detected, the congestion control would not have to take cross traffic into account, which allows for more flexibility. In this paper, we develop the first algorithm that continuously checks if fair queuing is present on a path. When fair queuing is detected, a different congestion control is chosen, which results in reduced latency. Unlike an algorithm proposed in a previous paper of us, the approach presented here does not only detect the presence of fair queuing once at flow startup but it does so continuously.**

I. Introduction

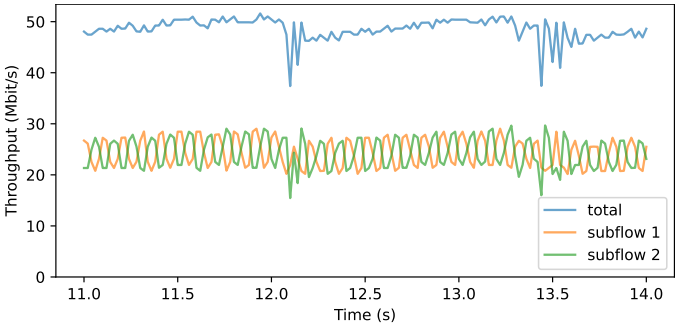
In our previous work we proposed a technique which determines the presence of fair queuing at flow startup [1], which worked as follows: If fair queuing is successfully detected at flow startup, a congestion control was used, which aimed to keep queuing delay low (delay-based congestion control). While this delay-based congestion control achieved high throughput and low delay, it was vulnerable to be outcompeted by other network flows sending more aggressively, such as [2]–[4], similar to the Vegas congestion control algorithm [5]. This means that our delay-based congestion control performed well but only when it wouldn't have to compete with other flows. Thus is only used if fair queuing is detected. If it is detected that there is no fair queuing, our approach uses a more aggressive congestion control (specifically PCC [3]), which can compete better with other network flows, but doesn't keep delay as low as our delay-based congestion control.

While our previous approach had high detection accuracy (98%), it also had some limitations:

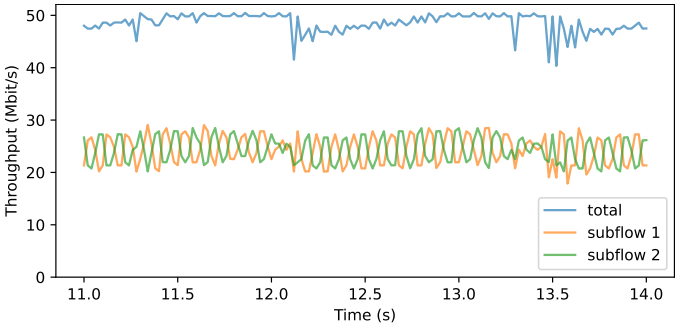
- It would **only** detect fair queuing at **flow startup**. But if the bottleneck link changes during a flow, it could be that the previous bottleneck had fair queuing while the new one doesn't. This wouldn't have been detected.
- It would detect fair queuing only after filling the queue at the bottleneck completely, **causing packet loss**.

We would thus like to have an approach which

- **continuously checks** for the presence or absence of fair queuing, not only at flow startup.
- **doesn't cause packet loss** while trying to determine if there is fair queuing or not.
- can be **transparently used** on top of any congestion control algorithm.



(a) Sending rate at the sender (fair queuing)

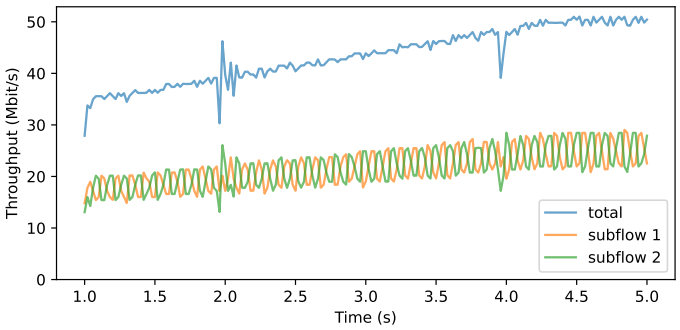


(b) Receiving rate at the receiver (fair queuing)

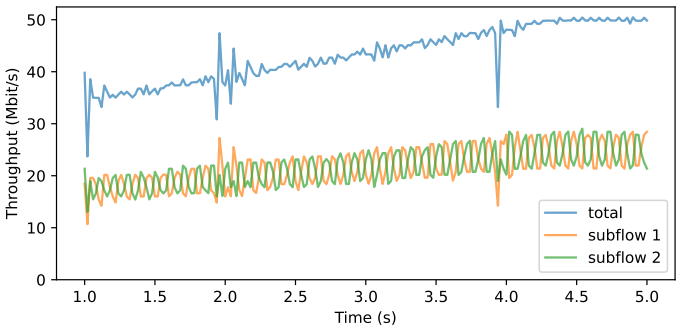
Fig. 1: Figure 1a shows the sending rate, 1b the receiving rate in case there's fair queuing. Around seconds 12 and 13.5, the sending rate reaches the maximum of the link – 50 Mbit/s – and fair queuing starts to limit the throughput of the flow that is sending more. Thus, in the lower figure, the dominant flow and the non-dominant flow achieve approx. the same receiving rate even though the dominant flow sends more (dominant flow). This is the effect of fair queuing.

	10 Mbit/s	50 Mbit/s	100 Mbit/s
10ms	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%
50ms	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%
100ms	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%

TABLE I: Detection accuracy in case there's **no fair queuing**. The average accuracy is 100%



(a) Sending rate at the sender (no fair queuing)



(b) Receiving rate at the receiver (no fair queuing)

Fig. 2: Figure 2a shows the sending rate, 2b the receiving rate in case there's no fair queuing. Even though the sender sends too much and a queue builds up, still the flow that sends more data also has a higher receiving rate. This is in contrast to Figure 1, where both flows have the same receiving rate once there is congestion at the bottleneck, thanks to fair queuing.

	10 Mbit/s	50 Mbit/s	100 Mbit/s
10ms	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 99% 3rd quart.: 100%
50ms	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%
100ms	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%	Median: 100% 1st quart.: 100% 3rd quart.: 100%

TABLE II: Detection accuracy in case there is **fair queuing**. The average accuracy is 96%

II. Concept

III. Evaluation

IV. Discussion

References

[1] M. Bachl, J. Fabini, and T. Zseby, "Detecting Fair Queuing for Better Congestion Control," Tech. Rep. arXiv:2010.08362, arXiv, Feb. 2021. arXiv:2010.08362 [cs] type: article.

[2] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, September-October, pp. 20 – 53, 2016.

[3] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting Congestion Control for Consistent High Performance," pp. 395–408, 2015.

[4] "CUBIC: a new TCP-friendly high-speed TCP variant: ACM SIGOPS Operating Systems Review: Vol 42, No 5."

[5] L. Brakmo and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1465–1480, Oct. 1995. Conference Name: IEEE Journal on Selected Areas in Communications.