# The Overhead of Being Fair

Zakaria Al-Qudah and Vincenzo Liberatore
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, Ohio 44106 USA
Email: {zma, vl}@case.edu
URL: http://vincenzo.liberatore.org/NetBots/

**Abstract**

The fairness of *Stochastic Fairness Queuing* (SFQ) is greatly affected by the number of queues to which flows are hashed. Specifically, a large queue pool results in a small probability of flow collisions and, hence, better fairness in allocating bandwidth to flows. However, the research community has assumed that a large number of queues poses an unsustainably large overhead on an SFQ router, even though such an overhead has never been investigated or quantified. This work investigates this overhead and proves its insignificance by means of emulations. For example, when the number of queues was increased by 3 orders of magnitude in our emulation setup, there was no statistically significant difference in the router processing delays.

Fair Queuing (FQ) aims at ensuring that flows will obtain an equitable share of bandwidth [1]. The basic idea of FQ is to assign a separate queue for each flow. In general, this perfect flow separation has many desirable properties such as preventing flows from obtaining an arbitrarily large share of link bandwidth and from increasing the delays experienced by other flows [1]. However, assigning a queue for each flow requires maintaining per-flow state and consequently does not scale well. *Stochastic Fairness Queuing* (SFQ) avoids these problems by a non-deterministic flow-to-queue hashing [2]. Although SFQ addresses FQ's scalability problems while roughly retaining its desired fairness properties, SFQ's performance is ill-understood. In particular, contrary to a common assumption in the literature, we argue that SFQ does not pose any statistically significant processing overhead on an SFQ router when a large number of queues is used.

In SFQ, flows are hashed into a fixed set of queues. When multiple flows result into the same hash value, those flows are placed into a single queue and, thus, are considered as a single flow—a *collision*. Therefore, colliding flows receive a smaller share of bandwidth than others.

The collision probability can be decreased by hashing flows into a large number of queues. It has been commonly assumed in the literature, however, that a large number of queues can place a large overhead on an SFQ router. Further, the source of this overhead is thought to arise from the management of an associated large data structure [3], [4]. Given that there exist a constant-time implementation for SFQ [2], one may ask whether such a computational overhead is a valid concern. In general, a constant-time algorithm does not necessarily mean an equal computational overhead for all implementation instances of this algorithm. To illustrate, the Big O notation expresses the asymptotic behavior of an algorithm. Therefore, it hides constants that not only can be large, but also can be different from one implementation instance to another. Hence, the objective of this work is to:

*Quantify the potential overhead of using a large number of queues in SFQ.*

To this end, we estimate the overhead associated with a large number of queues with emulations on Linux TC [5] and on Click [6].

The major finding of this work is that a large queue pool does not pose a statistically significant computational overhead on an SFQ router. To this end, we showed that when the number of queues is raised by 3 orders of magnitude in our emulation setup, there was no statistically significant difference in the router processing delays. Therefore, SFQ can be safely deployed on routers with large number of queues to reduce the likelihood of collisions.

## REFERENCES

[1] J. Nagle, "On packet switches with infinite storage," *IEEE Transactions on communication*, vol. 35, no. 4, pp. 435 – 438, 1987.
[2] P. E. McKenney, "Stochastic fairness queuing," in *IEEE INFOCOM*, June 1990.
[3] A. Mankin and K. Ramakrishnan, *Gateway Congestion Control Survey*, August 1991.
[4] R. Pan, B. Prabhakar, and K. Psounis, "CHOKE, a stateless active queue management scheme for approximating fair bandwidth allocation," in *INFOCOM (2)*, 2000, pp. 942–951.
[5] T. Graf, G. Maxwell, M. V. Oosterhout, P. B. Schroeder, J. Spaan, and P. Larroy, *Linux Advanced Routing and Traffic Control*, http://www.lartc.org/lartc.html.
[6] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000.