Load-Balancing Multipath Switching System with Flow Slice

Lei Shi, Bin Liu, Changhua Sun, Zhengyu Yin,
Laxmi N. Bhuyan, *Fellow*, *IEEE*, and H. Jonathan Chao, *Fellow*, *IEEE*

Abstract—Multipath Switching systems (MPS) are intensely used in state-of-the-art core routers to provide terabit or even petabit switching capacity. One of the most intractable issues in designing MPS is how to load balance traffic across its multiple paths while not disturbing the intraflow packet orders. Previous packet-based solutions either suffer from delay penalties or lead to $O(N^2)$ hardware complexity, hence do not scale. Flow-based hashing algorithms also perform badly due to the heavy-tailed flow-size distribution. In this paper, we develop a novel scheme, namely, Flow Slice (FS) that cuts off each flow into flow slices at every intraflow interval larger than a slicing threshold and balances the load on a finer granularity. Based on the studies of tens of real Internet traces, we show that setting a slicing threshold of $1-4$ ms, the FS scheme achieves comparative load-balancing performance to the optimal one. It also limits the probability of out-of-order packets to a negligible level (10^{-6}) on three popular MPSes at the cost of little hardware complexity and an internal speedup up to two. These results are proven by theoretical analyses and also validated through trace-driven prototype simulations.

Index Terms—Load balancing, traffic measurement, switching theory.

1 Introduction

Multipath Switching systems (MPS) play a pivotal role in fabricating state-of-the-art high performance core routers. A well-known paradigm is the deployment of Benes multistage switches in Cisco CRS-1 [1]. Other examples include the Vitesse switch chip family [3] implementing the Parallel Packet Switch (PPS), and the Load-balanced Birkhoff-von Neumann (LBvN) switches [9], [10]. In general, MPS is built by aggregating several lower speed switches and, therefore, exhibits multiple internal data paths.

One major open issue in MPS is the load-balancing problem defined as how to distribute incoming traffic *A(t)* across its k internal switching paths $\{\Gamma_l\}(l \in [1,k])$ to meet at least three objectives simultaneously:

1. *Uniform load sharing*. Traffic dispatched to each path should be uniform. Specifically in MPS, traffic destined for each output should be spread evenly to avoid output contention, minimize average packet delay, and maximize throughput. This requirement is formalized as

$$\text{Equalize } \left\{A_j^l(t)\right\}(l \in [1,k]) \text{ for any } j, \tag{1}$$

where $A_j^l(t)$ denotes the traffic rate destined for output port j through switching path l in MPS.
2. *Intraflow packet ordering*. Packets in the same flow should depart MPS as their arrival orders. (Unless otherwise stated, flow in this paper is defined by TCP/IP 5-tuple.) This ordering is essential since out-of-order packets will degrade the performance of higher level protocols [5], [24]. For any two packets P_1 and P_2 in the same flow with arrival time $T(P_1)$, $T(P_2)$, and departure time $D(P_1)$, $D(P_2)$, the formula below should be guaranteed:

$$D(P_1) < D(P_2) \text{ if } T(P_1) < T(P_2). \tag{2}$$

3. *Low timing and hardware complexity*. The load-balancing and additional resequencing mechanisms at MPS should work fast enough to match the line rate, and should introduce limited hardware complexity. MPS is most likely to hold hundreds of external ports operating at ultrahigh speed. To provide such scalability, the timing/hardware complexity of $O(1)$ is necessary.

As a rule of thumb, packet-based solutions are advocated where traffic is dispatched packet by packet to optimally balance the load. However, packets in the same flow may be forwarded in separate paths and experience different delays, thus violating the intraflow packet ordering requirement. A straightforward solution is to use an explicit resequencer at each output to restore packet orders.

- *L. Shi is with the IBM Research - China, Building 19, Zhouguancun Software Park, 8 Dongbeiwang West Road, Haidian District, Beijing 100193, China. E-mail: shijim@gmail.com.*
- *B. Liu is with the Lab of Broadband Network Switching Technology and Communications, Department of Computer Science and Technology, Room 9-416, East main building, Tsinghua University, Beijing 100084, China. E-mail: liub@tsinghua.edu.cn.*
- *C. Sun is with the IBM Research - China, Building 19, Zhouguancun Software Park, 8 Dongbeiwang West Road, Haidian District, Beijing 100193, China. E-mail: sunchanghua@tsinghua.org.cn.*
- *Z. Yin is with the Computer Science Department, University of Southern California, 3737 Watt Way, Los Angeles, CA 90089-0781. E-mail: zhengyuy@usc.edu.*
- *L.N. Bhuyan is with the Computer Science and Engineering Department, University of California, 351, Engineering Building II, Riverside, CA 92521. E-mail: bhuyan@cs.ucr.edu.*
- *H.J. Chao is with the Polytechnic Institute of New York University, 5 Metrotech Center, Brooklyn, NY 11201. E-mail: chao@poly.edu.*

Manuscript received 24 Feb. 2010; revised 8 Aug. 2010; accepted 22 Nov. 2010; published online 15 Dec. 2010.
Recommended for acceptance by V. Leung.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2010-02-0133.
Digital Object Identifier no. 10.1109/TC.2010.279.

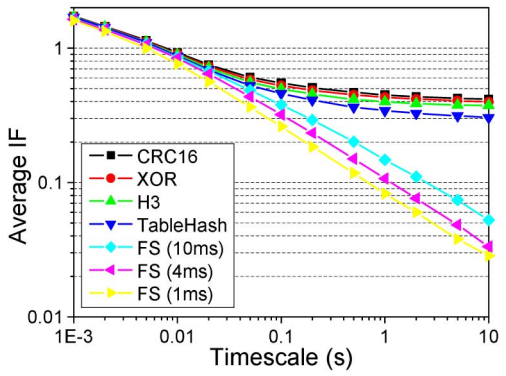

Fig. 1. Average IFs under different timescales.

In [17], [30], and [31], timestamp-based resequencers are developed. They delay each packet at output until the system delay upper bound is reached. Each packet is time shifted by the same offset before departing, thus preserving the arrival order. Nonetheless, the delay equalization method suffers from a huge penalty in magnifying the average delay. It is shown in our prototype simulations that even the latest adaptive resequencer [31] increases the average delay nearly 10 times to about 10 ms. A route passing through five such routers will lead to at least 50 ms average delay, almost violating the QoS requirement for delay-sensitive applications. Another kind of resequencers [14], [22], [23] records each packet's sequence number in the flow (defined by input, output port, and priority class), instead of absolute timestamp. By allowing only in-order packets with expected sequence number to depart, they preserve packet orders without penalizing packet delays, but at the cost of maintaining at least N resequencers (N is the number of input/output port in a square MPS) at each output, leading to $O(N^2)$ hardware complexity. In a 1,024-port/16-plane/8-priority-class three-stage Clos-network-based MPS, Virtual Input Queue (VIQ) resequencer [4] should maintain 4M resequencing FIFOs at each output, a scenario which is nearly impossible to implement.

To avoid the packet out-of-order, another choice is to use flow-based load-balancing algorithms [7], [15], [20], [26], [29]. They dispatch packets in the same flow to a fixed switching path by hashing its 5-tuple to path ID. However, hashing solutions intrinsically suffer from load-imbalance under every timescale. We give an illustration here using a trace collected in real life, but actually this behavior is universal irrespective of the input trace. If we denote the traffic volume dispatched to path l by $A^l(t)$, then we can define *load-Imbalance Factor* (IF) as the maximum deviation ratio:

$$\text{IF} = Max\{|A^l(t) - avg[A^l(t)]|/avg[A^l(t)]\} \quad l \in [1, k]. \quad (3)$$

We use the average of IFs in every nonoverlapped time period with length T to indicate overall load-imbalance degree under timescale T.

Three static hashing algorithms (CRC16 [7], XOR, and H3 [8]) and the dynamic table hashing [7] are evaluated in load balancing the same trace to eight identical paths. Results given in Fig. 1 demonstrate that their average IFs do not go asymptotically to zero as timescale increases, but stay stable at 0.2-0.3, showing their consistent load imbalances. This effect can be ascribed to the heavy-tailed flow-size distribution discussed in [6] and [16]. Adaptive hashing [20], [26] can be adopted to adjust flow mapping according to load status at each split path. But this does not work well for MPS since 1) the load balancer (LB) here is geographically separated from all the switching paths, making feedbacks outdated; and 2) there exists dual conflicting points at both the input and output of MPS and, therefore, feedback should be initiated from each input to each output, leading to $O(N^2)$ complexity.

In summary, previous solutions cannot gracefully deal with the load-balancing problem in MPS to meet the three objectives outlined above. In this paper, we develop a new scheme called *Flow Slice* (FS) that achieves our load-balancing goals perfectly. Based on the observations on tens of broadly located Internet traces, we find that the intraflow packet intervals are often, say in 40-50 percent, larger than the delay upper bound at MPS which can be calculated statistically. If we cut off each flow at every interval larger than a slicing threshold set to this bound and balance the load on the generated flow slices, all three objectives are met simultaneously:

1. Flow slices exhibit small average size, light-tailed size distribution, and large in total number; hence, the average load balancing of FS, measured by average packet delay and loss rate, is only moderately degraded from the optimal load balancing. In our simulations, FS receives nearly the same loss rate as the optimal even under a load rate of 0.95. Fig. 1 also depicts the average IF under FS. It indicates that the load-balancing uniformity improves quickly toward the optimal value as timescale increases.
2. As the slicing threshold is set to the statistical delay upper bound at MPS, the intraflow packet order is kept intact as they arrive. Exceptions only occur in negligible probability (below 10^{-6}) [5]. Hence, there is no need to use costly resequencing mechanisms. Throughout the paper, unless otherwise noted, the statistical delay (upper) bound is defined as a minimal value t that more than 99.9999 percent packet delays through MPS are smaller than t.
3. The active flow-slice number drops 1-2 magnitudes from the active flow number to at most 50 k even under full load at OC-768c port. As a result, the flow-slice table size in FS only requires 1.8 MB in total, which can be placed on-chip to provide ultrafast access speed. Here, the active flow/flow slice is defined as the one that has its last packet arrive within a time-out threshold. According to the flow/flow-slice table maintenance mechanism, only the active flows/flow slices are valid and taken into account for the table space.

At the time of our study, we noticed similar techniques developed for traffic splitting in multipath routing [28] and load balancing in the multicore Network Processor (NP) [25], [27]. Our major improvement over the existing works is to tailor the FS approach in the MPS scenario by introducing the offline delay bound calculation, while the previous solutions either use an empirical slicing threshold, e.g., 60 ms in [28], or maintain flow context to facilitate the slicing [25], [27]. The empirical slicing threshold will lead to

poor load-balancing performance for MPS, as shown in our simulations. Maintaining flow context is impractical in the MPS scenario as it requires a flow table for each input updated by all the outputs in real time (see Section 2 for details).

Other contributions besides the FS idea include:

1. The concept of smallest slicing threshold and the method to compute it. By configuring FS at this value, we find the best trade-off between the load-balancing and packet ordering performance.
2. A systematic study of the properties of flow slice. Based on these properties, the intrinsic reason for FS solution to achieve high performance is clarified.
3. An evaluation of FS performance, which to the best of our knowledge, is the first attempt to use multiport trace-driven switch simulation. The traces here are collected at backbone links of one of the largest commercial backbones worldwide, with average traffic rate beyond 3.5 Gbps. Previous studies [7], [25], [27], [28] rarely use traces with average speed beyond 1 Gbps.

The rest of this paper is organized as follows: Section 2 summarizes related works; Section 3 defines flow slice, studies its properties, and proposes our load-balancing scheme; Sections 4 and 5 calculate packet delay upper bound at three popular MPSes and further derive the smallest admissible slicing thresholds; Section 6 evaluates FS performance using trace-driven prototype simulations; Section 7 discusses implementation issues; and finally, we conclude the paper in Section 8.

2 Related Work

A basic timestamp-based resequencer was proposed by Turner [30] to deal with the cell out-of-order issue, when cells within a virtual circuit are allowed to take different paths in ATM switching systems. In each scheduling, the resequencer implements a smart contention resolution mechanism to select the oldest cell and then compares its age with a predefined threshold equal to the system delay upper bound. If the selected cell exceeds this age threshold, it will be sent without disturbing cell orders since all previously arrived cells have left the system. Henrion improved the timestamp-based resequencer by introducing time-wheel-like hardware [17] which does not need to compare cells' timestamps at output. It maintains an array of D pointer lists, where D is larger than delay upper bound at system measured by timeslots. Each pointer at the list stores location of a cell waiting for resequencing. At timeslot t, the pointer list at slot t modulo D is linked to the tail of the output cell list and removed from the array. After that, pointers of arrival cells at timeslot t are linked together and stored in this empty slot (t modulo D) of the array. This approach delays every cell by fixed D timeslots with $O(1)$ complexity and strictly guarantees cell orders.

Fixed-threshold timestamp-based resequencers work well for ATM switching systems where traffic is well defined and cell delay is moderate. However, in transition to an IP network, traffic becomes rather bursty, introducing a higher age threshold on these resequencers and equalizing cell delay to undesirable level. To cope with the traffic in the worst case, Turner proposed an adaptive resequencer [31] that dynamically adjusts threshold according to a real-time delay bound. Trace-driven simulations show that the resequencer performs well better than the fixed-threshold one.

Another kind of resequencer uses sequence number instead of timestamp. Cells with the same priority from one input to one output are numbered sequentially at their arrivals. At the output, an expected sequence number is maintained for all the packets in the same flow. Only the cells with an expected sequence can be sent after reaching output, thus preserving cell order. Predefined path scheduling protocol known by both input and output, such as Round-Robin (RR), can be employed to avoid attaching a sequence number to each cell. Resequencers using VIQ [4], [19], [21] can be categorized in this class. However, to recover from cell loss, more complicated mechanisms should be introduced, such as resequencers developed by Chiussi et al. [14] in designing Switched Connection Inverse Multiplexing for ATM (SCIMA) and the solution named *Rank* [22], which maintains *k*-1 fields in each cell header to deal with cell loss. However, *Rank* is not scalable for MPS having more than a hundred internal paths.

Resequencing techniques are also studied in designing popular MPS. In [18], Iyer et al. have proven that using a centralized scheduling algorithm, PPS rigorously emulates an Output-Queued (OQ) switch if only a speedup of two is provided. This approach does not disorder cells but suffers from huge communication complexity in sharing information among all the inputs and outputs. Another work [19] by Iyer and McKeown introduced distributed scheduling that does not exactly emulate OQ, hence also requires resequencing mechanisms at the output.

In Chang et al.'s work [9], [12], a jitter control mechanism is inserted before the second stage of LBvN to equalize delays experienced in its first stage and preserve cell order. This approach can be categorized into timestamp-based solution. Later in [11], by setting the configurations of LBvN's first and second stages to symmetrical patterns, each input and output is connected to each Virtual Output Queue (VOQ) between the two stages (called *mailbox*) in period. Then, the buffer lengths of all the VOQs, indicating cell delay at LBvN, can be known precisely. Each Head-Of-Line (HOL) cell is scheduled to the proper mailbox where it will not depart earlier than any previous cell in the same flow, thus keeps cell order intact. However, this method degrades throughput to 75 percent. Another solution by Keslassy and Mckeown [21] uses three-dimensional queues (3DQs) between LBvN's two stages. It arranges buffer by external input/output port and also by internal input port, with totally N^3 FIFOs. Full frames, defined by cells belonging to same input/output port and going to internal input 1 through N, are scheduled first (FFF) at LBvN's output; hence, it guarantees cell orders in most cases. FFF can be deemed as a generalization of the VIQ resequencer.

Other solutions on this issue are flow-based algorithms dispatching traffic of the same flow into a same path, which natively preserves packet order. In [7], Cao et al. evaluated the performance of static hashing. It concludes that 16 bit CRC achieves excellent load-balancing uniformity, but this

TABLE 1
Real Traces Studied in This Paper

Trace Description (Collected Point)		Trace Name	Collect Time	Link Speed / Load Rate	Trace Duration	Packet Count	TCP Flow Count	Packet Length Avg. / SCV[1]
One hour continuous packet headers collected at North China CERNET backbone, 11/10/2005. (One of largest commercial backbones in China.)		Cernet-10	Nov. 10, 2005	OC-192c / 35.0%	63.1s	4.32E7	1.77E6	639.6Byte / 1.02
		Cernet-30	Nov. 10, 2005	OC-192c / 34.5%	63.1s	4.32E7	1.77E6	629.1Byte / 1.05
		Cernet-40	Nov. 10, 2005	OC-192c / 35.4%	62.4s	4.32E7	1.79E6	640.3Byte / 1.02
Continuous packet headers collected at Tsinghua University egress link to CERNET at June to July, 2006. Trace segments are as long as 10 minutes.		Ts-1	June 23, 2006	GE / 27.1%	347.4s	2.65E7	1.58E6	443.9Byte / 1.62
		Ts-3	July 12, 2006	GE / 25.5%	469.4s	2.73E7	1.46E6	547.3Byte / 1.26
		Ts-5	July 12, 2006	GE / 24.7%	511.0s	2.73E7	1.25E6	577.6Byte / 1.20
Continuous packet headers at network of American univ. and research labs by NLANR [2] during 2004~2006.	Pittsburgh SC Center	PSC	Mar. 8, 2006	OC-48c / 10.5%	89.2s	3.50E6	6.38E4	835.6Byte / 0.68
	Front Range GigaPOP	FRG	Dec. 5, 2004	OC-12c / 56.9%	89.6s	5.55E6	1.93E5	717.4Byte / 0.84
	Merit Abilene	MRA	Mar. 9, 2004	OC-12c / 56.2%	90.1s	6.12E6	4.48E5	645.9Byte / 1.03
	Univ. Florida at Gain	UFL	Dec. 5, 2004	OC-12c / 53.7%	89.8s	5.68E6	2.72E5	663.4Byte / 1.03
	Columbia Univ.	BWY	Mar. 2, 2004	2×OC-3c / 53.5%	90.1s	2.60E6	5.50E4	723.8Byte / 0.79
	Old Dominion Univ.	ODU	Dec. 4, 2004	OC-3c / 52.3%	89.4s	1.33E6	6.80E4	684.5Byte / 0.89
	Colorado State Univ.	COS	Dec. 5, 2004	OC-3c / 50.3%	90.2s	2.04E6	1.79E5	435.0Byte / 1.62

[1] Squared Coefficient of Variation.

is only derived under large timescale (0.1 s) and two load-balancing paths. An adaptive table-based hashing was proposed by Dittmann and Herkersdorf [15] to deal with the load-balancing issue in clustered network processors. Each newly arrived flow is assigned to the NP with the lightest load. The major bottleneck of this method is to maintain a per-flow context table, which reaches up to a million entries at an OC-768c port.

A name-based hashing solution called Highest Random Weight (HRW) was proposed by Thaler and Ravishankar [29], which uses names of an object and each server to generate weights through static hashing. The object is handled by the server with the highest weight. The unique features of HRW lie in its graceful dealing with server failures, and it supports load balancing across heterogeneous servers by introducing the scaling vector. In [20], Kencl and Boudec generalized HRW to the multicore NP. They developed methods to detect NP's overload and optimally adjust a scaling vector to reshape flow distribution. However, flow remapping has negative impact on increasing packet out-of-order probability. To alleviate that problem, Shi and MacGregor proposed the idea of shifting only aggressive flows [26].

In [28], Sinha et al. found that TCP's burstiness can be utilized to improve load balancing. Based on this observation, flowlet switching [28] and adaptive burst shifting [25], [27] (by Shi et al.) were developed for traffic distributing at multipath routing and multicore NP designs. However, these approaches do not fit well into MPS, since 1) the slicing threshold at 60 ms [28] does not improve MPS performance much, and 2) the method in [25] and [27] records flow context of all the packets in the centralized or distributed flow table at each input. The method will remap the flows with all previous packets already departed to optimize the load-balancing performance. Also, this approach cannot be extended to MPS, as all the inputs/outputs of MPS are separated, making the centralized updates to each input difficult and costly to deploy.

3 Flow Slice

In this section, we study flow-slice properties using real traces listed in Table 1 [2]. The traces are collected at separate locations from the network core to edge/access points; hence, the traces represent comprehensive Internet traffic behaviors. Each data set records consecutive packet headers (TCP/IP) during an observation period of 1-10 min. The original 5-tuple at each header is remapped, using the longest prefix-preserving mapping, to new 5-tuple address spaces, but they still retain the original flow-level characteristics. The timestamp at each packet header has a resolution of at least 1 microsecond.

3.1 Definition

3.1.1 Flow Slice

A flow slice is a sequence of packets in a flow, where every intraflow interval between two consecutive packets is smaller than or equal to a slicing threshold Φ.

Flow slices can be seen as miniflows created by cutting off every intraflow interval larger than Φ. In Fig. 2, we depict the Cumulative Distribution Functions (C.D.F) of intraflow intervals in our traces. Most of the traces have more than 50 percent of their intervals larger than 1 ms, and more than 40 percent are larger than 4 ms. Two exceptions

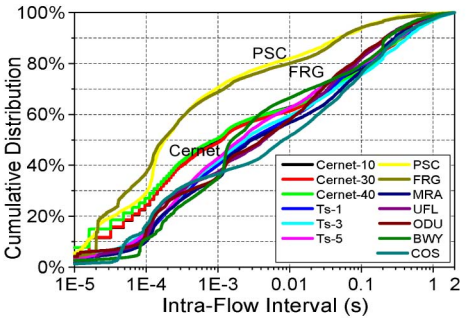

Fig. 2. Intraflow interval C.D.F.

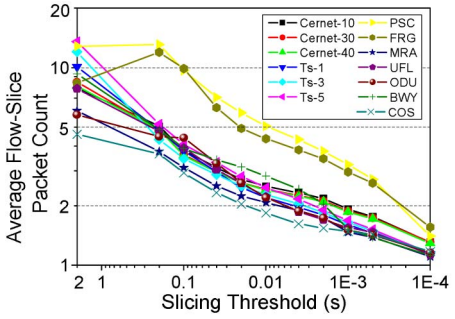

Fig. 3. Flow-slice packet count.

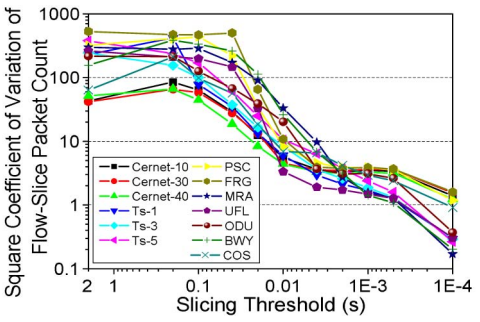

Fig. 5. SCV of flow-slice packet count.

are PSC and FRG. The former is under a light load ($\approx$10%); the latter is collected from a low-speed OC-12c network edge link. Thus, their weights are minimized. We denote the probability for an intraflow interval to be larger than Φ by P_C and the average packet count in the original flow by C_0. After slicing, the average packet count in flow slice, denoted by $\overline{F_C}$, is calculated by

$$\overline{F_C} = C_0/[1 + (C_0 - 1)P_C] < 1/P_C. \quad (4)$$

Setting $\Phi = 1$ ms, which leads to $P_C > 0.5$, we obtain $\overline{F_C} < 2$. That is, each flow slice has no more than two packets in average. Even under a modest setting of $\Phi = 4$ ms, we still have $\overline{F_C} < 2.5$. This reveals the very close load-balancing granularity of FS to the optimal packet-based solution.

3.2 Properties

The flow-slice characteristics are investigated at different slicing thresholds between 100 μs-200 ms. Compared with the original flow, which is equivalent to flow slice with a slicing threshold of 2 s (set to flow time-out value), three flow-slice specific properties are observed in all traces.

Property 1 (Small Size). *Both the average packet count ($\overline{F_C}$) and the average size ($\overline{F_S}$) of flow slice are much smaller than those of the original flow.*

Figs. 3 and 4 illustrate the average flow-slice packet count and flow-slice size, which drop linearly in log-log plot. (PSC/FRG are exceptions. They crest at $\Phi = 200$ ms since only flow slices that expire during trace collection time are counted, so the flow-slice size and packet count at $\Phi = 200$ ms or above are smaller than the actual values.)

At $\Phi \leq 4$ ms, all traces except PSC/FRG meet $\overline{F_C} \leq 2.43$; at $\Phi \leq 1$ ms, they meet $\overline{F_C} \leq 1.92$. These results fit very well to the analysis in Section 3.1. In contrast, the average metrics of original flow shown as intersections at the y-axis of Figs. 3 and 4 are much larger. Using the average flow-slice size in Fig. 4 to indicate load-balancing granularity, a flow-based algorithm is 3.5-12 times coarser than a packet-based one (compared with the average packet length given in Table 1), while a flow-slice-based algorithm is only 0.41-0.97 times coarser at Φ= 1 ms, and 1.95-2.42 times coarser at $\Phi = 4$ ms.

This property implies that the number of flow slices is much larger than the original flows.

Property 2 (Light-Tailed Size Distribution). *Flow-slice packet count/size distributions are light tailed while it is well-known that original flow-size distribution is heavy tailed.*

Figs. 5 and 6 show the SCV of flow-slice packet count (ρ^2_{FC}) and flow-slice size (ρ^2_{FS}). As proven in the later theorems, the delay bounds at MPS are directly subject to these metrics. Take ρ^2_{FS} as an example, at $\Phi \leq 4$ ms, it is below 48; at $\Phi \leq 1$ ms, it further drops to below 12. In the same figure, we observe that the SCV of the original flow size is much larger; some even beyond 1,000. This reveals the light-tailed property of flow-slice size distribution.

Property 3 (Fewer Active Flow Slices). *The active flow-slice number is* 1-2 *magnitudes smaller than that of active flow.*

Each flow slice remains active for an average length of $D_{FS} + \Phi$, where D_{FS} denotes its duration, and Φ denotes slicing threshold. So, the active flow-slice number, denoted by A_F, can be calculated by (5), where N_F denotes the total number of flow slices in the trace and D stands for the trace duration.

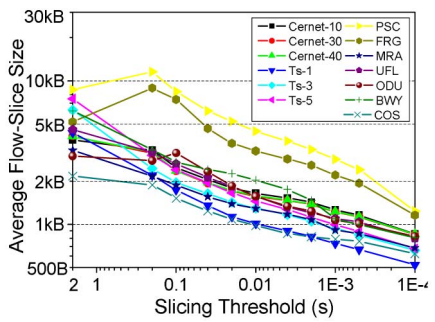

Fig. 4. Flow-slice size.

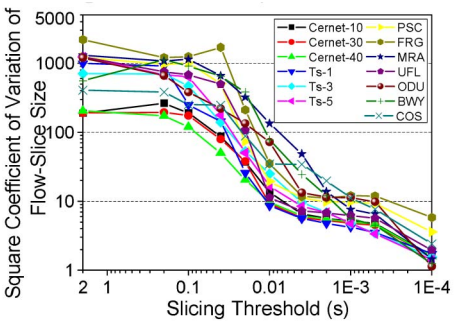

Fig. 6. SCV of flow-slice size.

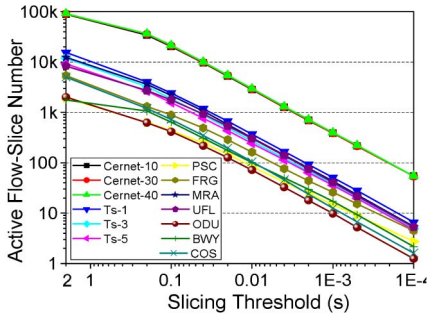

Fig. 7. Active flow-slice number.

$$A_F = (D_{FS} + \Phi)N_F/D. \tag{5}$$

Fig. 7 depicts the active flow-slice number. As we reduce Φ, it decreases linearly in log-log plot. At $\Phi = 4$ ms, it drops to less than 2.8 percent of the active flow number; further at $\Phi = 1$ ms, it is less than 1 percent of the active flow.

We also measure packet length distributions in all listed traces. Its average ($\overline{L}$) and SCV (ρ_L^2) are given in Table 1. In later analysis, we will assume packet length to be i.i.d and independent of individual flow slices.

3.3 Load-Balancing Scheme for MPS

Fig. 8 illustrates the proposed FS scheme and compares it to the packet-based and flow-based load-balancing schemes. We describe the FS scheme in two steps: 1) an introduction of the mechanism to detect and maintain flow-slice context, and 2) an algorithm to map new flow slices into individual switching paths.

First, we implement a hash table to maintain flow-slice information. Every table entry is expected to record the context of one active flow slice and is composed of two fields apart from the table index: *Latest Arrival Timestamp* (T_L), which holds the exact time of the latest packet arrival of this flow slice; and *Flow-slice Destination Path* (P_D), indicating the switching path taken by all the packets in this flow slice.

At any packet P_a's arrival, its corresponding entry Γ is retrieved from the table by hashing P_a's 5-tuples into the table index. Denote T_L in entry Γ by $T_L[\Gamma]$ and P_a's arrival time by T_A. In case T_A-$T_L[\Gamma] \leq \Phi$, P_a should be a *successive packet* of an active flow slice; otherwise, P_a should be an *initial packet* of a new flow slice. In the former case, P_a is dispatched to switching path $P_D[\Gamma]$ as its preceding packets in the same flow slice and $T_L[\Gamma]$ is updated to T_A; in the latter case, P_a is dispatched by the new flow-slice load-balancing algorithm: then both $T_L[\Gamma]$ and $P_D[\Gamma]$ are updated to the latest values: $T_L[\Gamma]$ to T_A and $P_D[\Gamma]$ to the generated path ID. With our scheme, each hash table entry will go dirty silently upon time-out (Φ) and be updated until the next packet hashed into this entry arrives, so there is no need for an extra space free-up mechanism. Note that the trade-off between table size and hash collision probability is discussed in Section 7.1.

Second, to load balance new flow slices, a simple round-robin algorithm called N-FS is applied, where N is the number of output ports in MPS. Denote the switching paths in MPS by 1-k; N-FS maintains a pointer $Pt_j \in [1, k]$ for output j, serving packets destined for output j. At the arrival of an initial packet P_I representing new flow slice destined for output j, P_I is dispatched to the switching path pointed by Pt_j, and after that, Pt_j is updated circularly.

Unless otherwise stated, we use N-FS to name our load-balancing scheme in the rest of this paper. By N-FS, the three objectives of the load-balancing issue are achieved simultaneously:

1. Since N-FS balances the load on the finer-grained flow slice, the switching performance—measured by packet delay and loss rate—is greatly improved from flow-based approaches.
2. Intraflow packet order is natively preserved by setting slicing threshold to the delay upper bound at MPS. This is because 1) any two packets in the same flow slice cannot be disordered as they are dispatched to the same switching path where FIFO processing is guaranteed; and 2) any two packets in the same flow but different flow slices will be in-order at departure, as the earlier packet will have depart from MPS before the latter packet arrives.
3. Due to the fewer number of active flow slices, the only additional overhead in N-FS, the hash table, can be kept rather small, say 1.6 MB in total even under the OC-768c line rate, and placed on-chip to provide ultrafast access speed. This table size depends only on system line rate and will stay unchanged even if MPS scales to more than a thousand external ports, thus guarantees system scalability. Meanwhile, the time complexity of N-FS in dispatching packet is shown to be $O(1)$.

Other advantages of N-FS are that

1. It is immune to packet loss, while other solutions like the VIQ resequencer require additional loss detection mechanisms.
2. It maintains a hash table to record active flow-slice context, a redirection mechanism can be added to provide robustness to system failure. When some switching path stops working, the load balancer can simply redirect all the active flow slices going to this path at their next packet arrivals, still by N-FS.

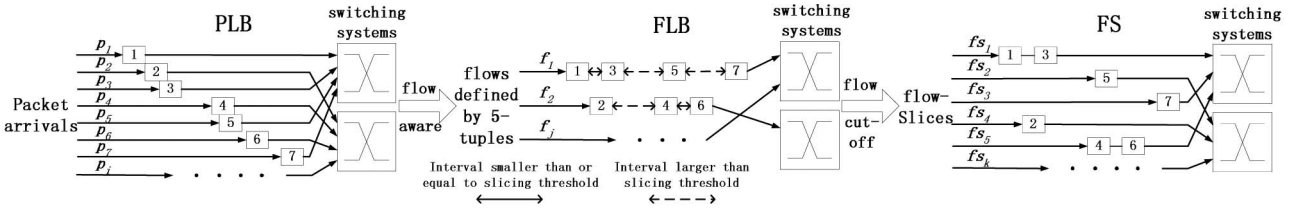

Fig. 8. FS Scheme and comparisons to Packet-based (PLB) and Flow-based (FLB) load-balancing schemes.

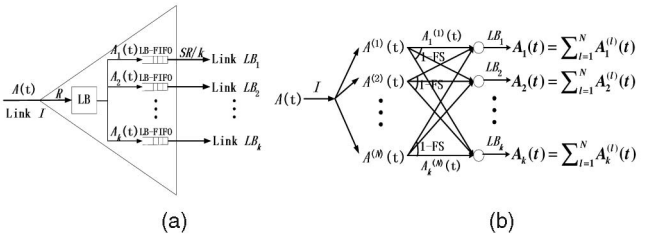

Fig. 9. (a) A basic demultiplexer. (b) N-FS mechanism.

3. It natively supports multicast. Multicast flows are treated in the same manner as unicast flows and still preserve packet orders. To load balance multicast flow, it is required to use $(N+M)$-FS, where M is the number of multicast destination port group.
4. N-FS supports load balancing across uneven switching paths by applying weighted round robin.

Now that the most critical remaining issue of N-FS is to calculate delay upper bound at MPS and determine their admissible slicing thresholds. This is settled in Sections 4 and 5 beginning with analyses of a basic multiplexer (mux) and three popular MPSes.

4 Performance Analysis on Basic Demultiplexers (Demux)

4.1 Notations

The structure of a basic demultiplexer is depicted in Fig. 9a. It is composed of a load balancer carrying out N-FS and k LB-FIFOs compensating for rate difference between external line rate R and internal line rate SR/k. Here, internal line rate is the rate each smaller switch at a switching path operates, S denotes its speedup.

We take a worst case assumption that all the arrival traffic is TCP packet, since the part of non-TCP traffic requiring in-order delivery can be modeled similarly with TCP by replacing the TCP/IP 5-tuple with the specific flow identifier; while the other best-effort non-TCP traffic such as UDP can be dispatched by the optimal load balancing without preserving packet order.

Denote packet arrivals at input link I during period $T = [t_0, t_0 + t]$ by $\{P_1, P_2, \ldots, P_{C(t)}\}$ and their packet lengths by $\{L_1, L_2, \ldots, L_{C(t)}\}$, a total of $C(t)$ packets. The aggregated traffic during T is summed by $A(t) = \sum_{i=1}^{C(t)} L_i$. Also denote the traffic dispatched to link LB_j by $A_j(t)$. Other definitions of flow-slice statistics follow those in Section 3: $\overline{F_C}$ and ρ_{FC}^2 for average packet count and its SCV; $\overline{F_S}$ and ρ_{FS}^2 for average size and its SCV; $\overline{L}$ for average packet length, σ_L and ρ_L^2 for its standard deviation and SCV.

Flow slices that have at least one packet arrival during T could be partitioned into two classes: new flow slice and active flow slice. New flow slice is the one with its first packet arrived during T; all the others are the active flow slices. The first packet of a new flow slice is called the initial packet; all the others are called successive packets. We denote β to be the ratio of initial packets in all arrival packets, having $\beta = 1/\overline{F_C}$. Given the i.i.d distribution assumption of the packet length in Section 4.2, β is also the traffic ratio of the initial packets.

As illustrated in Fig. 9b, N-FS can be deemed as aggregating $N1-$FS. The traffic destined for output l, denoted as $A^{(l)}(t)$, is load balanced independently by 1-FS. We start from an analysis of 1-FS's performance.

4.2 Analysis on Backlog and Delay Bound

Lemma 1 (1-FS Load-Balancing Bound). *By 1-FS, traffic dispatched to link LB_j during T is statistically bounded:*

$$A_j(t) \le [A(t)/k] + \overline{F_S} + 5\sqrt{A(t)/k}\sqrt{(\rho_L^2\overline{L}/\overline{F_C}) + (\rho_{FS}^2+1)\overline{F_S}}.$$

Proof. See Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2010.279, and can also be found on http://s-router.cs.tsinghua.edu.cn/~shilei/Appendix-FS.pdf. The proof is based on the assumption that the numbers of active and new flow slices in each link are large enough to hold the central limit theorem. This is also examined in Appendix A, which can be found on the Computer Society Digital Library. □

Lemma 2 (N-FS Load-Balancing Bound). *By N-FS, traffic dispatched to link LB_j during T is statistically bounded:*

$$A_j(t) \le [A(t)/k] + 5\sqrt{N/6}\times\overline{F_S} + 5\sqrt{A(t)/k}\sqrt{(\rho_L^2\overline{L}/\overline{F_C}) + (\rho_{FS}^2+1)\overline{F_S}}.$$

Proof. See Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2010.279, and can also be found on http://s-router.cs.tsinghua.edu.cn/~shilei/Appendix-FS.pdf. The assumption is the same to the proof of Appendix A. □

Theorem 1 (N-FS Delay Bound at Basic Demultiplexer). *When arrival traffic at input link L is constrained by arrival curve $A(t) \le Rt + b_I$, by N-FS, packet delay at demultiplexer is statistically bounded by*

$$d_1 = \frac{5k\sqrt{N/6}\times\overline{F_S}}{SR} + \frac{b_I}{R} + \frac{25k}{4S(S-1)R}\left[(\rho_L^2\overline{L}/\overline{F_C}) + (\rho_{FS}^2+1)\overline{F_S}\right].$$

Proof. In our analysis, we focus on traffic dispatched to link LB_j and study the delay bound at LB-FIFO j. Denote the length of LB-FIFO j at time t by $FL_j(t)$. We define $T_B = [t_0, t_1]$ to be a *backlog period* at LB-FIFO j if and only if $FL_j(t) > 0$ during $t \in (t_0, t_1)$ and $FL_j(t) = 0$ at $t = t_0, t_1$. Obviously, during any backlog period T_B, LB-FIFO j's length is calculated by $FL_j(t) = [A_j(t) - (t - t_0)\times RS/k]^+$, $[X]^+ = \max(0, X)$. According to Lemma 2 and $A(t) \le Rt + b_I$, we derive the statistical upper bound of $FL_j(t)$ by

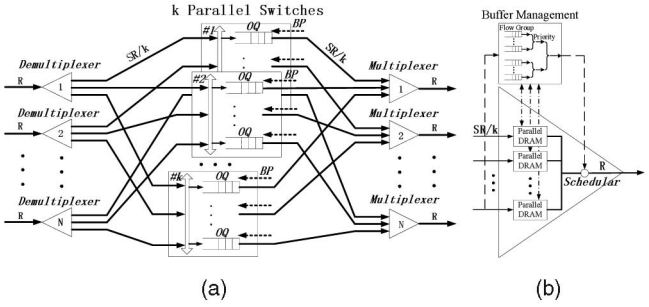

Fig. 10. (a) PPS model. (b) Output multiplexer in PPS.

$$
\begin{aligned}
&FL_j(t)\\
&\le \left[\begin{array}{l}[R(t-t_0)+b_I]/k-(t-t_0)\times SR/k+5\sqrt{N/6}\times\overline{F_S}+\\ 5\sqrt{[R(t-t_0)+b_I]/k}\sqrt{\left(\rho_L^2\overline{L}/\overline{F_C}\right)+\left(\rho_{FS}^2+1\right)\overline{F_S}}\end{array}\right]^+\\
&\le 5\sqrt{N/6}\times\overline{F_S}+Sb_I/k\\
&\quad+\left[\begin{array}{l}5\sqrt{\left(\rho_L^2\overline{L}/\overline{F_C}\right)+\left(\rho_{FS}^2+1\right)\overline{F_S}}\sqrt{[R(t-t_0)+b_I]/k}\\ -(S-1)\times[R(t-t_0)+b_I]/k\end{array}\right]^+.
\end{aligned}
$$

1. When

$$
\begin{aligned}
b_I/k &\le \frac{25}{4(S-1)^2}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right)+\left(\rho_{FS}^2+1\right)\overline{F_S}\right], \quad \text{at}\\
[R(t-t_0)+b_I]/k &= \frac{25}{4(S-1)^2}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right)\right.\\
&\quad\left.+\left(\rho_{FS}^2+1\right)\overline{F_S}\right], FL_j(t)
\end{aligned}
$$

reaches its statistical upper bound.

$$
\begin{aligned}
FL_j(t) &\le 5\sqrt{N/6}\times\overline{F_S}+Sb_I/k\\
&\quad+\frac{25}{4(S-1)}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right)+\left(\rho_{FS}^2+1\right)\overline{F_S}\right].
\end{aligned}
\tag{6}
$$

2. When

$$
\begin{aligned}
b_I/k &\in \left[\frac{25}{4(S-1)^2},\frac{25}{(S-1)^2}\right]\times\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right)\right.\\
&\quad\left.+\left(\rho_{FS}^2+1\right)\overline{F_S}\right]
\end{aligned}
$$

at $[R(t-t_0)+b_I]/k=b_I/k, FL_j(t)$ satisfies the bound in (6).

3. Otherwise, $FL_j(t)$ satisfies

$$
FL_j(t)\le 5\sqrt{N/6}\times\overline{F_S}+Sb_I/k. \tag{7}
$$

Delay upper bound at demultiplexer is calculated by $d_1=Max[FL_j(t)]/(SR/k)$. □

5 Extension to Multipath Switching System

5.1 Parallel Packet Switches

We consider the PPS model depicted in Fig. 10a, where the input demultiplexer is the same as in Fig. 9a and the output multiplexer is illustrated in Fig. 10b. Here, parallel DRAMs

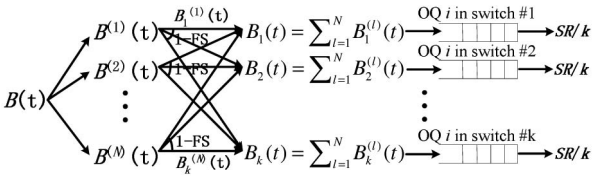

Fig. 11. Derivation of Lemma 3.

operating lower than line rate are used. Through lay-aside Buffer Management (BM) module, all packets are virtually queued at the output according to the flow group ID and the priority class in a hierarchical manner. The output scheduler fetches packets to the output line using information provided by BM. Packets in the same flow will be virtually buffered in the same queue and scheduled in FIFO discipline. Hence, intraflow packet departure orders hold as their arriving orders at the multiplexer. Central-stage parallel switches adopt an output-queued model.

By Theorem 1, we derive packet delay bound at first-stage LB-FIFO. We then study delay at second-stage switches. Define *native packet delay* at stage m of an MPS to be delay experienced at stage m on the condition that all the preceding $m-1$ stages immediately send all arrival packets out without delay.

Lemma 3 (N-FS Native Delay Bound at Second Stage of PPS). *Assume PPS with N outputs, k parallel switches and admissible traffic (i.e., traffic at each input is constrained by $A(t)\le Rt+b_I$, and traffic destined for each output is constrained by $B(t)\le Rt+b_O$). By N-FS, the native packet delay at second stage of PPS is statistically bounded by*

$$
\begin{aligned}
\widetilde{d_2} &= \frac{5k\sqrt{N/6}\times\overline{F_S}}{SR}+\frac{b_O}{R}\\
&\quad+\frac{25k}{4S(S-1)R}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right)+\left(\rho_{FS}^2+1\right)\overline{F_S}\right].
\end{aligned}
$$

Proof. Focus on the backlogs at OQ i of second-stage switches. Denote arrival traffic destined for output i during $T=[t_0,t_0+t]$ by *B(t)*. As in Fig. 11, *B(t)* is composed by arrival traffic from all N input ports. Denote the part from input l by $B^{(l)}(t)$. At first-stage demultiplexer, $B^{(l)}(t)$ is load balanced to all k second-stage switches by 1-FS. Denote the part of $B^{(l)}(t)$ going to switch #j by $B_j^{(l)}(t)$. Then, traffic of *B(t)* dispatched to switch #j, which is aggregated traffic at OQ i of switch #j, is computed by $B_j(t)=\sum_{l=1}^{N}B_j^{(l)}(t)$. Substituting $B(t)$, $B_j(t)$, $B_j^{(l)}(t)$, b_O for $A(t)$, $A_j(t)$, $A_j^{(l)}(t)$, b_I in the derivation of Theorem 1, the native delay bound at second-stage switch is calculated by the same technique. □

Lemma 4 (N-FS Delay Bound at Second Stage of PPS). *Under the same conditions as those of Lemma 3, the packet delay at the second stage of PPS is statistically bounded by*

$$
\begin{aligned}
d_2 &= \frac{10k\sqrt{N/6}\times\overline{F_S}}{SR}+\frac{b_I+b_O}{R}\\
&\quad+\frac{25k}{2S(S-1)R}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right)+\left(\rho_{FS}^2+1\right)\overline{F_S}\right].
\end{aligned}
$$

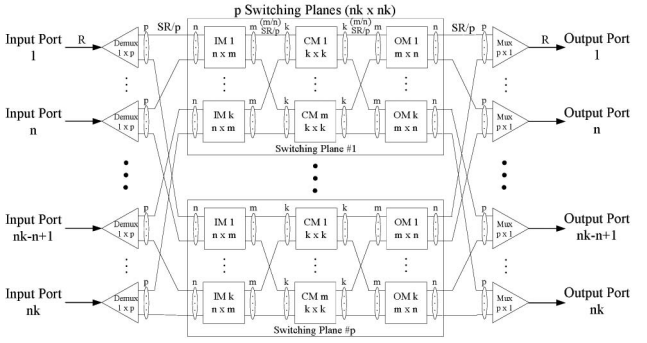

Fig. 12. Three-stage $\mathrm{M}^2\mathrm{Clos}$.

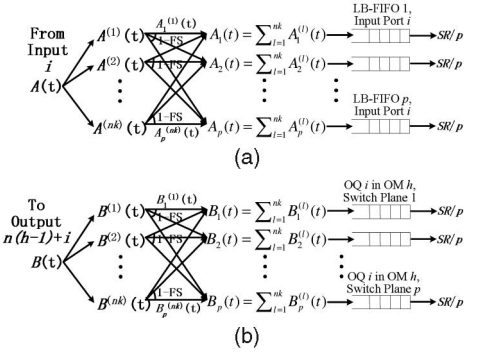

Fig. 13. $\mathrm{M}^2\mathrm{Clos}$ delay derivation: (a) Stage 1. (b) Stage 4.

Proof. Since packet delay at the first-stage demultiplexer is bounded by d_1 (Theorem 1), traffic arrivals at second-stage switches destined for output i are actually constrained by $B(t) \le R(t + d_1) + b_O = Rt + (b_O + Rd_1)$. Substituting $b_O + Rd_1$ for b_O in Lemma 3, we prove this lemma. □

By Theorem 1 and Lemmas 3 and 4, we have $d_2 = d_1 + \widetilde{d}_2$. It can be generalized as the additivity of a single-stage delay bound: the packet delay at stage m of MPS is bounded by the sum of the native packet delay bound at this stage and the delay bounds of all the $m-1$ preceding stages.

Theorem 2 (N-FS Delay Bound at PPS). *Packet delay at the multipath part of PPS is statistically bounded by*

$$d_{PPS} = \frac{15k\sqrt{N/6} \times \overline{F_S}}{SR} + \frac{2b_I + b_O}{R} + \frac{75k}{4S(S-1)R}\left[\left(\rho_L^2 \overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2 + 1\right)\overline{F_S}\right].$$

Proof. In our PPS model, the third-stage multiplexer does not introduce additional packet out-of-order. Hence, the multipath part of PPS only includes demultiplexers and second-stage switches. By Theorem 1 and Lemma 4, we have derived their statistical delay bounds. Then, overall delay bound holds as the theorem. □

5.2 Multistage Multiplane Clos Switches

We consider the Multistage Multiplane Clos-network-based switch by Chao et al. [13] depicted in Fig. 12 (called $\mathrm{M}^2\mathrm{Clos}$ in brief). It is constructed of five stages of switch modules with top-level architecture similar to a PPS with *nk* external input/output ports. The first and last stages of $\mathrm{M}^2\mathrm{Clos}$ are composed of *nk* input demultiplexers and output multiplexers, respectively, having similar internal structures as those in PPS. Stages 2-4 of $\mathrm{M}^2\mathrm{Clos}$ are constructed by p $nk \times nk$ parallel switching planes; however, each plane is no longer formed by a basic OQ switch, but by a three-stage Clos Network to support large port count. Inside each Clos Network, the first stage (stage 2 of $\mathrm{M}^2\mathrm{Clos}$) is composed by k identical Input Modules (IM). Each IM is an $n \times m$ packet switch, with each output link connected to a Central Module (CM). Thus, there are a total of m identical CMs in second stage of the Clos networks (stage 3 of $\mathrm{M}^2\mathrm{Clos}$). Each CM is a $k \times k$ packet switch. Symmetrically to IMs at the output side, each CM connects to k identical Output Modules (OM). Each OM is an $m \times n$ packet switch with n outputs working as outputs of its central switching plane; k OMs contribute *nk* outputs to each switching plane.

As with the modeling of PPS, we design each IM, CM, and OM as OQ switch. Let the input/output port of $\mathrm{M}^2\mathrm{Clos}$ run at line rate R and the central switching planes work at speedup S, and let both stage 1s demux at the transmitting side and stage 5s mux at the receiving side operate at rate *SR/p*. IM has input and output line rate of *SR/p* and *(m/n)SR/p*; CM has input and output line rate of both *(m/n)SR/p*; and OM has input and output line rate of *(m/n)SR/p* and *SR/p*.

Each input packet at $\mathrm{M}^2\mathrm{Clos}$ selects a path through which it transfers the switching system, including switching plane ID (1-p) and CM path ID (1-m). In this paper, we adopt the *nk*-FS scheme to load balance traffic. Note that the selections of switching plane and CM path are computed independently by *nk*-FS scheme.

As in PPS, in $\mathrm{M}^2\mathrm{Clos}$, the native delay bounds at stages 1 and 4 of $\mathrm{M}^2\mathrm{Clos}$ are calculated from delay bounds at stages 1 and 2 of PPS, respectively. The explicit illustrations are given in Figs. 13a and 13b. The only difference from analysis in PPS is the substitution of port number *nk* for N and switching plane number p for k. Therefore, by Theorem 1 and Lemma 3, we obtain native delay bound at stages 1 and 4 of $\mathrm{M}^2\mathrm{Clos}$ by

$$\widetilde{d}_1 = \frac{5p\sqrt{nk/6} \times \overline{F_S}}{SR} + \frac{b_I}{R} + \frac{25p}{4S(S-1)R}\left[\left(\rho_L^2 \overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2 + 1\right)\overline{F_S}\right], \tag{8}$$

$$\widetilde{d}_4 = \frac{5p\sqrt{nk/6} \times \overline{F_S}}{SR} + \frac{b_O}{R} + \frac{25p}{4S(S-1)R}\left[\left(\rho_L^2 \overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2 + 1\right)\overline{F_S}\right]. \tag{9}$$

Lemma 5 (N-FS Native Delay Bound at Stages 2 and 3 of M^2Clos). *In* $\mathrm{M}^2\mathrm{Clos}$ *with nk outputs, p switching planes, and admissible input traffic, when p and m are relatively prime, the native packet delays by* N-FS *at stages 2 and 3 of* $\mathrm{M}^2\mathrm{Clos}$ *are statistically bounded by*

$$\widetilde{d_2} = \frac{5pm\sqrt{k/6} \times \overline{F_S}}{SR} + \frac{b_I}{R} + \frac{25pm}{4S(S-1)nR}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2+1\right)\overline{F_S}\right],$$

$$\widetilde{d_3} = \frac{5pm\sqrt{k/6} \times \overline{F_S}}{SR} + \frac{b_O}{R} + \frac{25pm}{4S(S-1)nR}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2+1\right)\overline{F_S}\right].$$

Proof. See Appendix C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2010.279, and can also be found on http://s-router.cs.tsinghua.edu.cn/~shilei/Appendix-FS.pdf. □

Set $p = 2^a, m$ as odd; p and m will be relatively prime.

Theorem 3 (N-FS Delay Bound at M^2Clos). *The packet delay at the multipath part of* $\mathrm{M^2Clos}$ *is statistically bounded by*

$$d_{M^2Clos} = \frac{5(6m+9\sqrt{n})p\sqrt{k/6} \times \overline{F_S}}{SR} + \frac{12b_I + 3b_O}{R} + \frac{75p(3+2m/n)}{4S(S-1)R}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2+1\right)\overline{F_S}\right].$$

Proof. According to the additivity of the single-stage delay bound defined in Section 5.1, we obtain □

$$d_1 = \widetilde{d_1}, d_2 = d_1 + \widetilde{d_2}, d_3 = d_1 + d_2 + \widetilde{d_3}, d_4 = d_1 + d_2 + d_3 + \widetilde{d_4}.$$

Therefore, the delay bound at the multipath part (stages 1-4) of $\mathrm{M^2Clos}$ holds as the theorem.

Under a typical setting at $\mathrm{M^2Clos}$—$m = n = k = \sqrt{N} \gg 1$ (N denotes external port number)—Theorem 3 becomes

$$d_{M^2Clos} = \frac{5(6N^{\frac{3}{4}} + 9N^{\frac{1}{2}})p\overline{F_S}}{\sqrt{6}SR} + \frac{12b_I + 3b_O}{R} + \frac{375p}{4S(S-1)R}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2+1\right)\overline{F_S}\right]. \quad (10)$$

5.3 Two-Stage Load-Balanced Birkhoff-von Neumann Switches

The LBvN switch by Chang et al. [9], [10] was recently proposed and eliminates the need for input-output matching with online schedulers. It is composed of two identical switches connected in tandem, with first stage carrying out load balancing and second stage executing switching. However, since packets of the same flow may be dispatched to different switching path at the load-balancing stage, packet out-of-orders are quite possible to occur if no additional mechanism is applied.

Next, we demonstrate that the packet delay at LBvN switch with N-FS scheme is also bounded. We consider the LBvN switch with multistage buffering [9], as depicted in Fig. 14. In first stage, arrival traffic at each input is split into N L-VOQs. Packets in the ith L-VOQ are switched to output i at first switch. Each output here is connected to a C-VOQ, which buffers arrival packets according to their destination ports. Then, second switch retrieves packets from C-VOQ and forwards them to the correct output. Both switches set up connections corresponding to periodically changed permutation matrices. For example, denote switch connection at time slot t by matrix *P(t)*. A typical *P(t)* for the LBvN switch could be

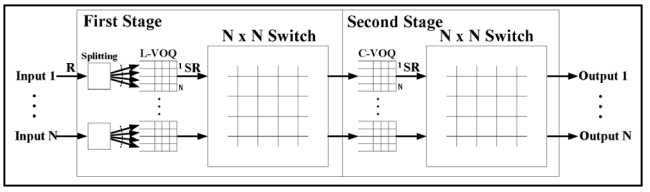

Fig. 14. Two-stage LBvN switch.

$$P_{ij} = 1(i,j \text{ connected}) \text{ where } i + j = t \text{ modulo } N$$
$$P_{ij} = 0(i,j \text{ disconnected}) \text{ otherwise.}$$

By Theorem 4, we show that packet delay bound at stages 1 and 2 of LBvN is analogous to those in PPS only by substituting port number N for switching plane number k.

Theorem 4 (N-FS Delay Bound at LBvN). *In LBvN with N output ports and a speedup S, assume traffic arrival to be admissible. By N-FS, the packet delay at LBvN is statistically bounded by*

$$d_{LBvN} = \frac{15N\sqrt{N/6} \times \overline{F_S}}{SR} + \frac{2b_I + b_O}{R} + \frac{75N}{4S(S-1)R}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2+1\right)\overline{F_S}\right].$$

Proof. At first stage of LBvN, traffic at input i is balanced to all N L-VOQs using $N-\mathrm{FS}$. By Lemma 2, we can calculate their load-balancing bound. Then, consider the output process at L-VOQ l in input i, under any backlog period $T = [t_0, t_0 + t]$, the departure traffic increases to at least *(SRt/N)-C(N-1)/N*, where C is the fixed-size cell length used internally. Therefore, the statistical backlog bound at L-VOQ is calculated by that from Theorem 1 plus *C(N-1)/N* (which can be omitted). Finally, delay bound at stage 1 of LBvN holds

$$d_1 = \frac{5N\sqrt{N/6} \times \overline{F_S}}{SR} + \frac{b_I}{R} + \frac{25N}{4S(S-1)R}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2+1\right)\overline{F_S}\right]. \quad (11)$$

At second stage of LBvN, packet arrival at each C-VOQs can also be compared to that at OQs of switches in central-stage PPS. We still substitute N for k, and derive the native packet delay bound at C-VOQ by

$$\widetilde{d_2} = \frac{5N\sqrt{N/6} \times \overline{F_S}}{SR} + \frac{b_O}{R} + \frac{25N}{4S(S-1)R}\left[\left(\rho_L^2\overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2+1\right)\overline{F_S}\right]. \quad (12)$$

Overall delay bound at LBvN sums up to $2d_1 + \widetilde{d_2}$. □

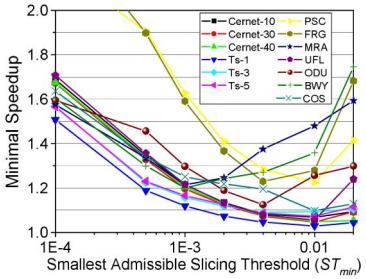

Fig. 15. Speedup requirements for PPS.

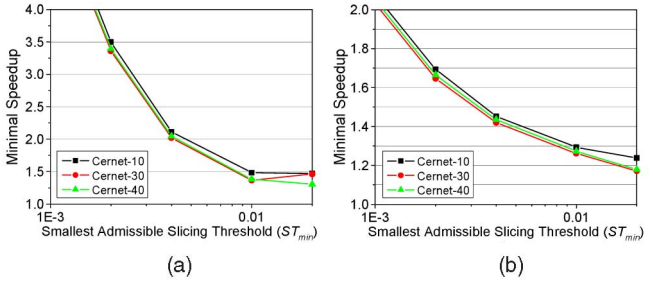

Fig. 16. Speedup requirements for (a) $\mathrm{M}^2\mathrm{Clos}$. (b) LBvN.

5.4 Admissible Slicing Threshold

Theorem 5 (Packet Out-of-Order Probability). *Setting a slicing threshold Φ for MPS with FS scheme, which leads to a statistical delay upper bound of $D_{1-\delta}$ in a $1-\delta$ confidence interval, the packet out-of-order probability in MPS is guaranteed to be no more than δ, if only we have $\phi \geq D_{1-\delta}$.*

Proof. For two packets P_1 and P_2 arriving in MPS at times $T(P_1)$ and $T(P_2)$ having $T(P_1) < T(P_2)$, packet delay $d(P_1), d(P_2)$, and departure time $D(P_1), D(P_2)$, the packets can be reordered in MPS by the FS scheme only if $[T(P_2) - T(P_1)] > \Phi$. Given $\phi \geq D_{1-\delta}$, this requirement becomes

$$[T(P_2) - T(P_1)] > D_{1-\delta}. \tag{13}$$

By the definition of $D_{1-\delta}$, for probability of no less than $1-\delta$, we have

$$0 \leq d(P_1) \leq D_{1-\delta}. \tag{14}$$

Combining (13), (14), and $d(P_2) \geq 0$, we obtain

$$\begin{aligned} D(P_2) - D(P_1) &= T(P_2) - T(P_1) + d(P_2) - d(P_1) \\ &\geq T(P_2) - T(P_1) - D_{1-\delta} > 0. \end{aligned} \tag{15}$$

Hence, the packet ordering criterion in (2) is satisfied for probability of at least $1-\delta$. In other words, packet out-of-order probability of no more than δ is guaranteed. □

In this paper, we define a slicing threshold Φ to be *admissible* if it guarantees a packet out-of-order probability of no more than 10^{-6}. We are most interested in the *smallest admissible slicing threshold* (Φ_{min}), as it provides the best load-balancing performance while satisfying the packet out-of-order requirement. By Theorem 5, we calculate Φ_{min} for all of the three MPSes with the flow-slice statistics measured in Section 3 and the delay bound computed in Sections 5.1-5.3. We assume arrival traffic to be strictly admissible ($b_I, b_O = 0$). (The extreme traffic cases are evaluated in the simulations and also discussed in Section 8.)

1. PPS. PPS is used to build switching systems with medium port count and ultrahigh line rate. Hence, we let $N \leq 32$, which is almost the maximal port number in single switch chip, and let $R/k = 5$ Gbps, representing a typical single switch line rate. By Theorem 2, delay bound at PPS holds

$$\begin{aligned} D_{1-\delta} = 0.0554 \times \overline{F_S}/S + 0.03 \times \left[\left(\rho_L^2 \overline{L}/\overline{F_C}\right)\right. \\ \left. + \left(\rho_{FS}^2 + 1\right)\overline{F_S}\right]/S(S-1)(D_{1-\delta}\text{: ms}, \\ \overline{L}\text{ : kB}, \overline{FS}\text{ : kB}, \delta = 10^{-6}). \end{aligned} \tag{16}$$

As in (16), for each trace statistic, delay bound at certain slicing threshold is only determined by the offered speedup. Given a larger speedup, the delay bound will drop and enable a smaller Φ_{min}. We calculate the minimal speedups to ensure different Φ_{min} by letting $d_{PPS} \leq \Phi_{min}$. These results are presented in Fig. 15. Speedup requirement drops first as Φ_{min} increases and then rises as Φ_{min} is beyond a point at about 1-10 ms. This is because the delay bound increases faster than the slicing threshold after this point.

We observe that a speedup of 1.409 is sufficient to ensure $\Phi_{min} \leq 2$ ms for all traces. Given a slightly larger speedup of 1.627, $\Phi_{min} \leq 1$ ms can be expected. If we only count results using backbone traces from CERNET, where a large capacity MPS is most probable to be deployed, a tiny speedup of 1.215 will ensure $\Phi_{min} \leq 1$ ms.

2. M^2Clos. $\mathrm{M}^2\mathrm{Clos}$ targets at switching systems with an ultrahigh line rate as well as large port count. Hence, we set $N \leq 1{,}024$ and $R/p = 5$ Gbps. By Theorem 3, delay bound at $\mathrm{M}^2\mathrm{Clos}$ holds

$$\begin{aligned} D_{1-\delta} = 4.488 \times \overline{F_S}/S + 0.15 \times \left[\left(\rho_L^2 \overline{L}/\overline{F_C}\right)\right. \\ \left. + \left(\rho_{FS}^2 + 1\right)\overline{F_S}\right]/S(S-1)(D_{1-\delta}\text{: ms}, \overline{L}\text{ : kB}, \\ \overline{F_S}\text{ : kB}, \delta = 10^{-6}). \end{aligned} \tag{17}$$

We depict the minimal speedups for $\mathrm{M}^2\mathrm{Clos}$ to ensure different Φ_{min} in Fig. 16a. This time, we only consider backbone traces from CERNET, since $\mathrm{M}^2\mathrm{Clos}$ will not likely be deployed in network edge/access point. Results show that a speedup of 2.112 ensures $\Phi_{min} \leq 4$ ms. Although this speedup is a little large (CRS-1 actually works at a speedup of 2.5), it only holds as an upper bound for packet out-of-order probability of 10^{-6}. In real deployments, smaller speedups will also provide excellent performance.

3. LBvN. LBvN is designed to replace single-chip switches by abandoning the use of online schedulers, so that it can support ultrahigh line rate. Hence, we let $N \leq 32$, $R \geq 40$ Gbps for LBvN. By Theorem 4, its delay bound holds

$$D_{1-\delta} = 0.221 \times \overline{F_S}/S + 0.12 \times \left[\left(\rho_L^2 \overline{L}/\overline{F_C}\right) + \left(\rho_{FS}^2 + 1\right)\overline{F_S}\right]/S(S-1)(D_{1-\delta}: \text{ms}, \overline{L}: \text{kB}, \overline{F_S}: \text{kB}, \delta = 10^{-6}). \quad (18)$$

We depict speedup requirements for LBvN in Fig. 16b, still only CERNET traces are considered. For a 32×32 LBvN, a speedup of 1.293 can ensure $\Phi_{min} \leq 4$ ms, and a speedup of 1.452 will grant a smaller Φ_{min} of 2 ms. In summary, under traffic characteristics of backbone traces, a slicing threshold at 1-4 ms is admissible for N-FS in all of the three MPSes studied above if only a speedup of no more than 2.112 is provided.

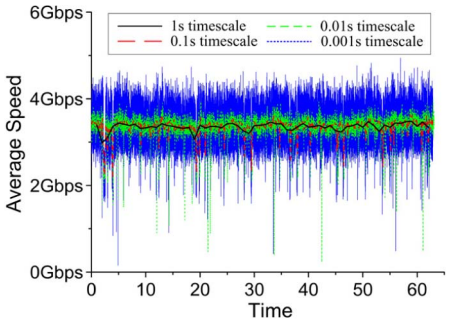

Fig. 17. Average speed of a trace segment.

6 Performance Evaluation

6.1 Prototype Configuration

We establish prototypes for all the three MPSes by software modeling. Our PPS prototype has the same architecture as the model given in Fig. 10a, except that there is no modeling of the queuing inside the multiplexers, since the multiplexers do not reorder packets. In our prototype, each packet will depart from PPS after the packet leaves the switching plane. (There is one exception for PPS with VIQ resequencer: the multiplexer is simulated since it plays a critical role in this case.) The PPS prototype has 32 external ports and eight switching planes. Each port works at a 40 Gbps line rate and each switching plane operates at 5 Gbps with no speedup. For the OQ at the central switching plane and the LB-FIFO at the demultiplexer, each has a buffer size of 10 MB each. Our LBvN prototype has 32 external ports, each working at 40 Gbps; our M^2Clos prototype has 64 external ports ($m = n = k = 8$), each working at 35 Gbps, and seven switching planes (p and m are relatively prime), each operating at 5 Gbps. The other specifications of LBvN and M^2Clos are the same as those of PPS.

We use homogeneous real traces collected at CERNET backbone (see Table 1, totally 60 segments) to generate input traffic. By default (PPS/LBvN), 32 segments are injected, respectively, to 32 ports of the prototypes. To operate the 64-port M^2Clos, we duplicate four of the segments. Each segment has similar trace speed on average, with a maximal deviation of eight percent and an average of 3.5 Gbps. This value is used to set the trace compression ratio; e.g., to emulate 40 Gbps traffic, each segment is compressed by $40\ \text{Gbps}/3.5\ \text{Gbps} = 11.4$ times. Each segment lasts 57-63 s, so we let prototypes work 55 s in trace time and measure their performance at 1-55 s. The first second is granted as warm-up time. Each arrival packet's information, including arrival time, packet length, and 5-tuples, is extracted from trace files. Packet's destination (output port) is set online according to the desired traffic pattern. Two patterns are simulated: uniform traffic, where packets have the same probability going to all 32 (64) outputs; and unbalanced traffic where packets at input i have a fixed probability of 0.8, namely, the unbalanced rate, going to output i. All other outputs average the remaining probabilities. Approximately 1.2 billion packets (2.4 billion for M^2Clos) are sent to our prototypes in each simulation slot. Note that the generated traffic is not strictly admissible but highly bursty. In Fig. 17, we depict the average speed of one trace segment in multiple timescale to reveal this fact.

Three classes of load-balancing schemes are evaluated. The first is hashing algorithms, including static hashing by XORing 5-tuples (S-Hash), dynamic table hashing (T-Hash), and dynamic table hashing with backpressure (TH-BP). The hash table here (as well as in FS) is set to 10,000 entries. In TH-BP, a backpressure signal is sent when the buffer length of a LB-FIFO increases beyond its half size. It then remaps all the flows going to the congested path at their packet arrivals. Another class is the packet-based algorithms, including RR, RR with adaptive-threshold timestamp-based resequencer (RRAT) [31], and RR with VIQ resequencer (RRVIQ). The last class is the FS scheme. Slicing thresholds are set to $0.1 - 4$ ms, as well as 60 ms to evaluate the method in [28]. FS with backpressure is also simulated (FSBP), with slicing threshold set to 1 ms.

We measure four performance metrics:

1. average packet delay;
2. packet loss rate;
3. intraflow packet out-of-order probability; and
4. average per-flow delay jitter (due to the space limit, given in Appendix D, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2010.279, and can also be found on http://s-router.cs.tsinghua.edu.cn/~shilei/Appendix-FS.pdf, as reference).

The packet out-of-order is defined in strict manner: an expected sequence number is maintained for each 5-tuple flow by received maximal sequence plus one, any packet departure is deemed as out-of-order if only its sequence is not expected. Lost packet's departure time is set to the loss time. Note that, as input traces are compressed in the simulations, each intraflow interval is shortened accordingly, leading to inaccurate out-of-order probabilities. To handle that, we restore each flow's packet departure process by decompressing intraflow intervals and then obtain the real packet out-of-order probabilities.

6.2 Results

6.2.1 PPS under Uniform Traffic Pattern

Fig. 18 depicts the average packet delay in PPS under a uniform traffic pattern. As expected, RR exhibits the smallest delay, as it optimally balances the load. RRVIQ is nearly the same as RR showing that the VIQ resequencer does not delay packets much. (However, the resequencer suffers from huge implementation complexity.) Closely on top of them in the figure lies the class of FS. At a load rate of 0.85, FS with a slicing threshold of 1 ms only increases the

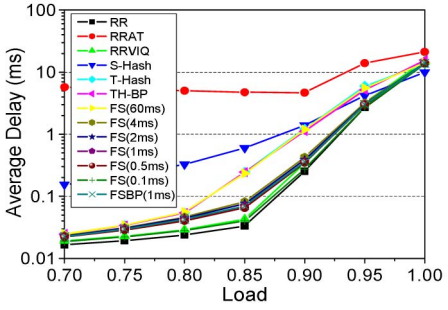

Fig. 18. Average delay (PPS, uniform).

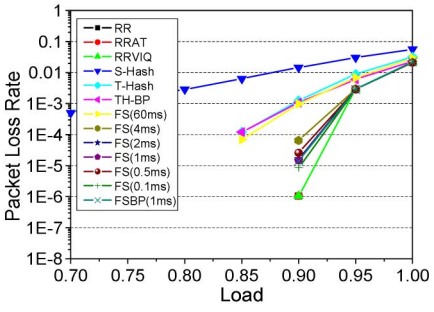

Fig. 19. Packet loss (PPS, uniform).

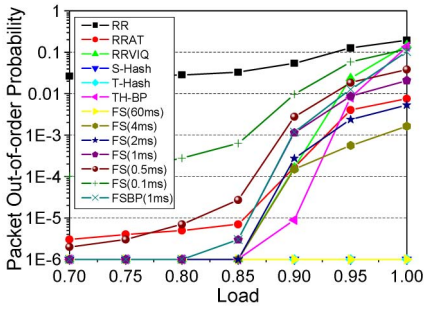

Fig. 20. Out-of-order (PPS, uniform).

average delay by one time from the optimal RR scheme, while T-Hash results in a delay of more than six times larger. FS with a 60 ms slicing threshold performs nearly the same as T-Hash, revealing the ineffectiveness of method in [28]. Also, the backpressure mechanism does not reduce delay for either T-Hash or FS. In this slot, S-Hash and RRAT perform the worst. They increase average delay one and two magnitudes, respectively, from RR even at moderate loads. This validates the load imbalance of static hashing and the delay penalty suffered by RRAT.

Fig. 19 gives the packet loss rates in PPS. FS schemes begin to drop packets at a load rate of 0.9, which is better than all hashing solutions. Fig. 20 depicts packet out-of-order probability. RR without a resequencer consistently disorders more than two percent packets, while FS limits packet out-of-order to a negligible level (below 10^{-6}) if only the slicing threshold is no less than 1 ms and the load rate is no larger than 0.8. This load bound corresponds to a speedup of 1.25, very close to the analytic result of the 1.215 speedup in Section 5.4.

6.2.2 PPS under Unbalanced Traffic Pattern

Results under unbalanced traffic are given in Figs. 21, 22, and 23. They are similar to the case under uniform traffic.

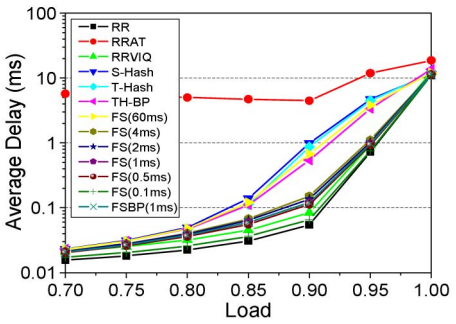

Fig. 21. Average delay (unbalanced).

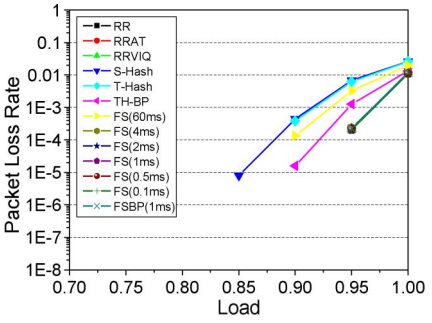

Fig. 22. Packet loss (unbalanced).

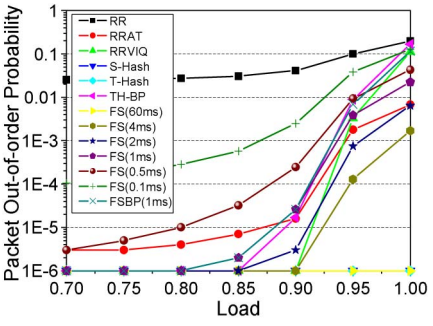

Fig. 23. Out-of-order (unbalanced).

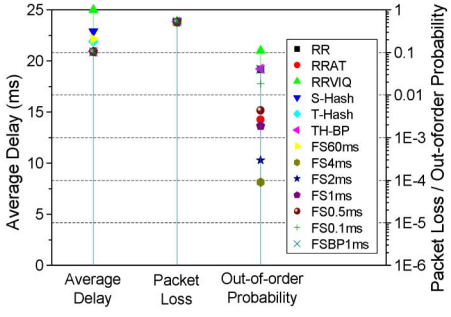

Fig. 24. Under overloaded traffic (PPS, uniform).

At a slicing threshold of 1 ms, FS limits packet out-of-order probability to below 10^{-6}, under a load rate of no larger than 0.8.

6.2.3 PPS under Overloaded Traffic

We also test the case in which load rate is set to 2.0 under a uniform traffic pattern. Results plotted in Fig. 24 show that both packet delay and loss of all the algorithms are similar as this time all the buffers are always full, leading to the largest delay and a 50 percent loss rate. Surprisingly, FS with a slicing threshold of 4 ms still limits out-of-order

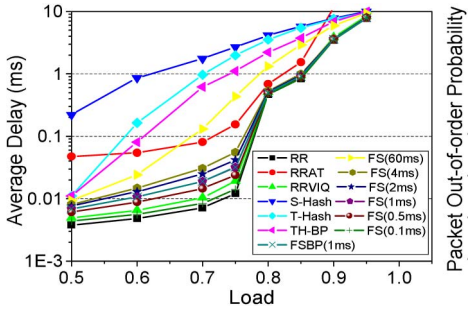

Fig. 25. Average delay (Tsinghua egress trace, PPS, uniform).

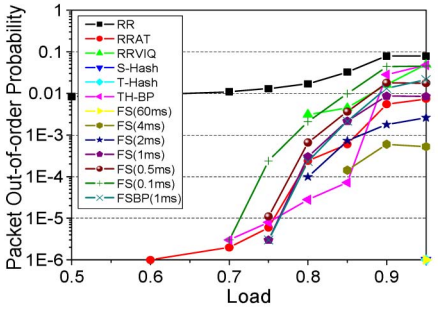

Fig. 26. Out-of-order (Tsinghua egress trace, PPS, uniform).

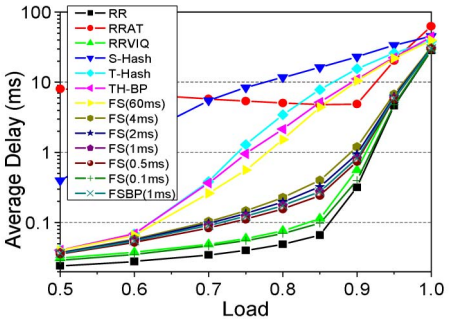

Fig. 27. Average delay (LBvN).

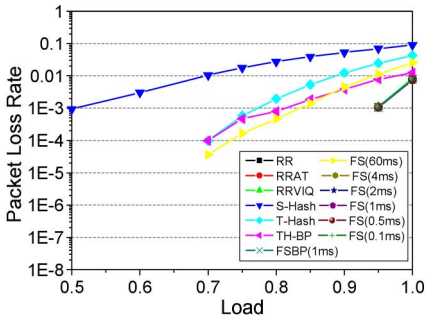

Fig. 28. Packet loss (LBvN).

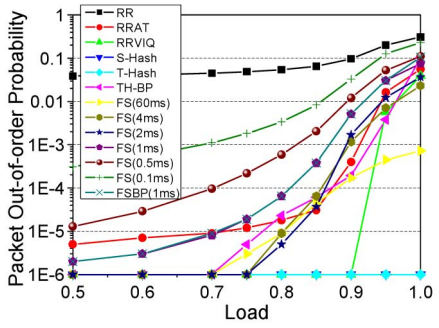

Fig. 29. Out-of-order (LBvN).

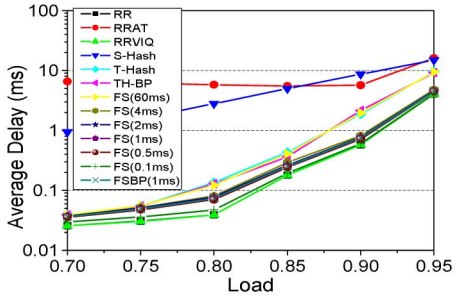

Fig. 30. Average delay (M^2Clos).

probability below 10^{-4} in this worst case, which reveals the excellent robustness of FS.

6.2.4 PPS under More Heavy-Tailed Flow-Size Distribution

As in Figs. 5 and 6, CERNET traces exhibit the lightest tail in flow-size distribution. It is also interesting to study how FS performs if input traffic is heavier tailed. We reiterate our simulation in the fist slot with traces collected at the campus gateway link of Tsinghua University to CERNET (see Table 1). Figs. 25 and 26 depict the average packet delay and out-of-order probability in this case. Comparing Fig. 25 with 18, we find that FS receives better comparable performance with hashing algorithms under more heavy-tailed traffic. Its average delay is at least five times smaller than those of T-Hash and TH-BP at a load rate of 0.6. However, in this case, FS requires a larger speedup of 1.33 (at 1 ms slicing threshold) to keep packet out-of-order probability below 10^{-6}.

6.2.5 LBvN under Uniform Traffic Pattern

Figs. 27, 28, and 29 depict experiment results of a 32-port LBvN switch. FS schemes this time achieve even better performance than in PPS. At a load rate of 0.7, FS lowers the average delay to 1/3 of hashing algorithms; at a load rate of 0.95, it begins to drop packets at the same packet loss rate as RR. Setting slicing threshold to 4 ms, FS limits packet out-of-order to a negligible level under a load rate of no more than 0.75. This is close to the theoretical result of a speedup of 1.293 in Section 5.4. Since 32-port LBvN can be compared to a 32-plane PPS, we can deduce that as plane number increases, PPS with an FS scheme will become more effective compared with hashing algorithms.

6.2.6 M^2Clos under Uniform Traffic Pattern

Performance metrics of a 64-port M^2Clos switch under uniform traffic pattern are illustrated in Figs. 30, 31, and 32. They are also similar to those in PPS. This time, FS under a slicing threshold of 4 ms requires a speedup of 1.25 to retain a negligible out-of-order probability, which is much smaller than the theoretical value of 2.112 in Section 5.4. This is because the latter result is for 1,024-port M^2Clos. By (10) and (17), we recalculate a theoretical speedup of 1.391 for 64-port case. This speedup is very close to the simulated one.

In summary, in the above cases, the FS scheme outperforms hashing algorithms and the adaptive-threshold resequencer in providing better load-balancing metrics

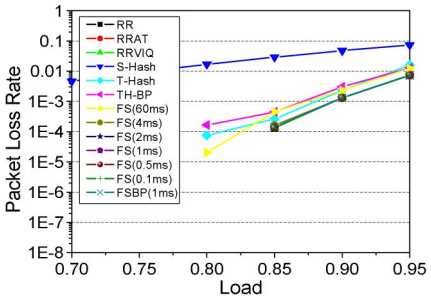

Fig. 31. Packet loss (M^2Clos).

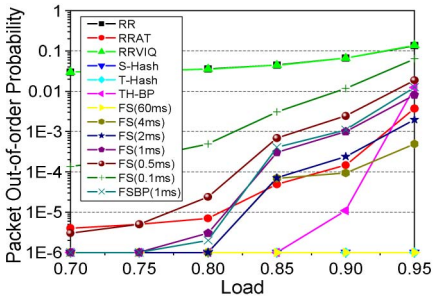

Fig. 32. Out-of-order (M^2Clos).

without paying out huge hardware complexity as the VIQ resequencer does. Under a slicing threshold of 1-4 ms, FS is able to keep packet out-of-order probability negligible with the provided speedup of no more than 1.33. These results, on one hand, reveal the effectiveness of FS; on the other hand, they validate the theoretical analyses in Section 5.

7 Implementation Issues

The major implementation cost introduced by FS is that of hash table maintaining an active flow-slice context. The size of this hash table is a trade-off 1) a large hash table reduces *hash collision* probability. Here, hash collision is defined as the situation when more than one flow slices are hashed into the same entry, which degrades load-balancing uniformity; while 2) a small hash table is desirable as the table is accessed in each load-balancing operation and on-chip SRAM with relatively small size allows higher access speed.

We calculate this trade-off in a Dual Hash Table (DHT) approach [25]. Assume a hash collision probability below 0.5 percent as negligible, where only one flow slice out of 200 is not load balanced independently. Then, at OC-768c port with an average packet length of $P_L \geq 400\text{B}$ and a slicing threshold of $\Phi \leq 4$ ms, we get the number of table entries, $F \leq 40\ \text{Gbps} \times 4\ \text{ms}/400\text{B} = 50\ \text{k}$. The hash table size is then bounded by $50\ \text{k} \times 5.35 \times 6\text{B} = 1.6\ \text{MB}$, where each entry occupies 6B size. This is small enough to be placed on-chip. As each FS load-balancing operation only needs one query at the hash table, $O(1)$ timing complexity is also achieved.

8 Conclusion, Discussion and Future Work

We propose a novel load-balancing scheme, namely, Flow Slice, based on the fact that the intraflow packet interval is often, say in 40-50 percent, larger than the delay upper bound at MPS. Due to three positive properties of flow slice, our scheme achieves good load-balancing uniformity with little hardware overhead and $O(1)$ timing complexity. By calculating delay bounds at three popular MPSes, we show that when the slicing threshold is set to the smallest admissible value at 1-4 ms, the FS scheme can achieve optimal performance while keeping the intraflow packet out-of-order probability negligible (below 10^{-6}), given an internal speedup up to two. Our results are also validated through trace-driven prototype simulations under highly bursty traffic patterns.

It is interesting to ask why flow slice exhibits such properties favoring load-balancing uniformity. Previous studies [26], [28] provide an answer in TCP's burstiness; i.e., a window of packets is transmitted at the very beginning of each RTT time, followed by a long silent period. This is the most critical factor when the slicing threshold is comparable with RTT. However, as the slicing threshold is set below 4 ms, much smaller than typical RTT, the dominating factor is most likely to be the random delay added along the previous routing path. If this delay is independent for intraflow packets and larger than the slicing threshold, the slicing probability will be about 0.5 and the average flow-slice packet count will be approximately two, coinciding with our trace observations.

We have proven FS to be effective under strictly admissible traffic, but it is also important to know how it behaves under extreme traffic. Our simulations with bursty real traces shed some light on this issue and suggest that it still work well. Actually, if only the slicing threshold is larger than the *delay variation bound* at all switching paths, packet order will be undisturbed. Under bursty input traffic, the delay at all switching paths may increase synchronously, leaving its delay variation bound nearly unchanged.

The FS scheme is validated in switches without class-based queues. As QoS provisioning is also critical in switch designs, one of our future works will be studying FS performance under QoS conditions.

Acknowledgments

This work was done while Lei Shi, Changhua Sun, and Zhengyu Yin were with Tsinghua University. This work was partially supported by NSFC (60625201, 60873250), Tsinghua University Initiative Scientific Research Program, and the Specialized Research Fund for the Doctoral Program of Higher Education of China (20100002110051).

References

[1] Cisco CRS-1, http://www.cisco.com/go/crs/, 2011.
[2] Real Traces from NLANR, http://pma.nlanr.net/, 2010.
[3] Vitesse Intelligent Switch Fabrics, http://www.vitesse.com, 2011.
[4] A. Aslam and K. Christensen, "Parallel Packet Switching Using Multiplexors with Virtual Input Queues," *Proc. Ann. IEEE Conf. Local Computer Networks (LCN)*, pp. 270-277, 2002.
[5] J. Bennett, C. Partridge, and N. Shectman, "Packet Reordering Is Not Pathological Network Behavior," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 789-798, Dec. 1999.
[6] N. Brownlee and K. Claffy, "Understanding Internet Traffic Streams: Dragonflies and Tortoises," *IEEE Comm. Magazine*, vol. 40, no. 10, pp. 110-117, Oct. 2002.
[7] Z. Cao, Z. Wang, and E. Zegura, "Performance of Hashing-Based Schemes for Internet Load Balancing," *Proc. IEEE INFOCOM*, pp. 332-341, 2000.
[8] L. Carter and M. Wegman, "Universal Classes of Hashing Functions," *J. Computer and System Sciences*, vol. 18, no. 2, pp. 143-154, 1979.

[9] C.S. Chang, D.S. Lee, and Y.S. Jou, "Load Balanced Birkhoff-von Neumann Switch, Part II: Multi-Stage Buffering," *Computer Comm.*, vol. 25, pp. 623-634, 2002.
[10] C.S. Chang, D.S. Lee, and Y.S. Jou, "Load Balanced Birkhoff-von Neumann Switches, Part I: One-Stage Buffering," *Computer Comm.*, vol. 25, pp. 611-622, 2002.
[11] C.S. Chang, D.S. Lee, and Y.J. Shih, "Mailbox Switch: A Scalable Two-Stage Switch Architecture for Conflict Resolution of Ordered Packets," *Proc. IEEE INFOCOM*, 2004.
[12] C.S. Chang, D.S. Lee, and C.Y. Yue, "Providing Guaranteed Rate Services in the Load Balanced Birkhoff-von Neumann Switches," *IEEE/ACM Trans. Networking*, vol. 14, no. 3, pp. 644-656, June 2006.
[13] H.J. Chao, P. Jinsoo, S. Artan, S. Jiang, and G. Zhang, "TrueWay: A Highly Scalable Multi-Plane Multi-Stage Buffered Packet Switch," *Proc. IEEE Workshop High Performance Switching and Routing (HPSR)*, 2005.
[14] F.M. Chiussi, D.A. Khotimsky, and S. Krishnan, "Generalized Inverse Multiplexing of Switched ATM Connections," *Proc. IEEE Conf. Global Comm. (GLOBECOM)*, pp. 3134-3140, 1998.
[15] G. Dittmann and A. Herkersdorf, "Network Processor Load Balancing for High-Speed Links," *Proc. Int'l Symp. Performance Evaluation of Computer and Telecomm. Systems (SPECTS)*, 2002.
[16] A.B. Downey, "Evidence for Long-Tailed Distributions in the Internet," *Proc. ACM SIGCOMM Workshop Internet Measurement (IMW)*, 2001.
[17] M. Henrion, "Resequencing System for a Switching Node," US Patent, 5,127,000, June 1992.
[18] S. Iyer, A. Awadallah, and N. McKeown, "Analysis of a Packet Switch with Memories Running Slower than the Line Rate," *Proc. IEEE INFOCOM*, pp. 529-537, 2000.
[19] S. Iyer and N. McKeown, "Analysis of the Parallel Packet Switch Architecture," *IEEE/ACM Trans. Networking*, vol. 11, no. 2, pp. 314-324, Apr. 2003.
[20] L. Kencl and J.-Y.L. Boudec, "Adaptive Load Sharing for Network Processors," *Proc. IEEE INFOCOM*, pp. 545-554, 2002.
[21] I. Keslassy and N. Mckeown, "Maintaining Packet Order in Two-Stage Switches," *Proc. IEEE INFOCOM*, pp. 1032-1041, 2002.
[22] D.A. Khotimsky, "A Packet Resequencing Protocol for Fault-Tolerant Multipath Transmission with Non-Uniform Traffic Splitting," *Proc. IEEE Conf. Global Comm. (GLOBECOM)*, pp. 1283-1289, 1999.
[23] D.A. Khotimsky and S. Krishnan, "Evaluation of Open-Loop Sequence Control Schemes for Multi-Path Switches," *Proc. IEEE Int'l Conf. Comm. (ICC)*, pp. 2116-2120, 2002.
[24] L. Shi, W. Li, B. Liu, and X. Wang, "Flow Mapping in the Load Balancing Parallel Packet Switches," *Proc. IEEE Workshop High Performance Switching and Routing (HPSR)*, pp. 254-258, 2005.
[25] W. Shi and L. Kencl, "Sequence-Preserving Adaptive Load Balancers," *Proc. ACM/IEEE Symp. Architecture for Networking and Comm. Systems (ANCS)*, 2006.
[26] W. Shi and M.H. MacGregor, "Load Balancing for Parallel Forwarding," *IEEE/ACM Trans. Networking*, vol. 13, no. 4, pp. 790-801, Aug. 2005.
[27] W. Shi, M.H. MacGregor, and P. Gburzynski, "A Scalable Load Balancer for Forwarding Internet Traffic: Exploiting Flow-Level Burstiness," *Proc. Symp. Architecture for Networking and Comm. Systems (ANCS)*, 2005.
[28] S. Sinha, S. Kandula, and D. Katabi, "Harnessing TCP's Burstiness with Flowlet Switching," *Proc. ACM SIGCOMM Workshop Hot Topics in Networks (HotNets)*, 2004.
[29] D.G. Thaler and C.V. Ravishankar, "Using Name-Based Mappings to Increase Hit Rates," *IEEE/ACM Trans. Networking*, vol. 6, no. 1, pp. 1-14, Feb. 1998.
[30] J.S. Turner, "Resequencing Cells in an ATM Switch," Technical Report WUCS-91-21, Feb. 1991.
[31] J.S. Turner, "Resilient Cell Resequencing in Terabit Routers," Technical Report WUCS-03-48, June 2003.

Lei Shi received the BS, MS, and PhD degrees in computer science from Tsinghua University in 2003, 2006, and 2008, respectively. He is now the research manager of visual analytics team in IBM Research - China where he joined in 2008. His research interests include information visualization, visual analytics, network science, and networked systems.

Bin Liu is now a full professor in the Department of Computer Science and Technology, Tsinghua University. His current research areas include high performance switches/routers, network processors, high-speed security, and greening the Internet. He has received numerous awards from China including the Distinguished Young Scholar of China.

Changhua Sun received the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2009. He is now a staff researcher in IBM Research-China.

Zhengyu Yin received the bachelor's degree from the Department of Computer Science, Tsinghua University, Beijing, China. He is now a graduate student in the Department of Computer Science, University of Southern California. He is currently a member of TEAMCORE research group focusing on multiagent systems directed by Professor Milind Tambe.

Laxmi N. Bhuyan is a distinguished professor and the chairman of the Computer Science and Engineering Department at the University of California, Riverside (UCR). He joined UCR in January 2001 as a professor. Prior to that, he was a professor of computer science at Texas A&M University (1989-2000) and the program director of the Computer System Architecture Program at the US National Science Foundation (1998-2000). He has also worked as a consultant to Intel and HP Labs. His current research interests are in the areas of computer architecture, network processors, Internet routers, and parallel and distributed processing. He has published more than 150 papers in related areas in reputed journals, and conference proceedings. He was the Editor-in-Chief of the *IEEE Transactions on Parallel and Distributed Systems* (TPDS) from 2006 to 2009. He is a fellow of the IEEE, a fellow of the ACM, and a fellow of the AAAS.

H. Jonathan Chao is the department head and a professor of electrical and computer engineering at Polytechnic Institute of New York University, Brooklyn, where he joined in January 1992. He has been doing research in the areas of terabit switches/routers, network security, network on the chip, and quality of service control in high-speed networks. He is a fellow of the IEEE for his contributions to the architecture and application of VLSI circuits in high-speed packet networks. He has published three networking/switching books.