Using Parallel DRAM to Scale Router Buffers

Feng Wang, *Student Member*, *IEEE*, Mounir Hamdi, *Senior Member*, *IEEE*, and Jogesh K. Muppala, *Senior Member*, *IEEE*

Abstract—This paper addresses the design of high-performance buffers for high-end Internet routers. The buffers are typically implemented using a combination of SRAM and DRAM technologies in order to simultaneously meet the routers' high speed and capacity requirements. The major challenge in designing router buffers is to maintain multiple flow queues in the memory, unlike computer memory buffers (i.e., memory system). The major objective is to minimize the use of expensive but fast SRAM while providing acceptable delay guarantees to packets. In this paper, we first investigate hybrid SRAM/DRAM solutions proposed in the past. We show that one of the architectural limitations of these solutions is that the required SRAM size grows linearly with the number of flows in the system. This prevents the solutions from scaling to support a large number of flows. We then break down this shortcoming by proposing a parallel hybrid SRAM/DRAM (PHSD) architecture. We design a series of memory management algorithms (MMAs) for PHSD, based on tradeoffs between the complexity of the MMAs and the guarantee of in-order delivery of packets (segmentations). We perform a detailed analysis of the proposed algorithms and conduct extensive simulations to show that PHSD can significantly outperform solutions proposed in the past in terms of the SRAM requirements and packet delay.

Index Terms—Router memory, SRAM/DRAM, packet scheduling.

1 Introduction

Packet switches/routers deployed in the Internet typically contain buffers, which temporarily hold packets that cannot be sent out immediately on an outgoing link. The buffers are especially useful in dealing with temporary congestion in the Internet. In general, the *speed* and *size* of the packet buffers (typically determined by the electronic memory technology used to implement them) have a significant impact on the switch performance [1], [2]. The memory speed is normally defined to be the reciprocal of its *access time*.

It has been reported that [3], [4], [5], with the ever increasing Internet line rate, current memory technologies available, viz., SRAM or DRAM alone cannot *simultaneously* satisfy both the router buffer's speed and size requirements. While SRAM is fast enough with an access time of around 4 ns, its largest size is limited by current fabrication technologies to only a few megabytes and also consumes a large amount of power. On the other hand, DRAM can be built with large capacity, but its typical memory access time[1] is too large, around 40 ns. This discrepancy in performance 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 on designing computer buffers, where a portion of data/instructions in the memories will be reused many times (normally referred to as the *locality* property [6]). Relatively less attention has been paid to router buffers, where data/packets are seldom reused and each packet 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 low-speed 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 thereafter, usually only once never to return.

To simultaneously meet the stringent speed and size requirements, researchers proposed the implementation of the router buffer as a combination of SRAM and DRAM in order to exploit the individual advantages of both the memory technologies. The basic idea is to use the SRAM in the *head* and *tail* for fast reading and writing and DRAM in the middle for buffering the majority of packets. Packets shuttle between the SRAM and DRAM under the control of a memory management algorithm (MMA). One difficulty of the MMA is to find a way to match the access speed gap between the SRAM and DRAM so that they can work together smoothly to meet the external speed requirements and provide sufficient buffering capacity. A generic model of such a combined SRAM/DRAM buffer implementation is shown in Fig. 1.

What makes the router buffer even harder to build is the fact that a typical router contains multiple flow queues. For example, input-queued (IQ) switches rely on virtual-output-queuing (VOQ) technique where an input buffer maintains N first-in, first-out (FIFO) queues corresponding to N outputs [7]. Even in output-queued (OQ) switches separate queues are maintained for different flows in order to support quality-of-service (QoS) requirements of the flows [8]. Each flow

1. Note that the access time of DRAM cannot be confused with its bandwidth. Access to the first byte of a block in DRAM usually has a long latency. Access to the subsequent bytes of the same row can be as fast as the SRAM. Therefore, we assume in this paper that accessing a packet in DRAM will cost 40 ns, no matter what the size of the packet is.

• *The authors are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.*
E-mail: {fwang, hamdi, muppala}@cse.ust.hk.

Manuscript received 23 Apr. 2008; revised 13 Aug. 2008; accepted 15 Aug. 2008; published online 21 Aug. 2008.
Recommended for acceptance by Y. Pan.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-04-0146.
Digital Object Identifier no. 10.1109/TPDS.2008.162.

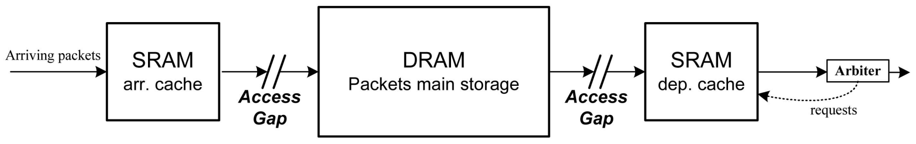


Fig. 1. A generic SRAM/DRAM combination for packet buffers.

queue may be as simple as FIFO. Queues management, however, becomes a challenging task for buffer designers when the number of flows increases.

1.1 Problem Statement

Our primary approach to router buffer design is to employ a combination of SRAM and DRAM together with an MMA in such a way that the buffers can meet both the speed and capacity requirements, while supporting multiple flow queues. The MMA should be able to successfully match the access speed gap between the SRAM and DRAM. The major *objective* of the design is to minimize the size of the expensive and power-hungry SRAM while providing reasonable performance (e.g., delay guarantees to packets).

We focus on the buffer's ability to scale with increasing number of flows. Ideally, the *scalability* of a combined SRAM/DRAM solution should have the following properties:

1. The SRAM size and packet delay in the system should not increase or increase slowly with the increase in the number of flows.
2. The MMA should have low complexity so as not to increase significantly with the increase in the number of flows.

1.2 Conventions

We define the two basic parameters in the router buffer design as follows:

1. b—The ratio of the DRAM access time to the SRAM access time. To simplify the analysis, we also set the access time of the SRAM as one time slot. Then, the access time of the DRAM is b time slots.
2. Q—The number of flows supported in the buffer system.

In practice, b can be regarded as a constant, around 10, but Q might range from hundreds to millions.

For the SRAM size analysis, we measure the SRAM size in terms of the number of packets it can hold rather than the actual size in bits. It is common practice for high-end routers to schedule fixed-length packets across their switch fabric. High-end routers normally employ Segmentation and Reassembly (SAR) modules in the line cards. Variable length packets are segmented as they arrive, buffered, and scheduled across the switch as fixed-length packets, and then reassembled back into original packets before they depart. The segmentation size is decided by the configuration of the switch fabric. Our memory architecture is mainly built for maintaining flow queues for the scheduler of switch fabric. The segmentation and reassembly overhead is usually taken by the SAR modules. It is easy to convert the number of packets (segmentations) to the memory size if packets (segmentations) are in fixed length.

1.3 Paper Organization

The rest of this paper is organized as follows: In Section 2, we briefly introduce the related work in literature in building high-performance packet buffers. We then show their architectural scalability limitations analytically in Section 3. We then propose a new buffering architecture (namely, parallel hybrid SRAM/DRAM (PHSD)) eliminating these limitations and design a primary MMA for it in Section 4. Its performance is studied in Section 5. In Section 6, we present a solution to the out-of-order problem in the primary MMA by designing a series of new MMAs, which show the tradeoffs between the performance and algorithm complexity. Simulations are carried out to verify these results in Section 7. We present some discussions in Section 8 and then conclude this paper in Section 9.

2 Related Work

The basic Hybrid SRAM/DRAM (HSD) architecture was first introduced by Iyer et al. [5] and further explained in detail in [9]. Fig. 2 shows the HSD, where two SRAMs hold heads and tails of all the flow queues and a DRAM maintains the middle part of the queues. Both the SRAM and DRAM individually maintain Q separate flow queues. When a packet arrives at HSD, it is first 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 function of the MMA is that it always transfers a block of b packets every time, *never smaller*, from an SRAM queue to the

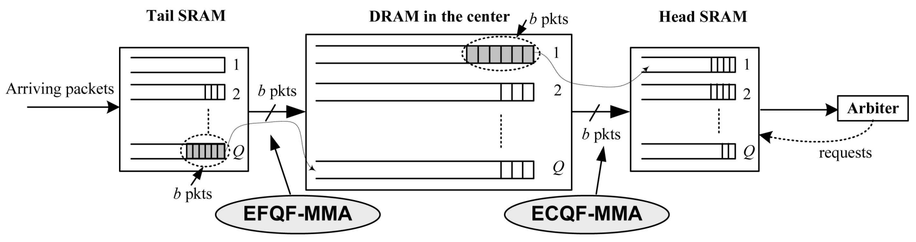


Fig. 2. The basic HSD architecture for packet buffers maintaining Q flows.

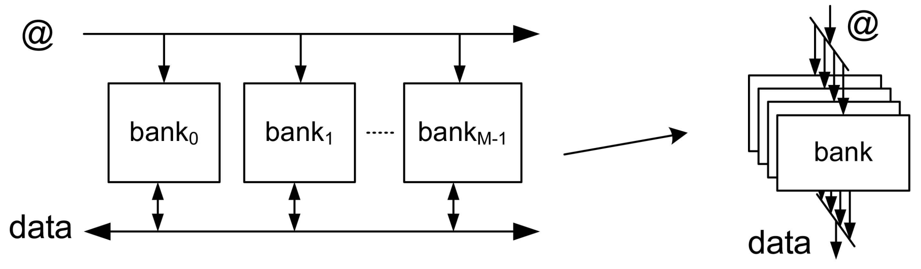


Fig. 3. M memory banks are interleaved to reduce the DRAM's effective access time.

corresponding DRAM queue. Similarly, in the head, the MMA always transfers a block of b packets, *never smaller*, from a DRAM queue to the corresponding SRAM queue to fulfill the output arbiter requests. These b packets should be from the same queue so that they can be accessed simultaneously. Each transmission costs b time slots. In other words, the DRAM is always accessed by b packets from the same queue in b time slots. We can see that the DRAM is accessed in a larger granularity, b packets per operation, while the SRAM's access granularity is one packet (one packet per operation from outside). *Larger access granularity to the DRAM helps it overcome the speed gap with the SRAM so that they can work together smoothly.*

Intuitively, since the MMA always transfers b packets in a batch, the SRAM should be sized to absorb DRAM's larger granularity. The head SRAM should be sized to temporarily hold packets that are not required by the output 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 proposed an *earliest-critical-queue-first* (ECQF) MMA, which uses a *look-ahead* scheme to replenish the queues in the head SRAM. The ECQF-MMA waits and looks sufficiently ahead at many packet requests from the output arbiter, then combines this information with the packets already in the head SRAM and calculates which queue will be the first to become empty under the current sequence of requests. This queue is named the *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. Since HSD is symmetric, a similar MMA can be performed between the tail SRAM and the DRAM. The tail MMA just waits until a flow queue has *first* accumulated b packets and then transfers them in a batch to the corresponding DRAM queue. The MMA in the tail SRAM can thus be called *earliest-full-queue-first* (EFQF) MMA.

It is proved in that the worst-case size required for the SRAM, both at the tail and head, is $Q(b-1)$ packets and the possible delay a packet may experience is bounded by $Q(b-1)+1$ time slots under the continuously incoming traffic assumption.

2.1 The HSD with Interleaved DRAM Banks

The basic HSD is the first attempt to build router buffers with a combination of SRAM and DRAM. As we can see from above, however, both the SRAM size and packet delay amounts to $O(Qb)$. To reduce the SRAM size requirement or to make HSD capable of supporting more flows under a limited SRAM size budget, Garcia-Vidal et al. [3] tried to reduce the DRAM's access time, b. In particular, they redesigned the DRAM part by using interleaved DRAM banks. They showed that the effective DRAM access time can be reduced by overlapping multiple accesses to interleaved DRAM banks. For example, as shown in Fig. 3, if using M interleaved banks, the effective DRAM access time can possibly be reduced to b/M, thus reducing the SRAM size to $O(Qb/M)$. The key challenge there is to design a DRAM bank management algorithm to deal with bank conflicts while the accesses are in flight. The authors used an *issue-queue-like* mechanism to avoid the bank conflicts.

Both their analysis and simulations showed that their improved HSD system can support thousands of queues for line rates up to 160 Gbps.

3 The Scaling Limitations of HSD

In general, the router buffer's scaling problem can be addressed in two dimensions: *line rate scaling* and *flow number scaling*. The HSD design was targeted at addressing the line rate scaling problem, at the cost of possible packets pipeline delay. When the line rate is low, simply using DRAM is sufficient for routers. As the line rates scale, SRAM should be introduced to keep up with the increasing line rate. The design of HSD, however, did not consider scaling with the flow number Q as well. In particular, as shown above, the SRAM size in HSD scales *linearly* with Q. Given current SRAM fabrication limitations (a few megabytes) and a normal packet size of 1,000 bytes, a simple calculation shows that HSD can only work well with $Q < 1,000$. Even with the interleaved DRAM, the improved HSD can support no more than 10,000 flows. In this section, we show that it is the *intrinsic* limitation for HSD to scale *linearly* with Q.

The authors in [5] derived the worst-case performance of HSD, that is, the maximum SRAM required and maximum packet delay under any possible traffic conditions. However, in practice, we are more concerned with its performance under practical traffic conditions, which can help us better understand the system. By *practical* traffic, we mean the following weak assumptions about 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 a flow source normally does not interfere with other sources. Furthermore, the stationary and ergodic properties are shown in nearly all types of practical traffic, such as uniform, hotspot, diagonal, and even bursty traffic in the long term. We should also note that all the flows do not necessarily have identical distributions.

To analyze the size of the SRAM required under this practical traffic, we assume an unlimited SRAM size and derive its expected occupancy. Focus on the tail SRAM in HSD. The EFQF-MMA keeps receiving packets from outside and puts them in their individual flow queues in the tail SRAM. The EFQF-MMA waits until a queue has accumulated packets and then transfers the block of packets as a whole from that queue to the DRAM. Based on this behavior, we now derive some properties of the HSD system.

Theorem 1. *The expected occupancy of the tail SRAM in HSD with EFQF-MMA is at least $Q(b-1)/2$, with incoming traffic conforming to the aforementioned two assumptions.*

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

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

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 SRAM should be large enough *at least* to hold these $\sum_{i=1}^{Q} q_i$ residual packets.

By rewriting $\sum_{i=1}^{Q} q_i$, we obtain

$$\begin{aligned}\sum_{i=1}^{Q} q_i &= \sum_{i=1}^{Q} \mathbf{I_A}(q_i = 1) \cdot 1 + \sum_{i=1}^{Q} \mathbf{I_A}(q_i = 2) \cdot 2 + \cdots \\ &\quad + \sum_{i=1}^{Q} \mathbf{I_A}(q_i = b-1) \cdot (b-1) \qquad (1) \\ &= \sum_{j=1}^{b-1} \sum_{i=1}^{Q} \mathbf{I_A}(q_i = j) \cdot j.\end{aligned}$$

Here, $\mathbf{I_A}(\cdot)$ is an indicator function defined as follows:

$$\mathbf{I_A}(\mathbf{x}) = \begin{cases} 1 & \text{if } x \text{ is true,} \\ 0 & \text{if } x \text{ is false.} \end{cases}$$

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

$$\mathbf{E}\left[\sum_{i=1}^{Q} \mathbf{I_A}(q_i = j)\right] = Q/b, (0 \leq j < b).$$

Therefore, by taking expectations on both sides in (1), we obtain

$$\begin{aligned}\sum_{i=1}^{Q} q_i &= \sum_{j=1}^{b-1} \mathbf{E}\left[\sum_{i=1}^{Q} \mathbf{I_A}(q_i = j)\right] \cdot j \\ &= Q/b \cdot \sum_{j=1}^{b-1} j \\ &= Q/b \cdot (b \cdot (b-1))/2 \\ &= Q(b-1)/2.\end{aligned}$$

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 just *half* of its worst-case requirement. □

Corollary 1. *In HSD, the expected packet delay in the head SRAM with ECQF-MMA is at least $Q(b-1)/2$ time slots, with outgoing traffic conforming to the aforementioned two assumptions.*

Proof. Using similar analysis, we can see that in expectation the ECQF-MMA in the head SRAM looks ahead *at least* $Q(b-1)/2$ packet requests to issue a transmission from the DRAM. Therefore, the expected delay a packet may experience is at least $Q(b-1)/2$ time slots. □

From above analysis, we can see that under practical traffic, both the expected SRAM occupancy and packet delay in HSD scale *linearly* with Q. Even with the interleaved DRAM banks, this linearity does not disappear. It shows to be an intrinsic limitation for HSD with ECQF/EFQF-MMA to scale with the increasing number of flows. Our first task is to remove this *linear* dependency on Q.

Another problem to be considered is about the MMA. The ECQF/EFQF-MMA in HSD requires selecting the most critical or full queue, which involves *sort*-related operations. When Q increases, these sort operations quickly become unfavorable for practical hardware implementations.

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

By carefully investigating the operations of the ECQF/EFQF-MMA and the proof above, we can intuitively see that the *larger* access granularity of the DRAM causes the SRAM occupancy to increase and consequently increases packet delay. Specifically, the larger access granularity of the DRAM makes HSD *non-work-conserving*. That is, packets/requests may wait for the MMA to accumulate b packets/requests in one queue. All flows may experience non-work-conserving behavior and the SRAM should hold all the waiting packets/requests for each flow. Consequently, the SRAM size or packet delay should be proportional to the number of flows, which is at least linear with Q.

4.1 The Parallel Hybrid SRAM/DRAM Architecture

We try to remove this disadvantage, viz., the non-work-conserving property of HSD by employing PHSD architecture. In PHSD, both the SRAM and DRAM have the same access granularity—one packet per operation, so that packets do not wait unnecessarily. To match the access speed gap between the SRAM and DRAM, we employ round-robin dispatchers to divert the traffic from individual flows to different queues.

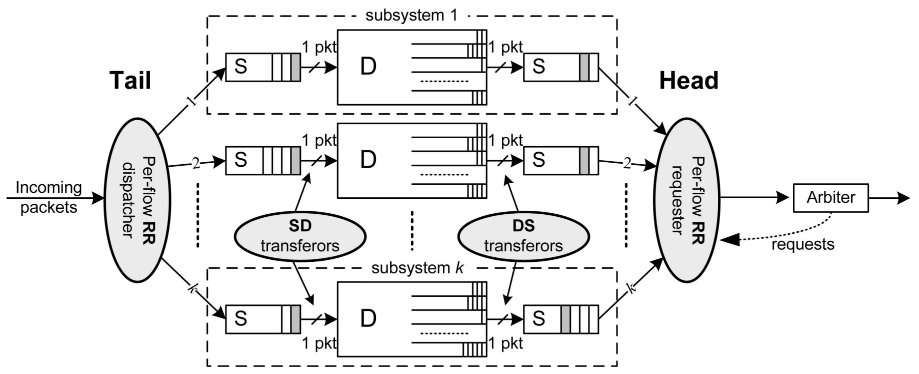


Fig. 4. The PHSD system.

We use Fig. 4 to illustrate the PHSD system. Basically, it consists of $k (k \geq b)$ parallel *subsystems*, each being a combined SRAM/DRAM structure. We must point out the differences between the subsystems here and the basic HSD:

1. In the subsystem in PHSD, only the DRAM maintains Q FIFO flow queues, while each SRAM maintains just a single FIFO queue for all flows;
2. Packet transmission between the SRAM and DRAM is one by one, not b packets in a batch as in the basic HSD system. Thus, the access granularity of the DRAM is also one packet.

4.2 The RRSD-MMA

We first design a very simple yet efficient MMA on PHSD. We focus on the tail part of the MMA. As shown in the figure, the MMA consists of two components: the *per-flow round-robin* (RR) packet dispatcher and the *SRAM to DRAM (SD) transferor* in each subsystem. Therefore, we call this MMA as RRSD-MMA.

ALGORITHM RRSD-MMA

The RR and SD components work independently:

RR—the per-flow round-robin packet dispatcher

When a packet arrives at the memory system, the RR dispatcher simply adds it to a subsystem according its flow ID and the round-robin rule. For example, if a packet is the $i\text{th}$ packet in a flow, then it should be dispatched into subsystem j, where $j = i \bmod k$. The dispatcher completes writing a packet into an SRAM in just *one* time slot.

SD—the SD transferor between the SRAM and DRAM

In each subsystem, the SD keeps transferring the head-of-the-queue packet of the SRAM *one by one* into the corresponding flow queue in the DRAM whenever the SRAM is *nonempty*. The SD completes a packet transfer in b time slots.

PHSD is symmetric. A similar MMA can be employed in PHSD head. If we view the arbiter requests as *virtual* packets and being dispatched into the k head SRAMs in a per-flow round-robin fashion, then transferring packets from the DRAM to the head SRAM can be viewed as transferring *virtual* packets from the head SRAM to the DRAM. The requests (virtual packets) are queued in the end of the flow queues in the head SRAM and get replenished in a first-queued-first-served way. It is obvious to see that they are mirror operations of the RRSD-MMA in the tail, and therefore, the head MMA shares the same performance analysis. In the following, we mainly focus on analyzing the tail SRAM unless otherwise stated.

5 Performance Analysis for PHSD with RRSD-MMA

It is easy to see that the RRSD-MMA is quite simple and its time complexity is $O(1)$ since the RR only has to decide which subsystem the packet should go to and then dispatch it. The SD is work conserving and does not involve sorting or searching in the SRAM.

To analyze the SRAM size requirements for PHSD with RRSD-MMA, we note that the speed of the SD is (b times) slower than that of the RR. Although the long-term speed of the RR feeding the subsystem should be divided by k, the SRAM should still be sized to absorb the possible bursty packets from different flows that might simultaneously feed the same subsystem. In this part, we first analyze the worst-case SRAM size for PHSD and then investigate the expected SRAM occupancy under practical traffic conditions.

5.1 The Worst-Case SRAM Size and Packet Delay in PHSD

To analyze the performance of PHSD, we should first have a definition of *critical period* for an SRAM.

Definition 1 (critical period). *A time period is called a critical period for an SRAM, if in this period the SRAM does not ever become empty. The length of the critical period is represented as T time slots. T may be infinity if the SRAM is never emptied.*

Theorem 2. *In PHSD with RRSD-MMA, for any tail SRAM A, in any of its critical period T, the occupancy S in A satisfies the following equation: $S \leq Q(1 - 1/k) + T/k - T/b$.*

Proof. In the critical period T, there are at most T packets arriving to the system. They belong to Q flows individually. We assume that $q_i (q \leq i \leq Q)$ packets belong to the $i\text{th}$ flow.

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

In particular, each q_i can always be decomposed into $q_i = n_i \cdot k - m_i (0 \leq m_i \leq k - 1)$.

This form can tell us that at most n_i packets from the ith flow will go to the SRAM A.

Combining the above two equations, we can obtain

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

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 critical period, the SRAM A is nonempty during the T time slots. Therefore, the SD transferred T/b packets into the DRAM.

Therefore, the maximum occupancy of the SRAM is

$$\begin{aligned} S &= \sum_{i=1}^{Q} n_i - T/b, \\ &\le Q(1-1/k) + T/k - T/b. \end{aligned}$$

□

From this theorem, we can immediately find the following two facts:

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 $k > b$, $Q(1-1/k) + T/k - T/b \le Q(1-1/k)$. This means that the occupancy S of SRAM A is bounded by $Q(1-1/k)$.[2]

Further, we can derive the following corollary from fact 2.

Corollary 2. *In PHSD with RRSD-MMA, if $k \ge b$, then the total tail SRAM size in all k subsystems is bounded by $Q(k-1)$ and the packet request delay is bounded by $bQ(1-1/k)$.*

Proof. *There are k tail SRAMs in total and each SRAM size is bounded by $Q(1-1/k)$ from the above fact 2. Therefore, the total maximum SRAM size in the tail of PHSD is*

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

For the packet request delay analysis, the maximum delay happens when the arbiter issues a request to the head SRAM, the request queues in the $Q(1-1/k)$ position. It will cost that MMA $b \cdot Q(1-1/k)$ time slots to service that request, since it costs b time slots for the MMA to read out a packet from the DRAM. Therefore, the maximum delay a request may experience is $bQ(1-1/k)$ time slots. □

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

5.2 The Expected Performance of PHSD under Practical Traffic Conditions

We have seen that the worst-case performance of PHSD with RRSD-MMA is the same as HSD with ECQF/EFQF-MMA. So, besides the reduced MMA complexities, what else does PHSD gain from employing the parallelism? We show here that the expected performance of PHSD *significantly outperforms* that of HSD under practical traffic conditions.

Intuitively, PHSD is a fully distributed and asynchronous system, and it breaks down the scaling limitations in HSD by smaller access granularity to the DRAM. The RRSD-MMA is *work conserving*, which means that whenever an SRAM is nonempty, the RRSD-MMA is able to transfer packets in PHSD. On the contrary, always waiting for b packets makes the ECQF/EFQF-MMA non-work conserving in HSD. This means that HSD needs more SRAM to hold waiting packets, while PHSD can keep the SRAM as small as possible.

However, it is very difficult to analyze the *exact* performance of the RRSD-MMA under practical traffic conditions. In general, it cannot be modeled using any stochastic process since there is a deterministic indicator function in it—the round-robin process. We resort to a *heuristic* method here, which gives rather good insights into the behavior of PHSD.

We focus on one tail SRAM A in a subsystem and see how many packets it can accumulate. We start the analysis from simple situations with fewer flows. For example, if there is only one flow in the system, the SRAM A can only accumulate at most one packet since the flow is dispatched into k subsystems in a *round-robin* way and $k \ge b$. If a second flow comes to the PHSD system, A might accumulate two packets if the two flows are feeding A simultaneously; otherwise, A can only accumulate one packet. The first situation is less likely to happen since there are k subsystems and the two flows are independent. Continuing to add more flows sequentially to the PHSD system, we can see that the possibility that they are synchronized to A is very small since they are all independent. In other words, each new flow has a marginal bursty effect on A's occupancy. However, this marginal bursty effect decreases as the number of flows in PHSD increases. The aggregated burstiness builds up the occupancy of the SRAM.

In particular, we define the burstiness of a newly added flow as follows:

Definition 2 (burstiness). *Assume there are already $q-1$ flows in PHSD. When adding the qth new flow, the new flow adds a burstiness of $1/k - (q-1)/qb$ to each of the k SRAMs.*

We elaborate on this definition below:

1. The contribution of the newly added flow's burstiness to PHSD is $1/k$, since the flow feeds the k SRAMs in a round-robin fashion.
2. When there are $q-1$ flows already in the PHSD system, adding a new flow will decrease the burstiness of each of these $q-1$ flows by $1/q$. This

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 surely 0. Anyway, $Q(1-1/k)$ serves as a good and sufficient upper bound.

is because packets from the newly added flow interleave into the existing $q-1$ flows and thus decrease their effective speed to $(q-1)/q$ of their original ones. This fact allows the SD to virtually gain an additional $(q-1)/q \cdot (1/b) = (q-1)/qb$ speed, since the original SD transferor speed is $1/b$.

In summary, the qth flow adds a burstiness of $1/k - (q-1)/qb$ to each of the k SRAMs.

Note that this definition of burstiness is only of the *incremental* effect of the flow. It cannot be applied to the individual flows that are already in the system. Accordingly, we take an incremental approach to analyze the SRAM occupancy. That is, for an equilibrium PHSD system with Q flows, we assume it has evolved from the state of having $1, 2, \ldots, Q-1$ flows, each state being an equilibrium. It is certainly not a practical assumption. However, it is valid enough to analyze the equilibrium state.

Based on this definition, we introduce the following two propositions.

Proposition 1. *When $k = b$, the expected SRAM occupancy and packet delay in PHSD with RRSD-MMA scales in a speed of $O(\ln Q)$ under practical traffic conditions.*

Proposition 2. *When $k > b$, the expected SRAM occupancy and packet delay in PHSD with RRSD-MMA does not scale with Q under practical traffic conditions.*

Proof. Using the incremental analysis, we aggregate all Q flows' burstiness as follows:

$$\begin{aligned}\sum_{q=1}^{Q}\left(\frac{1}{k}-\frac{q-1}{qb}\right) &= \sum_{q=1}^{Q}\left(\frac{1}{k}-\frac{1}{b}+\frac{1}{qb}\right)\\ &= \left(\frac{1}{k}-\frac{1}{b}\right)\cdot Q+\frac{1}{b}\cdot\sum_{q=1}^{Q}\frac{1}{q}\\ &= \left(\frac{1}{k}-\frac{1}{b}\right)\cdot Q+\frac{1}{b}\cdot\mathbf{H_Q}\\ &= \left(\frac{1}{k}-\frac{1}{b}\right)\cdot Q+\frac{1}{b}\cdot(\ln Q+o(1)).\end{aligned} \tag{2}$$

Refer to the footnote for the details of the Harmonic function $\mathbf{H_Q}$.[3]

We can see from (2) that

1. When $k = b$, the first part in (2) is 0. Then, the aggregated burstiness in the system is $\frac{\ln Q}{b}$. Since b is a constant, the SRAM occupancy should then scale with $O(\ln Q)$. This supports Proposition 1.
2. When $k > b$ and Q is sufficiently large, the first part in (2) becomes over negative compared with the second part, and hence, the aggregated burstiness is less than 0. That is to say, the SRAM occupancy cannot accumulate when Q is over some threshold, which means that the SRAM occupancy does not scale with Q. This supports Proposition 2.

A similar analysis can be performed for the head MMA. Using the concept of virtual packets, we can conclude that in PHSD with RRSD-MMA the packet delay scales with $O(\ln Q)$ when $k = b$ and does not scale with Q when $k > b$. □

3. The Harmonic function $\mathbf{H_Q}$ is a summation of series $\sum_{q=1}^{Q}\frac{1}{q}$. Its accurate representation is $\mathbf{H_Q} = \gamma + \ln Q + \frac{1}{2}Q^{-1} - \frac{1}{12}Q^{-2} + \frac{1}{120}Q^{-4} + O(Q^{-6})$, where γ is a constant of $0.57721\ldots$

The above analysis shows that unlike the HSD's *linear* scaling with Q, PHSD can distribute the traffic into k parallel subsystems and reduce both the SRAM size and packet delay to an $O(\ln Q)$ scaling speed. Furthermore, if we employ more than b subsystems in PHSD, the average SRAM occupancy and packet delay can be made independent of Q. We verify these two properties by extensive simulations later in this paper.

6 Modifying the PHSD Scheme to Maintain Packets' Order

6.1 The Out-of-Order Problem in PHSD with RRSD-MMA

As we have seen above, PHSD can significantly reduce the SRAM size and packet delays under practical traffic with very simple RRSD-MMA. These advantages, however, come at a small price, the possibility of packets being *out of order*. The out-of-order problem can occur inside the PHSD due to two reasons. First, since large packets are segmented into fixed-length segments and buffered in different DRAM, their relative order might be altered. Second, the order of whole packets in a flow might also be altered after reassembly. We note here that the first type of out-of-order problem is of specific interest to us in this paper. The out-of-order problem caused due to segmentation will result in unnecessary delays or losses of packets inside the router. The packets order in a flow (e.g., TCP) is not a fundamental requirement in the Internet. However, when designing routers, we normally try to keep the packet order inside the router [10], [11], if this can be achieved at a reasonable cost even though it may not be a strict requirement. This would prove beneficial for more advanced feature of routers, such as QoS and various AQM techniques.

In general, we formulate the out-of-order problem as follows. The notation $p(f_i, n)$ represents the nth packet in flow f_i.

Definition 3 (out of order). *For any flow f_i in PHSD, if there exist two integers n and m $(n > m)$, such that packet $p(f_i, n)$ arrives at the arbiter earlier than $p(f_i, m)$, then we say the PHSD has an out-of-order problem.*

We proved that the packet delay is bounded in RRSD-MMA, which indicates that the degree of the out-of-order problem is controllable. A direct solution to this problem is to use a sorting memory in the head side. However, this requires additional SRAM for storage and sorting. In this section, we solve this out-of-order problem within the MMA itself.

6.2 The Modified PHSD

To maintain packet order, the basic idea is to guarantee in-order transmission between the SRAM and DRAM. We slightly modify the basic PHSD architecture. The new PHSD system is shown in Fig. 5. Unlike the basic PHSD in Fig. 4, each SRAM here maintains flow queues as well so that the SD transferors can transfer packets from any flow

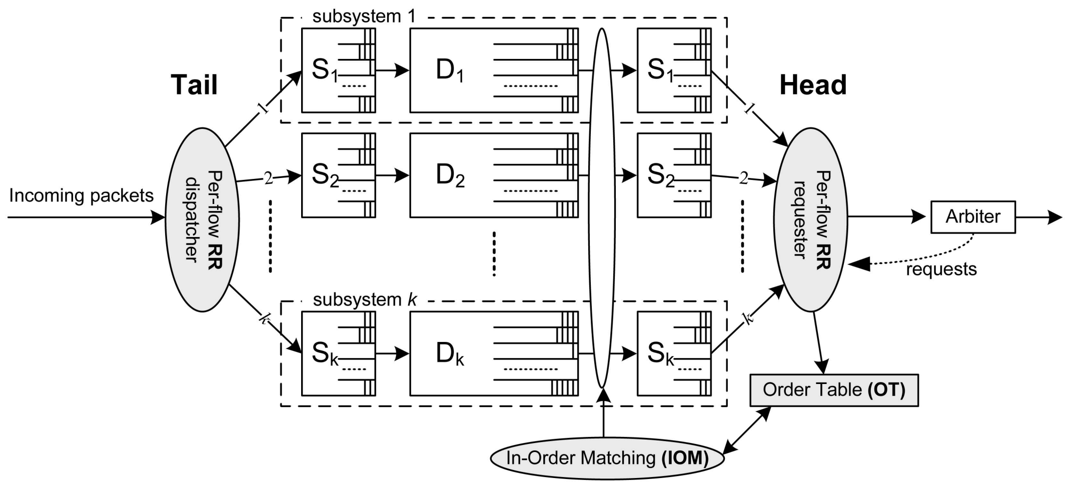


Fig. 5. The PHSD with IOM and OT.

queue head, instead of only from the head of a single FIFO queue.

It is still a symmetric system, in terms of the number of SRAM and DRAM needed in the head and tail sides. To address the out-of-order problem, we focus on the head part. The k SD transferors in the k subsystems no longer work independently as in the RRSD-MMA. Instead, they are coordinated by an *In-Order Matching* (IOM) scheduler, as shown in Fig. 5. The IOM scheduler utilizes the Order Table (OT) to determine how to control the individual SD transferors. The OT remembers the *least-ordered* request L_{f_i} for each flow f_i from all the k head SRAMs, which also indicates the most *urgent* packet that is requested to be transferred from the DRAM to the SRAM for that flow. For example, Table 1 shows an instance of an OT. $L_{f_2} = 92$ means currently the least-ordered (most urgent) packet request for flow 2 is 92.

We formulate the in-order scheduling between the DRAM and head SRAM using a bipartite graph shown in Fig. 6a. There are $2k$ vertices representing k DRAMs and k head SRAMs. The links represent packet requests from the head SRAM to the DRAM and each link is labeled as $r(f_i, n)$, which means the SRAM requests flow f_i's nth packet from the DRAM.

We define an in-order match in the request graph as follows:

Definition 4 (In-Order Match). *In the request graph in Fig. 6a, an in-order match is a link set U, in which the links conform to the following two conditions:*

1. *No two links share a DRAM or SRAM vertex.*
2. *If a link $r(f_i, n) \in U$, then $\forall m \ (L_{f_i} \leq m < n)$, the link $r(f_i, m)$ should also be $\in U$, where L_{f_i} is the least-ordered packet request for flow f_i in the OT.*

TABLE 1
OT for the Head SRAMs Scheduling

f_i	1	2	3	4	$\cdots$	Q
L_{f_i}	1	92	51	20	$\cdots$	31

The first condition is due to the one-packet access granularity of the DRAM. The second condition is essential to guarantee fulfilling the requests in order. For example, Fig. 6b is an in-order match of Fig. 6a. However, Fig. 6c is not. The request $r(f_2, 93)$ is in the matching set U, while request $r(f_2, 92)$ is not. This violates the above condition 2.

6.3 Maximum In-Order Matching (MIOM)-MMA

We first design a Maximum IOM (MIOM)-MMA for PHSD. We prove that the MIOM-MMA not only keeps packets *in order* for each flow but inherits the *same* performance of that in the RRSD-MMA as well. The MIOM is defined simply as follows:

Definition 5 (MIOM). *An MIOM is an in-order match, where there is no SRAM, which has requests, left unmatched in the final in-order match.*

The most interesting result we found in the PHSD request graph in Fig. 6a is that an MIOM is always achievable for *any* request patterns in the head SRAMs in PHSD. In particular, we state it as a theorem that follows.

Theorem 3. *There always exists an MIOM for any request graph in PHSD.*

Proof. Basically, the theorem tells that for any request graph, we can find a maximum matching (no SRAM left with requests), while it is also an IOM. The proof is lengthy and we defer it to the Appendix. □

Based on Theorem 3, we are now able to present the MIOM-MMA between the DRAM and head SRAM in PHSD.

ALGORITHM MIOM-MMA

```
In each $b$ time slots, the MMA finds an MIOM in the
request graph of the head SRAMs. It then transfers a batch
of packets according to the match. The transmission costs
$b$ time slots.
After the transmission, the MIOM-MMA updates each
flow's least-ordered request in OT with the just
transferred packets information.
```

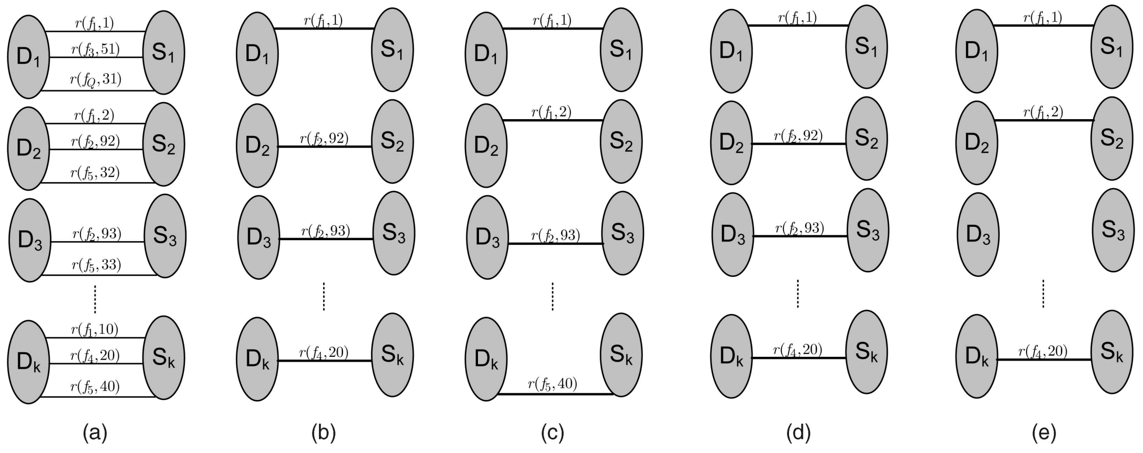


Fig. 6. Request graph between the DRAM and SRAMs and their matches. (a) Request graph. (b) In-order match. (c) Out-of-order match. (d) Maximum in-order match. (e) Maximal in-order match.

The similar MIOM-MMA can be applied in PHSD's tail, where the requests in the MIOM-MMA are packets in the tail SRAM.

The following theorem gives results on the worst-case and expected SRAM occupancy for PHSD with MIOM-MMA.

Theorem 4. *In PHSD with MIOM-MMA, the total size of the tail (head) SRAM is bounded by $Q(k-1)+kb$. The expected SRAM occupancy scales with $O(\ln Q)$ if $k=b$ and does not scale with Q if $k>b$.*

Proof. We focus on PHSD tail to calculate the SRAM sizes. Consider a tail SRAM A. A is work conserving except the following situation. When an MIOM is to be found, A is empty. However, while the batch of packets is to be transferred according to the MIOM, A starts to accumulate packets. A can only accumulate up to b packets, since the MIOM-MMA runs every b time slots and when A is having packets afterward, it is guaranteed to be matched by the MIOM-MMA.

That is to say, the occupancy of the SRAM A under the MIOM-MMA is at most b packets larger than that under the RRSD-MMA at any time. Since there are k SRAM in the tail, the total additional SRAM needed by the MIOM-MMA to absorb the burst is bounded by kb.

Using Corollary 2, we can derive that the maximum SRAM size is bounded by $Q(k-1)+kb$. Using the two propositions in Section 5, we can derive that the expected SRAM occupancy is $O(\ln Q+kb)=O(\ln Q+b^2)=O(\ln Q)$ when $k=b$ and stays as constant when $k>b$, since k and b can be regarded as constants for a specific PHSD system. □

Since PHSD is symmetric, the packet delay in the head SRAM shares the same analysis as the SRAM size in the tail.

The proof of Theorem 3 also presents a method to find an MIOM and we can derive that the MIOM-MMA complexity is $O(k^2)$. It is explained as follows: To add one link to the final matching, the algorithm might go through checking/removing at most $k-1$ other matched links to maintain the in-order property. There are at most k links to be added in. Therefore, the algorithm complexity is $O(k(k-1))=O(k^2)$. If $k=b$, the complexity becomes $O(b^2)$.

6.4 The Request-Grant (RG)-MMA

We have seen that MIOM-MMA maintains the low requirements for the SRAM while providing packet's in-order delivery. However, it has two problems in practice:

1. The MIOM-MMA requires finding the *maximum* in-order match. As we can see in the proof of Theorem 3, the MIOM-MMA requires sequentially checking/adding/removing requests, which is normally difficult to implement in hardware.
2. Some flow under the MIOM-MMA may experience starvation. A constantly incoming flow may monopolize the MIOM-MMA process and exclude all other flows from being matched.

In this section, we address these two problems by designing a practical Request-Grant (RG)-MMA that is ready for hardware implementation. To find a match, the RG-MMA uses nearly the same idea as those RG-accept negotiation algorithms well known in IQ switches [12].

ALGORITHM RG-MMA

The RG-MMA works every b time slots. It works in rounds and each round consists of two phases: *request* and *grant*.

Request: Each flow f_i sends a request $r(f_i, L_{f_i})$ to the $(L_{f_i} \bmod k)$th DRAM.

Grant: After a DRAM receives all requests from the flows, if the DRAM is matched already, it does nothing. If the DRAM is not matched, it selects one flow according to a round-robin priority list.[4]

In a round, a match is formed and each granted flow updates its least-ordered packet request in the OT to be $L_{f_i}+1$ and goes to the next round.

The RG-MMA keeps running the *RG* round until no match is found in the round.

It is straightforward to see that the match found by the RG-MMA is an in-order match, since the match for a flow f_i is always started from L_{f_i} in the OT. In addition, we have the following facts about this RG-MMA:

4. The round-robin priority list is maintained like that in the iSLIP [12] matching algorithm for the IQ switches. How to optimally select the starting point in the priority list for a new round is out of the scope of this paper. We only need the round-robin scheme to prevent flow starvation.

1. The RG-MMA guarantees no starvation to each flow.
2. The RG-MMA stops after at most k rounds.

The first fact is due to the round-robin selection among all the flows. Each flow has a chance to get granted in at most Q rounds. The second fact is also easy to show. In each round, at least one link can be matched. Since the maximum number of links in the match is k, the RG-MMA stops after at most k rounds. That is to say, the RG-MMA complexity is $O(k)$. It is therefore very efficient to implement in hardware, since each RG-MMA matching only involves at most k rounds of RG processes, which are also distributed in k subsystems.

To analyze the performance of the RG-MMA, we first define the following maximal in-order match for the request graph.

Definition 6 (Maximal In-Order Match). *A maximal in-order match is an in-order match where no links can be added to the match without violating the in-order property.*

Fig. 6e shows a maximal in-order match where we cannot add any request to DRAM D_3. For comparison, we can see that Fig. 6d shows an MIOM for the same request graph.

It is obvious to see that the RG-MMA finds a *maximal* in-order match in every b time slots. Since it is only *maximal*, not MIOM, the RG-MMA might be non-work conserving. With 100 percent traffic loads, the SRAM occupancies in PHSD with RG-MMA could become unbounded.

This is a similar situation to the scheduling in the IQ switches, where only the *maximum* weight matching algorithms are proven to make the input queues bounded. In practice, however, we can only employ heuristic matching algorithms, such as iSLIP [12], FIRM [13], and DRRM [14], which have less complexity to approximate the *maximal* size matching. A common way to prevent the IQ switches from having unbounded queues is to employ speedup inside the switch fabric [15], which can also be viewed as decreasing the traffic load.

In PHSD, we can prevent the unbounded queues in a similar way and can decrease the effective traffic load by employing more subsystems. In fact, the effective traffic load to each subsystem is b/k, which is less than 100 percent when $k > b$. We will see in simulation results presented later that the performance of the maximal IOM is comparable to the MIOM when $k > b$, even when k is only $b + 1$.

6.5 The Request-Transfer (RT)-MMA

RT-MMA presents a fast and efficient way to find the IOM. In addition, it can be pipelined to further reduce the hardware complexity. We describe the pipelined MMA as follows:

ALGORITHM RT-MMA

RT-MMA works *every* time slot. In every time slot, it just performs *one* round that consists of two phases: *request* and *transfer*.

Request: Each flow f_i sends a request $r(f_i, L_{f_i})$ to the $(L_{f_i} \bmod k)$th DRAM.

Transfer: After a DRAM receives all requests from the flows, if the DRAM is not busy transferring a packet, it just selects one flow according to the round-robin priority list and starts to transfer a packet from the flow queue head. If the DRAM is busy transferring a packet, it does nothing. Each flow updates its least-ordered request in the OT to be $L_{f_i} + 1$ after it transfers a packet.

It is obvious to see that Request-Transfer (RT)-MMA is simply a pipelined version of the RG-MMA. Therefore, they share the same performance characteristics except that RT-MMA might introduce additional pipeline delay to packets. As we can see from the second fact about the RG-MMA, the pipeline delay a packet request might incur is bounded by k time slots. Similarly, the additional SRAM occupancy for each subsystem is at most k larger with RT-MMA than with the RG-MMA. Since k can be regarded as a constant, RT-MMA and RG-MMA share the same scaling properties with regard to Q.

The algorithm complexity of RT-MMA is $O(1)$, since it only involves a round of *RT* in each time slot.

7 Simulations and Discussions

We perform extensive simulations for both HSD and PHSD systems. The performance metrics we focus on in our simulations are the maximum SRAM occupancies and packet delays with respect to the increasing number of flows in both systems. The SRAM occupancy is measured by the number of packets it holds. The packet delay is measured by the number of time slots from when the packet is requested by the outside arbiter to when it actually leaves the system. We simulate both systems under uniform traffic, unbalanced (hotspot) traffic, and bursty traffic. For the uniform traffic, incoming packets are uniformly distributed across all the Q flows. For the unbalanced (hotspot) traffic, 90 percent of the traffic aggregates to only 10 percent of the flows. In our simulations, we find the same performance under both traffics. The simple reason is that they all conform to the *stationary* and *ergodic* conditions. The round-robin dispatcher in the tail preprocesses all traffic to be sufficiently uniform to all the *subsystems*, regardless of their nonuniformity to the *flows*. For the sake of space, we only show results for both systems with the uniform traffic.

In all the simulations, we set $b = 10$ and Q ranges from 100 to 10,000. All simulations run for 10^9 time slots, and we record the *maximum* SRAM occupancies and packet delays. For comparison purposes, we feed both HSD and PHSD with exactly the same traffic in every simulation.

7.1 PHSD versus HSD under 100 Percent Traffic Loads

We first compare the performance of HSD with ECQF/EFQF-MMA and PHSD with MIOM-MMA under 100 percent traffic loads, i.e., in every one time slot there is always one incoming packet. To make them comparable, we set $k = b$ in PHSD.

We can see from Fig. 7 that both the maximum SRAM occupancy and packet delays in HSD with ECQF/EFQF-MMA scales linearly with the number of flows Q. The SRAM occupancy quickly scales beyond 10,000 packets when Q reaches 2,000. With a normal packet size of over 1,000 bytes, an SRAM size of a few megabytes can hardly hold over 10,000 packets. While in PHSD with MIOM-MMA, both the maximum SRAM occupancy and packet delay scale only in an $O(\ln Q)$ speed, which conforms to our Proposition 1. As we can see from the figure, the SRAM

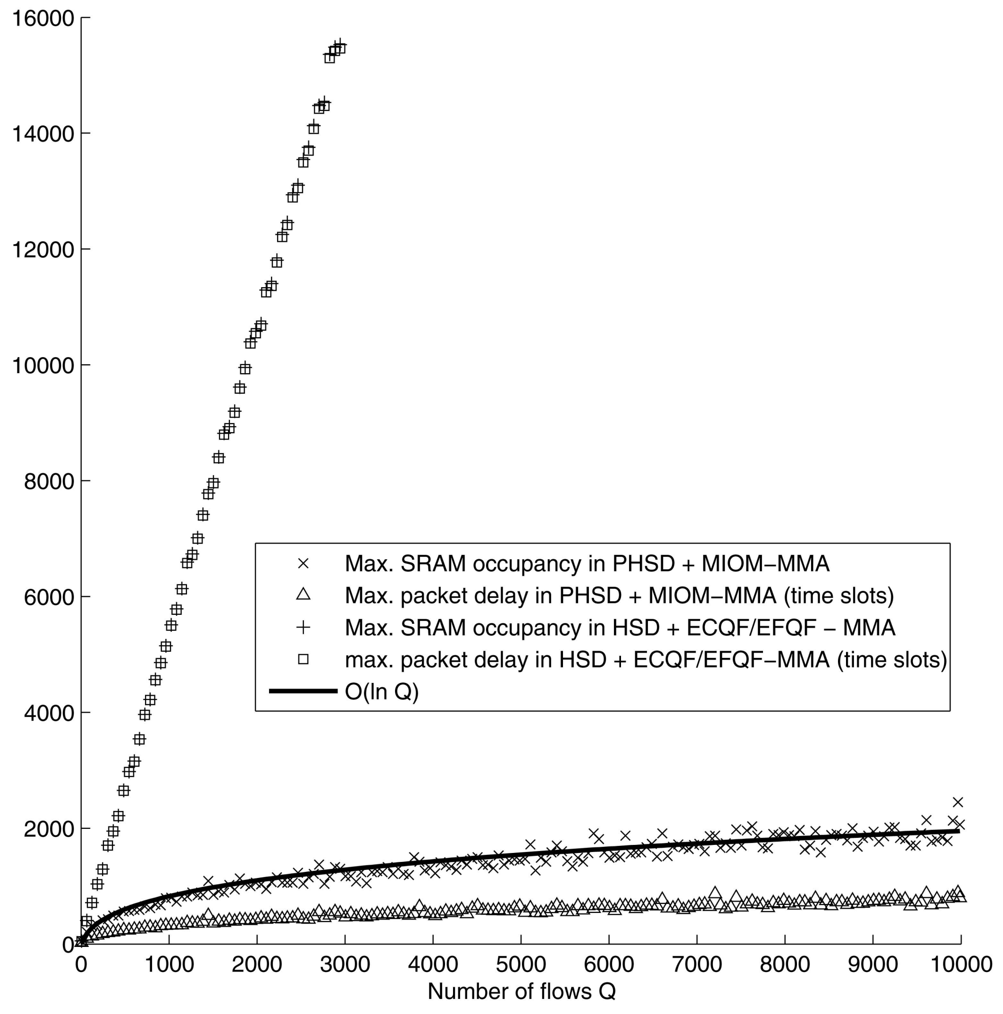


Fig. 7. Performances of HSD and PHSD with increasing Q.

occupancy is still under 3,000 packets even when Q reaches 10,000.

7.2 PHSD versus HSD under Different Traffic Loads

We also test the performance of both systems under varying traffic loads. We fix Q to be 5,000 for both HSD and PHSD, and the simulation results are shown in Figs. 8 and 9. Fig. 8 shows that for PHSD with MIOM-MMA, the SRAM occupancy and packet delay decrease significantly as the traffic loads decrease.

However, for HSD in Fig. 9, the SRAM occupancy is kept nearly unchanged even if the traffic load decreases to 50 percent. For the packet delays, they even increase when the traffic load decreases. The intuition follows. In light traffic loads, the ECQF/EFQF-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 maximum SRAM occupancy essentially does not change. Consequently, this fact also causes every packet to wait in the SRAM for a longer time with lighter traffic. Therefore, the average packet delay increases when the traffic load decreases.

7.3 Increasing k in PHSD

The performance of PHSD can be significantly improved when $k > b$. We first simulate the MIOM-MMA performance with $k > b$ and the results are shown in Fig. 10. By comparing it with Fig. 7, we can see that the SRAM occupancy decreases dramatically when we increase k, even when k is only one larger than b. In addition, when $k > b$, the SRAM occupancy remains nearly as a *constant*, which conforms precisely to our Proposition 2.

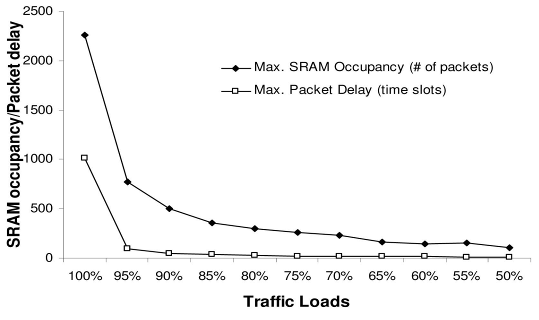


Fig. 8. Performance for PHSD with MIOM-MMA under decreasing traffic loads.

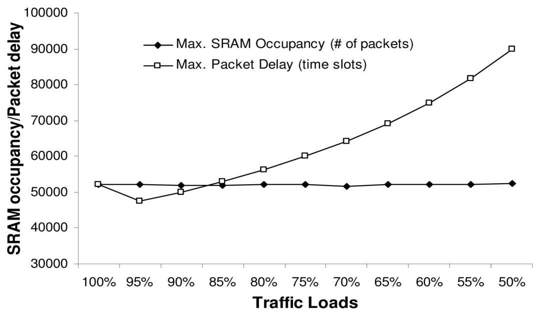


Fig. 9. Performance for HSD with ECQF/EFQF-MMA under decreasing traffic loads.

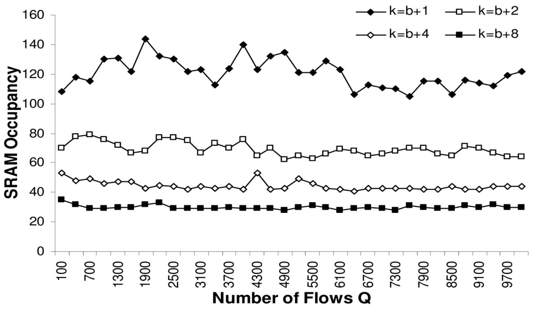


Fig. 10. Maximum SRAM occupancy for PHSD with MIOM-MMA when k increases.

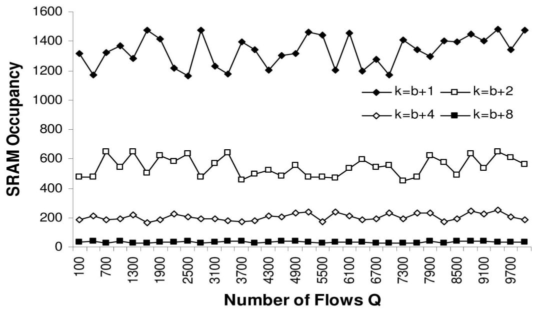


Fig. 11. Maximum SRAM occupancy for PHSD with RT-MMA when k increases.

We also simulate the RT-MMA for PHSD with $k > b$ and the results are shown in Fig. 11. We can see that the RT-MMA has nearly the same *constant* scaling property as that of the MIOM-MMA when $k > b$. And, the maximum SRAM occupancy does not scale with Q even when k is only one larger than b.

We can also find in Figs. 10 and 11 that the marginal gain from increasing k further gets smaller and smaller. This fact suggests that in practice it is sufficient to set k to be $b + 1$ or $b + 2$.

7.4 The MIOM-MMA versus RG and RT-MMA

The RT-MMA is a pipelined version of the RG-MMA, and therefore, their performance characteristics essentially share the same property. As we have shown, RT/RG-MMA is only a *maximal* version of the MIOM-MMA that is a *maximum* matching. In this section, we conduct simulations to find out the performance gap between the RT-MMA and the MIOM-MMA. As we can see from Figs. 12 and 13, when k becomes larger than b, or the traffic loads become lighter, their performance gap becomes smaller. In particular, when

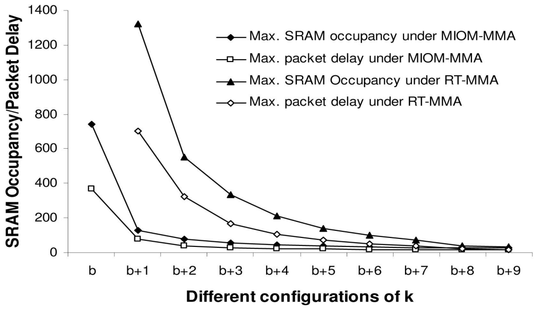


Fig. 12. Performance comparison between MIOM and RT-MMA when k increases.

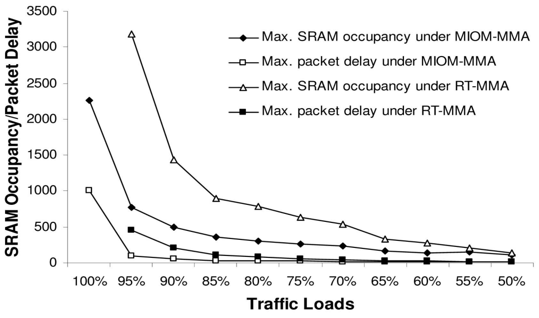


Fig. 13. Performance comparison between MIOM and RT-MMA when traffic load decreases.

$k > b + 2$ or the traffic load becomes less than 90 percent, the RT-MMA performs comparably to MIOM-MMA with $k = b$ or 100 percent traffic. This fact makes PHSD with RT-MMA more appealing under lighter traffics, or we can leverage the performance of RT-MMA by employing more subsystems even when the traffic load is 100 percent.

8 Summary and Discussions

The main performance results are summarized in Table 2 for both HSD and PHSD. We can see that PHSD with RRSD-MMA is our starting point and achieves very good performance with less MMA complexity. However, it causes the packet out-of-order problem. The MIOM-MMA maintains the packet order with PHSD. It also preserves the same SRAM occupancy property as that of RRSD-MMA, at the cost of a higher algorithm complexity $O(b^2)$. To make the MIOM-MMA practical to implement, we use a heuristic maximal matching algorithm RG-MMA to approximate it. The RG-MMA can decrease the matching complexity to $O(b)$. The RG-MMA complexity can be further ecreased to $O(1)$ by employing pipelining, which we call RT-MMA. We note that the SRAM occupancy in PHSD with RG-/RT-MMA might be unbounded if $k = b$, since the maximal matching might be nonwork conserving. However, when $k > b$, as we have seen in the simulations, the SRAM occupancy decreases significantly and becomes comparable to that of the maximum matching MIOM-MMA.

From both analysis and simulations, we can see that PHSD is not only a parallel (linear) improvement over the basic HSD system. Its performance, in terms of SRAM requirement, packet delays, and MMA complexities, significantly outweighs that of HSD when there are a large

TABLE 2
Performance Summary of All MMAs in Both HSD and PHSD

	HSD		PHSD					
			RRSD		MIOM		RG	RT
	ECQF	M banks	$k = b$	$k > b$	$k = b$	$k > b$	$k > b$	$k > b$
SRAM Occupancy	$O(Qb)$	$O(Qb/M)$	$O(\ln Q)$	const	$O(\ln Q)$	const	const	const
MMA Complexity	$O(\ln Q)$	$O(\ln Q)$	$O(1)$	$O(1)$	$O(b^2)$	$O(k^2)$	$O(k)$	$O(1)$
Packet in-order?	Yes	Yes	No	No	Yes	Yes	Yes	Yes

number of flows Q in the system. In fact, the PHSD removes the scaling dependency on the flow number Q under practical traffic conditions. Although Theorem 4 states that the worst-case performance of PHSD is the same as that of HSD, this worst case can hardly happen because 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. We have estimated the possibility of the occupancy exceeding half of the number of flows in the Appendix. We show that it is extremely small. We have also shown this by simulations. In all our simulations, we recorded all the *maximum* SRAM occupancy and the *maximum* packet delays, both of which comply with the average analysis very well.

The improvement of the expected performance enables us to design the router memory practically and efficiently with less SRAM. In practice, if we have the SRAM overflow in some rare case, we can choose to drop the packet. This would prove beneficial to the upper layer when the packet may experience rather large delay if it was buffered in the memory system. Nevertheless, from the simulations, we can have an estimation of the maximum SRAM occupancy. For the MIOM-MMA with $k = b$ and 100 percent traffic load, or the RT-MMA with $k = b + 2$ or less than 90 percent traffic, the maximum SRAM occupancy is less than 2,500 packets. Assuming an average of 100 bytes per packet (for TCP traffic, most packets are of 40 bytes), we can calculate that the maximum SRAM occupancy is $2,500 \times 100 \times 8 = 2\ \mathrm{Mbits}$, which is well bounded by current SRAM fabrication technology.

We can also see the tradeoffs between the complexity of the MMAs and the guarantee of in-order delivery in Table 2. Besides the MMA complexity, the in-order delivery also cost two data structures, the per-flow queues in SRAM and the OT, which scale linearly with Q. We believe some future improvements can be made here. From the SRAM size analysis, we know that packets residing in the SRAM is significantly smaller than Q, which means that most of the flow queues are empty most of the time. Therefore, some overlap techniques can be used to reduce the complexities of the two data structures.

9 Conclusions

In this paper, we first investigate existing proposals for the router buffers that appeared in literature. We analyze their performance and highlight their architectural scalability limitations in supporting a large number of flows in the system. Then, we address the limitations by proposing new buffering architecture based on PHSD.

For PHSD, we first design a simple yet efficient MMA. We prove that the SRAM size and packet delay can be significantly reduced compared with previous solutions with the same buffering capacity. However, this simple MMA incurs the out-of-order problem for packets. We proceed to solve this problem by designing a series of new MMAs that show tradeoffs between the SRAM sizes and the MMA complexities. Extensive analysis and simulations are performed to assess the improvement obtained by PHSD over previous solutions in terms of packet delay and the SRAM size required in the system.

Having shown its nearly $O(1)$ MMAs and better packet delay and SRAM requirements, we believe PHSD is among the best candidates to build the next generation router buffers, where they are expected to accommodate a large number of flows at very fast line rates.

Appendix

Proof of Theorem 3

Proof. We construct the MIOM in the following way.

```
Initialize: $U = \phi$
Loop: (if there exists an SRAM $A$ with request but unmatched)
1) Select an arbitrary request $r(f_i, L_{f_i})$ in $A$.
2) Add links $\{r(f_i, L_{f_i}), r(f_i, L_{f_i}+1), r(f_i, L_{f_i}+2), \dots, r(f_i, m)\}$ to $U$.
3) If the new link $r(f_j, t)$, $n < t \le L_{f_i}$ is already in $U$, then
   remove the duplicates.
4) If the new link $r(f_j, t)$, $n < t \le L_{f_i}$, shares the same
   SRAM with one link $r(\cdot,\cdot)$ that is already in $U$, then
   remove $r(\cdot,\cdot)$ from $U$.
End Loop
```

It is easy to see that in the final set U, no two links share the same SRAM or DRAM. We now prove that all the links in U preserves the in-order property.

After the first loop, $U = \{r(f_i, L_{f_i}), r(f_i, L_{f_i}+1), r(f_i, L_{f_i}+2), \dots, r(f_i, m)\}$. It obviously preserves the in-order property and is an in-order match.

For the following loops, assume U is already an in-order match and is about to add links $\{r(f_j, L_{f_j}), r(f_j, L_{f_j}+1), r(f_j, L_{f_j}+2), \ldots, r(f_j, n)\}$. We prove that after removing the conflicts, U is still an in-order match.

We prove it by gradually removing link conflicts from $r(f_j, n-1), r(f_j, n-2), \ldots,$ to $r(f_j, L_{f_j})$, since $r(f_j, n)$ has no conflicts.

For $r(f_j, n-1)$, if it has no conflicts with links already in U, it surely does not cause an out-of-order problem. If it is a duplicated link in U, removing it actually does not change links in the original U and thus does not cause an out-of-order problem. If $r(f_j, n-1)$ shares an SRAM with a link in U, say $r(f_s, t), s \neq j$, we assert that removing $r(f_s, t)$ still does not cause an out-of-order problem. Otherwise, if removing $r(f_s, t)$ causes an out-of-order problem, it indicates that $\exists t'$, $t' > t$, and $r(f_s, t')$ is in U. According to the definition of in-order match, $r(f_s, t+1)$ is also in U. Since packet requests are dispatched into the subsystems in a round-robin way, we know that $r(f_s, t+1)$ shares the same SRAM with $r(f_j, n)$, which is a contradiction.

After removing conflicts with $r(f_s, n-1)$, we proceed with considering $r(f_s, n-2)$ and the same analysis holds that removing conflicts with $r(f_s, n-2)$ does not cause an out-of-order problem in U.

Since the newly added links $\{r(f_j, L_{f_j}), r(f_j, L_{f_j}+1), r(f_j, L_{f_j}+2), \ldots, r(f_j, n)\}$ are in order and removing the conflict links with them does not cause any out of order, we can see that after one loop, U is still an in-order match.

In each loop, the size of U increases by at least one. Since the maximum size match is k, the loop is surely to stop. And, when it stops, the resulted U forms an MIOM according to the definition. □

Acknowledgments

This work was supported by the Hong Kong RGC Grant HKUST6200/02E.

References

[1] J. Garcia, J. Corbal, L. Cerda, and M. Valero, "Design and Implementation of High-Performance Memory Systems for Future Packet Buffers," *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture (MICRO-36 '03),* pp. 372-384, 2003.

[2] Y. Ganjali and N. McKeown, "Update on Buffer Sizing in Internet Routers," *ACM SIGCOMM Computer Comm. Rev.,* vol. 36, no. 5, pp. 67-70, 2006.

[3] J. Garcia-Vidal, M. March, L. Cerda, J. Corbal, and M. Valero, "A DRAM/SRAM Memory Scheme for Fast Packet Buffers," *IEEE Trans. Computers,* vol. 55, no. 5, pp. 588-602, May 2006.

[4] J. Garcia, M. March, L. Cerda, J. Corbal, and M. Valero, "On the Design of Hybrid DRAM/SRAM Memory Schemes for Fast Packet Buffers," *Proc. Workshop High Performance Switching and Routing (HPSR '04),* pp. 15-19, 2004.

[5] S. Iyer, R. Kompella, and N. McKeowa, "Analysis of a Memory Architecture for Fast Packet Buffers," *Proc. IEEE Workshop High Performance Switching and Routing (HPSR '01),* pp. 368-373, 2001.

[6] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 2003.

[7] Y. Tamir and G. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communications Switches," *ACM SIGARCH Computer Architecture News,* vol. 16, no. 2, pp. 343-354, 1988.

[8] A. Demers, S. Keshavt, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. ACM SIGCOMM '89,* pp. 3-12, 1989.

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

[10] C. Chang, D. Lee, Y. Shih, and C. Yu, "Mailbox Switch: A Scalable Two-Stage Switch Architecture for Conflict Resolution of Ordered Packets," *IEEE Trans. Comm.,* vol. 56, no. 1, pp. 136-149, 2008.

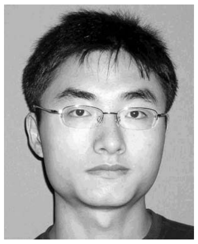
[11] I. Keslassy and N. McKeown, "Maintaining Packet Order in Two-Stage Switches," *Proc. IEEE INFOCOM '02,* vol. 2, 2002.

[12] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking,* vol. 7, no. 2, pp. 188-201, 1999.

[13] D. Serpanos and P. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-Speed ATM Switches with Multiple Input Queues," *Proc. IEEE INFOCOM '00,* vol. 2, 2000.

[14] H. Chao and J. Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM Switch," *Proc. IEEE ATM Workshop,* pp. 11-16, 1998.

[15] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input/Output-Queued Switch," *IEEE J. Selected Areas in Comm.,* vol. 17, no. 6, pp. 1030-1039, 1999.



Feng Wang received the BSc degree in informatics, minor in economics, from Peking University, Beijing, in 2003. He is currently working toward the PhD degree in computer science and engineering at the Hong Kong University of Science and Technology, Hong Kong. His research interests are generally in computer networks. In particular, he is focusing on the design and analysis of high-performance switches/routers. He is also interested in wireless networks. He is a student member of the IEEE and the IEEE Communications Society.

Mounir Hamdi received the BS degree (with distinction) in computer engineering from the University of Louisiana in 1985 and the MS and PhD degrees in electrical engineering from the University of Pittsburgh in 1987 and 1991, respectively. From 1985 to 1991, he was a teaching/research fellow in the Department of Electrical Engineering, University of Pittsburgh, where he was involved in major research projects as well as teaching undergraduate courses. He is currently a full professor and the head of the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, a director of the Master of Science in Information Technology, and a director of the Computer Engineering and Networking Laboratory. In 1999 to 2000, he held visiting professor positions at Stanford University, Stanford, California, and the Swiss Federal Institute of Technology, Lausanne, Switzerland. His general area of research is in high-speed wired/wireless networking in which he has published more than 300 research publications, received numerous research grants, and graduated more 30 postgraduate students. In addition, he has frequently consulted for companies in the US, Europe, and Asia on high-performance Internet routers and switches as well as high-speed wireless LANs. Currently, he is working on the design, analysis, scheduling, and management of high-performance Internet switches/routers, algorithm/architecture codesign, wavelength division multiplexing (WDM) networks/switches, and high-speed wireless networks. In particular, he is leading a research team at the Hong Kong University of Science and Technology that is designing one of the highest capacity chip sets for Terabit switches/routers in the world. This chip set is targeted toward 256 $\times$ 256 OC-192 Internet switches and includes a crossbar fabric chip, a scheduler/arbiter chip, and a traffic management chip. He has been on the editorial board of the *IEEE Transactions on Communications*, *IEEE Communication Magazine*, *Computer Networks*, *Wireless Communications and Mobile Computing*, and *Parallel Computing* and has been on the program committees of more than 70 international conferences and workshops. He was a guest editor of three *IEEE Communications Magazine* special issues, a guest editor-in-chief of two special issues of *IEEE Journal on Selected Areas of Communications*, and a guest editor of *Optical Networks Magazine*. He has chaired more than 11 international conferences and workshops including the IEEE GLOBECOM/ICC Optical Networking Workshop, the IEEE ICC High-Speed Access Workshop, and the IEEE IPPS HiNets Workshop. He has been the chair of IEEE Communications Society Technical Committee on Transmissions, Access and Optical Systems and vice-chair of the Optical Networking Technical Committee, as well as member of the ComSoc technical activities council. He has been on the technical program committees of more than 100 international conferences and workshops. He received the Best Paper Award at the International Conference on Information and Networking in 1998 out of 152 papers. He also supervised the Best PhD Paper Award among all universities in Hong Kong. In addition to his commitment to research and professional service, he is also a dedicated teacher. He received the Best 10 Lecturers Award (through university-wide student voting for all university faculties held once a year), the Distinguished Engineering Teaching Appreciation Award from the Hong Kong University of Science and Technology, and various grants targeted toward the improvement of teaching methodologies, delivery, and technology. He is a senior member of the IEEE and the ACM.

Jogesh K. Muppala received the PhD degree in electrical engineering from Duke University, Durham, North Carolina, in 1991. He is currently an associate professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong. From 1991 to 1992, he was a member of the technical staff at Software Productivity Consortium, Herndon, Virginia, where he was involved in the development of modeling techniques for systems and software. While at Duke University, he participated in the development of two modeling tools, the Stochastic Petri Net Package (SPNP) and the symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE), both of which are being used in several universities and industry in the USA. He was the program cochair for the 1999 Pacific Rim International Symposium on Dependable Computing held in Hong Kong in December 1999. He also cofounded and organized the Asia-Pacific Workshop on Embedded System Education and Research. He has also served on the program committees of many international conferences. He received the Excellence in Teaching Innovation Award in 2007. He was also awarded the Teaching Excellence Appreciation Award by the Dean of Engineering, HKUST. He is a senior member of the IEEE and the IEEE Communications Society, and a participating representative from HKUST with EDUCAUSE.