Safe Randomized Load-Balanced Switching By Diffusing Extra Loads

Sen Yang

Georgia Institute of Technology
sen.yang@gatech.edu

Bill Lin

University of California, San Diego
billlin@ece.ucsd.edu

Jun Xu

Georgia Institute of Technology
jx@cc.gatech.edu

ABSTRACT

Load-balanced switch architectures are known to be scalable in both size and speed, which is of interest due to the continued exponential growth in Internet traffic. However, the main drawback of load-balanced switches is that packets can depart out of order from the switch. Randomized load-balancing of application flows by means of hashing on the packet header is a well-known simple solution to this packet reordering problem in which all packets belonging to the same application flow are routed through the same intermediate port and hence the same path through the switch. Unfortunately, this method of load-balancing can lead to instability, depending on the mix of flow sizes and durations in the group of flows that gets randomly assigned to route through the same intermediate port. In this paper, we show that the randomized load-balancing of application flows can be enhanced to provably guarantee both stability and packet ordering by extending the approach with *safety* mechanisms that can *uniformly diffuse* packets across the switch whenever there is a build-up of packets waiting to route through some intermediate port. Although simple and intuitive, our experimental results show that our extended randomized load-balancing approach outperforms existing load-balanced switch architectures.

CCS CONCEPTS

• **Networks** → **Packet scheduling**; *Routers; Bridges and switches; Packet-switching networks*; • **Mathematics of computing** → *Queueing theory*;

KEYWORDS

Load-balanced switches; packet reordering; throughput guarantees; low latency

ACM Reference Format:

Sen Yang, Bill Lin, and Jun Xu. 2018. Safe Randomized Load-Balanced Switching By Diffusing Extra Loads. In *SIGMETRICS '18 Abstracts: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems Abstracts, June 18–22, 2018, Irvine, CA, USA*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/3219617.3219673>

1 INTRODUCTION

Internet traffic continues to grow exponentially, fueled by an increasing adoption of cloud computing and video streaming and

Figure 1: Generic load-balanced switch.

by an explosion of network-connected devices with increasing access speeds. To keep up with the relentless traffic growth with reliable service, network operators need high-performance switch architectures that can scale well in both size and speed, provide throughput guarantees, achieve low latency, and maintain packet ordering. However, conventional switch architectures like centrally-scheduled input-queued crossbar switches are not scalable.

Alternatively, a promising scalable class of switch architecture is the load-balanced switch, which was first introduced by Chang et al. [1, 2], and later further developed by others (e.g. [5–8]). They rely on two switching stages for routing packets. Figure 1 shows a diagram of a generic two-stage load-balanced switch. The first switching stage connects the first stage of input ports to the center stage of intermediate ports, and the second switching stage connects the center stage of intermediate ports to the final stage of output ports. Both switching stages execute a deterministic connection pattern such that each input is connected to each output of a switching stage $\frac{1}{N}$ th of the time. This can be implemented for example using two identical $N \times N$ crossbar switching stages where each switching stage goes through the same predetermined cyclic-shift connection pattern such that each input is connected to each output of a switching stage exactly once every N cycles (time slots). Alternatively, as shown in [7], the deterministic connection pattern can also be efficiently implemented using optics in which all inputs are connected to all outputs of a switching stage in parallel at a rate of $\frac{1}{N}$ the line rate.

Although the basic load-balanced switch originally proposed in [1] is capable of achieving throughput guarantees, it has the critical problem that packet departures can be badly out of order. In the basic load-balanced switch, consecutive packets at an input port are spread to all N intermediate ports upon arrival. Packets going through different intermediate ports may encounter *different queueing delays*. Thus, some of these packets may arrive at their output ports out-of-order. This is detrimental to Internet traffic since the widely used TCP transport protocol falsely regards out-of-order

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGMETRICS '18 Abstracts, June 18–22, 2018, Irvine, CA, USA

© 2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5846-0/18/06.

<https://doi.org/10.1145/3219617.3219673>

packets as indications of congestion and packet loss. Therefore, a number of researchers have explored this packet ordering problem.

One simple approach for ensuring packet ordering, called Application Flow-Based Routing (AFBR) algorithm [7], is based on the following insight: To prevent harmful effects in TCP performance due to out-of-order packets, only packets belonging to the same application flow (e.g. a TCP/IP flow) have to depart from their output port in order. This can be achieved by forcing all packets that belong to the same application flow to go through the same intermediate port. In doing so, all packets belonging to the same application flow are guaranteed to take the same path through the switch, which avoids reordering among them. The selection of intermediate port can be easily achieved by hashing on the header field of every packet (source and destination IP addresses, source and destination ports, and protocol identification) to obtain a value from 1 to N . Hence this approach is nicknamed TCP hashing. Although simple and intuitive, the main drawback of TCP hashing is that stability cannot be guaranteed, as we will elaborate shortly.

Alternatively, most existing approaches that can guarantee both stability and packet ordering are based on some form of complete or partial aggregation of packets into frames or stripes. Uniform Frame Spreading (UFS) [7], Full-Order Frames First (FOFF) [7], Padded Frames (PF) [6], and Sprinklers [5] are representative examples of such approaches. However, these methods pay a significant price for ensuring packet ordering in that they perform significantly worse than the originally proposed basic load-balanced switch [1].

2 OUR APPROACH

In this paper, we investigate the sources of instability in the TCP hashing approach in order to derive mechanisms that can mitigate them. In particular, at the first switching stage, each input port i is only connected to an intermediate port once every N time slots, or equivalently at $\frac{1}{N}$ of the line rate, via a deterministic connection pattern. Persistent overloading occurs at an input port when the arrival rate of packets hashed to the same intermediate port exceeds $\frac{1}{N}$ of the line rate for a long period of time, which can occur depending on the mix of flow sizes and durations in the group of flows that gets randomly hashed to route through the same intermediate port. For example, such persistent overloading can happen if there is a long-lived elephant flow in their midst. Although the notion of instability used here is a practical one (with respect to the limited packet buffer a switch has on each input or intermediate port), we will explain that TCP hashing could become unstable also under a theoretical and more restrictive notion of stability called rate-stability [4]. In comparison, the approach that we propose in this paper is provably rate-stable.

Similarly, at the second switching stage, each intermediate port m is also only connected to an output port j at $\frac{1}{N}$ of the line rate. Packets queued at an intermediate port may come from different input ports, possibly from all N of them. Overloading occurs at an intermediate port when the arrival rate of packets destined to the same output port, from all N inputs, exceeds $\frac{1}{N}$ of the line rate.

Two Safety Mechanisms. To remedy these problems, we extend the basic flow randomization scheme with two *safety* mechanisms. First, let λ_{ij} be the arrival rate for VOQ_{ij} , the Virtual Output Queue (VOQ) of packets arriving at input port i with output destination

j . Depending on the hash values of their flow identifiers, the set of TCP/UDP flows within VOQ_{ij} can be partitioned into N subsets called bins. Each bin m , $m = 1, 2, \dots, N$, corresponds to the set of flows that are hashed (and hence need to be switched) to intermediate port m . Using a simple credit scheme, without any knowledge or measurement of the value of λ_{ij} , we can limit the rate at which packets in each bin are served (switched) to at most $\frac{\lambda_{ij}}{N}$. The first safety mechanism, when enforced on every VOQ, ensures no overloading at any input or intermediate port, by the “normal” (i.e., rate-limited) traffic, under any *admissible* arrival traffic, and that it does so in the least restrictive manner in the following sense: $\frac{\lambda_{ij}}{N}$ is indeed the maximum traffic rate that can be granted to each bin safely (i.e., without overloading any input or intermediate port).

However, by limiting the service rate of each such bin at an input port to $\frac{\lambda_{ij}}{N}$, those bins with traffic arrival rates exceeding that limit (e.g., bins that contain elephant flows as mentioned above) can grow in size. To ensure that these bins do not grow infinitely, thus leading to instability, we implement a second safety mechanism in which once a build-up of packets at a bin exceeds some threshold $W \geq N$ in size, we “evacuate” the excess load by *uniformly diffusing* the build-up of packets across all intermediate ports (i.e., a full-frame of N packets are uniformly spread one-to-one to the N intermediate ports). We introduce an easy-to-implement technique to ensure packet ordering when this evacuation mechanism kicks in, which involves requiring a “to-be-evacuated” bin to wait till it is safe (from packet reordering) to do so. Due to this waiting, careful scheduling is needed to coordinate an “orderly evacuation” of all bins being evacuated at any input port to ensure that every bin has a fair chance to have its backlog duly cleared.

The Stability Proof. We prove that our Safe Randomization Switch (SRS) scheme can achieve 100% throughput (i.e., rate-stability), while guaranteeing packet order, under any admissible input traffic that is allowed to change rapidly dynamically over time. Proving the stability of SRS is very challenging partly because it appears hard to make the standard machinery of fluid analysis [3] work for this problem, despite our considerable efforts.

In arriving at this proof, we have invented a general methodology for proving the stability of queues. Our proof is based on the following extremal argument. Suppose SRS is not stable so that the total length $Q(t)$ of a subset of queues in SRS, as a function of time t , does not satisfy the stability condition $\lim_{t \rightarrow \infty} \frac{Q(t)}{t} = 0$. Then we define $\gamma \equiv \limsup_{t \rightarrow \infty} \frac{Q(t)}{t} > 0$. Let t_i , $i = 1, 2, \dots$, be a sequence of times such that $\lim_{i \rightarrow \infty} t_i = \infty$ and $\lim_{i \rightarrow \infty} \frac{Q(t_i)}{t_i} = \gamma$. Starting with this time sequence t_i , by the properties of the aforementioned credit scheme (for rate-limiting “TCP hashed” traffic into each intermediate port) and the aforementioned scheduler for the orderly evacuation, and through standard busy period arguments, we can construct another sequence of times t'_i , $i = 1, 2, \dots$, such that $\lim_{i \rightarrow \infty} t'_i = \infty$ and $\limsup_{i \rightarrow \infty} \frac{Q(t'_i)}{t'_i} > \gamma$, which contradicts the definition of γ .

The basic ideas of SRS were first proposed in an extended abstract [9]. In this paper, we expand on these ideas significantly and provide a stability proof of SRS. The full version of this paper is

available in [10]. We summarize the contributions of this work as follows.

- First, we propose a new load-balanced switch architecture, called SRS, that solves the packet reordering problem of load-balanced switches and has a delay performance that compares favorably with other solutions.
- Second, we prove that, like other load-balanced switch solutions, the SRS architecture can achieve 100% throughput under any admissible arrival traffic, while guaranteeing packet order. The methodology used in the proof, described above, is novel and general, and is a major contribution by itself.
- Third, although the basic flow randomization approach has long been regarded as a simple and intuitive solution to the load-balanced switching problem, its lack of stability guarantees thus far has led many researchers to develop more complex solutions. We show that the basic flow randomization approach can indeed be made stable through the proposed safety extensions.

3 EVALUATION

In this section, we compare the performance of our proposed SRS approach with other existing load-balanced switching algorithms, including the basic load-balancing scheme [1], Uniform Frame Spreading (UFS) [7], Full-Ordered Frame First (FOFF) [7], Padded Frames (PF) [6], the Sprinklers scheme [5]. The basic load-balancing scheme (labeled “Basic”) does not guarantee packet ordering, but it provides the lower bound of the delay that a load-balanced switch can achieve. In the full paper [10], we also propose a variant of the baseline SRS architecture, named SRS-UFS, that can improve the delay performance when the traffic load is high, the simulation result of which is also included in this section.

Our first set of experiments assumes uniform distribution of the destination ports for the arrival flows – i.e. a new flow goes to output j with probability $\frac{1}{N}$. The results are shown in Figure 2a. The second set of experiments assumes a quasi-diagonal distribution. A new flow arriving at input port i goes to output $j = i$ with probability $\frac{1}{2}$, and goes to any other output port with probability $\frac{1}{2(N-1)}$. The results are shown in Figure 2b. Readers may refer to our full paper [10] for more evaluation experiments.

The following three observations can be made from the average delay results in the above experiments. First, for uniform traffic, the average delay of baseline SRS is significantly better than the existing methods of UFS, FOFF, PF, and Sprinklers for all traffic loads up to about $\lambda < 0.85$. Similarly, for quasi-diagonal traffic, the average delay of baseline SRS is significantly better than the existing methods of UFS, FOFF, PF, and Sprinklers for all traffic loads up to about $\lambda < 0.8$. Second, in comparison to the basic LBS that does not guarantee packet order, the performance of SRS follows roughly the same trend in both experiments. Third, our SRS-UFS variant improves the performance of the baseline SRS at high loads. In both experiments, the SRS-UFS variant scheme performs almost the same as the baseline SRS when the traffic load is low to moderate ($\lambda \leq 0.75$), and as the UFS scheme when traffic load is high ($\lambda > 0.75$).

Acknowledgment: This project is supported in part by NSF grants CNS-1423182 and CNS-1248117.

(a) Average delay under uniform traffic

(b) Average delay under quasi-diagonal traffic

Figure 2: Average delay under different traffic patterns.

REFERENCES

- [1] Cheng-Shang Chang, Duan-Shin Lee, and Yi-Shean Jou. 2002. Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering. *Computer Communications* 25, 6 (2002), 611–622.
- [2] Cheng-Shang Chang, Duan-Shin Lee, and Ching-Ming Lien. 2002. Load balanced Birkhoff-von Neumann switches, part II: multi-stage buffering. *Computer Communications* 25, 6 (2002), 623–634.
- [3] J. G. Dai. 1998. *Stability of fluid and stochastic processing networks*. University of Aarhus. Centre for Mathematical Physics and Stochastics (MaPhySto)[MPS].
- [4] J. G. Dai and Balaji Prabhakar. 2000. The throughput of data switches with and without speedup. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 2. IEEE, 556–564.
- [5] Weijun Ding, Jun Xu, J. G. Dai, Yang Song, and Bill Lin. 2014. Sprinklers: A randomized variable-size striping approach to reordering-free load-balanced switching. In *ACM CoNext, the 10th International Conference on Emerging Networking EXperiments and Technologies*. ACM.
- [6] Juan José Jaramillo, Fabio Milan, and R Srikant. 2008. Padded frames: a novel algorithm for stable scheduling in load-balanced switches. *Networking, IEEE/ACM Transactions on Networking* 16, 5 (2008), 1212–1225.
- [7] Isaac Keslassy. 2004. *The load-balanced router*. Ph.D. Dissertation. Stanford University.
- [8] Sen Yang, Bill Lin, Paul Tune, and Jun Jim Xu. 2017. A simple re-sequencing load-balanced switch based on analytical packet reordering bounds. In *INFOCOM 2017-IEEE Conference on Computer Communications*, IEEE. IEEE, 1–9.
- [9] Sen Yang, Bill Lin, and Jun Xu. 2016. Safe Randomized Load-Balanced Switching by Diffusing Extra Loads. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*. ACM, 397–398.
- [10] Sen Yang, Bill Lin, and Jun Xu. 2017. Safe Randomized Load-Balanced Switching By Diffusing Extra Loads. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 2, Article 29 (Dec. 2017), 37 pages. <https://doi.org/10.1145/3154487>