A High-Speed Large-Capacity Packet Buffer Scheme for High-Bandwidth Switches and Routers

Ling Zheng[1], Zhiliang Qiu[1], Weitao Pan[1](✉), and Ya Gao[2]

[1] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China
wtpan@mail.xidian.edu.cn

[2] School of Internet of Things Technology, Wuxi Institute of Technology, Wuxi, China

Abstract. Today's switches and routers require high-speed and large-capacity packet buffers to guarantee a line rate up to 100 Gbps as well as more fine-grained quality of service. For this, this paper proposes an efficient parallel hybrid SRAM/DRAM architecture for high-bandwidth switches and routers. Tail SRAM and head SRAM are used for guaranteeing the middle DRAMs are accessed in a larger granularity to improve the bandwidth utilization. Then, a simple yet efficient memory management algorithm is designed. The memory space is dynamically allocated when a flow arrives, and a hard timeout is assigned for each queue. Hence, the SRAM space is utilized more efficiently. A queueing system is used to model the proposed method, and theoretical analysis is performed to optimize the timeout value. Simulation shows that the proposed architecture can reduce packet loss rate significantly compared with previous solutions with the same SRAM capacity.

Keywords: Switching system · Packet buffer · SRAM · DRAM Queueing system

1 Introduction

The rapid development of the Internet as well as appearance of the 40/100GE standard have placed higher demands on network switches and routers. With the emergence of new generation data centers and various killer applications such as cloud service, high-definition video streaming and social media, it requires the switching devices to develop towards larger capacity, higher bandwidth and more sophisticated Quality of Service (QoS) guarantees to meet a variety of demands of new applications.

Most current packets switches deploy Store-and-Forward architecture. The incoming packets are first stored in the packet buffers and then forwarded to the destination ports. The packet buffers are critical for network congestion control. The access time and capacity of the switch memory have significant influences on the performance of a switch [1, 2]. On one hand, the ever-increasing bandwidth requires the memory to operate extremely fast to keep up with the line rate of the input port. On the other hand, the novel network techniques, such as software-defined networking

[3] and OpenFlow [4], need fine-grained flow classifications for sophisticated QoS. As a result, the switch memory often maintains several thousands of queues. Therefore, a large amount of memory space is commonly required. Current memory technologies, such as Static Random Access Memory (SRAM) or Dynamic Random Access Memory (DRAM), cannot simultaneously satisfy both the speed and capacity requirements.

To simultaneously meet these requirements, hybrid SRAM/DRAM (HSD) architecture has been proposed [5]. Fast but small-size SRAM meets the random access time and large but slow DRAM can satisfy the capacity requirement. The main idea of HSD is to use two SRAM caches. The tail SRAM is used to store packets at the tail of each FIFO queue, while the head SRAM holds packets at the head. The bulk DRAM maintains the large middle part of the queues. Both SRAM and DRAM maintain Q separate flow queues, and each queue store the input packet from a certain flow. The incoming packets are first stored in the tail SRAM. A memory management algorithm (MMA) is responsible for the transmission between Tail/Head SRAM and DRAM. The basic HSD architecture is shown in Fig. 1.

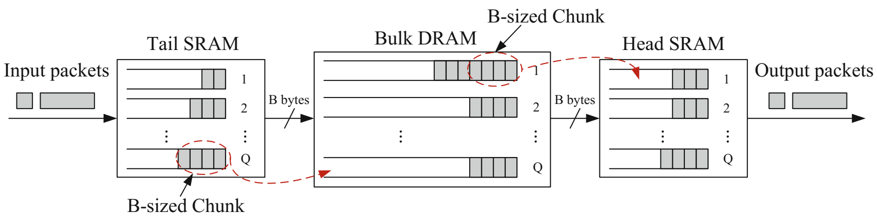

Fig. 1. Basic hybrid SRAM/DRAM architecture

The key function of MMA is to aggregate the input packets into B-sized chunks, and transfer a chunk from tail SRAM to DRAM each time. Similarly, the MMA always read a chunk of B bytes from DRAM and then send it to the head SRAM. When a SRAM queue accumulates B bytes of data, it is transferred to DRAM through a single write. The chunk size B is usually a constant and generally can be determined by $B = 2RA$, where R is the input line rate and A is the random access time of DRAM. Reference [6] proved that the required tail and head SRAM size is $Q(B-1)$ bytes. However, this buffer size is too large for current SRAM chip, due to the fact that today's data center switch may maintain from hundreds to tens of thousands of flow queues [7] and the input line rate is upgrading towards 100/400G. Besides, today's DRAM device cannot provide such a bit width of B bytes for a single data transfer.

To reduce the SRAM capacity requirement and the chunk size, researchers introduced interleaved DRAM [8, 9] and parallel DRAMs architectures [10, 12]. However, the interleaved DRAMs need additional effort for DRAM bank management and the modification of DRAM controller. This task may be too complex for real time processing. In addition, in the parallel DRAM architecture, the bandwidth utilization of each DRAM is not optimized because it often use single FIFO queue in the tail SRAM [11]. To reduce the chunk size to 64 bytes, we need $k = 20$ parallel DRAMs for $R = 100$ Gbps line rate processing [13], which cannot be implemented now and in near future.

In this paper, we introduce a high-speed large-capacity packet buffer scheme for high-bandwidth switches and routers. Parallel DRAMs are used to build the bulk memory, and SRAMs are used as caches for tail and head packets. Per-flow data aggregation is employed to achieve sequential access to DRAMs, so that the effective bandwidth can be improved and in-sequence packet service is guaranteed. Besides, we design a simple yet efficient MMA. The main idea is that the memory space of tail SRAM is only allocated for the most recent active flows, increasing the memory utilization. Theoretical analysis is performed and the optimal timeout value can be obtained. Numerical results show that the proposed architecture can reduce the overall packet loss rate at lower hardware costs compared with existing methods.

The rest of the paper is organized as follows. In Sect. 2, we introduce the proposed HSD architecture. In Sect. 3, theoretical analysis is performed, then the closed-form formulas is derived for analysis and optimization. Performance simulation are presented in Sect. 4. Finally, the conclusions are given in Sect. 5.

2 System Architecture

The proposed HSD architecture is shown in detail in Fig. 2. Basically, it consists of *k* parallel subsystems, each being a combined SRAM/DRAM structure. Each subsystem consists of one DRAM, tail SRAM, head SRAM, tail/head buffer management module. Besides, this architecture also has dynamic memory allocation module, dispatcher, re-assembler and output scheduler.

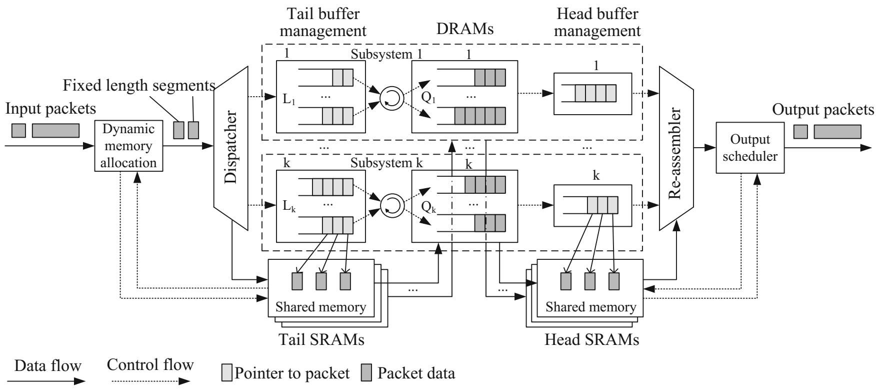

Fig. 2. Proposed HSD system architecture

Dynamic memory allocation: When a packet comes into the switch, the dynamic memory allocation module request free memory space form the tail SRAM, then pointers to the free memory are returned if the request is successful. The memory space in tail SRAM and head SRAM is managed by fixed length buffer, for purpose of fully utilize the limited SRAM size. So the incoming variable length packet are segmented into fixed length segments, typically 64 bytes.

Dispatcher and tail buffer management: The dispatcher module distributes the fixed length segments to their destination queue according to the flow classification results. The tail buffer management module maintains k queue groups, one group for one DRAM. The i-th queue group maintains L_i logic queues, $i = 1, \cdots, k$. The total number of logic queues is $L = \sum_{i=1}^{k} L_i$. These logic queues are actually linked lists of pointers. The packet data is physically stored in the SRAM. The tail SRAM is used for the aggregation of packet data *per-flow*, and the transmission to DRAM in granularity of chunks.

Parallel DRAMs: There are k parallel DRAMs, and each DRAM provides $1/k$ of the required bandwidth. The i-th DRAM contains Q_i physical queue for *per-flow* buffering, where $i = 1, \cdots, k$. Let Q denote to the number of flows the switch can classify, then $Q = \sum_{i=1}^{k} Q_i$. This Q queues should be uniformly distributed in this k DRAMs. Due to the fact that the DRAMs have large enough memory space, the Q physical queues is pre-allocated for each flow. The input packets are transmitted to DRAM in granularity of B-sized chunks to improve the effective bandwidth.

Head buffer management and output scheduler: The output scheduler polls each flow queue according to some schedule algorithm, and fetch the packet data from it to the head SRAM. The read granularity of the DRAM is B-sized chunks too. Similar to the tail buffer management, the head buffer management also uses linked list to manage the logic queue, and the packet is physically stored in the head SRAM. The difference is that the head buffer management only manage k logic queue, one queue per DRAM, because there is no need for per-flow data aggregation in the output side. The outgoing fixed length segments are re-assembled into variable length packet before output.

Memory Management Algorithm: The MMA works as follows. When a flow f arrives, the MMA requests a chunk of B bytes from tail SRAM for data buffering of this flow. If there is no enough memory space, this flow is blocked and dropped. If successfully allocated, a pointer to the allocated memory space is returned and the subsequent packets of this flow are stored in the memory as a queue for data aggregation. Meanwhile, a countdown timer with a default timeout T is assigned to the queue. The timer starts count down once it is enabled. When it reaches 0, a timeout signal is generated. The subsequent packets of flow f are stored in the allocated queue. When data in the queue reaches to B bytes, the aggregated data is transferred into DRAM with sequential address access. If a timeout signal of a queue is received, the data in this queue is transferred to DRAM, even if the queue length is smaller than the chunk size B. Then the memory space of the queue is freed and the timer is disabled. The MMA is simple enough to be implemented in high-speed switches and routers, because it only needs several timers to identity the timeout.

3 Performance Analysis

In the memory management algorithm, the timeout value T should be set a proper value. If T is too long, more memory space is cost for stale flows, and the blocking probability of newly arrive flow will increase. If T is too short, we cannot aggregate enough data for sequential DRAM access, leading to low DRAM transfer speed. In this section, we perform theoretical analysis and wish to obtain the optimal value of T.

Since the k subsystems operate completely parallel and the input traffic is uniformly distributed in the k subsystem, the k subsystem can be regarded as equally. We only analyze a behavior of a certain subsystem. The system analysis model is shown in Fig. 3.

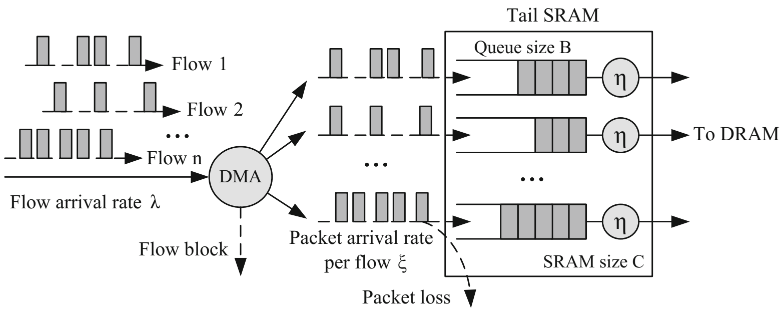

Fig. 3. System analysis model

The traffic applied to the subsystem is composed of different network flows, and the stochastic process of each flow is independent and identically distributed. The behavior of tail SRAM can be regarded as a queueing system where the arriving flows are customers and the memory space allocated for each flow is service station [14]. When the memory space is dynamically allocated for an arriving flow, a customer gets service in the queueing system. When the memory space is freed, this customer leaves the system. The arriving process of flows can be modeled as Poisson process with parameter λ, and the flow durations are exponentially distributed with parameter μ [15]. Let random variable W denote the flow size (number of packets) of each flow, and R denote the total packet arrival rate.

Within each flow, the packet arrival also can be modeled as Poisson process with parameter ξ, and the service time of each packet is exponentially distributed with parameter $1/\eta$. Denote variable X to express the flow duration. Let T denote the timeout value, and the total service time of a flow is denoted as Y. When a new flow come but there is no enough free memory space to be allocated, this flow is *blocked* and dropped. The flow blocking probability is denoted as P_C and C is the total size of the tail SRAM (the number of queues the SRAM can accommodate). The overall packet loss rate is $P_{overall}$.

First, at the view of flow arrival, we analyze the flow blocking probability. If a flow's duration is less than T, it will get a service time of T. If the duration is larger than T, a new buffer will be allocated for this flow and the flow will get another service time of T. The flow duration is exponentially distributed, so that the cumulative distribution function (CDF) is:

$$F(x) = 1 - e^{-\mu x} \tag{1}$$

And the expectation of X is $E[X] = 1/\mu$. Then we have the probability distribution of Y as follows

$$P(Y = iT) = P\{(i-1)T < X \leq iT\} = e^{-(i-1)\mu T} - e^{-i\mu T}, \quad i = 1, 2, \cdots \tag{2}$$

And the expectation of Y is

$$\begin{aligned} E[Y] &= \sum_{i=1}^{\infty} iT \times P\{Y = iT\} = T(1 + e^{-\mu T} + e^{-2\mu T} + \cdots) \\ &= T \times \lim_{n\to\infty} \frac{1 - e^{-n\mu T}}{1 - e^{-\mu T}} = \frac{T}{1 - e^{-\mu T}} \end{aligned} \tag{3}$$

The above queueing system is an M/G/C/C system, where the flow arrival rate is λ and the service rate is $1/E[Y]$. So that the blocking probability can be obtained by the Erlang-B formula

$$P_C = \frac{\rho^C}{C!}\left[\sum_{i=0}^{C} \frac{\rho^i}{i!}\right]^{-1} \tag{4}$$

where $\rho = \lambda E[Y]$.

Next, at the view of packet arrival per flow, we study the packet loss rate within a flow queue in tail SRAM. Once a flow arrives and memory space is allocated for it, the flow queue can be modeled as an M/M/1/B queue, where B is the allocated memory space for each queue.

According to the characteristic of compound Poisson process, we have the following relation

$$R = \lambda E[W] \tag{5}$$

The average packet arrive rate per flow can be obtained by the flow size divided by the flow duration, which is

$$\xi = \frac{E[W]}{E[X]} = \frac{\mu R}{\lambda} \tag{6}$$

And the service rate of this M/M/1/B queue depends on the transmission speed of DRAM and average number of queues. Let random variable N denote the number of queues in the system, then the expectation of N can be obtained as follows

$$E[N] = \sum_{n=0}^{C} np_n = \sum_{n=0}^{C} \left(n \times \frac{\rho^n}{n!} \left[\sum_{i=0}^{C} \frac{\rho^i}{i!} \right]^{-1} \right) = \frac{\rho \left(\sum_{n=0}^{C-1} \frac{\rho^n}{n!} \right)}{\sum_{i=0}^{C} \frac{\rho^i}{i!}} = \rho(1 - P_C) \tag{7}$$

It can be considered that the total bandwidth of DRAM is equally shared by these N queues. As mentioned above, the more data aggregated in the SRAM queue, the higher speed of DRAM. In this study, we assume that the speed of DRAM has a linear relationship with the aggregated queue length. According to Little's theorem, the average queue length equals to packet arrival rate multiplied by the timeout value, so we have the service rate of each M/M/1/B queue by the following equation

$$\eta = \begin{cases} \frac{S_{\min} + \alpha\xi T}{E[N]}, & 0 < T \le \frac{S_{\max} - S_{\min}}{\alpha\xi} \\ \frac{S_{\max}}{E[N]}, & T > \frac{S_{\max} - S_{\min}}{\alpha\xi} \end{cases} \tag{8}$$

where $S_{\max}$ and $S_{\min}$ is the maximum and minimum speed of DRAM, and α is a constant. When $T > (S_{\max} - S_{\min})/(\alpha\xi)$, there is no improvement in DRAM speed since the DRAM can operate at its peak bandwidth, so we only consider the situation where $0 < T \le (S_{\max} - S_{\min})/(\alpha\xi)$.

Therefore, in the M/M/1/B queue, the packet loss rate can be obtained by

$$P_B = \frac{(\xi/\eta)^B (1 - \xi/\eta)}{1 - (\xi/\eta)^{B+1}} \tag{9}$$

Together with (4), (5), (6), (7), (8) and (9), we have the overall packet loss rate as follows

$$P_{overall} = P_C + (1 - P_C)P_B = \frac{\rho^C}{C!} \left[\sum_{i=0}^{C} \frac{\rho^i}{i!} \right]^{-1} + \left(1 - \frac{\rho^C}{C!} \left[\sum_{i=0}^{C} \frac{\rho^i}{i!} \right]^{-1} \right) \frac{(\xi/\eta)^B (1 - \xi/\eta)}{1 - (\xi/\eta)^{B+1}} \tag{10}$$

The optimal value of T can be obtained by solving the optimization problem

$$T* = \arg\ \min_{0 < T < (S_{\max} - S_{\min})/(\alpha\xi)} P_{overall} \tag{11}$$

4 Simulation

In this section, we evaluate the proposed architecture through software simulation. The following parameters are assumed throughout the simulations. We assume that the input packets have a fixed length and time is split into discrete time slots. The SRAM size is 1600 packets, and the chunk size allocated for each arriving flow is 16 packets. The total packet arrive rate R is fixed to 100, i.e. 100 packets arrive in the system each time slot on average. Each simulation is performed using 10^5 time slots.

In the first simulation, we consider two traffic scenario: "light flow arrival with large flow size" and "heavy flow arrival with small flow size". We use triple $(R, \lambda, 1/\mu)$ to describe traffic condition, then the first traffic scenario is express by (100, 8, 5), while the second one is (100, 18, 5). The results are shown in Table 1 and the following results can be found. First, in the light flow arrival traffic pattern, a relatively long timeout (T = 6) should be set to perform better data aggregation, so that the DRAM's access speed can be improved. Second, the heavy flow arrival traffic will cause a higher packet loss rate. This is because heavy flow arrive rate will cause higher blocking probability, and data aggregation is more difficult. But we can choose a relatively small timeout (T = 2) to let the SRAM to accommodate more flows. In addition, we use Distributed Packet Buffers (DPB) [11] and Semi-Parallel Hybrid SRAM/DRAM (SPHSD) [13] as the benchmarking methods. It is found that overall packet loss rates of DPB and SPHSD are independent of T. Under the same traffic scenario, if a proper T is set, the proposed method has a better performance in packet loss rate compared with DPB and SPHSD.

Table 1. Results of simulation-1, comparison of different traffic scenario.

Traffic Scenario		Overall packet loss rate		
		Proposed	DPB [11]	SPHSD [13]
(100, 8, 5) light flow arrival with large flow size	T = 1	0.11058	0.111438	0.361268
	T = 2	0.0310028	0.109163	0.363204
	T = 4	0.0304534	0.107947	0.358306
	T = 6	0.0297913	0.108581	0.361994
	T = 8	0.0398018	0.111425	0.360386
	T = 10	0.0919443	0.109131	0.358593
(100, 18, 5) heavy flow arrival with small flow size	T = 1	0.377449	0.32771	0.358723
	T = 2	0.263501	0.327208	0.358788
	T = 4	0.30571	0.325895	0.357888
	T = 6	0.412023	0.328218	0.362129
	T = 8	0.496174	0.328878	0.359489
	T = 10	0.559625	0.326562	0.362606

In the second simulation, different HSD methods is compared under different flow arrive rate. The results as shown in Fig. 4. For all given λ, the proposed method achieves better performance in packet loss rate compared with DPB and SPHSD. When

$\lambda = 16$, the packet loss rate performance is improved by about 50% compared with DPB and about 60% compared with SPHSD.

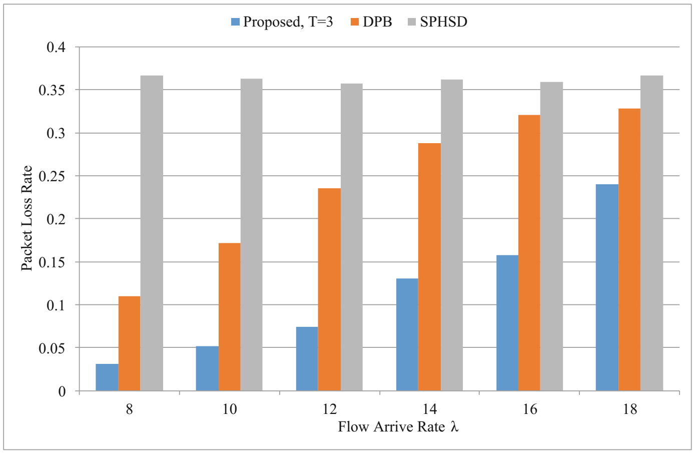

Fig. 4. Result of simulation-2, comparison of different HSD methods.

5 Conclusion

Today's most switches and routers employ Store-and-Forward architecture, and packets need to be stored in memory before sent to the destination ports. To simultaneously meet the high-speed and large-capacity requirements for packet buffers, this paper proposes an efficient parallel hybrid SRAM/DRAM architecture for high-bandwidth switches and routers. Tail SRAMs and head SRAMs are used for guaranteeing the middle DRAMs are accessed in a larger granularity. To improve the SRAM utilization, a simple yet efficient memory management algorithm is designed. When a flow arrives, memory space is dynamically allocated for it, and a hard timeout is assigned for the allocated queue. After the specific period, the memory space is freed, in order to accommodate the most recent active flows. A queueing system is used to model the proposed method, and theoretical analysis is performed to optimize the timeout value. Simulation results indicate that with a proper timeout value, packet loss rate can be significantly reduced compared with existing methods with the same SRAM capacity.

In the future work, we will develop a prototype of the proposed architecture and MMA based on Field Programmable Gate Array (FPGA) platform. Then further evaluate its performance using real traffic loads, making the proposal more practical.

Acknowledgements. This work was supported in part by the project of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory (KX152600010/ITD-U15001), the National Natural Science Foundation of China (61502204, 61306047), the Fundamental Research Funds for the Central Universities (JB140112), and the Qing Lan Project of Jiangsu.

References

1. Wang, F., Hamdi, M.: Memory subsystems in high-end routers. IEEE Micro **29**(3), 52–63 (2009)
2. Ganjali, Y., McKeown, N.: Update on buffer sizing in internet routers. ACM SIGCOMM Comput. Commun. Rev. **36**(5), 67–70 (2006)
3. Kreutz, D., Ramos, F.M.V., Verissimo, P.E., et al.: Software-Defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2014)
4. Kao, S.C., Lee, D.Y., Chen, T.S., Wu, A.Y.: Dynamically updatable ternary segmented aging bloom filter for OpenFlow-compliant low-power packet processing, IEEE/ACM Trans. Netw. **26**(2), 1004–1017 (2018)
5. Iyer, S., Kompella, R., Mckeown, N.: Analysis of a memory architecture for fast packet buffers. In: Proceedings of IEEE International Conference on High Performance Switching and Routing (HPSR), pp. 368–373. Dallas, TX, USA (2001)
6. Iyer, S., Kompella, R., McKeown, N.: Designing packet buffers for router linecards. IEEE/ACM Trans. Netw. **16**(3), 705–717 (2008)
7. Juniper E Series Router (2011). http://juniper.net/products/eseries/
8. Garcia, J., March, M., Cerda, L., Corbal, J., Valero, M.: A DRAM/SRAM memory scheme for fast packet buffers. IEEE Trans. Comput. **55**(5), 588–602 (2006)
9. Wang, F., Hamdi, M.: Scalable router memory architecture based on interleaved DRAM: analysis and numerical studies. In: Proceedings of IEEE International Conference on Communications (ICC), pp. 6380–6385. Glasgow, UK (2007)
10. Lin, D., Hamdi, M., Muppala, J.: Designing packet buffers using random round robin. In: Proceedings of IEEE Global Telecommunications Conference (GLOBECOM), pp. 1–5. Miami, FL, USA (2010)
11. Lin, D., Hamdi, M., Muppala, J.: Distributed packet buffers for high-bandwidth switches and routers. IEEE Trans. Parallel Distrib. Syst. **23**(7), 1178–1192 (2012)
12. Wang, F., Hamdi, M., Muppala, J.: Using parallel DRAM to scale router buffers. IEEE Trans. Parallel Distrib. Syst. **20**(5), 710–724 (2009)
13. Mutter, A.: A novel hybrid memory architecture with parallel DRAM for fast packet buffers. In: Proceedings of IEEE International Conference on High Performance Switching and Routing (HPSR), pp. 44–51. Richardson, TX, USA (2010)
14. Zhang, L., Lin, R., Xu, S., Wang, S.: AHTM: achieving efficient flow table utilization in software defined networks. In: Proceedings of IEEE Global Telecommunications Conference (GLOBECOM), pp. 1897–1902. Austin, TX, USA (2014)
15. Rai, I.A., Urvoy-Keller, G., Biersack, E.W.: Analysis of LAS scheduling for job size distributions with high variance. In: Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), pp. 2018–228. San Diego, CA, USA (2003)