Matching the Speed Gap between SRAM and DRAM

Feng Wang and Mounir Hamdi
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{fwang, hamdi}@cse.ust.hk

Abstract[1]— **With the constantly increasing Internet traffic, buffers are becoming major bottlenecks for today's high-end routers. In particular, router buffers are required to have both high speed and large capacities, which are hard to build with current single memory technology, such as SRAM or DRAM. A general approach is to make a combination of the SRAM and DRAM and exploit advantages from both. The main obstacle is to find a way matching the speed gap between them. And the requirement to maintain multiple flows in the system further complicates the problem.**

In this paper, we first investigate previous solutions that use different access *granularities* to match the speed gap. We point out their intrinsic scaling problems when the number of flows increases. Then, we propose to use *parallelism* to match the speed gap. Numerical studies and simulations both show that our proposal can theoretically support any number of flows in the router with just little SRAM under practical traffic. In addition, the memory management algorithm is also more scalable compared to those in previous solutions.

I. INTRODUCTION

A. Background

Current researches on high-performance routers mainly focus on efficient packet scheduling algorithms [1] or scalable switching architecture [2], implicitly assuming underlying fast-accessible buffers with enough capacities. However, in practice it is not an easy task to build a buffer with both *fast speed* and *large capacity*. With current available memory technologies [3], neither the SRAM nor the DRAM *alone* meets both requirements simultaneously. The SRAM is fast enough with an access time of around 4 *ns*, but it cannot be built with large capacities (SRAM's normal capacity is only a few *MB*) and is power-hungry as well. The DRAM can be built with a larger capacity, but its access time is too large, which is around 40 *ns*. This dilemma may be largely due to the fact that current memory technologies are mainly optimized for computers, not for routers. Extensive research has been carried out for computer buffers, where a portion of data/instructions in the memories will be considerably reused many times (normally referred to as the *locality* property [4]). Relatively less attention has been paid to router buffers, where data/packets are seldom reused and each one is treated equally in terms of processing. The locality property enables computers to work well using a *hierarchical* cache-based buffering system, which consists of small fast high-level memories (e.g., register and/or the SRAM) and large slow low-level buffers (e.g., DRAM and/or hard disks). However, the locality property does not hold for routers; each packet comes into the memory and leaves sometime afterwards, usually only once never to return.

A specific feature of router buffers imposes another difficulty. That is, router buffers are required to maintain multiple flow queues for incoming packets. This feature is required in at least the two following situations:

1. The wide adoption of input-queued (IQ) switches relies on the virtual-output-queuing (VOQ) technique, where an input buffer maintains N packet queues corresponding to N outputs [1].
2. Even in the output side buffers, in order for the router to provide quality-of-service (QoS) packets are normally arranged in their independent flow queues to reduce possible interference between each other [5].

Each queue itself may be as simple as first-come-first-served (FCFS). However, when the flow number increases, queue management becomes a challenging task for buffer designers.

We summarize the issues in the router buffer design as following: *The high-performance router buffers require both fast speed and large capacity, which are not met by current available single SRAM or DRAM. In addition, the router buffers are required to maintain multiple FCFS queues in the system.*

B. Related work

To simultaneously meet these two stringent requirements of high speed and large capacity, a natural idea is to combine the SRAM and DRAM and exploit the individual advantages from both. The general approach is to use the SRAM in the head and tail for fast reading and writing and the DRAM in the middle for majority buffering. Packets shuttle between the SRAM and DRAM under the control of some memory-management-algorithm (MMA). One key function of the MMA is to find ways to match the access speed gap between the SRAM and DRAM so that they can co-work smoothly.

Along this idea, the basic Hybrid SRAM/DRAM (HSD) architecture was first introduced by Iyer [6] and further explained in detail in [7]. Fig.1 shows the HSD system, where two small SRAM hold the head and tail of a flow queue and a DRAM maintains its middle part. All the SRAM and DRAM individually maintain Q separate flow queues. When a packet arrives at the HSD, it is immediately written to its flow queue in the tail SRAM, waiting for the MMA to transfer it to its corresponding queue in the DRAM. The key constraint of the MMA is that it always transfers a batch of b packets every time, *never smaller*, from a SRAM queue to the corresponding DRAM queue. Similarly, in the head the MMA always transfers a batch of b packets, *never smaller*, from a DRAM queue to the corresponding SRAM queue to fulfill the outside arbiter requests. These b packets should be from the same queue so that they can be accessed simultaneously.

The parameter b is normally configured to be the ratio of the access time of DRAM to that of SRAM, which is around 10 ($= 40ns / 4ns$) and can be regarded to be a constant with current technology. For the simplicity of analysis, we define one *time slot* to the access time of SRAM. Therefore, the access time to DRAM is b time slots and each transmission in the above MMA costs b time slots. In the HSD, the DRAM is always accessed with b packets from a same queue in b time

[1] This work was supported under Hong Kong RGC Grant HKUST6200/02E.

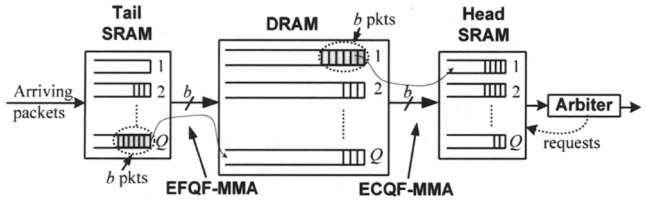

Fig. 1: The basic HSD architecture

slots. We can say that the DRAM's access *granularity* is b packets, while the SRAM's access *granularity* is one packet (one-by-one packets from outside). Basically, different access *granularities* to the SRAM and DRAM help match the access speed gap between them so that they can co-work smoothly.

Intuitively, since the MMA always transfers b packets in a batch, the SRAM should be sized to absorb the different granularities. In particular, the head SRAM should be sized to hold temporary packets that are not required by the outside arbiter but read out from the DRAM in batches. The tail SRAM should also be sized to hold packets that are still waiting to accumulate up to b packets. The authors in proposed an *earliest-critical-queue-first* (ECQF) MMA to schedule packets in the head SRAM. The ECQF-MMA uses a *look-ahead* scheme to replenish the queues. It waits and looks sufficiently ahead at many packet requests from the outsider arbiter, then combines this information with the packets already in the head SRAM and calculates which queue will become the first to be empty under the current sequence of requests. That queue is named *earliest critical* queue. The ECQF-MMA chooses the most critical queue and replenishes it by transferring b packets from the corresponding queue in the DRAM.

The HSD architecture is symmetric. A similar MMA can be used between the tail SRAM and the DRAM. The tail MMA just waits until an earliest queue has accumulated b packets and then transfers them in a batch to the DRAM queues. The tail MMA can be called *earliest-full-queue-first* (EFQF) MMA.

It is proved that the worst-case requirement for the SRAM is the capacity of holding $Q(b-1)$ packets and the possible delay a packet may experience is at most $Q(b-1)+1$ time slots under the continuously incoming traffic assumption.

C. Scaling problems of the HSD and our motivation

In general, router buffer scaling problem can be investigated in two dimensions: line rate scaling and flow number scaling. When the line rate is slow, simply using DRAM is enough. However, when the line rates scales, the faster SRAM has to be introduced, thus comes the HSD.

However, given current fabrication limitations on SRAM (a few *MB*) and a normal packet size of 1000 bytes, a simple calculation can show that the basic HSD system is only feasible with less than a thousand flows. Even with the interleaved DRAM banks, the HSD can hardly support flows over ten thousand. This flow number scaling limitation is simply due to the fact that the required SRAM in the HSD is $O(Qb)$, which increases *linearly* with the number of flows Q.

Another problem with scaling Q is the memory management algorithm. The ECQF- or EFQF-MMA in the HSD requires selecting the most critical/full queue, which involves sort-related operations. When Q increases, these selecting operations certainly become unfavorable in practical hardware implementations.

In this paper, we plan to design buffers supporting a large number of flows for next generation routers. Therefore, we concern these scalability issues foremost. In detail, we define the *scalability* problem as following:

1. The buffer design should minimize the SRAM requirement while providing reasonable performance guarantees to packets. In particular, the SRAM size is expected not to scale with Q.
2. The memory management algorithm (MMA) should be simple to implement and is expected not to scale with Q, since it has a limited time to be executed.

D. Paper Contributions

The rest of this paper is organized as follows. In section II, we show that it is HSD's *intrinsic* limitation that the required SRAM scales *linearly* with Q. We then propose to use parallelism to scale the HSD and call the resulted system PHSD. We analyzed the worst-case and average performance of PHSD, which outperform HSD significantly. We carry out extensive simulations in section IV and discussions in section V. Then we conclude the paper.

II. Intrinsic Limitations of the HSD

The authors in [6] derived the worst-case performance of the HSD, that is, the maximum SRAM requirement and maximum packet delay under any possible traffic conditions. In practice, however, we are more interested in its average performance, which can help us better understand the system. In this section, we put the HSD under practical traffic and analyze its average performance. By saying practical traffic, we make the following weak assumptions to the traffic:

1. *All the Q flows are independent of each other.*
2. *Each flow is a stationary and ergodic process.*

These are reasonable and practical assumptions, since in reality one flow source normally does not interfere with other sources. A large range of traffic conforms to the stationary and ergodic properties, such as uniform, hot-spot,..., and even the bursty traffic in the long term. Please also note that all the flows do not necessarily have identical distributions.

To analyze the SRAM requirement, we assume an infinite SRAM and perform numerical study on its average occupation. Focus on HSD's tail part in Fig.1. The EFQF-MMA works as following. The tail SRAM keeps receiving packets from outside and puts them in their flow queues. The EFQF-MMA waits until one flow queue has reached b packets, and then transfers the block of b packets as a whole from that queue to the DRAM via just one write operation. This operation costs b time slots.

Theorem 1: *In the HSD, the average occupation of the tail SRAM with EFQF-MMA is at least $Q(b-1)/2$, and the average packet delay in the head SRAM with ECQF-MMA is at least $Q(b-1)/2$ time slots, with traffic observing the above two practical assumptions.*

Proof: The tail SRAM should be sized to hold the residual packets that are still waiting for the EFQF-MMA for each of the Q flows. Suppose the EFQF-MMA has run for a sufficiently long time, and each flow i has $p_i (1 \le i \le Q)$ packets that have arrived to the system. Then, each p_i can be represented as following:

$$p_i = b \cdot m_i + q_i \ (0 \le q_i < b)$$

This representation tells us that for each flow i, there are *at least* q_i packets residing in the tail SRAM, since the EFQF-MMA only transfers packets in a batch of b packets.

Therefore, the tail SRAM should be sized *at least* to hold these $\sum_{i=1}^{Q} q_i$ residual packets.

Write $\sum_{i=1}^{Q} q_i$ in another way:

$$\sum_{i=1}^{Q} q_i = \sum_{i=1}^{Q} \mathrm{I}_A(q_i = 1) \cdot 1 + \sum_{i=1}^{Q} \mathrm{I}_A(q_i = 2) \cdot 2 + \ldots + \sum_{i=1}^{Q} \mathrm{I}_A(q_i = b-1) \cdot (b-1) = \sum_{j=1}^{b-1} \sum_{i=1}^{Q} \mathrm{I}_A(q_i = j) \cdot j \quad (*)$$

The $\mathrm{I}_A(\cdot)$ is an indicator function defined as following:

$$\mathrm{I}_A(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{if } x \text{ is false} \end{cases}$$

Each q_i ranges between $[0, b-1]$. Since all the flows are *independent* of each other and each flow is *stationary* and *ergodic*, all their residuals q_i should be uniformly distributed across the interval $[0, b-1]$, which indicates

$$\mathrm{E}[\sum_{i=1}^{Q} \mathrm{I}_A(q_i = j)] = Q/b, \forall j$$

Therefore, by taking expectations of both sides in (*), we can get:

$$\begin{aligned} \mathrm{E}[\sum_{i=1}^{Q} q_i] &= \sum_{j=1}^{b-1} \mathrm{E}[\sum_{i=1}^{Q} \mathrm{I}_A(q_i = j)] \cdot j \\ &= Q/b \cdot \sum_{j=1}^{b-1} j \\ &= Q/b \cdot (b \cdot (b-1))/2 \\ &= Q(b-1)/2 \end{aligned}$$

That is to say, the expected number of packets in the tail SRAM to initiate a transmission is *at least* $Q(b-1)/2$, which is half of its maximum boundary requirement.

Similarly, in the head SRAM the ECQF-MMA should look ahead expected *at least* $Q(b-1)/2$ packet requests to issue a transmission from the DRAM. Therefore, the average delay a packet may experience is *at least* $Q(b-1)/2$ time slots. ■

We can see that even with practical traffic, the SRAM occupation in the HSD still scales *linearly* with the number of flows Q. By carefully investigating the operations in the ECQF- and EFQF-MMA and the proof above, we can intuitively see that it is the different access granularities that build up the SRAM occupation. In particular, different access granularities make the HSD *non-work-conserving*. That is, packets may wait in the SRAM for the MMA to accumulate b packets in one queue. All flows may experience non-work-conserving and the SRAM should hold all the waiting packets for each flow. Consequently, the SRAM should be sized with a factor of the flows number, which is at least linear with Q.

III. The Parallel Hybrid SDRA/DRAM (PHSD) and the RRSD-MMA

We have seen in the previous section that although different granularities help the HSD match the access speed gap between SRAM and DRAM, they also make the system non-work-conserving. This fact poses an intrinsic limitation for the HSD's SRAM to scale *linearly* with Q. We break this non-work-conserving limitation in our proposed buffer architecture. In particular, we employ parallel DRAM to match up the speed gap and force the same access granularities in the MMA, i.e., one packet per access to the SRAM and one packet per access to the DRAM as well.

Parallelism is a natural approach used to fill in the speed gaps. However, for router buffers a key issue is to maintain as many as Q FCFS flows. In this section, we first describe the parallel hybrid SRAM/DRAM architecture for scaling router buffers. Then, we perform numerical studies on it, including the worst-case performance and expected performance under practical traffic conditions.

A. The PHSD and the RRSD-MMA

We use Fig.2 to illustrate the PHSD system. Basically, it consists of k ($\geq b$) parallel *subsystems*, each one being a hybrid SRAM/DRAM structure. However, there are still differences between the subsystems here and the basic HSD system: 1) In the subsystem in Fig.2, only the DRAM maintains Q FIFO flow queues, while each SRAM maintains just one single FIFO queue according to a packet's arriving order; 2) Packet transmission between the SRAM and DRAM is in *one-by-one* mode, not b packets in a batch as in the basic HSD system. That is to say, the access granularity of the DRAM is also *one*. We call this architecture parallel hybrid SRAM/DRAM, namely PHSD.

We design a RRSD-MMA for the PHSD system. To explain how it works, we focus on the tail part. It has two components: the *per-flow round-robin (RR) dispatcher* and the *SD transferor* between the SRAM and DRAM. They work as follows:

The per-flow round-robin (RR) dispatcher

When a packet comes to the memory system, the per-flow RR dispatcher first determines which subsystem the packet should go according its flow ID and the round-robin rule, and then sends it to the tail SRAM in the subsystem. The round-robin rule is following: if a packet is the i-th packet in a flow, then it should be dispatched into the j-th subsystem, where $j = i \bmod k$. The dispatcher finishes writing to a SRAM in just *one* time slot.

The SD transferor between the SRAM and DRAM

The SD transferor keeps transferring the head packet of the SRAM *one-by-one* into the corresponding flow queue in the DRAM whenever the SRAM is *non-empty*. The SD transferor completes a packet transfer in b time slots.

The PHSD architecture is symmetric. A similar MMA can be employed in the head part. If we view the continuous outside requests as *virtual* packets and buffer them in the head SRAM. Transferring packets from the DRAM to the head SRAM can be translated into transferring *virtual* packets from the head SRAM into the DRAM. It is the same as the tail MMA and shares the same performance analysis. Therefore, in the following, we only focus on analyzing the tail MMA unless otherwise stated.

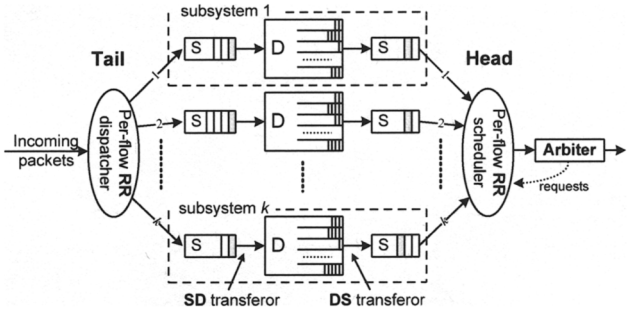

Fig. 2: The PHSD architecture

The PHSD is a very simple system. It maintains the flows queue information while breaks the non-work-conserving limitations as well. That is to say, whenever there are packets (requests) in the tail (head) SRAM, the SD (DS) transferor is busy transferring packets between the SRAM and DRAM. This will save a lot space in the SRAM.

In addition, the RRSD-MMA is as simple as having $O(1)$ complexity. This is because the PHSD is a distributed, asynchronous system and complexity is amortized into every packet. While the HSD system can be viewed as a centralized synchronous system, where a central MMA should determine which queue among Q flows can be selected and then transfer b packets from it synchronously.

B. Worst-case performance of the PHSD with RRSD-MMA

We first analyze the worst-case performance of the PHSD, i.e. the maximum SRAM requirement and packet delay under any traffic conditions.

We can note that the speed of the SD transferor is (b times) slower than that of the per-flow RR dispatcher. Although the long-term speed of the per-flow RR dispatcher feeding the subsystem should be divided by k, the SRAM should still be sized to absorb the bursty packets from different flows that simultaneously feed the same subsystem. Similarly in the PHSD head, the SRAM should be sized to buffer bursty requests from the outside arbiter and the SRAM occupation dictates the delay a packet request may experience.

We define an SRAM's *critical period* as following:

Definition: A time period is called a *critical period* for an SRAM, if in the beginning of this period the SRAM starts accumulating packets and does not ever become empty in this period. The critical period is measured by T time slots. T can be infinity if the SRAM occupation never becomes 0.

Theorem 2: *For an SRAM A, in any critical period of T, the occupation S in the SRAM satisfies the following:*

$$S \le Q(1-1/k) + T/k - T/b$$

Proof: In the period of T, there are at most T packets arriving at the per-flow RR dispatcher. They belong to Q flows individually. We assume that $q_i (1 \le i \le Q)$ packets belong to the i-th flow. Therefore

$$\sum_{i=1}^{Q} q_i = T$$

In particular, each q_i can always be represented by

$$q_i = n_i \cdot k - m_i (0 \le m_i \le k-1)$$

This form can tell us that *at most* n_i packets from the i-th flow will go to the SRAM A.

Combining the above two equations, we can get

$$\sum_{i=1}^{Q} n_i \cdot k - \sum_{i=1}^{Q} m_i = T$$

Therefore,

$$\sum_{i=1}^{Q} n_i = \frac{T + \sum_{i=1}^{Q} m_i}{k}$$

Since $0 \le m_i \le k-1$,

$$\sum_{i=1}^{Q} n_i \le \frac{T + Q(k-1)}{k} = Q(1-1/k) + T/k$$

According to the definition of the critical period, the SRAM A is non-empty during the T time slots, which indicates that the SD transferor has transferred T/b packets into the DRAM.

Therefore, the maximum occupancy of the SRAM A is:

$$S = \sum_{i=1}^{Q} n_i - T/b \le Q(1-1/k) + T/k - T/b$$

■

We can immediately find two properties from this theorem.

1. If $k < b$, $Q(1-1/k) + T/k - T/b$ can be infinite if T goes to infinity. This means that the occupancy S of SRAM A is unbounded.
2. If $k \ge b$, then $Q(1-1/k) + T/k - T/b \le Q(1-1/k)$. This means that the occupancy S of SRAM A is bounded by $Q(1-1/k)$.[2]

We can then have the following corollary.

Corollary: *In the PHSD with RRSD-MMA, if $k \ge b$, then the total SRAM size in the tail (head) is bounded by $Q(k-1)$ and the packet delay is bounded by $bQ(1-1/k)$.*

Proof: From above property 2, we can see that if $k \ge b$, each SRAM occupation is bounded by $Q(1-1/k)$. There are k SRAM in total. Therefore, the total SRAM required in the *tail* of the PHSD is:

$$k \cdot Q(1-1/k) = Q(k-1)$$

The same SRAM requirement holds in the *head* of PHSD.

For the packet delay analysis in the head, the maximum delay a packet request may experience happens when the outside scheduler issues the request to the head SRAM, and that request queues in the $Q(1-1/k)$ position. It will cost the DS transferor $b \cdot Q(1-1/k)$ time slots to service that request, since the DS transferor can only read out one packet in b time slots. Therefore, the maximum delay a request may have is $bQ(1-1/k)$ time slots.

■

To be comparable to the HSD, we set $k = b$. Therefore, in the PHSD, the total SRAM size in the tail (head) is bounded by $Q(b-1)$, and the request delay is also $Q(b-1)$, which are the same as those in the HSD.

We note here that his worst case can hardly happen; it requires all the Q flows synchronize perfectly (Q flows feeding the same SRAM simultaneously all the time), which is of very little possibility when Q is large. However, it is very difficult to exactly analyze the average performance of PHSD, we perform extensive simulations in the next section to test the SRAM occupation and packet delay. We can see that they are at the order of $O(\ln Q)$.

IV. SIMULATIONS

We perform extensive simulations to both the HSD and PHSD systems. We test the maximum SRAM occupation and average packet delays in both systems with regard to the in-

[2] To be accurate, when $k > b$, this upper bound $Q(1-1/k)$ cannot be achieved. The reason is that this upper bound $Q(1-1/k)$ can only be achieved when $T = 0$. However, if $T = 0$, the occupancy of SRAM A is 0. Nevertheless, $Q(1-1/k)$ serves as a good and sufficient upper bound.

creasing number of flows Q. SRAM occupation in the PHSD adds up from all the k SRAM. We also test both systems under different traffic load situations. In addition, we test the impact of increasing the parameter k in the PHSD system.

In all the simulations, we set $b = 10$ and Q ranges from 5 to 10000. We set $k = b$ in all simulations otherwise stated explicitly. All simulations run for 10^9 time slots. For each case of Q, we repeat the simulation for 100 times and take the maximum of all 'maximum SRAM occupation' and 'average packet delay'. The SRAM occupation is measured by the number of packets it holds. The packet delay is measured by time slots between the time when the packet is requested by outside arbiter and the time when it actually leaves the system. For the purpose of comparison, in every simulation we feed both the HSD and PHSD system with exactly the same traffic.

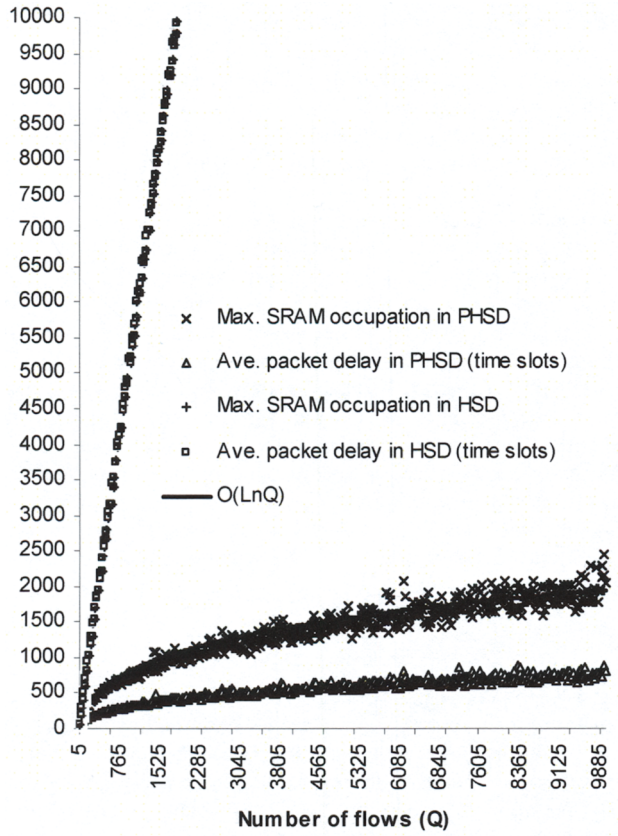

Fig.3: Performance of HSD and PHSD with increasing Q

A. SRAM occupation and packet delays

We first test both systems under uniform traffic and unbalanced traffic. For the uniform traffic, incoming packets are uniformly distributed across all the Q flows. For the unbalanced traffic, 90% traffic aggregates to only 10% of the flows. We have got nearly the same results for the uniform and unbalanced traffic, since they both conform to the *stationary* and *ergodic* properties. For the sake of limited space, we only show the results under uniform traffic in Fig.3.

We can see from the figure that in the HSD both the maximum SRAM occupation and average packets delay scale *linearly* with Q. The SRAM occupation quickly scale beyond 10000 packets when Q reaches 2000. With a normal packet

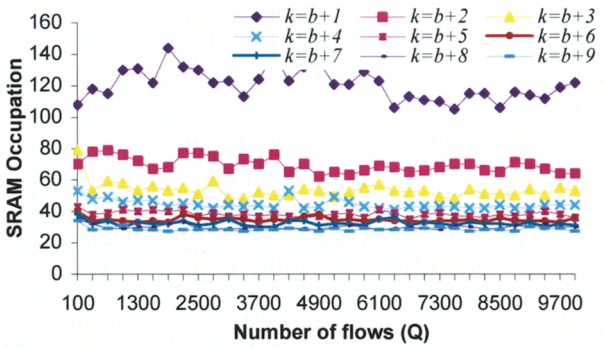

Fig. 4: Maximum SRAM occupation in PHSD when $k > b$

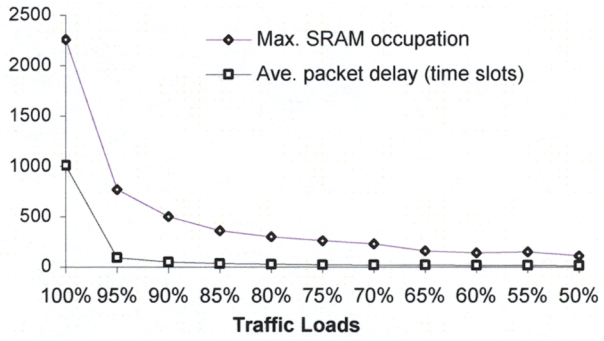

Fig.5: PHSD performance with decreasing traffic loads

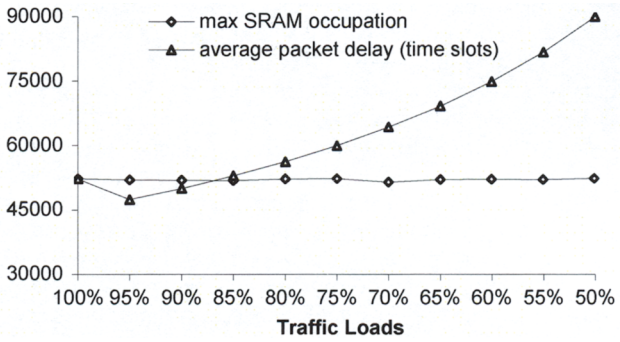

Fig.6: HSD performance with decreasing traffic loads

size of over 1000 bytes, a SRAM size of a few MB can hardly hold over 10000 packets. While in the PHSD, both the maximum SRAM occupation and average packet delay scale only with a $O(\ln Q)$ speed. Even when Q reaches 10000, the SRAM occupation is still under 3000 packets. This is the result with $k = b$.

B. Increasing k in the PHSD

We test the PHSD performance with increasing k. From the simulation results shown in Fig.4 and comparing them to those in Fig.3, we can find that the maximum SRAM occupation in the PHSD decreases dramatically when we increase k, even when k is only one larger than b. In addition, when $k > b$, the maximum SRAM occupation remains nearly unchanged when Q increases.

We can also find in Fig.4 that the marginal gain from further

increasing k gets smaller. This fact suggests that in practice it is enough to set k to be just one or two larger than b. This also gives incentives for network operators to minor over-provisioning.

C. Different traffic loads

We also test the performance of both systems under varying traffic loads. The simulation results are shown in Fig.5 and 6. Fig.5 shows that for the PHSD system, the SRAM occupation decreases significantly as the traffic loads decreases.

However, for the HSD system in Fig.6, the SRAM occupation is kept nearly unchanged even when the traffic load decreases to 50%. For the packet delays, they even surprisingly increase when the traffic load decreases. Intuitively, in light traffic loads, the EFQF- or ECQF-MMA still needs to wait to gather b packets in each flow queue to initiate a transmission. The lighter traffic does not necessarily decrease residual packets in each flow queue. Therefore, the SRAM occupation sustained by the HSD essentially does not change. Consequently, this fact also causes every packet tending to waiting in the SRAM for a longer time with a lighter traffic. Therefore, the average packet delay increases when the traffic load decreases.

V. Discussions

A. Other practical advantages of PHSD over HSD

When we design router buffers in literature, we normally assume the traffic load to be heavy. In fact, as also stated in [6], the HSD was designed under the assumption of continuously incoming packets for its MMA to work desirably. However, in practice, more valuable questions are: what if the traffic is light, or varying? We discuss about these two questions.

The PHSD removes the starvation problem in light traffic. It is easy to see a flow in the HSD may be starving if it has less than b packet requests in a certain period. This is due to that the DRAM access granularity in ECQF-MMA is fixed to be as large as b packets. While in the PHSD, the DRAM access granularity is one packet, which means that even one packet request in a flow can still initiate a transmission from the DRAM.

The PHSD responds promptly to varying traffic. We have seen from the simulations that the HSD performs poorly under light traffic loads. This is due to the fact that the ECQF-MMA is non-work-conserving, which makes it non-prompt to light traffics, or sluggish to busty traffic. On the other hand, the RRSD-MMA in PHSD is work-conserving and more prompt to light traffic. As we can see from Fig.6, the packet delay decreases significantly when the traffic load is below 95%. This is normal behavior of a network component, based on which network operator can improve the performance significantly by over-provisioning. While in the HSD system, there is no way to improve the performance by any over-provisioning.

B. Out-of-sequence problem in the PHSD

As we have seen above, besides significantly reducing the SRAM requirements, the PHSD is also more desirable in practice with varying traffic conditions.

However, these advantages do not come for free. They are at the cost of the possibility of packets 'out-of-sequence' problem. That is to say, for a flow, some packet may respond to the arbiter earlier than its preceding ones, since they are dispatched into different subsystems and each may experience different delays. While in the HSD, packets sequence is strictly maintained, since all packets have the same route to the final arbiter.

This is similar to the load-balanced switches proposed by Chang [8]. However, we argue that the out-of-sequence problem is much less severe in the PHSD. The load-balanced switches also employ round-robin to dispatch incoming packets, there are N sources of packets, each one with N VOQ. This fact makes the out-of-sequence order amount to $O(N^2)$. While in the PHSD system, there is only one source of packets and the worse-case out-of-sequence order is $O(Q)$ ($Q = N$ for the VOQ). From our analysis, we can see that in practice, the out-of-sequence order is only $O(\ln Q)$. A simple way to solve this problem is to use a sorting memory in the arbiter. Or, we can design alternative MMA to take considerations of packets order. However, that complicates the MMA. We will investigate this kind of MMA in our future work.

VI. Conclusions

We build high performance router buffers in this paper. In particular, we address the scaling problem in the buffers with regards to the increasing number of flows. It is proven by our numerical model that the HSD in literature is *intrinsically* unscalable to support large number of flows. We then propose the PHSD to scale it.

In summary, the PHSD is a *parallel*, *distributed*, *asynchronous*, and *work-conserving* system, while the HSD can be comparably regarded as a *sequential*, *centralized*, *synchronous*, and *non-work-conserving* system. From both the analysis and simulations, we show that the PHSD out-performs HSD significantly in terms of the SRAM requirement and packet delay.

In particular, when we set $k > b$ in the PHSD, the SRAM size and packet delay remains as a *small constant* under practical traffic, however large the Q is. This makes the PHSD able to virtually support any number of flows. We believe it is a promising solution for the next generation router buffers.

References

[1] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking,* vol. 7, pp. 188-201, 1999.

[2] F. Wang and M. Hamdi, "Analysis on the Central-stage Buffered Clos-network for packet switching," in *Proceedings of IEEE International Conference on Communications*, Seoul, Korea, 2005.

[3] Samsung semiconductor, Available online: "http://www.samsung.com/Products/Semiconductor/Products.htm"

[4] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*: Morgan Kaufmann, 2006.

[5] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in *Proceedings of ACM SIGCOMM*, 1995.

[6] S. Iyer, R. R. Kompella, and N. McKeown, "Analysis of a memory architecture for fast packet buffers," in *Proceedings of IEEE Workshop on High Performance Switching and Routing*, 2001, pp. 368-373.

[7] S. Iyer, R. Kompella, and N. McKeown, "Designing Buffers for Router Line Cards," *Technical Report TR02-HPNG-031001, Stanford University,* 2002.

[8] C. S. Chang, D. S. Lee, and Y. S. Jou, "Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering," *Computer Communications,* vol. 25, pp. 611-622, 2002.