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 (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. Hence, researchers have sought alternative solutions to ensure both stability and packet ordering, but known solutions either have poor packet delays, require a priori knowledge of the traffic pattern, or employ sophisticated matching mechanisms. 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 some intermediate port. Although simple and intuitive, our experimental results show that our extended randomized load-balancing approach significantly outperforms existing load-balanced switch architectures.

1. INTRODUCTION

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

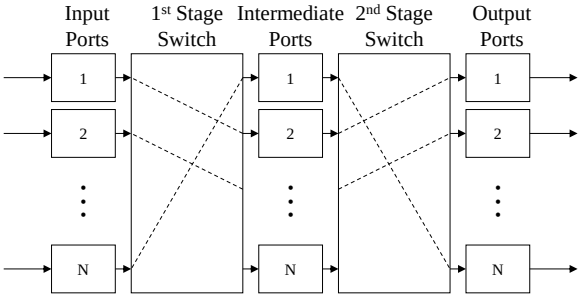

Figure 1: Generic load-balanced switch.

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, 7, 8, 9]). These architectures rely on two switching stages for routing packets. Fig. 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 at $1/N$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 [8], 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 $1/N$th 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 packets as indications of congestion and packet loss.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Therefore, a number of researchers have explored this packet ordering problem.

One simple approach for ensuring packet ordering is based on randomization. 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. Although simple and intuitive, the main drawback of this randomized load-balancing 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) [8], Full-Order Frames First (FOFF) [8], Padded Frames (PF) [7], 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].

1.1 Our Approach

In this paper, we investigate the sources of instability in the basic randomized load-balancing scheme 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 m $1/N$th of the time (or equivalently at $1/N$th of the line rate) via a deterministic connection pattern. Overloading occurs at an input port when the arrival rate of packets hashed to the same intermediate port exceeds $1/N$th of the line rate, 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, this could happen if an elephant flow [6] happens to be hashed to one of the bins.

Similarly, at the second switching stage, each intermediate port m is only connected to an output port j $1/N$th of the time (or equivalently at $1/N$th 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 $1/N$th of the line rate.

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, numbered $1, 2, \cdots, N$. Each bin m, $m = 1, 2, \cdots, N$, corresponds to the set of flows that are hashed (and hence need to be switched) to intermediate port m. Using a simple credit scheme that we will describe in Section 3.1.1, 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 as mentioned above) 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, which we will explain in Section 3.1.2.

We prove that our Safe Randomization Switch (SRS) scheme can achieve 100% throughput (i.e., rate-stable), while guaranteeing packet order, under any admissible arrival traffic that is allowed to change rapidly dynamically over time. In this work, we choose to prove only the rate-stability – a rather weak form of stability – because only this weak form is used and proved in all other LBS works. Proving the stability of SRS is very challenging partly because the standard machinery of fluid analysis [3] does not appear to work for this problem, at least not in a straightforward manner. More specifically, in this problem, due to the aforementioned waiting-for-evacuation, some queues need to be completely emptied (i.e., actual zero queue length) before some other queues can start service. It appears hard to faithfully express this constraint in the fluid model, because zero queue length in the fluid sense is not the actual zero queue length we need here.

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 $\lim_{t\to\infty} \frac{Q(t)}{t} = 0$. Then we define $\gamma \equiv \limsup_{t\to\infty} \frac{Q(t)}{t} > 0$.[1] Let t_i, $i = 1, 2, \cdots$, be a sequence of time such that $\lim_{i\to\infty} t_i = \infty$ and $\lim_{i\to\infty} \frac{Q(t_i)}{t_i} = \gamma$. 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, $i = 1, 2, \cdots$, such that $\lim_{i\to\infty} t'_i = \infty$ and $\limsup_{i\to\infty} \frac{Q(t'_i)}{t'_i} > \gamma$, 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

[1]Clearly $\gamma < \infty$ because even if the switch does not process/serve any packet, γ is bounded above by the total incoming traffic rate.

it hard to use the standard fluid analysis.

1.2 Contributions of the Paper

This paper makes the following three major contributions:

- First, we propose a new LBS architecture, called SRS, that solves the packet reorder problem of LBS. Compared to other solutions, SRS has the lowest possible computational and implementation complexities, yet delivers the best empirical performance, in terms of average packet delay.
- Second, we prove that, just like other 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 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.

The rest of the paper is organized as follows. In Sections 2 and 3, we introduce the structure of the SRS architecture and explain how it works. In Section 4, we provide rigorous proofs of stability for the proposed SRS architecture. In Section 5, we compare the average delay performance of the SRS architecture with existing load-balanced switch architectures. In Section 6, we provide a brief review of existing load-balanced swtich architectures. Finally, we conclude the paper in Section 7.

2. OVERVIEW OF SRS

Before we can describe how packets are forwarded in SRS and why, we need to first provide a brief description of the periodic sequences of connections executed at both switching fabrics shown in Fig. 1, explain the packet reorder problem of the basic Load-Balanced Switch (LBS) in more details, and describe the Uniform Frame Spreading (UFS) algorithm, a solution to this packet reorder problem and building block of our SRS architecture. We will do so in Section 2.1, before providing a high-level overview of the SRS architecture in Section 2.2.

Throughout the paper, we make the following standard homogeneity assumptions about an SRS. All packets have a fixed length and time is slotted. Every input, intermediate, or output port operates at the same speed: Each can process and transmit exactly one packet per time slot. We refer to this service rate as 1. Every connection made in a switching fabric also has speed of 1 (i.e., one packet can be switched per time slot). Since N connections are made by a switching fabric at any time slot, up to N packets can be switched by it during each time slot.

2.1 Background on the basic LBS and UFS

The first switching fabric executes a periodic "increasing" sequence, that is, at any time slot t, each input port i is connected to the intermediate port $((i + t) \mod N) + 1$. The second switching fabric, on the other hand, executes a periodic "decreasing" sequence, that is, at any time slot t, each intermediate port ℓ is connected to the output port $((\ell - t) \mod N) + 1$. Following the sequence of connection patterns in the first switching fabric, each input port i can "stripe" a frame of N packets, or all packets that are ready for service if there are less than N of them, to intermediate ports $1, 2, \cdots, N$ respectively, in N consecutive time slots. As explained before, two packets belonging to the same VOQ can go to two different intermediate ports, experience different queueing delays, and arrive at the output port out of order.

Unlike the basic LBS, in which a frame of packets could come from different VOQs and be only partially filled (when there are less than N of them as mentioned above), Uniform Frame Spreading (UFS) [8] requires that every frame of packets belong to the same VOQ and the frame is fully filled, before it can be served. Each such full frame is then striped across the N intermediate ports like in a basic LBS. This ensures that for each full frame of N packets, the set of N queues at the N intermediate ports that they travel through are of equal length, and hence induce the same queueing delay on these N packets. Consequently, these N packets will (shortly) start to appear respectively at the heads of these N queues in N consecutive time slots and be striped to their respective destination output ports in the correct order by the second switching fabric. A drawback of UFS, however, is that a low-rate VOQ could incur a long buffering delay waiting for a full frame of N packets to fill in order for it to be eligible for service. SRS does not inherit this buffering delay issue from UFS, because as mentioned earlier, it uses UFS only for "evacuating" a bin (consisting of a subset of flows in a VOQ) that has too many (rather than too few) packets in it.

2.2 Overview of the switch operation

As described earlier, in SRS, by default, packets are routed through the switch via a *randomly* selected intermediate port, where the random selection is based on hashing on the packet identifier, which for the Internet Protocol is a combination of source and destination IP addresses, source and destination port numbers, and protocol number. We refer to this mode of operation as the Random Single Path (RSP) mode. With RSP, all packets belonging to the same application (TCP or UDP) flow are routed along the *same path* through the switch, thereby ensuring their packet order. As explained earlier, the packets – or rather the application (TCP/UDP) flows – in a VOQ are thus partitioned into N bins, one for flows in the VOQ that are hashed to each intermediate port; To guarantee the stability of the switch (more specifically that of queues at intermediate ports), we *limit* the rate at which packets in each bin can be served under the RSP mode to $\frac{\lambda_{ij}}{N}$, where λ_{ij} is the arrival rate of VOQ_{ij}. We achieve this rate-limiting using a simple credit scheme, to be described in Section 3.1.1. In the meantime, we simply refer to a nonempty bin of RSP packets that conforms to the respective rate-limits – a time-varying set – as RSP-ready.

As explained earlier, when the packet arrival rate to a bin exceeds the aforementioned rate limit $\frac{\lambda_{ij}}{N}$ for an extended period of time, the nonconforming (to the rate-limit) packets have to be queued at the bin and the length of the queue can become very long. Our solution is, once such a bin builds up to a certain level, the switch will allow all packets in the bin to "evacuate" through all intermediate ports simultaneously, one frame (of N packets) at a time and hence at a very high rate, until the queue is cleared. We refer to this mode of operation as UFS, named after the prior work (described

earlier) that serves packets within each VOQ frame by frame to avoid packet reorder [8]. As discussed earlier, when the service of a bin is switched from the RSP mode to the UFS mode, care needs to be taken so that packet reorder will not happen during the transition, which we will discuss in details in Section 3.1.2. In the meantime, we simply call a bin UFS-ready when it is safe (from packet reorder) to do so.

At an input port, when there are both RSP-ready and UFS-ready bins waiting for their respective services, UFS-ready bins take priority because UFS kicks in only when a bin has a very long queue and needs to be "evacuated" quickly. When there are multiple UFS-ready bins competing for service, however, careful scheduling among them is critical to ensure an "orderly evacuation" of all bins. We say an evacuation is orderly when "no evacuee is left behind", that is, every UFS-ready bin can evacuate at least roughly as fast as new packets arrive at the bin. We would like the evacuation process to be orderly because evacuation of some long UFS-ready bins could otherwise prevent or slow down that of other bins, possibly leading to significant buildup of their (queue) lengths and hence their instabilities. In SRS, this "orderly evacuation" is achieved using a simple low-complexity ($O(1)$ per packet) scheduling algorithm that requires no knowledge of bin rates.

While careful scheduling is needed among UFS frames, as far as the stability of the SRS switch is concerned, the order in which the HOL packets from RSP-ready bins are served does not matter. In fact, our stability proof assumes only that the RSP-scheduler at any input port is work-conserving with respect to RSP-ready traffic. However, careful RSP scheduling is desired for providing the best possible empirical performance, in terms of the average packet delay, and the best possible fairness to all RSP-ready bins. In SRS, an input port simply serves RSP-ready bins in the round-robin order, as we will explain in Section 3.1.1.

3. DETAILED SRS OPERATIONS

As explained in Section 2.1, in SRS, like in all other LBS schemes, the connection patterns at both switching fabrics are deterministic periodic sequences. Therefore, the action of an SRS switch (on an arrival process of incoming packets to its input ports) is completely determined by its policies of scheduling packets (for switching service) at both the input ports and the intermediate ports. They will be described in Sections 3.1 and 3.2 respectively.

3.1 Operations at an Input Port

In an SRS switch, like in most other LBS schemes, the input ports drive the policy decisions in the sense that the scheduling decisions made by the input ports to a large degree determine those made by the intermediate ports. Therefore, the scheduling policies at an input port, and the data structures and algorithms for implementing them, require much more explanations than those at an intermediate port. As explained earlier, an input port performs two important functions: Rate-limiting the traffic inside each bin that shall be served in the RSP mode and evacuating the UFS-ready frames in an orderly way. They are described in Sections 3.1.1 and 3.1.2 respectively.

In this section, we describe only the operations at an input port i; Operations at other input ports are identical. We would like to first precisely specify the set of bins that carries out these operations. Network traffic entering each input port i belongs to one of the N VOQs originated at port i, namely $VOQ_{i1}, VOQ_{i2}, \cdots, VOQ_{iN}$. As mentioned earlier, application flows inside each of these N VOQs will be further divided into N bins via hashing, one corresponding to each intermediate port. More specifically, for each output (destination) port j, packets in VOQ_{ij} are divided into N bins, which we denote as $B_{ij1}, B_{ij2}, \ldots, B_{ijN}$. For any $1 \le m \le N$, packets in bin B_{ijm}, if switched under the RSP mode, should go through intermediate port m. Hence there are a total of N^2 bins at each input port i, namely B_{ijm}, for $j, m = 1, 2, \cdots, N$. As explained earlier, a bin is in the RSP mode to start with, and enters the UFS mode only when its length reaches or exceeds a threshold.

3.1.1 Random Single Path Mode

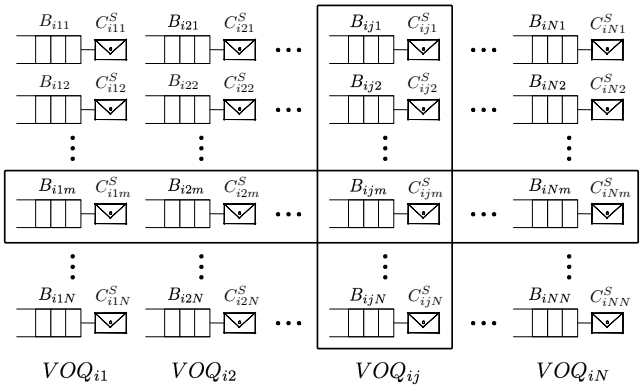

Figure 2: How bins are served and corresponding credit counters updated under the RSP mode.

Algorithm 1 Scheduling RSP packets at input port i

1: **Initialize:** Set all RSP credit counters to a positive integer $C^S_{ijm} = C \ge 1$, $j, m = 1, 2, \ldots, N$;

When connected to intermediate port m:

2: Pick an nonempty the RSP mode bin B_{ijm} with $C^S_{ijm} \ge 1$, in round-robin order, from $\{B_{i1m}, B_{i2m}, \cdots, B_{iNm}\}$ (i.e., bins in row m as highlighted in Figure 2);
3: **if** such a B_{ijm} exists **then**
4: Switch the HOL packet of B_{ijm};
5: Decrement counter C^S_{ijm} by 1;
6: Increment counters $C^S_{ij1}, C^S_{ij2}, \ldots, C^S_{ijN}$ each by $1/N$;
7: **else**
8: Idle;

A logical arrangement of the aforementioned N^2 bins at an input port i is illustrated in Figure 2. Each column of queues (depicting bins) plus "wallets" (depicting credit counters used for performing rate-limiting) correspond to a VOQ. For example, the j-th column, highlighted in the figure, contains N bins B_{ij1}, B_{ij2}, $\cdots$, B_{ijN} that belong to VOQ_{ij}. Upon the arrival of a packet that belongs to VOQ_{ij}, it will be placed in the corresponding input bin B_{ijm} depending on its destination output port j and the hash value of its flow identifier m, waiting to be switched through the corresponding intermediate port m in the RSP mode. Each bin

should be viewed as a queue: All packet arrivals to the bin will be serviced in the FIFO order.

Since the switching of UFS-ready frames takes strict priority over that of RSP-ready packets, input port i serves packets in the RSP mode for a "frame" of N consecutive time slots only when none of the N^2 bins at the input port is UFS-ready right before the first such time slot. During such a "frame" of N time slots, up to N RSP-ready packets destined respectively for intermediate ports $1, 2, \ldots, N$ will be switched. This corresponds to rows $1, 2, \ldots, N$ (of bins) taking turns to receive a unit (packet) of switching service. When it is the turn of row m (highlighted in Figure 2), containing bins $B_{i1m}, B_{i2m}, \ldots, B_{inm}$, to receive service (i.e., currently connected to intermediate port m), if at least one such bin is RSP-ready, one of the RSP-ready bins will be selected to have its HOL packet switched during the time slot. As mentioned earlier, our policy is for the subset of RSP-ready bins in row m – or rather their respectively HOL packets – to receive service in a round-robin fashion, when the turn of row m comes. Such round-robin scheduling can be implemented – with $O(1)$ per packet complexity – by maintaining a linked list of RSP-ready bins. Note this idea of maintaining a linked list (say of active flows) can be traced as far back as to the Deficit Round Robin (DRR) packet scheduler [10].

As mentioned earlier, when in the RSP mode, we need to limit the servicing rate of each bin to $1/N$ of the arrival rate of the VOQ to which the bin belongs. This is achieved through a credit redistribution mechanism, shown in Algorithm 1. Each bin B_{ijm} is associated with a credit counter C^S_{ijm}, depicted as a "wallet" in Figure 2. As shown in Line 1 of Algorithm 1, initially all credit counters are initialized to a positive constant C. Now we are finally ready to precisely define the notion "RSP-ready": A bin is RSP-ready when it is currently in the RSP mode, has a credit amount of at least 1 in its wallet, and is not empty. Once an RSP-ready bin (say B_{ijm}) is chosen for service in the RSP mode – in the round-robin fashion as described above (Line 2 in Algorithm 1) – a credit is taken from its wallet, and a credit of $1/N$ each is deposited to the wallets of all the N bins in the j-th column (Lines 5 - 6). In other words, the unit of credit paid by B_{ijm} for the RSP service, will be evenly distributed to all N bins belonging to VOQ_{ij}, including B_{ijm} itself.

This credit consumption and redistribution mechanism guarantees that, for each VOQ, the difference in the amounts of RSP service rendered to any two of the N bins belonging to the VOQ, during any time interval, is bounded by a constant (more precisely by NC), as we will prove in Lemma 2. Such guarantee implies that during any time interval long enough, the amount of RSP service provided to any bin belonging to a VOQ is about $1/N$ of that provided to the VOQ, which effectively achieves the aforementioned goal of limiting the RSP servicing rate of the bin to at most $1/N$ of the incoming traffic rate of the VOQ. While this credit mechanism is easy to describe, it is very computationally expensive ($O(N)$ per packet) to implement. This complexity can however be easily reduced to $O(1)$ per packet as follows. We increment, instead of all these N counters by $1/N$ each, a single counter by 1 in the round-robin order.[2] In other words, if we last incremented credit counter C^S_{ijm}, then we should increment counter $C^S_{ij[m+1]}$ this time.

[2]It can be shown that Lemma 2 continues to hold with a slightly larger constant ($NC + 1$ instead of NC).

3.1.2 Uniform Frame Spreading (UFS) Mode

In this section, we present the details of the UFS (service) mode at an input port i. We first describe how a bin enters the UFS mode and what happens to it afterwards. Then we explain how the aforementioned orderly evacuation is achieved by carefully scheduling (UFS) service to all UFS-ready bins.

UFS Waiting and Evacuation Phases.

When the queue length of B_{ijm} reaches or exceeds the aforementioned evacuation threshold W, due to the traffic arrival rate of the bin exceeding $1/N$ of that of VOQ_{ij}, its service mode is changed to UFS, meaning that B_{ijm} is no longer eligible for RSP service, even if it still has unused credits, until its queue is eventually cleared by UFS. In this case, the intermediate port m is informed of this change. This notification can be piggybacked to the next packet (RSP or UFS) destined for intermediate port m, and hence does not have to consume a time slot. It is not eligible for the UFS service right away (i.e. not UFS-ready) for the following reason. One or more packets sent earlier from the same input bin B_{ijm} to the intermediate port may still be queued at the intermediate port m, more specifically, at the intermediate bin H_{ijm} as will be explained in Section 3.2. Suppose B_{ijm} is allowed to start receiving UFS service right away and sends out one or more UFS frames to the intermediate ports. Because UFS packets also take strict priority over RSP packets at intermediate ports, as we will explain in Section 3.2, those UFS packets sent to intermediate port m from B_{ijm} may arrive at the output port j earlier than those RSP packets queued in H_{ijm}, causing packet reorder. We say that a bin is in the *UFS waiting phase* when it enters the UFS mode, but is not yet eligible for UFS service.

When the aforementioned intermediate bin H_{ijm} has been cleared at intermediate port m, a notification message[3] is sent back to input port i. Upon receiving the notification, B_{ijm} exits the UFS waiting phase, becomes UFS-ready, and joins the ranks of other UFS-ready bins for the "orderly evacuation". We say that the bin enters the *UFS evacuation phase*. By waiting for H_{ijm} to clear before evacuating B_{ijm}, which prevents the reorder between an earlier RSP packet and a later UFS packet within the same VOQ, SRS ensures correct packet order in every VOQ because, as discussed earlier, reorder cannot happen between two RSP packets or two UFS packets within the same VOQ.

Orderly Evacuation via Careful Scheduling.

Recall there are altogether N^2 bins at input port i, and many of them can be UFS-ready (i.e., in the UFS evacuation phase) at the same time, especially when the traffic load is heavy. As discussed in Section 2.2, these UFS-ready bins need to be evacuated (via UFS) in an orderly fashion to ensure switch stability. The idealized goal of this orderly evacuation is that, for any bin, once it enters the UFS evacuation phase, its queue length should be strictly non-

[3]Note the overhead caused by such notifications is quite small, considering that even an excessively overloaded bin (say with a traffic rate close to 90% of the VOQ it belongs to) triggers such a notification no more frequent than once every $O(W)$ time slots, where $W \geq N$ is the aforementioned UFS evacuation threshold.

increasing over time despite new packet arrivals to this bin, until it exits the UFS mode (after its queue length drops under N). This idealized goal is however unrealistic in practice due to the following service granularity restriction: The smallest unit of UFS service is a frame (of N consecutive time slots), and while a bin is being serviced during these N time slots, queue lengths of some other UFS-ready bins can increase due to new packet arrivals. Hence our (realistic) goal is to ensure that the queue length of any UFS-ready bin is non-increasing over time except for a fluctuation – caused by this service granularity restriction – which we prove (see Lemma 6) is bounded by N^3.

Algorithm 2 Scheduling UFS packets at input port i

1: **Initialize:** Set all UFS pressure counters to $C^U_{ijm} = 0$, $j, m = 1, 2, \ldots, N$;

Upon the arrival of a packet to B_{ijm}:

2: **if** B_{ijm} is in UFS evacuation phase **then**
3: $\quad C^U_{ijm} \leftarrow C^U_{ijm} + 1$;

Upon the departure of a packet from B_{ijm}:

4: **if** B_{ijm} is in UFS evacuation phase **then**
5: $\quad C^U_{ijm} \leftarrow \max\left(0, C^U_{ijm} - 1\right)$;

When connected to intermediate port 1**:**

6: Pick a UFS-ready bin B_{ijm} with the largest UFS pressure counter value at input port i;
7: **if** such a B_{ijm} exists **then**
8: $\quad$ Transmit the HOL frame (of N packets) of B_{ijm} in the next N time slots, triggering the execution of Lines 4 - 5 with each such transmission;
9: **else**
10: $\quad$ Switch packets at input port i in RSP mode in the following N time slots, as described in Algorithm 1;

We propose a scheduler, that is low in both computational and implementation complexities, to achieve this goal. As shown in Algorithm 2, each bin B_{ijm} is associated with a UFS "pressure" counter C^U_{ijm} which is set to 0 (Line 1 in Algorithm 2) when B_{ijm} becomes UFS-ready. This counter C^U_{ijm} tracks, since the last time it was set or decreased to 0, whether the queue length of B_{ijm} has increased – due to the number of new packet arrivals ("back-pressure") exceeding the number of B_{ijm} packets that has been served ("pressure relief") – and if so, by how much, as follows. This pressure counter remains 0 when B_{ijm} is in the UFS waiting phase. After B_{ijm} enters the UFS evacuation phase, we increment C^U_{ijm} by 1 with each new packet arrival to B_{ijm} (Line 3). When a frame of N packets depart from a UFS-ready bin B_{ijm}, we decrement C^U_{ijm} by N; If the value of C^U_{ijm} becomes negative after this decrement, we reset it to 0. This is equivalent to executing Lines 4 - 5 in Algorithm 2 N times. We purposefully write the pseudo code this way to make it consistent with the proof of Lemma 5.

As is clear from our orderly evacuation goal, the larger the value of C^U_{ijm} is, the more urgently B_{ijm} should be serviced. Hence our scheduler adopts the following simple service discipline: Among all UFS-ready bins, the one with the largest UFS pressure counter value is serviced next (Line 6). To implement this scheduler, we need only maintain a heap of UFS-ready bins indexed by their corresponding pressure counter values. The complexity of this implementation, however, is $O(\log N)$ per packet, because every new packet arrival could increase the value of a pressure counter and potentially trigger a "heapify" operation. We can further reduce this complexity to $O(\log N)$ per frame (of N packets) – or $O(1)$ per packet – by making the increment process more coarse, the details of which will be shown in Appendix A.

3.2 Operations at an intermediate port

In SRS, intermediate ports play a passive role in packet and frame scheduling. In fact, they simply follow the RSP and UFS scheduling decisions made by the input ports. More specifically, like input ports, intermediate ports also prioritize UFS frames over RSP packets. In addition, intermediate ports serve both UFS frames and RSP packets in the order dictated by the input ports, which, as far as the intermediate ports are concerned, is the FIFO order.

We describe operations at an intermediate port m: Operations at other intermediate ports are identical. At intermediate port m, N^2 RSP bins H_{ijm} are maintained for $i, j = 1, 2, \cdots, N$. Bin H_{ijm} buffers RSP packets switched from input port i that are destined for output port j. It is clear that all these packets come from input bin B_{ijm}. In addition, N UFS bins U_{jm}, for $j = 1, 2, \cdots, N$, are maintained for buffering UFS packets sent to it from all N input ports, with all UFS packets destined for output port j appended to the end of the bin U_{jm}. Whenever intermediate port m is connected to output port j, the intermediate port m first checks if the corresponding UFS queue U_{jm} is non-empty. If so, it *first* services its HOL packet. Otherwise, it services a RSP packet from one of $H_{1jm}, H_{2jm}, \ldots, H_{Njm}$ in FIFO order[4]. When a bin H_{ijm} is cleared and B_{ijm} is in the UFS waiting phase, a notification to input port i is triggered.

4. STABILITY ANALYSIS

In this section, we prove that SRS is throughput-optimal in the sense that it can achieve 100% throughput. Equivalently, we prove that none of the B_{ijm}, H_{ijm} or U_{jm} bins/queues in SRS can grow in length linearly with time, even when the switch is 100% loaded. This notion of (queue) stability is known as rate-stability [4], because it implies that the long-run average of packet departure rate is equal to the long-run average of packet arrival rate when the switch is no more than 100% loaded. With a slight abuse of notation, we define the queue length of B_{ijm} at time slot t as $B_{ijm}(t)$, $i, j, m = 1, \ldots, N$; $H_{ijm}(t)$ and $U_{jm}(t)$ are defined similarly. Our main result is:

THEOREM 1.

$$(a) \quad \lim_{t\to\infty} \frac{B_{ijm}(t)}{t} = 0, \quad i, j, m = 1, \ldots, N$$

$$(b) \quad \lim_{t\to\infty} \frac{H_{ijm}(t)}{t} = 0, \quad i, j, m = 1, \ldots, N$$

$$(c) \quad \lim_{t\to\infty} \frac{U_{jm}(t)}{t} = 0, \quad j, m = 1, \ldots, N$$

for any deterministic packet arrival process that satisfies a mild admissible condition (to be stated next) and for any arbitrary initial queue lengths of B, H, and U (at time 0).

[4]In fact, our stability proof, shown in the next section, assumes only that the service discipline is work-conserving. We choose a FIFO scheduler simply because it is a "no brainer" for intermediate port m and can be implemented with a computational complexity of $O(1)$ per packet.

Note that in the proof, we actually assume all queues in the switch are empty at time 0. It is however not hard to extend this proof to accommodate arbitrary initial queue lengths, as will be explained at the end of Section 4.2.

We now state the only (mild) admissible condition that we have to exogenously impose on the traffic arrival process. Let $A_{ijm}(t)$ be the cumulative number of packet arrivals into bin B_{ijm} by time slot t (since time 0). Define $A_j(t) \equiv \sum_{i,m=1}^{N} A_{ijm}(t)$. $A_j(t)$ is the total number of packets that have arrived at all input ports by time slot t and are destined for output port j. The admissible condition, which we exogenously impose, is that, for each output port j, that there exist a constant $\lambda_j \in [0,1]$ such that

$$\lim_{t\to\infty} \frac{A_j(t)}{t} = \lambda_j \quad j = 1, 2, \ldots, N \tag{1}$$

In other words, the long-run-average total rate of all traffic destined for output port j must exist and is no more than 1 (i.e., 100% loaded). In the worst case, however, up to N packets destined for output j can arrive, one at each input port, during a single time slot. Therefore, for any $t \geq 0$, we always have

$$A_j(t) \leq Nt \tag{2}$$

Note that, in our proof, we do assume another admissible condition that at most one packet can arrive at each input port in a single time slot. This condition is however imposed "endogenously" by the maximum rate of the network link and the clock speed of the input line card circuitry, not "exogenously" by us.

In the following, we first develop a property of the aforementioned packet arrival process $A_j(t)$ in the form of a general lemma, and then describe some important queuing dynamics of the switch that result from the aforementioned R-SP credit redistribution mechanism and the aforementioned UFS orderly evacuation scheduler. Then we prove Theorem 1 in Section 4.2.

4.1 System dynamics and notations

We develop only the dynamics of queues that hold packets destined for (i.e., associated with) an arbitrary (but fixed) output port j; Queues associated with any other output port have the same dynamics. To facilitate the following presentation, for $i, m = 1, 2, \ldots, N$, we define a set of queue groups as follows

$$\begin{aligned} B_j &\equiv \left\{ B_{ijm} \,\middle|\, i, m = 1, 2, \ldots, N \right\} \\ B_{ij} &\equiv \left\{ B_{ijm} \,\middle|\, m = 1, 2, \ldots, N \right\} \\ G_{jm} &\equiv \{U_{jm}\} \cup \left\{ H_{ijm} \,\middle|\, i = 1, 2, \ldots, N \right\} \end{aligned}$$

Let $B_j(t) = \sum_{i,m=1}^{N} B_{ijm}(t)$ be the total number of packets in queue group B_j at time slot t, $j = 1, \ldots, N$. Similarly we define $G_{jm}(t) = \sum_{i=1}^{N} H_{ijm}(t) + U_{jm}(t)$, $j, m = 1, \ldots, N$.

As mentioned earlier, throughout this paper, time is slotted and is numbered by nonnegative integers, and we use the terms "time" and "time slot" interchangeably. By convention, time starts at (slot) 0 and packets start to arrive at or after (slot) 1. When we say "at/by time (slot) t", we mean "at/by *the end of* time slot t". For example, the queue length of B_{ijm} at time t refers to that at the end of time slot t, which accounts for any (packet) arrival and/or any departure that has happened during time slot t.

4.1.1 Arrival process dynamics

In this section, we state a useful lemma that applies to any deterministic arrival process whose long-run average converges to a constant, including the aforementioned $A_j(t)$. It will be used in the stability proof in Section 4.2.

LEMMA 1. *Let $\{X(t),\ t \geq 0\}$, be an arbitrary deterministic time series. If there exists a constant $\lambda \in \mathbb{R}$ such that $\lim_{t\to\infty} \frac{X(t)}{t} = \lambda$, then for any $\epsilon > 0$ and $p \in (0, 1]$, there exists a constant $T_X > 0$, such that*

$$\left\{ \begin{array}{l} T \geq T_X \\ 0 \leq t_1 < t_2 \leq T \\ t_2 - t_1 \geq pT \end{array} \right\} \Rightarrow\ X(t_2) - X(t_1) \leq (\lambda + \epsilon)(t_2 - t_1)$$

PROOF. As $\lim_{t\to\infty} \frac{X(t)}{t} = \lambda$, for $\epsilon' = p\epsilon/4$, there exists a constant $T' > 0$, such that

$$t \geq T' \Rightarrow \left| \frac{X(t)}{t} - \lambda \right| < \frac{\epsilon'}{2}$$

Let $T_X = \max\left(\frac{2T'}{p}, \frac{T'}{p} + \frac{X(T')}{\frac{\epsilon}{2}\cdot p}\right)$.

As $T \geq T_X \geq \frac{2T'}{p}$ and $t_2 \geq pT$, we have $t_2 - T' \geq t_2 - pT/2 = pT/2$. Then we must have $T' \leq (1 - \frac{1}{2}p)t_2$ or equivalently $T' \leq \frac{2-p}{p}(t_2 - T')$. Otherwise, we'll have $t_2 \geq T' + \frac{1}{2}pT > (1 - \frac{1}{2}p)t_2 + \frac{1}{2}pT$, which implies $t_2 > T$. But this contradicts our assumption that $t_2 \leq T$. Therefore,

$$\begin{aligned} X(t_2) - X(T') &< (\lambda + \frac{\epsilon'}{2})t_2 - (\lambda - \frac{\epsilon'}{2})T' \\ &= (\lambda + \frac{\epsilon'}{2})(t_2 - T') + \epsilon' T' \\ &\leq (\lambda + \frac{\epsilon'}{2})(t_2 - T') + \epsilon' \frac{2-p}{p}(t_2 - T') \\ &= \left(\lambda + \frac{\epsilon'}{2} + \frac{2-p}{p}\epsilon'\right)(t_2 - T') \\ &= \left(\lambda + \frac{4-p}{2p}\epsilon'\right)(t_2 - T') \\ &\leq \left(\lambda + \frac{2}{p}\epsilon'\right)(t_2 - T') \\ &= \left(\lambda + \frac{\epsilon}{2}\right)(t_2 - T') \end{aligned}$$

Furthermore, we have $t_2 \geq pT \geq pT_X \geq \frac{X(T')}{\epsilon/2} + T'$, which implies $X(T') - X(t_1) \leq X(T') \leq \frac{\epsilon}{2}(t_2 - T')$. Hence,

$$\begin{aligned} X(t_2) - X(t_1) &= \big(X(t_2) - X(T')\big) + \big(X(T') - X(t_1)\big) \\ &= \left(\lambda + \frac{\epsilon}{2}\right)(t_2 - T') + \frac{\epsilon}{2}(t_2 - T') \\ &\leq (\lambda + \epsilon)(t_2 - T') \\ &\leq (\lambda + \epsilon)(t_2 - t_1) \end{aligned}$$

□

4.1.2 RSP rate-limiting dynamics

Let $D_{ijm}(t)$ be the cumulative number of packet departures from B_{ijm} by time slot t. Let $D^S_{ijm}(t)$ be the cumulative number of packet departures from B_{ijm} in RSP mode by time slot t. Let $D^S_{ij}(t) = \sum_{m=1}^{N} D^S_{ijm}(t)$ be the cumulative number of packet departures from queue group B_{ij} in RSP mode by time slot t. Let $D_j(t) = \sum_{i,m=1}^{N} D_{ijm}(t)$

be the cumulative number of packet departures from queue group B_j by time slot t. Some properties of the RSP credit redistribution mechanism shown in Algorithm 1 are characterized in the following lemmas. Lemma 2 was mentioned earlier in Section 3.1.1.

LEMMA 2. *For any two input port bins in the same queue group B_{ij}, say B_{ijm} and $B_{ijm'}$, we have*

$$|D^S_{ijm}(t) - D^S_{ijm'}(t)| \le NC$$

PROOF. Let $C^S_{ijm}(t)$ and $C^S_{ijm'}(t)$ denote the value of credit counter C^S_{ijm} and $C^S_{ijm'}$ at time slot t respectively. Note that for any input port queue group B_{ij}, the total RSP credits stay unchanged at anytime. So we always have $C^S_{ijm}(t) + C^S_{ijm'}(t) \le NC$. From Lines 5 - 6 in Algorithm 1, we know that

$$C^S_{ijm}(t) = C - D^S_{ijm}(t) + \frac{1}{N} D^S_{ij}(t)$$

i.e., $D^S_{ijm}(t) = C + \frac{1}{N} D^S_{ij}(t) - C^S_{ijm}(t)$. Similarly, we have $D^S_{ijm'}(t) = C + \frac{1}{N} D^S_{ij}(t) - C^S_{ijm'}(t)$. Thus,

$$\begin{aligned} |D^S_{ijm}(t) - D^S_{ijm'}(t)| &= |C^S_{ijm}(t) - C^S_{ijm'}(t)| \\ &\le |C^S_{ijm}(t) + C^S_{ijm'}(t)| \\ &\le NC \end{aligned}$$

□

Let $I^S_{ijm}(t)$ be the cumulative number of packets that arrive at H_{ijm} in RSP mode by time slot t. Let $I_{jm}(t)$ be the cumulative number of packets that arrive at queue group G_{jm} by time slot t. $I_{jm}(t)$ consists of two types of traffic: One is the traffic sent to $H_{1jm}, H_{2jm}, \ldots, H_{Njm}$ in RSP mode, and the other is the traffic sent to U_{jm} in UFS mode. We denote them by $I^S_{jm}(t)$ and $I^U_{jm}(t)$, respectively. Note that $I^S_{ijm}(t) = D^S_{ijm}(t)$ and $D_j(t) = \sum_{m=1}^N I_{jm}(t)$.

LEMMA 3. *For any two intermediate ports m and m', we always have $|I_{jm}(t) - I_{jm'}(t)| \le N^2C + 1$.*

PROOF. As $I^S_{jm}(t) - I^S_{jm'}(t) = \sum_{i=1}^N \left(I^S_{ijm}(t) - I^S_{ijm'}(t)\right) = \sum_{i=1}^N \left(D^S_{ijm}(t) - D^S_{ijm'}(t)\right)$, by Lemma 2, we have

$$\left|I^S_{jm}(t) - I^S_{jm'}(t)\right| \le \sum_{i=1}^N \left|D^S_{ijm}(t) - D^S_{ijm'}(t)\right| \le N^2C$$

Furthermore, if an input port bin is in UFS mode, around the time it sends one packet to intermediate port m, it must also send one packet from the same frame to intermediate port m'. Thus, we always have $\left|I^U_{jm}(t) - I^U_{jm'}(t)\right| \le 1$ for any $t \ge 0$. Therefore,

$$\begin{aligned} |I_{jm}(t) - I_{jm'}(t)| &= \left|(I^S_{jm}(t) + I^U_{jm}(t)) - (I^S_{jm'}(t) + I^U_{jm'}(t))\right| \\ &\le \left|I^S_{jm}(t) - I^S_{jm'}(t)\right| + \left|I^U_{jm}(t) - I^U_{jm'}(t)\right| \\ &= N^2C + 1 \end{aligned}$$

□

LEMMA 4. *For $0 \le t_1 < t_2$, if there exists an intermediate port m' such that $G_{jm'}(t_1) = 0$ and $G_{jm'}(t) > 0$ for any $t \in [t_1+1, t_2]$, we have $D_j(t_2) - D_j(t_1) \ge (t_2 - t_1) + NG_{jm'}(t_2) - 5N^3C$.*

PROOF. Let $O_{jm'}(t)$ be the cumulative number of packets departure from queue group $G_{jm'}$ by time slot t. Since $G_{jm'}$ is non-empty from time slot $(t_1 + 1)$ to t_2, it must send out 1 packet per N time slots. We have

$$\begin{aligned} &I_{jm'}(t_2) - I_{jm'}(t_1) \\ &= O_{jm'}(t_2) - O_{jm'}(t_1) + G_{jm'}(t_2) - G_{jm'}(t_1) \\ &\ge \left\lfloor \frac{1}{N}(t_2 - t_1) \right\rfloor + G_{jm'}(t_2) \\ &\ge \frac{1}{N}(t_2 - t_1) - 1 + G_{jm'}(t_2) \end{aligned}$$

where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x. From Lemma 3, we know that $I_{jm}(t_2) \ge I_{jm'}(t_2) - (N^2C+1)$ and $I_{jm}(t_1) \le I_{jm'}(t_1) + N^2C + 1$ (which will be used in the first inequality below). Therefore,

$$\begin{aligned} &D_j(t_2) - D_j(t_1) = \sum_{m=1}^N I_{jm}(t_2) - \sum_{m=1}^N I_{jm}(t_1) \\ &= \sum_{m=1}^N \left(I_{jm}(t_2) - I_{jm}(t_1)\right) \\ &\ge \sum_{m=1}^N \left(I_{jm'}(t_2) - (N^2C+1) - (I_{jm'}(t_1) + N^2C + 1)\right) \\ &= \sum_{m=1}^N \left(I_{jm'}(t_2) - I_{jm'}(t_1) - 2(N^2C+1)\right) \\ &\ge \sum_{m=1}^N \left(\frac{1}{N}(t_2 - t_1) - 1 + G_{jm'}(t_2) - 2(N^2C+1)\right) \\ &\ge (t_2 - t_1) + NG_{jm'}(t_2) - 5N^3C \end{aligned}$$

□

4.1.3 UFS orderly evacuation dynamics

In this section, we investigate the effects of the aforementioned "orderly evacuation" scheduler on the dynamics of B bins/queues, They are characterized in Lemma 6 and Corollary 1 below. For convenience of presentation, we simply assume that B_{ijm} experiences a (degenerated) UFS waiting period of length 0 if H_{ijm} is already cleared (i.e., has length 0) when B_{ijm} enters the UFS mode. Recall from Section 3.1.2 that C^U_{ijm} is the UFS pressure counter associated with bin B_{ijm}. Let $C^U_{ijm}(t)$ denote the value of C^U_{ijm} at time t. As mentioned earlier, when B_{ijm} is in the UFS evacuation phase, C^U_{ijm} tracks the change of its queue length except that it could be allowed to transmit a UFS packet (as a part of a UFS frame) at time t, even if $C^U_{ijm}(t)$ is 0 (Line 5 in Algorithm 2), and $C^U_{ijm}(t)$ remains 0 after such a transmission. We say B_{ijm} "steals service" at such moments (without having $C^U_{ijm}(t)$ decremented). Our scheduler purposefully allows such a behavior, since Algorithm 2 guarantees that B_{ijm} needs UFS evacuation most urgently whenever it's scheduled (Line 6). If B_{ijm} steals service at time t, UFS pressure counters of all other UFS-ready bins at input port i should also have values equal or close to 0 at that time, indicating none of them needs evacuation urgently anyway. Hence B_{ijm} should not be punished for stealing service at time t. To prove Lemma 6, we need the following technical lemma. Its proof is provided in Appendix B.

LEMMA 5. *If none of the bins at input port i has ever stolen service throughout time slots t_1 to t_2, we must have*

$$\sum_{j,m=1}^{N} C_{ijm}^{U}(t_2) \le \max\left(N, \sum_{j,m=1}^{N} C_{ijm}^{U}(t_1)\right)$$

Recall that the goal of the orderly (UFS) evacuation process is that, once the evacuation starts on a bin, its queue length should be non-increasing over time, except for a small unavoidable fluctuation amount. The following lemma states that our scheduler achieves this goal.

LEMMA 6. *If B_{ijm} remains in UFS evacuation phase throughout time slots t_1 to t_2, we have $B_{ijm}(t_2) - B_{ijm}(t_1) \le N^3$.*

PROOF. Whenever B_{ijm} is in UFS evacuation phase, C_{ijm}^{U} will be incremented by 1 upon the arrival of every packet and be decremented by *at most* 1 upon the departure of every packet. Thus we have

$$B_{ijm}(t_2) - B_{ijm}(t_1) \le C_{ijm}^{U}(t_2) - C_{ijm}^{U}(t_1) \le C_{ijm}^{U}(t_2)$$

Therefore it's sufficient to prove that $C_{ijm}^{U}(t_2) \le N^3$.

If none of the bins at input port i has ever stolen service throughout $[0, t_2]$, by Lemma 5, we have

$$\sum_{j,m=1}^{N} C_{ijm}^{U}(t_2) \le \max\left(N, \sum_{j,m=1}^{N} C_{ijm}^{U}(0)\right) = N \le N^3$$

Otherwise, let $t' > 1$ be the time such that none of the UFS-ready bins at input port i has ever stolen service throughout time slots t' to t_2, but some bin, say $B_{ij'm'}$, steals service in time slot $(t'-1)$ to send out a packet *pkt*. If so, $B_{ij'm'}$ should have been scheduled to send the first packet of the frame, to which *pkt* belongs, at some time slot $t'' \in [t'-N,\ t'-1]$. At that time, we must have $C_{ij'm'}^{U}(t'') \le N-1$ and $C_{ij'm'}^{U}(t'') = \max_{j,m=1}^{N} C_{ijm}^{U}(t'')$. Thus we have

$$\begin{aligned}\sum_{j,m=1}^{N} C_{ijm}^{U}(t') &\le \sum_{j,m=1}^{N} C_{ijm}^{U}(t'') + (t'-t'') \\ &\le N^2(N-1) + N \\ &\le N^3\end{aligned}$$

Since no bin has stolen services throughout $[t', t_2]$, by Lemma 5, we have

$$C_{ijm}^{U}(t_2) \le \sum_{j,m=1}^{N} C_{ijm}^{U}(t_2) \le \max\left(N, \sum_{j,m=1}^{N} C_{ijm}^{U}(t')\right) \le N^3$$

□

COROLLARY 1. *If B_{ijm} has never been in UFS waiting phase throughout time slots t_1 to t_2, we have*

$$B_{ijm}(t_2) - B_{ijm}(t_1) \le W^3$$

PROOF. If B_{ijm} is in RSP mode at time t_2, we have $B_{ijm}(t_2) < W$ and thus

$$B_{ijm}(t_2) - B_{ijm}(t_1) \le B_{ijm}(t_2) < W \le W^3$$

Otherwise, if B_{ijm} is in UFS evacuation phase at t_2, it must be in UFS evacuation phase throughout $[t_1, t_2]$. By Lemma 6, we have $B_{ijm}(t_2) - B_{ijm}(t_1) \le N^3 \le W^3$. □

4.2 Proof of Theorem 1

In this section we present the proof of Theorem 1 in details. We first prove Theorem 1(a) (i.e., the stability of B bins) using the aforementioned extremal argument. Theorem 1(b) and 1(c) (i.e., the stability of H and U bins) are then much easier to prove, which will be presented in Appendix C.

LEMMA 7. *Suppose B_{ijm} remains in UFS waiting phase throughout time slots t_1 to t_2, there must exist a time slot $t_0 \le t_1$ such that $G_{jm}(t_0) = 0$ and $G_{jm}(t) > 0$ for any $t \in [t_0+1, t_2]$.*

PROOF. Since H_{ijm} should be non-empty whenever B_{ijm} is in the UFS waiting phase, we have $G_{jm}(t) \ge H_{ijm}(t) > 0$ for any $t \in [t_1, t_2]$. Note that initially we have $G_{jm}(0) = 0$. Thus there must exist a time slot $t_0 \le t_1$ such that $G_{jm}(t_0) = 0$ and $G_{jm}(t) > 0$ for any $t \in [t_0+1,\ t_2]$. □

THEOREM 2.

$$\limsup_{t\to\infty} \frac{B_j(t)}{t} = 0 \quad j = 1, 2, \ldots, N$$

PROOF. We prove the theorem by contradiction. Suppose there exists an output port j such that

$$\limsup_{t\to\infty} \frac{B_j(t)}{t} = \gamma > 0 \tag{3}$$

Let $f(x) = \frac{\gamma+5x}{2\left(1+\frac{1}{\gamma+x}\right)^{N^2}} - 3x$. Since $f(x)$ is a continuous function of x in a neighborhood of 0 and $f(0) > 0$, there must exist an $\epsilon' > 0$ such that $f(x) > 0$ for $x \in (0, \epsilon')$. Let $\epsilon = \min(\frac{\gamma}{4}, \frac{\epsilon'}{2})$, by (3), there exists an integer $T' > 0$, such that

$$t > T' \Rightarrow \frac{B_j(t)}{t} < (\gamma + \epsilon) \tag{4}$$

By Lemma 1, for $p = \min\left(\frac{f(\epsilon)}{2(\gamma+\epsilon)}, 1\right)$, there exists an integer $T_A > 0$, such that

$$\left\{\begin{array}{l} T > T_A \\ 0 \le t_1 < t_2 \le T \\ t_2 - t_1 \ge pT \end{array}\right\} \Rightarrow A_j(t_2) - A_j(t_1) \le (1+\epsilon)(t_2 - t_1) \tag{5}$$

From (3), for the same ϵ, there exists another integer

$$T_2 > \max\left(T_A, \frac{16W^5C}{f(\epsilon)}, \frac{4W^5}{\gamma-\epsilon}, \frac{2(NT' + 8W^5C)}{\gamma - 3\epsilon}\right) \tag{6}$$

such that

$$\frac{B_j(T_2)}{T_2} > (\gamma - \epsilon) \tag{7}$$

Define two number sequences $\{a_k\}_{k=0}^{N^2}$ and $\{S_k\}_{k=0}^{N^2}$ as

$$a_k = \begin{cases} f(\epsilon)T_2 + 8W^5C\left(\frac{1}{\left(1+\frac{1}{\gamma+\epsilon}\right)^{N^2}} - 1\right) & \text{if } k = 0 \\ \frac{1}{\gamma+\epsilon}\left(1+\frac{1}{\gamma+\epsilon}\right)^{k-1}\left(a_0 + 3\epsilon T_2 + 8W^5C\right) & \text{if } k \ge 1 \end{cases}$$

$$S_k = \sum_{i=0}^{k} a_i \qquad k = 0, 1, \ldots, N^2$$

It's not hard to verify the following properties of these two sequences.

- The following equation holds for $k = 1, 2, \ldots, N^2$

$$a_k = \frac{1}{\gamma+\epsilon} S_{k-1} + \frac{1}{\gamma+\epsilon}(3\epsilon T_2 + 8W^5 C) \qquad (8)$$

- Since $\epsilon < \epsilon'$ and $T_2 > \frac{16W^5C}{f(\epsilon)}$, we have $f(\epsilon) > 0$ and

$$a_0 \geq f(\epsilon)T_2 - 8W^5 C \geq \frac{f(\epsilon)}{2} T_2$$

Note that $a_{k+1} > a_k$ for $k \geq 1$, thus for $k = 1, 2, \ldots, N^2$ we have

$$a_k \geq a_1 \geq \frac{1}{\gamma+\epsilon} a_0 \geq \frac{f(\epsilon)}{2(\gamma+\epsilon)} T_2 \geq pT_2$$

- When $k = N^2$, we have

$$S_{N^2} = \frac{\gamma - \epsilon}{2} T_2$$

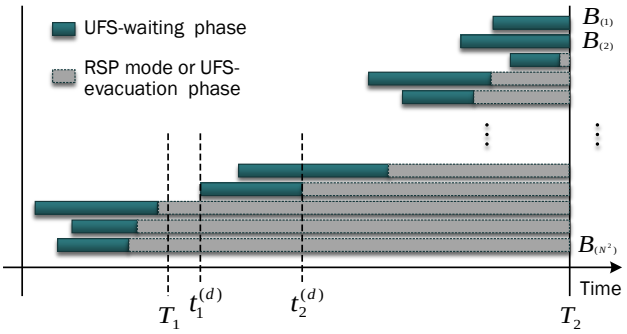

Figure 3: Last UFS waiting phases of $B_{(k)}$'s before T_2.

Let $[t_1^{[ijm]}, t_2^{[ijm]}]$, $i, m = 1, 2, \ldots, N$, be the last UFS waiting phase of B_{ijm} before T_2. If B_{ijm} has never been in UFS waiting phase throughout $[0, T_2]$, we simply set $t_1^{[ijm]} = t_2^{[ijm]} = 0$. If the last UFS waiting phase of B_{ijm} does not end before T_2, we set $t_2^{[ijm]} = T_2$. Now we temporarily rewrite the term $t_2^{[ijm]}$ as $t_2^{(i,j,m)}$, and B_{ijm} as $B_{(i,j,m)}$, in order to introduce the following "order statistics" of $t_2^{[ijm]}$ and the relabeling accordingly of B_{ijm}, for $i, m = 1, 2, \ldots, N$. Define $\mathcal{V}_j$ as

$$\mathcal{V}_j = \{(i, j, m) \mid i, m = 1, 2, \ldots, N\}$$

Let $\sigma : \mathcal{V}_j \to \{1, 2, \ldots, N^2\}$ be a bijection such that $t_2^{\sigma^{-1}(1)} \leq t_2^{\sigma^{-1}(2)} \leq \cdots \leq t_2^{\sigma^{-1}(N^2)}$. To make our presentation more succinct, we use $B_{(k)}$, $t_1^{(k)}$ and $t_2^{(k)}$ as shorthands for $B_{\sigma^{-1}(k)}$, $t_1^{\sigma^{-1}(k)}$ and $t_2^{\sigma^{-1}(k)}$, respectively, in the sequel. Also to make the reasoning easier, we define a dummy bin $B_{(0)}$ with $B_{(0)}(t) = 0$ forever, and a corresponding dummy interval $t_1^{(0)} = t_2^{(0)} = T_2$. Thus, for any integer $k \in [1, N^2]$, we always have $t_2^{(k-1)} \leq t_2^{(k)} \leq T_2$, and $B_{(k)}$ should never be in UFS waiting phase throughout $[t_2^{(k)}, T_2]$, as shown in Figure 3. From Corollary 1, we know that

$$B_{(k)}(T_2) - B_{(k)}(t_2^{(k)}) \leq W^3 \quad k = 0, 1, \ldots, N^2 \qquad (9)$$

Hence

$$B_j(T_2) = \sum_{i,m=1}^{N} B_{ijm}(T_2) = \sum_{k=0}^{N^2} B_{(k)}(T_2)$$

$$\begin{aligned}
&= \sum_{k=0}^{N^2} \Big(B_{(k)}(t_2^{(k)}) - B_{(k)}(t_1^{(k)}) \Big) + \sum_{k=0}^{N^2} B_{(k)}(t_1^{(k)}) \\
&\quad + \sum_{k=0}^{N^2} \Big(B_{(k)}(T_2) - B_{(k)}(t_2^{(k)}) \Big) \\
&\leq \sum_{k=0}^{N^2} \Big(t_2^{(k)} - t_1^{(k)} \Big) + \sum_{k=1}^{N^2} W + \sum_{k=1}^{N^2} W^3 \qquad (10) \\
&\leq \sum_{k=0}^{N^2} \Big(t_2^{(k)} - t_1^{(k)} \Big) + 2W^5
\end{aligned}$$

which implies that $\sum_{k=0}^{N^2} \left(t_2^{(k)} - t_1^{(k)} \right) \geq B_j(T_2) - 2W^5 \geq (\gamma - \epsilon)T_2 - 2W^5 \geq \frac{\gamma-\epsilon}{2} T_2 = S_{N^2} \equiv \sum_{k=0}^{N^2} a_k$. The last inequality holds since $T_2 \geq \frac{4W^5}{\gamma-\epsilon}$ (due to (6)). Therefore, there must exist an integer $k' \in [1, N^2]$ such that $t_2^{(k')} - t_1^{(k')} \geq a_{k'}$ (note that $t_2^{(0)} - t_1^{(0)} = 0 < a_0$). Let d be the smallest such integer. In other words, we have $t_2^{(k)} - t_1^{(k)} < a_k$ for $k = 0, 1, \ldots, d-1$ and $t_2^{(d)} - t_1^{(d)} \geq a_d$. Note that for any $k \geq d$, $B_{(k)}$ must never be in UFS waiting phase throughout $[t_2^{(d)}, T_2]$. By Corollary 1, we have $B_{(k)}(T_2) - B_{(k)}(t_2^{(d)}) \leq W^3$, $k = d, \ldots, N^2$. This inequality and (9) will be used in the first inequality below. Similar to (10), we can prove that

$$\begin{aligned}
B_j(T_2) &= \sum_{k=0}^{N^2} B_{(k)}(T_2) = \sum_{k=0}^{d-1} B_{(k)}(T_2) + \sum_{k=d}^{N^2} B_{(k)}(T_2) \\
&= \sum_{k=0}^{d-1} B_{(k)}(t_1^{(k)}) + \sum_{k=0}^{d-1} \Big(B_{(k)}(t_2^{(k)}) - B_{(k)}(t_1^{(k)}) \Big) \\
&\quad + \sum_{k=0}^{d-1} \Big(B_{(k)}(T_2) - B_{(k)}(t_2^{(k)}) \Big) \\
&\quad + \sum_{k=d}^{N^2} B_{(k)}(t_2^{(d)}) + \sum_{k=d}^{N^2} \Big(B_{(k)}(T_2) - B_{(k)}(t_2^{(d)}) \Big) \\
&\leq \sum_{k=0}^{d-1} W + \sum_{k=0}^{d-1} \Big(t_2^{(k)} - t_1^{(k)} \Big) + \sum_{k=0}^{d-1} W^3 \\
&\quad + \sum_{k=d}^{N^2} B_{(k)}(t_2^{(d)}) + \sum_{k=d}^{N^2} W^3 \\
&\leq \sum_{k=d}^{N^2} B_{(k)}(t_2^{(d)}) + \sum_{k=0}^{d-1} \Big(t_2^{(k)} - t_1^{(k)} \Big) + 3W^5 \\
&\leq \sum_{k=0}^{N^2} B_{(k)}(t_2^{(d)}) + \sum_{k=0}^{d-1} a_k + 3W^5 C \\
&= B_j(t_2^{(d)}) + S_{d-1} + 3W^5 C
\end{aligned}$$

By Lemma 7, there exists a time slot $T_1 \leq t_1^{(d)}$ such that $G_{jm}(T_1) = 0$ and $G_{jm}(t) > 0$ for any $t \in [T_1 + 1, t_2^{(d)}]$. Note that $T_2 \geq T_A$ and $t_2^{(d)} - T_1 \geq a_d > pT_2$, by (5), we know that $A_j(t_2^{(d)}) - A_j(T_1) \leq (1+\epsilon)(t_2^{(d)} - T_1)$. By Lemma 4, we have $D_j(t_2^{(d)}) - D_j(T_1) \geq (t_2^{(d)} - T_1) - 5N^3 C$. Hence,

$$\begin{aligned}
B_j(t_2^{(d)}) &= B_j(T_1) + \big(A_j(t_2^{(d)}) - A_j(T_1) \big) - \big(D_j(t_2^{(d)}) - D_j(T_1) \big) \\
&\leq B_j(T_1) + (1+\epsilon)(t_2^{(d)} - T_1) - \big((t_2^{(d)} - T_1) - 5N^3 C \big)
\end{aligned}$$

$$= B_j(T_1) + \epsilon\,(t_2^{(d)} - T_1) + 5N^3C$$
$$\leq B_j(T_1) + \epsilon T_2 + 5W^3C$$

Therefore

$$B_j(T_2) \leq B_j(t_2^{(d)}) + S_{d-1} + 3W^5C$$
$$\leq B_j(T_1) + \epsilon T_2 + S_{d-1} + 8W^3C$$

We need only to consider the following two cases. Both lead to conclusions that contradict (7).

(i) If $T_1 > T'$, we have $T_1 \leq T_2 - (t_2^{(d)} - t_1^{(d)}) \leq T_2 - a_d$ and $B_j(T_1) \leq (\gamma+\epsilon)T_1 \leq (\gamma+\epsilon)(T_2 - a_d)$ (due to (4)). We have

$$B_j(T_2) \leq B_j(T_1) + \epsilon T_2 + S_{d-1} + 8W^5C$$
$$\leq (\gamma+\epsilon)(T_2 - a_d) + \epsilon T_2 + S_{d-1} + 8W^5C$$
$$= (\gamma + 2\epsilon)T_2 - \big((\gamma+\epsilon)a_d - S_{d-1}\big) + 8W^5C$$
$$= (\gamma + 2\epsilon)T_2 - (3\epsilon T_2 + 8W^5C) + 8W^5C$$
$$= (\gamma - \epsilon)T_2$$

The second equality above is from the relationship between a_d and S_{d-1} given in (8).

(ii) If $T_1 \leq T'$, by (2), we have $B_j(T_1) \leq A_j(T_1) \leq NT_1 \leq NT'$ and then

$$B_j(T_2) \leq B_j(T_1) + \epsilon T_2 + S_{d-1} + 8W^5C$$
$$\leq NT' + \epsilon T_2 + S_{N^2} + 8W^5C$$
$$= NT' + \epsilon T_2 + \frac{\gamma - \epsilon}{2} T_2 + 8W^5C$$
$$= NT' + \frac{\gamma + \epsilon}{2} T_2 + 8W^5C$$
$$\leq (\gamma - \epsilon)T_2$$

The last inequality holds since $\epsilon = \min(\frac{\gamma}{4}, \frac{\epsilon'}{2})$, which implies $\gamma - 3\epsilon > 0$, and $T_2 \geq \frac{2(NT' + 8W^5C)}{\gamma - 3\epsilon}$.

□

Theorem 2 implies that

$$\lim_{t\to\infty} \frac{B_{ijm}(t)}{t} = 0 \quad i, j, m = 1, 2, \ldots, N$$

which is Theorem 1(a).

Remark. As mentioned at the beginning of Section 4, SRS remains stable when B, H, U queues are not empty to start with (at time 0). More specifically, it can be shown that all results in Sections 4.1 and 4.2 continue to hold with some minor changes, when initial queue lengths can be nonzero. For example, in that case, we can only claim $G_{jm'}(t_1) \leq G_{jm'}(0)$ in Lemma 4, (instead of $G_{jm'}(t_1) = 0$) so its conclusion has to be changed to $D_j(t_2) - D_j(t_1) \geq (t_2 - t_1) + N(G_{jm'}(t_2) - G_{jm'}(0)) - 5N^3C$.

5. 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) [8], Full-Ordered Frame First (FOFF) [8], Padded Frames (PF) [7], and the recently proposed Sprinklers scheme [5]. The basic load-balancing scheme (labeled "Basic") does not guarantee packet ordering, but it provides the lower bound of the average delay that a load-balanced switch can achieve. UFS, FOFF, PF, and Sprinklers are known to provide reasonably good performance and all of them guarantee packet ordering.

To simulate the effects of grouping application flows (i.e., TCP/UDP flows) into bins by hashing in SRS, we generate application flows over the simulation period using flow statistics measured from real-world Internet traffic traces. Specifically, we assume that the arrival of new application flows to an input port follows a Poisson process, and the rate and duration of each new application flow, viewed as a random vector, follows the corresponding joint empirical distribution measured from the traffic traces. The rate of this Poisson process is set according to the intended traffic rate λ of the input port in a simulation run, and the measured empirical average size (in number of packets) of an applications flow.

The traces that we used were collected by University of North Carolina (UNC) on a 1 Gbps access link connecting the campus to the rest of the Internet on April 24, 2003. It contains 198,944,706 packet headers and around 13.5 million flows. In our simulation study, traffic into each input port is generated according to the empirical flow statistics measured from this trace. We also use different traffic patterns in our evaluation. The size of the switch in the simulation study is $N = 64$. The RSP-to-UFS mode threshold is set to $W = N$ and the initial RSP credits is set to $C = 100N$.

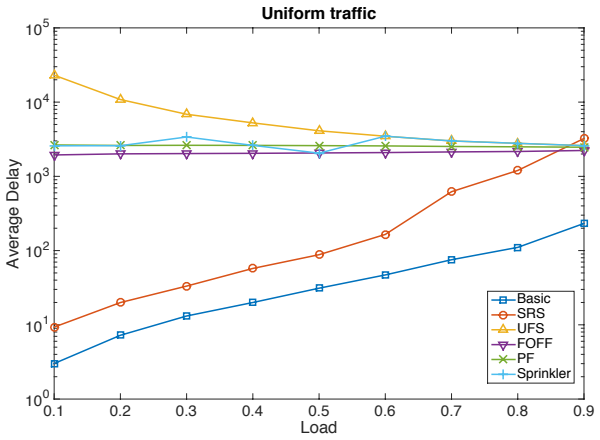

Figure 4: Average delay under uniform traffic.

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 Fig. 4. 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 Fig. 5.

Several observations can be made from the above experiments. First, for uniform traffic, the average delay of SRS is significantly better than the existing methods of UFS, FOFF, PF, and Sprinklers for all traffic loads up to about $\lambda < 0.9$ and the average delay is only slightly more for $\lambda \geq 0.9$. Similarly, for quasi-diagonal traffic, the average delay of SRS is significantly better than the existing methods of UFS, FOFF, PF, and Sprinklers for all traffic loads up to about $\lambda < 0.8$, and the average delay is only slightly more for $\lambda \geq 0.8$. Actually, when the traffic load is heavy, SRS

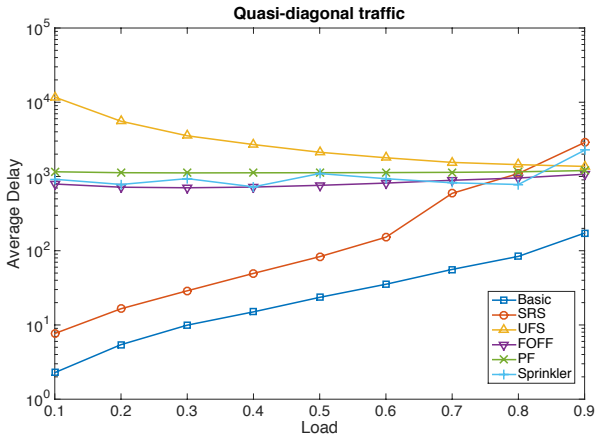

Figure 5: Average delay under quasi-diagonal traffic.

should mainly work in the UFS mode. Comparing with the basic UFS scheme, the SRS maintains N bins for each VOQ instead of 1, making it slower to accumulate a full-frame of N packets. But this could be mitigated by allowing an input port to wait for a VOQ to accumulate W packets rather than wait for an individual bin to accumulate W packets when the the packet rate is fast enough. 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.

6. RELATED WORK

This section briefly reviews existing solutions to the packet reordering problem in load-balanced switches. As already discussed, Uniform Frame Spreading (UFS) [8] prevents reordering by requiring that each VOQ first accumulates a full-frame of N packets before they are uniformly spread across the N intermediate ports. The main drawback of UFS is that it suffers from $O(N^3)$ delay in the worst-case. Full Ordered Frames First (FOFF) [8] also uniformly spreads full-frames when available. When no full-frame is available, FOFF will serve incomplete frames in a round-robin manner, but it suffers from $O(N^2)$ delay in the worst-case for packet reordering at the output. Padded Frames (PF) [7] is another method that avoids the need to accumulate full-frames. When no full-frame is available, PF will pad the longest incomplete frame with fake packets to create a full-frame, which is then uniformly spread across the N intermediate ports, just like UFS. However, its worst-case delay is still $O(N^3)$. Recently, an approach called Sprinklers [5] was proposed based on the idea of variable-size striping. Sprinklers uses the arrival rate of packets to a VOQ to determine a variable stripe size L rather than always requiring a full-frame. It then only requires the accumulation of L packets in a VOQ before uniformly spreading them across a randomly chosen contiguous block of L intermediate ports, where VOQs with slower arrival rates are given smaller stripe sizes. Finally, packet ordering can be guaranteed via another approach called a Concurrent Matching Switch (CMS) [9], which enforces packet ordering throughout the switch by using a fully distributed load-balanced scheduling approach.

7. CONCLUSIONS

In this paper, we proposed SRS, a simple randomized load-balanced switch architecture based on the hashing of application flows, combined with safety mechanisms to prevent unstable built-up of packets at intermediate queues throughout the switch. We rigorously proved that the proposed SRS guarantees both stability under arbitrary admissible traffic and packet ordering. We showed experimentally that SRS significantly outperforms existing load-balanced switch architectures. Finally, we believe that this work will serve as a catalyst to a rich family of solutions based on the simple principles of flow randomization.

8. 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] Dai, J. G. *Stability of fluid and stochastic processing networks*. University of Aarhus. Centre for Mathematical Physics and Stochastics (MaPhySto)[MPS], 1998.

[4] Dai, J. G., and Prabhakar, B. 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* (2000), vol. 2, IEEE, pp. 556–564.

[5] 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.

[6] Estan, C., and Varghese, G. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS) 21*, 3 (2003), 270–313.

[7] 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.

[8] Keslassy, I. *The load-balanced router*. PhD thesis, Stanford University, 2004.

[9] Lin, B., and Keslassy, I. The concurrent matching switch architecture. *Networking, IEEE/ACM Transactions on 18*, 4 (2010), 1330–1343.

[10] Shreedhar, M., and Varghese, G. Efficient fair queueing using deficit round robin. In *ACM SIGCOMM Computer Communication Review* (1995), vol. 25, ACM, pp. 231–242.

[11] Zhao, Q., Xu, J., and Liu, Z. Design of a novel statistics counter architecture with optimal space and time efficiency. *ACM SIGMETRICS Performance Evaluation Review 34*, 1 (2006), 323–334.

APPENDIX

A. EFFICIENT IMPLEMENTATION OF UFS PRESSURE COUNTERS

We associate another counter, called lazy pressure counter, with each bin. We increment the value of a lazy pressure counter only once by N after every N increments to the corresponding normal pressure counter. Hence, on average only one lazy pressure counter is incremented (by N) every N time slots. Then by indexing the heap nodes instead by the lazy pressure counter values of the corresponding UFS-ready bins, on average only one heapify operation, with $O(\log N)$ complexity, is triggered every frame (i.e., N time slots).[5] It can be shown that, by making scheduling decisions based on the slightly outdated "pressure readings" indicated by the lazy pressure counter values, our streamlined scheduler increases the aforementioned fluctuation bound N^3 only slightly. Our stability proof, based on the guarantees of the original scheduler, only needs to be slightly modified to accommodate this increase.

B. PROOF OF LEMMA 5

PROOF. Let $\mathcal{B}_i^U(t)$ be the set of bins at input port i that are in the UFS evacuation phase at time t. If $\mathcal{B}_i^U(t_2) = \emptyset$, we must have $C_{ijm}^U(t_2) = 0$ for $j, m = 1, 2, \ldots, N$ and thus $\sum_{j,m=1}^N C_{ijm}^U(t_2) = 0 \leq N$. Otherwise, let $t' \in [0, t_2]$ be the (unique) time such that $\mathcal{B}_i^U(t') = \emptyset$ and $\mathcal{B}_i^U(t) \neq \emptyset$ for any time slot $t \in [t'+1, t_2]$. Note that t' must exist since $\mathcal{B}_i^U(0) = \emptyset$. Then we must have $\sum_{j,m=1}^N C_{ijm}^U(t') = 0$. Note that when input port i is connected to intermediate port 1 at a time slot $t'' \in [t'+1, t'+N]$, one of the following statements must be true.

- If $t_2 \leq t''$, we have $t_2 - t' \leq N$. Note that at most one packet can arrive at each input port in a single time slot and the value of $\sum_{j,m=1}^N C_{ijm}^U$ is incremented by at most 1 per packet arrival (Line 3 in Algorithm 2), we can obtain

$$\sum_{j,m=1}^N C_{ijm}^U(t_2) \leq \sum_{j,m=1}^N C_{ijm}^U(t') + (t_2 - t') \leq N$$

- If $t_2 > t''$ and $t_1 \leq t''$, input port i must send out exactly one packet in UFS mode every time slot throughout $[t'', t_2]$. $\sum_{j,m=1}^N C_{ijm}^U$ should then be decremented by 1 upon every departure of such packets since none of the UFS evacuation phase bins at input port i has stolen service from t'' to t_2. On the other hand, $\sum_{j,m=1}^N C_{ijm}^U$ can be incremented by at most 1 every time slot. Therefore $\sum_{j,m=1}^N C_{ijm}^U$ is strictly not increased throughout $[t'', t_2]$. Hence,

$$\sum_{j,m=1}^N C_{ijm}^U(t_2) \leq \sum_{j,m=1}^N C_{ijm}^U(t'')$$

$$\leq \sum_{j,m=1}^N C_{ijm}^U(t') + (t'' - t')$$

$$\leq N$$

- If $t_2 > t''$ and $t_1 > t''$, similarly we can prove that $\sum_{j,m=1}^N C_{ijm}^U$ is strictly non-increasing throughout $[t_1, t_2]$ and thus $\sum_{j,m=1}^N C_{ijm}^U(t_2) \leq \sum_{j,m=1}^N C_{ijm}^U(t_1)$.

In summary, we always have

$$\sum_{j,m=1}^N C_{ijm}^U(t_2) \leq \max\left(N, \sum_{j,m=1}^N C_{ijm}^U(t_1)\right)$$

□

C. PROOF OF THEOREM 1(B) AND 1(C)

In the following we'll prove Theorem 3, which implies Theorems 1(b) and 1(c).

THEOREM 3.

$$\limsup_{t\to\infty} \frac{G_{jm}(t)}{t} = 0 \quad j, m = 1, 2, \ldots, N$$

PROOF. We prove the theorem by contradiction. Suppose there exists a pair of integers $j > 0$ and $m > 0$ such that

$$\limsup_{t\to\infty} \frac{G_{jm}(t)}{t} = \gamma > 0 \tag{11}$$

Thus, for $\epsilon = \frac{\gamma}{4}$, there exists an integer $T' > 0$, such that

$$t > T' \Rightarrow \frac{G_{jm}(t)}{t} < (\gamma + \epsilon)$$

By Lemma 1, for $p = \frac{\gamma - \epsilon}{N}$, there exists an integer $T_A > 0$, such that

$$\left\{ \begin{array}{l} T > T_A \\ 0 \leq t_1 < t_2 \leq T \\ t_2 - t_1 \geq pT \end{array} \right\} \Rightarrow A_j(t_2) - A_j(t_1) \leq (1+\epsilon)(t_2 - t_1) \tag{12}$$

From (11), for the same ϵ, there exists an increasing sequence $\{T_2^k\}_{k=1}^\infty$ such that $\lim_{k\to\infty} T_2^k = \infty$, and for $k = 1, 2, \ldots$ we have $T_2^k > T_A$ and $\frac{G_{jm}(T_2^k)}{T_2^k} > (\gamma - \epsilon)$.

As $G_{jm}(0) = 0$ and $G_{jm}(T_2^k) > 0$, for each T_2^k, there must exist a $T_1^k < T_2^k$ such that $G_{jm}(T_1^k) = 0$ and $G_{jm}(t) > 0$ for any $t \in [T_1^k + 1, T_2^k]$.

Note that in the following proof, k (as well as k') is not an exponent in the terms T_1^k and T_2^k, but 3 in the term $5N^3C$ is. By Lemma 4, we have

$$\begin{aligned} & D_j(T_2^k) - D_j(T_1^k) \\ & \geq (T_2^k - T_1^k) + NG_{jm}(T_2^k) - 5N^3C \\ & \geq (T_2^k - T_1^k) + N(\gamma - \epsilon)T_2^k - 5N^3C \end{aligned} \tag{13}$$

Furthermore, since $G_{jm}(T_1^k) = 0$, we have

$$N \cdot (T_2^k - T_1^k) \geq I_{jm}(T_2^k) - I_{jm}(T_1^k) \geq G_{jm}(T_2^k) \geq (\gamma - \epsilon)T_2^k$$

Therefore $T_2^k - T_1^k \geq \frac{(\gamma-\epsilon)}{N} T_2^k = pT_2^k$. Since $T_2^k > T_A$, by (12), we have

$$A_j(T_2^k) - A_j(T_1^k) \leq (1+\epsilon)(T_2^k - T_1^k) \tag{14}$$

[5] Although in the worst case there can be a burst of N^2 consecutive increments to the lazy counters, using a (counter) "seed value" randomization technique introduced in [11], we can ensure that, with overwhelming probabilities, at most $O(1)$ increments to lazy pressure counters can be triggered during every N consecutive time slots.

Combining (13) and (14), we have

$$\begin{aligned}
B_j(T_1^k) &= B_j(T_2^k) + \big(D_j(T_2^k) - D_j(T_1^k)\big) - \big(A_j(T_2^k) - A_j(T_1^k)\big) \\
&\geq \Big(D_j(T_2^k) - D_j(T_1^k)\Big) - \Big(A_j(T_2^k) - A_j(T_1^k)\Big) \\
&\geq (T_2^k - T_1^k) + N(\gamma - \epsilon)T_2^k - 5N^3C - (1+\epsilon)(T_2^k - T_1^k) \\
&\geq N(\gamma - \epsilon)T_2^k - 5N^3C - \epsilon T_2^k \\
&\geq N(\gamma - 2\epsilon)T_2^k - 5N^3C
\end{aligned}$$

We consider two cases:

(i) If $\max_{k=1}^{\infty}\{T_1^k\} < \infty$, there must exist an integer $k' > 0$ such that $T_2^{k'} > \frac{N \max_{k=1}^{\infty}\{T_1^k\} + 5N^3C}{N(\gamma - 2\epsilon)}$, since $\lim_{k\to\infty} T_2^k = \infty$. This however implies $B_j(T_1^{k'}) \geq N(\gamma - 2\epsilon)T_2^{k'} - 5N^3C > N\max_{k=1}^{\infty}\{T_1^k\} \geq NT_1^{k'}$, which contradicts the fact that $B_j(T_1^{k'}) \leq A_j(T_1^{k'}) \leq NT_1^{k'}$ (due to (2)).

(ii) Otherwise, we must have $T_1^k \to \infty$ as $k \to \infty$, thus

$$\begin{aligned}
\limsup_{k\to\infty} \frac{B_j(T_1^k)}{T_1^k} &\geq \limsup_{k\to\infty} \frac{N(\gamma - 2\epsilon)T_2^k - 5N^3C}{T_1^k} \\
&\geq \limsup_{k\to\infty} \frac{N(\gamma - 2\epsilon)T_1^k - 5N^3C}{T_1^k} \\
&= N(\gamma - 2\epsilon) \\
&> 0
\end{aligned}$$

This contradicts Theorem 2.

□