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
billin@ece.ucsd.edu

Jun (Jim) 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 the same intermediate port. Although simple and intuitive, our experimental results show that our extended randomized load-balancing approach significantly outperforms existing load-balanced switch architectures.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Packet-switching networks*; C.2.6 [Computer-Communication Networks]: Internetworking—*Routers*

Keywords

Load-balanced switches; throughput guarantees

1. INTRODUCTION

To keep up with the relentless growth of the Internet traffic, network operators need high-performance switch archi-

tectures 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 (LBS), which was first introduced by Chang et al. [1, 2], and later further developed by others. These architectures rely on two switching stages for routing packets. 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 at $1/N$ th of the time.

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. This is detrimental to Internet traffic since TCP falsely regards out-of-order packets as indications of congestion and packet loss. One simple approach for ensuring correct packet order in an application flow, which is all that matters, is randomized load-balancing. This approach, also known as “TCP hashing,” force all packets that belong to the same application flow to go through the same intermediate port, by determining the intermediate port each packet should be headed to via hashing (on its header). Although simple and intuitive, the main drawback of the randomized approach is that stability cannot be guaranteed.

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) [5], Full-Order Frames First (FOFF) [5], Padded Frames (PF) [4], and Sprinklers [3] 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

To remedy the problems that lead to instability in the randomized load-balancing approach, we extend it 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

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 '16 June 14-18, 2016, Antibes Juan-Les-Pins, France

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4266-7/16/06.

DOI: <http://dx.doi.org/10.1145/2896377.2901480>

bins, numbered $1, 2, \dots, N$. 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 of the value of λ_{ij} , we can limit the rate at which packets in each bin are served (switched) to at most λ_{ij}/N . In fact, this credit scheme guarantees something even stronger: The actual long-term service rates of these bins, after the rate-limiting, are identical. The first safety mechanism, when enforced on every VOQ, clearly ensures no overloading at any input port by the “normal” (i.e., rate-limited) traffic.

However, by limiting the service rate of each such bin at an input port to λ_{ij}/N , those bins with traffic arrival rates exceeding that limit (e.g., bins that contain elephant flows) can grow in size. To ensure that these bins don’t 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 switch stability under 100% traffic load.

We prove that our Safe Randomization Switch (SRS) scheme can achieve 100% throughput (i.e., rate-stable), while guaranteeing packet order, under any admissible¹ arrival traffic that is allowed to change rapidly dynamically over time. 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 in spirit (but not exactly in this form). 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 time 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 time 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 . Given how general this methodology is, we believe it can be applied to some other queueing theory problems, especially where circumstances make it hard to use the standard fluid analysis.

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], UFS [5], FOF [5], PF [4], and the recently proposed Sprinklers scheme [3]. The basic load-balancing scheme (labeled

¹Here “admissible” means the long-run-average total rate of all traffic destined for any output port must exist and is no more than 1 (i.e., 100% loaded).

“Basic”) does not guarantee packet ordering, but it provides the lower bound of the average delay that a load-balanced switch can achieve.

Figure 1: Average delay under uniform traffic.

In the experiments we assume uniform distribution² 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 Fig. 1. Several observations can be made from the above experiments. First, the average delay of SRS is significantly better than the existing methods of UFS, FOF, PF, and Sprinklers for all traffic loads up to about $\lambda < 0.9$ and the average delay is only slightly more for $\lambda \geq 0.9$. Furthermore, in comparison to the basic load-balanced switch approach that does not guarantee packet order, the performance of SRS follows roughly the same trend as the basic LBS approach. In summary, perhaps surprisingly, the SRS solution is able to achieve significantly better results than existing methods, especially under low-to-moderate loads, based on the simple mechanism of hashing.

Acknowledgement: This work has been supported in part by NSF projects CNS 1248117, CNS 1218092, CNS 1423182, CNS 1422286.

4. REFERENCES

- [1] CHANG, C.-S., LEE, D.-S., AND JOU, Y.-S. Load balanced birkhoff-von neumann switches, part i: one-stage buffering. *Computer Communications* 25, 6 (2002), 611–622.
- [2] CHANG, C.-S., LEE, D.-S., AND LIEN, C.-M. Load balanced birkhoff-von neumann switches, part ii: multi-stage buffering. *Computer Communications* 25, 6 (2002), 623–634.
- [3] DING, W., XU, J., DAI, J. G., SONG, Y., AND LIN, B. 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* (2014), ACM.
- [4] JARAMILLO, J. J., MILAN, F., AND SRIKANT, R. Padded frames: a novel algorithm for stable scheduling in load-balanced switches. *Networking, IEEE/ACM Transactions on Networking* 16, 5 (2008), 1212–1225.
- [5] KESLASSY, I. *The load-balanced router*. PhD thesis, Stanford University, 2004.

²We also conducted experiments for quasi-diagonal traffic and the conclusion is roughly the same.