

QOS GUARANTEE IN INPUT-QUEUED SWITCHES WITH NONITERATIVE SCHEDULERS*

Kevin F. Chen, Edwin H.-M. Sha, S. Q. Zheng
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75083, USA
{kchen, edsha, sizheng}@utdallas.edu

Abstract

We report two fast and scalable scheduling algorithms that provide exact bandwidth guarantee, low delay bound, and reasonable jitter in input-queued switches. The two schedulers find a maximum input/output matching in a single iteration. They sustain 100% throughput under both uniform and bursty traffic. They work many times faster than existing scheduling schemes and their speed does not degrade with increased switch size. SRA and SRA+ algorithms are of $O(1)$ time complexity and can be implemented in simple hardware. SRA tends to incur different delays to flows of different classes of service due to their different subscribed portions of the total bandwidth. SRA+ is a weighted version of SRA. SRA+ improves over SRA in that all flows undergo the same delays regardless of their bandwidth shares. The schedulers operate on queue groups at the crossbar arbiters in a distributed manner.

Key Words

Input-queued switch, switching, quality of service

1 Introduction

Real-time network services impose stringent *quality of service* (QoS) requirements on switches. Switches must guarantee bandwidth (or rate), bound delay, and smooth jitter. This paper presents two novel and efficient switch fabrics that support these QoS functions. We show how exact bandwidth guarantee, low delay bound, and reasonable jitter can be provided in a switch fabric.

Our switch fabrics are based on the input-queued (IQ) paradigm. We consider them as using crossbars with unbuffered crosspoints. IQ switches have large packet buffers at the input ports. Packets arriving at an input port are organized into separate queues according to their destination output and their *class of service* (COS). Such a queue is called a virtual output queue (VOQ). Using VOQs removes head-of-line (HOL) blocking that occurs when FIFO queues are used. In providing QoS, our switch fabrics operate on traffic aggregates.

A traffic aggregate can be a number of flows arriving at different input ports but going to the same destination; all the flows require the same COS.

Robust and efficient switch fabrics that support QoS are still lacking. Existing schemes found in the literature for the IQ switch that support certain QoS have limitations. They often are too complex, too slow, and not scalable. Some schemes are based on the Slepian-Duiguid algorithm (e.g. [1, 8, 10]). Some are based on the Gale-Shapley algorithm (e.g. [7, 9, 11, 16, 17, 20]). Both types of algorithms are not effective in reserving bandwidth and can cause severe degradation in throughput. They are also too complex for hardware implementation and to be scalable. Iterative maximal matching schemes (e.g. [2, 12, 19, 21]) tend to incur long delay and are not scalable. There are also maximum-weight matching algorithms [13–15] to process traffic of different priorities. These algorithms are too complex (of time complexity $O(N^3 \log N)$) and difficult to be implemented in hardware. The Birkoff-von Neumann switch runs an offline matching algorithm that is based on matrix decomposition and of $O(N^{4.5})$ complexity [3, 4] which is clearly too complex.

Our work offers better solutions that could be used in future-generation switches. In previous work, we studied a best-effort switch fabric that implements a fast noniterative fabric scheduling algorithm called *single round-robin arbitration* (SRA) and its corresponding scalable architectures [5, 6]. SRA finds a maximum matching in a single iteration. Its time complexity is $O(1)$. SRA runs many times faster than existing algorithms for crossbar arbitration while operating at line rate. SRA is simple to implement in hardware as are its supporting architectures.

In this work, we evaluate how well SRA supports QoS. We show that SRA is as efficient as it is for best-effort traffic. In addition, we follow the framework of SRA for arbitration and add a credit-based weighting mechanism to form a new system, called SRA+, for QoS assurance. The SRA+ scheme still runs at line rate and is capable of fast arbitration and exact QoS guarantee. SRA+ also finds a maximum matching in a single iteration. It is many times faster than existing QoS schemes as it uses SRA for arbitration. Thus it can satisfy most stringent delay requirements. Furthermore, SRA/SRA+ algorithms are all of constant time which enables simple hardware implementation. SRA and SRA+ employ simple queuing structure and disciplines and match simple architectures.

*Work supported in part by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001, NSF CCF-0309461, NSF CCF-0514092, and Microsoft, USA.

2 Switch Architecture

Figure 1 illustrates a switch model that represents the IQ switch. It shows the overall architecture and the queuing structure of the IQ switch. The model consists of N input ports, an $N \times N$ crossbar with no buffer at cross-points, and N output ports. Arbiters sit between the input ports and the crossbar to arbitrate access to the crossbar.

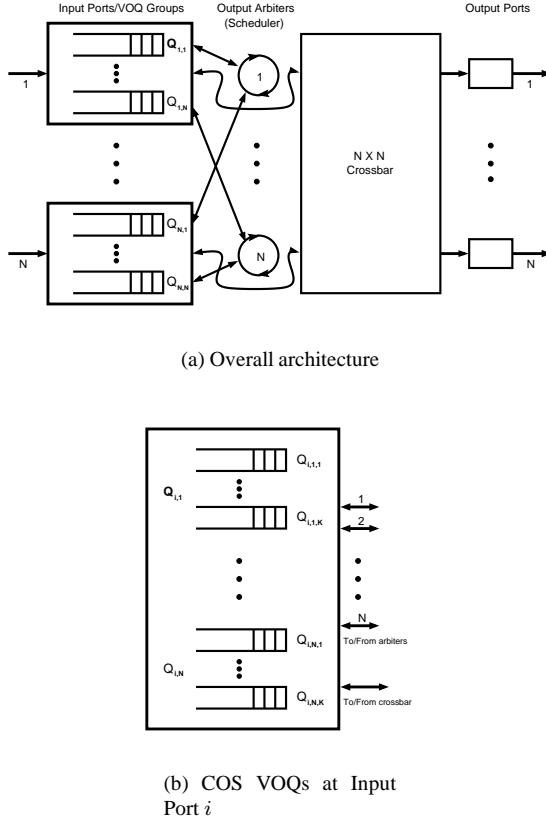


Figure 1. The architecture of the IQ switch.

An input port of an IQ switch will have buffering of around 150 ms which is approximately the round-trip time of traffic along a typical Internet path. An output port of an IQ switch needs a buffer to hold cells of a number of packets because it may take a few time slots for all the cells of a packet to arrive before the packet can be reassembled and sent out. There will be only one cell arriving and to send out in each time slot when the fabric operates at the line rate.

Queues are organized differently in input ports and output ports. At input ports are VOQs which are organized in accordance with the COSs; traffic going to the same output port but belonging to different COSs form a virtual *queue group* ($Q_{i,j}$, $1 \leq i, j \leq N$, in Figure 1(a)). Traffic over the entire switch is classified into K COSs. There are a total of NK VOQs in each input port. Traffic of a COS can be striped over all N input ports and all N output ports. As an example, $Q_{i,1,1}, Q_{i,2,1}, \dots, Q_{i,N,1}$, $1 \leq i \leq N$, in Figure 1(b) con-

stitute one COS (COS 1). We could specify as many COSs as there are VOQs, but defining K COSs as above is most practical. For instance, DiffServ classifies traffic into only 6 COSs [21]. Also, VOQs are FIFO queues.

Time in the switch fabric is slotted. In each time slot, at most one cell can arrive at each input port, each arbiter completes one arbitration process, selected cells are switched through the crossbar, and each output port sends at most one cell out. There is no speedup, i.e. the value of the speedup is 1. The fabric operates on fixed-sized cells. Packets of variable lengths are segmented and padded into cells.

SRA or SRA+ is used for arbitration for all the schemes discussed in this paper. SRA or SRA+ is run at the arbiters. There is an arbiter for each output. Arbiters are distributed vertically across the fabric and operate independently of each other. SRA itself can also be used to guarantee bandwidth and to provide an average delay bound to all COSs.

SRA+ is a fusion of SRA and a credit-based weighting scheme for arbitration and enforcing QoS. It eliminates the difference in delays of different COSs. SRA+ employs a rationing mechanism at distributed arbiters that operate in parallel. SRA+ is the first switch fabric scheme to use many instances of a rationing mechanism in parallel and thereby verify the feasibility of such a mechanism's parallelization. A serial credit-based weighting scheme is deficit round-robin (DRR) [18]. By initial design, DRR works on a few flows that are pooled together. DRR has been adapted into use in the Cyclone switch where it is combined with iSLIP and used for arbitration [22].

SRA and SRA+ are scalable in that cell delay does not degrade when N increases. Our data show that the overall cell delay remains about the same when we vary N among 8, 16, 32, and 64. Variation in standard deviation is also negligible. Similar variation exists in variance and coefficient of variance. Throughput remains 100% for all N values under all loads. In addition, exact bandwidth guarantee holds for all the values of N . In all the simulation runs, input blocking has shown to be low. Cell multiplicity is no higher than 3 in about 99% of the time at even high loads for high N values and is almost never higher than 5.

Because the arbiters in SRA and SRA+ act independently of each other, there is a possibility that each input port will receive more than one grant to send in each time slot. That an input port sends more than one cell going to different outputs in a time slot is called *input blocking*. We use k to notate this cell multiplicity. In switching best-effort traffic, k has been shown to be low [5, 6]. For any N , the occurrence of $k > 2$ is only about 7%. In most cases, k is 1 or 2. Instances of $k > 5$ virtually do not occur. To abate input blocking, we proposed two crossbars to purposely and perfectly support SRA [5, 6], although SRA and SRA+ can work on any architecture that supports any existing scheduling scheme.

SRA and SRA+ use only one layer of round-robin arbiters (for outputs only), unlike some maximal matching schemes that use two layers (at both inputs and outputs). SRA and SRA+ all have a time complexity of $O(1)$ since all the operations take constant time to complete. Particularly, the way the quantum

accretes in SRA+ ensures when the FIFO head element gets its turn, at least one cell or a burst of cells will be sent. Thus there is no queue traversal in a time slot which would take linear time. SRA and SRA+ find a maximum matching in each time slot in a single iteration. We proved in [5, 6] that SRA sustains 100% throughput and is fair. We showed in simulations that SRA is many times faster than traditional maximal matching schemes [5, 6]. Following the same frameworks of proof in [5, 6] for best-effort traffic, we can easily prove that these properties of SRA hold true for COS-differentiated traffic and for SRA+ for cells and packets. These properties were clearly shown to be true in our simulations to be described in the following sections. Moreover, SRA and SRA+ are far simpler than other existing QoS schemes.

3 SRA for QoS

SRA was initially devised to fast switch best-effort traffic, but it has the ability to guarantee bandwidth. In addition, its delay performance does not degrade when used to guarantee bandwidth. Furthermore, it guarantees bandwidth exactly. We can find a delay bound that is the same for all COSs. Service contracts can be practically enacted by specifying the bandwidth required.

3.1 The SRA Algorithm

The switch scheduling problem can be modeled as a bipartite matching problem as follows. Let I_i and O_j denote input port i and output port j , respectively. Let $VOQ_{i,j}$ denote the VOQ at I_i holding cells destined for O_j , and $VOQ_{i,j}(t)$ the length of $VOQ_{i,j}$ at time slot t . In each time slot t , we construct a bipartite graph $G(V, E)$ such that $V = V_1 \cup V_2$, $V_1 = \{VOQ_{i,j} | 1 \leq i, j \leq N\}$, $V_2 = \{O_j | 1 \leq j \leq N\}$, and $E = \{(VOQ_{i,j}, O_j) | VOQ_{i,j}(t) > 0, 1 \leq i, j \leq N\}$. Graph G is called a *modified I/O mapping graph*. Note that $VOQ_{i,j}$ represents the queue group of K VOQs at input port I_i destined to the same output port O_j . SRA works in such a way that at most one VOQ in a queue group can transmit in a time slot. A *matching* is defined as the set $\mathcal{M} \subseteq E$ such that no two edges in \mathcal{M} are incident to the same node in V_2 but multiple edges are allowed to be incident to the same node in V_1 . A *maximum matching* is one with the maximum number of edges that can be matched in a time slot. SRA finds a maximum matching in G in one iteration [5, 6].

With COS distinction, the SRA algorithm works as follows.

(1) At the outputs: Each output arbiter maintains a FIFO queue of status information of the input VOQs that have cells destined to the output. This queue can be no longer than NK at any time. The arbiter always chooses the VOQ (corresponding to one input) at the head of the queue to send in each time slot. Then the arbiter sends a grant to the input that the chosen VOQ belongs to and removes the VOQ from the head of the status queue. After the input has sent a cell from the VOQ, if the VOQ still has cells queued, that VOQ is queued again into

the tail of the status queue, else the status element for that VOQ is gone.

(2) At the inputs: Upon receiving a grant, the input checks if the corresponding VOQ is to become empty if the cell has been sent. If yes, it sends a status signal to the output arbiter indicating the VOQ is to be empty, so the output arbiter will not keep an element for this VOQ in its FIFO queue again. Then the input port sends a cell from the VOQ to the crossbar with the designated output information. The input sends status information about any of its VOQs to the corresponding output only when the VOQ changes from being empty to having a cell arrived and from having cells to becoming empty.

3.2 Bandwidth Guarantee

The fairness property of SRA implies that a stream of traffic going from an input port i to an output port j always gets its fair share of service. By the same token, with traffic differentiation, a COS will always get its fair share of bandwidth. This can be viewed as an extension of our previous fairness result (Theorem 3 of [5,6]). In fact, we found by simulation that the guarantee of bandwidth is exact. In the simulations, we also found that SRA performs well regardless of the values of N and is therefore scalable.

We simulated the working of SRA with $N = 8, 16, 32, 64$ and $K = 6$ under incoming traffic of i.i.d. Bernoulli with destination uniformly distributed. We set the bandwidth portions of the K COSs to be 0.5, 0.3, 0.1, 0.05, 0.03, and 0.02. At all loads, the bandwidth of every COS is exactly guaranteed. On all accounts, the overall throughput is sustained at 100%.

Figures 2 and 3 show the results when N is 8. The COSs underwent different mean delays due to their bandwidth shares as can be clearly seen in Figure 2. COS 1, with a bandwidth share of 50% suffered the most delay whereas COS 5 the least. While each COS gets its bandwidth share, the one with a higher portion will only get the same $1/m$ of the chances as the others if there are m COSs all having queued cells. This variation is also evident in some other statistic properties of cell delay we obtained.

Figure 3 shows the standard deviation of the COS delays which clearly varies among the COSs. The variance and coefficient of variance of the COS delays display the same variation. These statistics measure the degree of dispersion which is indicative of jitter. Jitter measures variation in cell delay. The most effective jitter indicator of all is standard deviation which with a value of s plainly means that the jitter is s cell times above or below the mean delay. We see that dispersion and hence jitter is high at high load values but is within 10 cell times when the load is lower than 95%. When the load approaches 1, the system becomes less stable and is said to have been overloaded. Overcharge should not occur in normal provisioning of the system.

If the sole purpose of a system is to guarantee bandwidth, then SRA is workable. But it is usually more desirable to subject all COSs to the same overall delay or a COS to a delay less than the overall delay. The delay dispersion problem of SRA in ensuring bandwidth will be remedied in a weighted version of

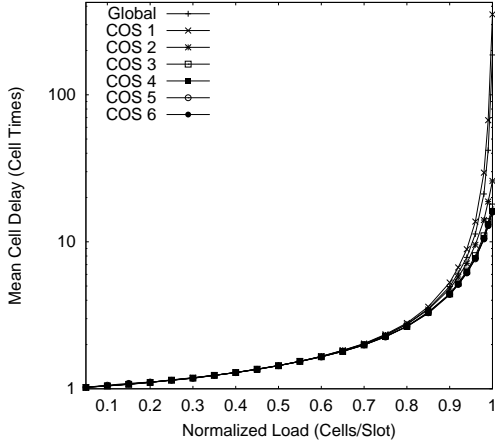


Figure 2. Mean cell delays under SRA.

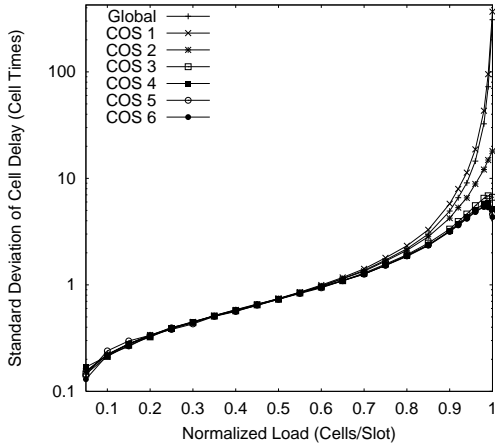


Figure 3. Standard deviation of cell delays under SRA.

SRA, called SRA+, which we present in the next section.

4 SRA+

SRA+ is a weighted version of SRA. It micrifies the dispersion of statistics of COSs taking different bandwidth proportions at high loads observed in using SRA as discussed in the last section. SRA+ maintains the low delay and low jitter properties of SRA in bandwidth guarantee. It sustains 100% throughput as SRA. The SRA+ algorithm runs in $O(1)$ time and is amenable to simple hardware implementation.

4.1 The SRA+ Algorithm

SRA+ follows the graph-theoretic definition of the switching problem as for SRA which is described in Subsection 3.1. Only an input port now has to handle K COS flows with weight constraint such that each COS gets its share of the bandwidth with the same delay as other COSs across the switch fabric. We as-

sume that every packet has been segmented to cells and SRA+ operates on these cells without regard to their adjacency in the original packet. We consider SRA+ switching cells of a packet contiguously in the next section. SRA+ for cells works as follows.

(1) At the outputs: Each output arbiter maintains a FIFO queue of status information of the input VOQs that have cells destined to the output. This queue can be no longer than NK at any time. The arbiter always picks the head element of the queue (corresponding to $VOQ[i, f]$ of input i and COS f) to send in the time slot. Then the arbiter sends a grant to input port i . When the credit for $VOQ[i, f]$ is exhausted or insufficient for the next burst or when $VOQ[i, f]$ has become empty, the arbiter dequeues the element representing $VOQ[i, f]$ off the status queue and gets to its next element. When the credit for $VOQ[i, f]$ has been exhausted or become insufficient for the next burst, if $VOQ[i, f]$ is still backlogged, it notifies the input i to stop sending and inserts the element representing $VOQ[i, f]$ back into the status queue immediately or after as many time slots as the number of cells in the burst. When $VOQ[i, f]$ has become empty, its credit is reset to zero until it gets backlogged again. The quantum of credit for each COS is its weight divided by the smallest of weights of all the COSs (times the maximum packet length in terms of number of cells if traffic is bursty).

(2) At the inputs: Upon receiving a grant, the input checks if the corresponding VOQ is to become empty if the cell has been sent. If yes, it sends a status signal to the output arbiter indicating the VOQ is to be empty, so the output arbiter will not keep an element for this VOQ in its FIFO queue again. Then the input port sends a cell in each time slot onward from the VOQ to the crossbar with the designated output information until it receives a notification from the arbiter that the VOQ's credit has been exhausted or become insufficient for the next burst or the VOQ has become empty. The input sends status information about any of its VOQs to the corresponding output only when the VOQ changes from being empty to having a cell arrived and from having cells to becoming empty.

4.2 QoS Guarantees

In simulating SRA+, we found that SRA+ matches almost exactly to SRA on all of the overall performance metrics. For individual COSs, SRA+ shows that the dispersion of their statistics found in running SRA is greatly alleviated.

We simulated the working of SRA+ with $N = 8, 16, 32, 64$ and $K = 6$ under incoming traffic of i.i.d. Bernoulli with destination uniformly distributed. We set the bandwidth portions of the K COSs to be 0.5, 0.3, 0.1, 0.05, 0.03, and 0.02. As in running SRA, at all loads, the bandwidth of every COS is exactly guaranteed. On all accounts, the overall throughput is sustained at 100%.

Figures 4 and 5 show the mean and standard deviation of cell delays for the COSs. The dispersion of cell delays is almost completely gone. Variance and coefficient of variance of the delays show the same effect. This can be clearly seen if these figures are compared with those for SRA shown in Figures 2 and 3.

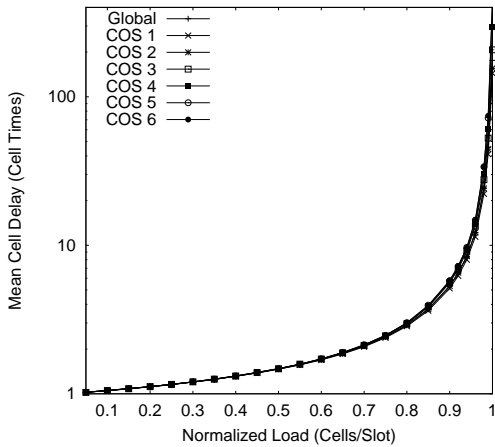


Figure 4. Mean cell delays under SRA+.

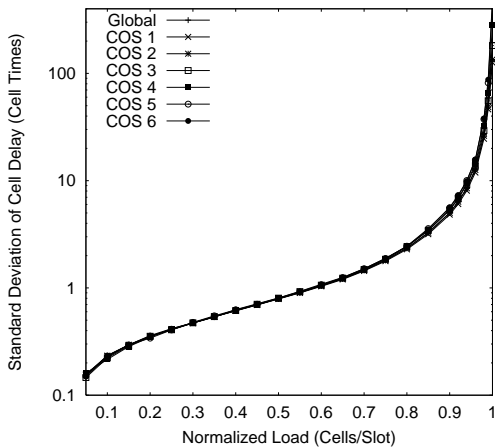


Figure 5. Standard deviation of cell delays under SRA+.

The global mean cell delay shown in Figure 4 matches nearly perfectly with that in Figure 2 as does the global standard deviation in Figure 5 with Figure 3. Global metrics are summary measures of all cells switched through the fabric during the simulation interval.

The trajectory of global mean cell delay represents the amount of delay that every COS experiences under a series of loads. Similarly, the global standard deviation curve indicates how much jitter there will be for every COS under various loads. For both metrics, the values for a COS are independent of its subscribed rate.

We also tested how SRA+ performs under bursty traffic. We devised a special bursty loading process which can be modeled as a modified Interrupted Bernoulli Process (IBP) with an average burst length of 10.885536 cells. Each burst corresponds to a packet-worth of cells; the cells of each arrived packet are sent back-to-back. We simulated the working of SRA+ with $N = 8, 16, 32, 64$ and $K = 6$. We set the bandwidth portions of the K COSs to be 0.5, 0.3, 0.1, 0.05, 0.03, and 0.02. At all

loads, the bandwidth of every COS is exactly guaranteed. On all accounts, the overall throughput is sustained at 100%.

Figures 6 and 7 show the mean and standard deviation of the COS cell delays. Variance and coefficient of variance show the same effect. The dispersion of cell delays is almost nonexistent except at low-end loads where a COS with lower bandwidth proportion gets far fewer packets arriving and thereby shows lower variation.

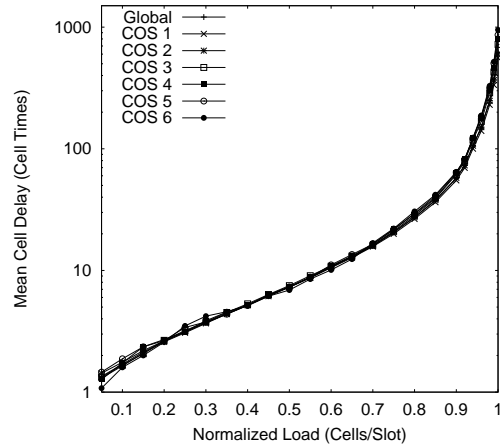


Figure 6. Mean cell delays under SRA+ for packets.

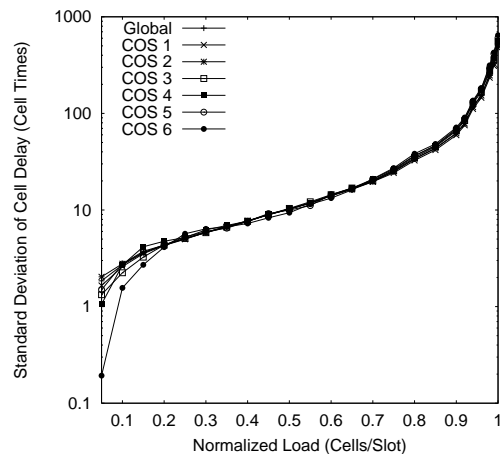


Figure 7. Standard deviation of cell delays under SRA+ for packets.

In terms of cell delay, SRA+ is a number of times faster than HDS and PQWRR under bursty traffic [21] although burst length we used is a bit shorter than theirs. But SRA+ shows higher jitter under high loads than HDS and PQWRR.

5 Concluding Remarks

Our two noniterative schedulers SRA and SRA+ are many times faster than known conventional QoS schedulers. They

have good QoS performance. They are capable of exact bandwidth guarantee and tight delay bound. They sustain 100% throughput and make subscribing out total available bandwidth possible. They are also scalable. Their performance does not degrade with switch size. They can all operate on the same supporting switch fabric architectures. Their constituent algorithms run in constant time and are easy to implement in hardware.

With SRA+, we subjected the switch fabric to two modes of cell switching. The first mode is to switch cells as they are dispatched to the switch fabric in an i.i.d. Bernoulli pattern. In this mode, cells of packets can not be sent as soon as the packets arrive and are segmented because they have to be shaped according to the i.i.d. Bernoulli pattern. In the second mode, cell traffic is bursty as cells of each packet are sent contiguously to the fabric as the packet arrives and is segmented. It can be estimated that the delays suffered by the packets in the two modes are only slightly different. But overall, SRA+ is more efficient than other existing schemes regardless of traffic patterns.

Two quantities make SRA and SRA+ to be many times faster than conventional QoS schedulers. In simulations, cell delay incurred by SRA and SRA+ is several times less than that by a conventional scheduler such as one using iterative maximal matching. Secondly, SRA and SRA+ all finds a maximum matching in one iteration and their time complexity is $O(1)$, whereas existing schedulers need much more time to find a matching. To compute a matching, iterative matching schedulers such as PIM and iSLIP are shown to run $O(\log N)$ iterations to converge (their worst case time complexity is $O(N^2)$), SDA-based schedulers take $O(N^3)$ time, and GSA-based ones are $\Omega(N^2)$ times slower. Thus, compared with any of these schedulers, our SRA/SRA+ algorithms are much faster.

References

- [1] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems*, 11(4):319–352, Nov. 1993.
- [2] H. Balakrishnan, S. Devadas, D. Ehlert, and Arwind. Rate guarantees and overload protection in input-queued switches. In *Proceedings of IEEE INFOCOM 2004*, volume 4, pages 2185–2195, Mar. 2004.
- [3] C.-S. Chang, W.-J. Chen, and H.-Y. Huang. On service guarantees for input-buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann. In *Proceedings of IEEE/IFIP IWQoS '99*, pages 79–86, May 1999.
- [4] C.-S. Chang, W.-J. Chen, and H.-Y. Huang. Birkhoff-von Neumann input buffered crossbar switches. In *Proceedings of IEEE INFOCOM 2000*, pages 1614–1623, Mar. 2000.
- [5] K. F. Chen, E. H.-M. Sha, and S. Q. Zheng. Fast and noniterative scheduling for input-queued switches with unbuffered crossbars. Submitted to *Journal of Systems Architecture*.
- [6] K. F. Chen, E. H.-M. Sha, and S. Q. Zheng. A fast noniterative scheduler for input-queued switches with unbuffered crossbars. In *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2005)*, pages 230–235, Las Vegas, NV, USA, Dec. 2005.
- [7] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input/output-queued switch. *IEEE Journal on Selected Areas in Communications*, 17(6):1030–1039, June 1999.
- [8] A. Hung, G. Kesidis, and N. McKeown. ATM input-buffered switches with the guaranteed-rate property. In *Proceedings of IEEE ISCC '98*, pages 331–335, June 1998.
- [9] A. C. Kam and K.-Y. Siu. Linear-complexity algorithms for QoS support in input-queued switches with no speedup. *IEEE Journal on Selected Areas in Communications*, 17(6):1040–1056, June 1999.
- [10] C. E. Koksal, R. G. Gallager, and C. E. Rohrs. Rate quantization and service quality over single crossbar switches. In *Proceedings of IEEE INFOCOM 2004*, volume 3, pages 1962–1973, Mar. 2004.
- [11] S. Li and N. Ansari. Input-queued switching with QoS guarantees. In *Proceedings of IEEE INFOCOM '99*, pages 1152–1159, Mar. 1999.
- [12] N. McKeown. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201, Apr. 1999.
- [13] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. In *Proceedings of IEEE INFOCOM '96*, pages 296–302, Mar. 1996.
- [14] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47(8):1260–1267, Aug. 1999.
- [15] A. Mekkittikul and N. McKeown. A practical scheduling algorithm to achieve 100% throughput in input-queued switches. In *Proceedings of IEEE INFOCOM '98*, pages 792–799, Mar. 1998.
- [16] G. Nong and M. Hamdi. On the provision of integrated QoS guarantees of unicast and multicast traffic in input-queued switches. In *Proceedings of IEEE GLOBECOM '99*, volume 3, pages 1742–1746, Dec. 1999.
- [17] B. Prabhakar and N. McKeown. On the speedup required for combined input- and output-queued switching. *Automatica*, 35(12):1909–1920, Dec. 1999.
- [18] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, June 1996.
- [19] D. Stiliadis and A. Varma. Providing bandwidth guarantees in an input-buffered crossbar switch. In *Proceedings of IEEE INFOCOM '95*, volume 3, pages 960–968, Apr. 1995.
- [20] I. Stoica and H. Zhang. Exact emulation of an output queueing switch by a combined input output queueing switch. In *Proceedings of IEEE/IFIP IWQoS '98*, pages 218–224, May 1998.
- [21] M. Yang, J. Wang, E. Lu, and S. Q. Zheng. Hierarchical scheduling for DiffServ classes. In *Proceedings of IEEE GLOBECOM 2004*, volume 2, pages 707–712, Nov. 2004.
- [22] K. Y. Yun. A terabit multiservice switch. *IEEE Micro*, 21(1):58–70, Jan./Feb. 2001.