Flexible Bandwidth Allocation in Terabit Packet Switches

Dr. Aleksandra Smiljanić, AT&T Laboratories, USA

Abstract

Switches with input buffers can potentially provide high capacity, because they do not involve multiplexing at output ports, and inputs can operate at high bit-rates. We have recently proposed an algorithm for packet scheduling in high-capacity switches, termed round-robin greedy scheduling (RRGS) [8]. RRGS completely removes head-of-line (HOL) blocking at terabit capacity. However, RRGS cannot provide arbitrary bandwidth shares to input-output pairs. We propose a simple extension of RRGS, termed weighted RRGS (WRRGS), which can flexibly share the bandwidth of any output among the inputs at terabit switching capacity. We prove that WRRGS can share at least 50% of the total switch capacity. It exploits the fact that RRGS finds a maximal matching between inputs and outputs.

1 Introduction

Most generally, packet switches transfer packets from their inputs to the specified outputs. It is important to be able to flexibly share an output bandwidth among the inputs. In other words, inputs should be guaranteed to get the negotiated bandwidth even if some other inputs are overloaded. A switch with output buffers is usually a set of statistical multiplexers. Packets coming from different inputs are stored in the output buffer, and transmitted according to some scheduling policy. For example, weighted round-robin (WRR) policy would provide to the inputs their reserved bandwidth shares. But, the capacity of a switch with output buffers is limited by the speed of the output buffer. In contrast, the capacity of a switch with input buffers is not limited similarly because packets are stored at the line bit-rate. So, switches with input buffers can provide a much higher switching capacity, which is why they attract much of the interest recently [1]-[10]. In a switch with input buffers, a packet competes not only with the packets of other inputs bound for the same output, but also with the packets of the same input bound for other outputs. Several proposed protocols calculate the maximal matching between inputs and outputs that does not leave input-output pair unmatched if there is a traffic between them [1, 5, 8, 10]. However, they do not provide flexible sharing of the output bandwidth among the inputs in a switch with input buffers. Few protocols have been proposed for this purpose [1, 3, 7, 9]. We propose a new protocol which is simpler than the previously proposed ones, and can, consequently support packet switching of higher capacity. We discover that the maximal matching of inputs and outputs not only removes the head-of-line (HOL) blocking [4], but also simplifies the flexible bandwidth sharing in a switch with input buffers.

The simplest way to share the bandwidth in a switch with input buffering is to precompute a schedule in advance based on the reservations made in a connection setup phase [1]. Time is divided into frames that consist of time slots. The schedule determines input-output pairs that will be connected in each time slot of a frame. Each input-output pair is assigned a certain number of time slots within a frame, which ensures the requested bandwidth share. It can be shown that requests can be accommodated as long as

$$\sum_m a_{im} \leq F,$$

$$\sum_m a_{mj} \leq F,$$
$$0 \leq i, j \leq N-1,$$

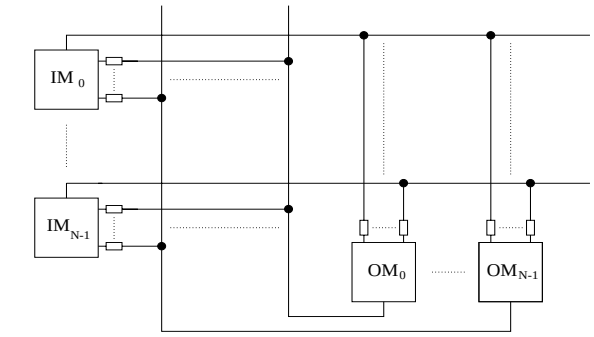
where a_{ij} is the number of time slots requested by input-output pair (i,j), F is the frame length, and N is the number of input and output ports. As a result, the bandwidth reserved for input-output pair (i,j) is $p_{ij} = a_{ij}/F$ times the line bit-rate. However, computing the schedule has a complexity on the order of $O(FN^2)$, and may become impracticable for the fast varying traffic. For this reason, Anderson et al. propose the statistical matching algorithm to arbitrarily share the switch capacity [1]. In the statistical matching algorithm, output j grants input i with probability $p_{ij} = a_{ij}/F$. Each input chooses one output from which it received a grant in a specified probabilistic way. It has been shown that the statistical matching uses 63% of the total switch capacity, or 72% if two iterations are performed. Stiliadis and Varma propose weighted probabilistic iterative matching (WPIM) instead of statistical matching [9]. They argue that the computing of several distribution functions within one time slot, as in statistical matching, becomes impractical in high-capacity switches. In WPIM, time is divided into frames, and input-output pair (i,j) is assigned a_{ij} credits within each frame. Namely, a counter associated to input-output pair (i,j) is set to $c_{ij} = a_{ij}$ at the beginning of a frame, and is decremented whenever this queue is served. Queues with positive counters compete for transmission with higher priority. They are rewarded according to the parallel iterative matching (PIM) algorithm. Remaining queues compete for the rest of the bandwidth, and they are again rewarded according to the PIM algorithm. The performance of the WPIM protocol has been assessed only through simulations. We will show that the WPIM protocol exploits at least 50% of the switch capacity. Recently, Kam et al. proposed a scheduling algorithm for flexible bandwidth reservations in a WDMA optical network with input buffering [3]. If the number of wavelengths equals the number of users, such WDMA network is equivalent to a switch with input buffering. Kam et al. also associate to each input-output queue a counter which is increased in each time slot by p_{ij}, and decreased by 1 if this queue has been served. Queues with positive counters compete for service, and they are served according to some efficient maximal weighted matching algorithm. For example, queues are considered for service in the order in which their counters decrease. Since it processes N^2 input-output pairs, this algorithm can also become a bottleneck in high-capacity switches. It was shown in [3] that this algorithm guarantees 50% of the switch capacity.

In this paper, we propose a new protocol, the weighted round-robin greedy scheduling (WRRGS), that provides flexible bandwidth sharing in switches with terabit capacity. Terabit switches involve more than 100 ports, line bit-rates as high as 10Gb/s, and processing times (equal to packet transmission times) of $10 - 100$ns. Our approach is similar to the WPIM, only it is based on the RRGS protocol instead of the PIM. In this way, the WRRGS implementation is further simplified in a comparison to the WPIM. The PIM algorithm performs $2\log_2 N + 2$ selections in order to find maximal matching, and involves the full interconnection between input and output modules of the central controller. On the other side, the RRGS algorithm performs only one selection per time slot, and involves simple structure of the central controller. So, WRRGS can potentially be used in a switch with a larger number of ports and/or higher line bit-rate, i.e. in a switch with a higher capacity. We prove that WRRGS can flexibly allocate at least 50% of the total switch capacity.

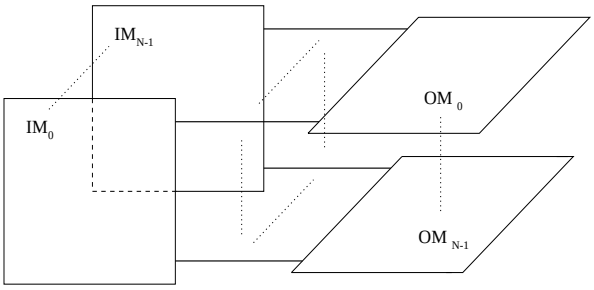
2 Weighted Round Robin Greedy Scheduling

2.1 Protocol Description

The WPIM and WRRGS protocols compare similarly as the PIM and RRGS protocols. We will briefly review them for the sake of completeness. The PIM protocol consists of several iterations: all inputs send requests to the outputs for which they have packets to send, requested outputs send acknowledge-



(a)



(b)

Figure 1: Central controller for the PIM protocol

ments to their selected inputs, and selected inputs choose one output each [1]. Inputs and outputs that have not been selected in the previous iterations compete in the next iteration in the same way. It has been shown that the PIM algorithm finds a maximal matching after no more than $\log_2 N + 3/4$ iterations [1]. Each iteration involves two selections, and all iterations have to be completed one after another within one packet transmission time. The planar and two-dimensional designs of the central controller that execute the PIM algorithm are shown in **Figure 1** (a) or (b), respectively. Each input module (IM) sends a request to each output module (OM) and each OM sends an acknowledgement to each IM. There should be $2N^2$ wires connecting input and output modules. Such central controllers may become difficult for implementation as N grows. On the other side, the RRGS protocol consists of N steps. In the first step, some particular input chooses one of the outputs for which it has packets to send. In each following step, the next input chooses one of the remaining outputs for which it has packets to send. Clearly, RRGS can be implemented by using a pipeline technique [8]. Each step of the algorithm is completed within a separate time slot, and the algorithm is completed within N time slots. But, in each time slot, all inputs choose outputs for different time slots in future, so, the central controller is calculating in parallel schedules for N future time slots. As a result, only one selection has to be performed within one time slot (other $N-1$ simultaneous selections are done in parallel). A simple structure of the central controller that executes the RRGS algorithm is shown in **Figure 2**. A round-robin (RR) arbiter associated to each input module communicates only with the RR arbiters associated to adjacent input modules, and the complex interconnection between input and output modules is avoided. It stores addresses of the reserved outputs into the memory (M). Price that RRGS pays for its simplicity is the additional pipeline delay, which is on average equal to $N/2$ time slots. This pipeline delay is not critical for assumed very short packet transmission time.

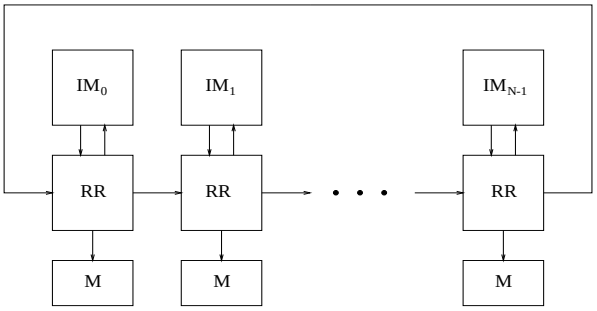


Figure 2: Central controller for the RRGS protocol

The RRGS protocol needs to be further modified in order to provide flexible sharing of the total switch capacity. We propose that time is divided into frames, and counter associated with input-output queues are set to their negotiated values at the beginning of each frame, as in WPIM. Queues with positive counters compete with higher priority according to RRGS. Then, the remaining queues contend according to RRGS for the available bandwidth.

Consider an $N \times N$ cross-bar switch, where

each input port i, $i \in \{0, 1, \cdots, N-1\}$, has N logical queues, corresponding to each of the N outputs. All packets are fixed size cells. The input of the WRRGS protocol is the state of all input-output queues, or a set $C = \{(i,j) \mid$ there is at least one packet at input i for output $j\}$. The output of the protocol is a schedule or a set $S = \{(i,j) \mid$ packet will be sent from input i to output $j\}$. Note that in each time slot, an input can only transmit one packet, and an output can receive only one packet. The schedule for the kth time slot is determined as follows:

- Step 1: If $k = 1 \bmod F$ then $c_{ij} = a_{ij}$;
- Step 2: $I_k = O_k = \{0, \cdots, N-1\}$; $i = 0$; $h = 0$;
- Step 3: Input i chooses in a round-robin fashion output, if any, j from O_k such that $c_{ij} > 0$, and $(i,j) \in C_k$;
- Step 4: Remove j from O_k and i from I_k; Add (i,j) to S_k; $c_{ij} = c_{ij} - 1$; $h = h + 1$;
- Step 5: If $h < N$ choose $i = i + 1$ and go to Step 3;
- Step 6: $i = 0$; $h = 0$;
- Step 7: If $i \in I_k$ choose j from O_k such that $(i,j) \in C_k$;
- Step 8: Remove j from O_k and i from I_k; Add (i,j) to schedule S_k; $h = h + 1$;
- Step 9: If $h < N$ go to Step 7;

In steps 1-5, prioritized packets compete for a service according to RRGS. Then, in steps 6-9, the remaining packets compete once again for the given time slot according to RRGS. Steps 6-9 are optional, they will increase the efficiency of WRRGS, but introduce an additional average pipeline delay of $N/2$ time slots. They, actually, represent a service for the best-effort traffic. Note that in RRGS input 0 is always the first to pick up an output, while in originally proposed RRGS all inputs get chance to be the first to choose an output [8]. In the latter case an input might reserve an output in the earlier time slot for the later time slot in future, in other words, it might interchangeably reserve outputs for different frames. So, each queue should be assigned multiple counters related to different frames.

2.2 Pipelined WRRGS

Let us first consider steps 1-5 of the pipelined WRRGS. WRRGS as outlined in the previous section is easy to implement by using a pipeline technique. In time slot k, input i reserves an output for time slot $k+N-i$ within frame $\lfloor (k+N-i)/F \rfloor$, where $\lfloor x \rfloor$ is the largest integer not exceeding x. Also, input i resets its counters c_{ij}, $0 \le j \le N-1$, in time slots $lF+1-N+i$, where $l \ge \lceil N/F \rceil$, and $\lceil x \rceil$ is the smallest integer not smaller than x. Time diagram for this first case of WRRGS applied in a 5×5 switch is shown in **Figure 3**. This figure shows the relation between inputs and the time slots for which they are choosing their outputs. For example, in time slot T_5, input I_1 is scheduling or choosing an output for transmission during time slot T_9, while I_3 is scheduling for time slot T_7 and so on. After it chooses an output, e.g., input I_1 forwards the control information (about available outputs) to input I_2 which reserves an output for time slot T_9 in the next time slot T_6. Bold vertical line denotes that input I_0 starts a new schedule choosing any of the outputs, i.e. it does not receive the control information from input I_4. Pipelining proposed for RRGS in [8] might be applied to WRRGS in order to equalize inputs. Time diagram for this case of WRRGS applied in a 5×5 switch is shown in **Figure 4**. Here, in each time slot another input starts a schedule. But, an input might interchangeably reserve outputs for different frames. For example, input I_0 reserves an output for time slot T_{11} in time slot T_6, and it reserves an output for time slot T_9 in the next time slot T_7. If the frame length is $F = 5$, then input I_0 interchangeably reserves outputs for frames F_1 and F_2. For a reasonable assumption that $F \ge N$, an input might interchangeably reserve outputs for at most two consecutive frames. So, each queue should be assigned two counters related to these two frames. A counter of the earlier frame will be reset every F time slots.

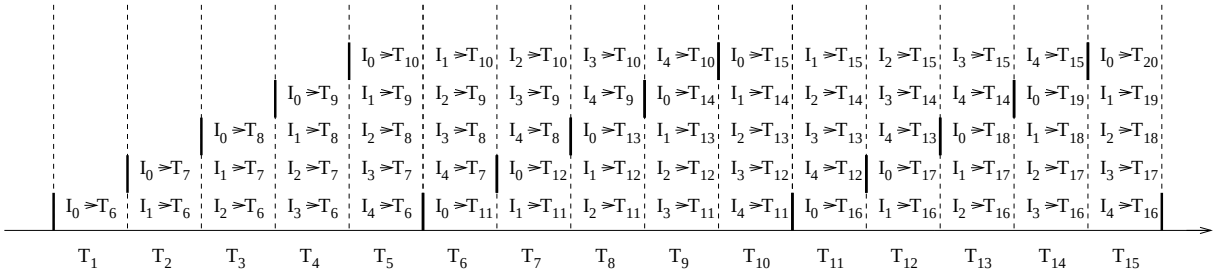


Figure 3: Time diagram for the switch controller. $N = 5$.

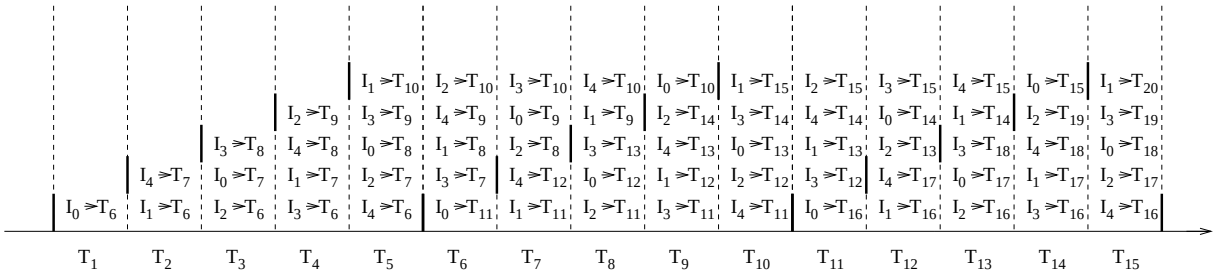


Figure 4: Time diagram for the switch controller with input equalization. $N = 5$.

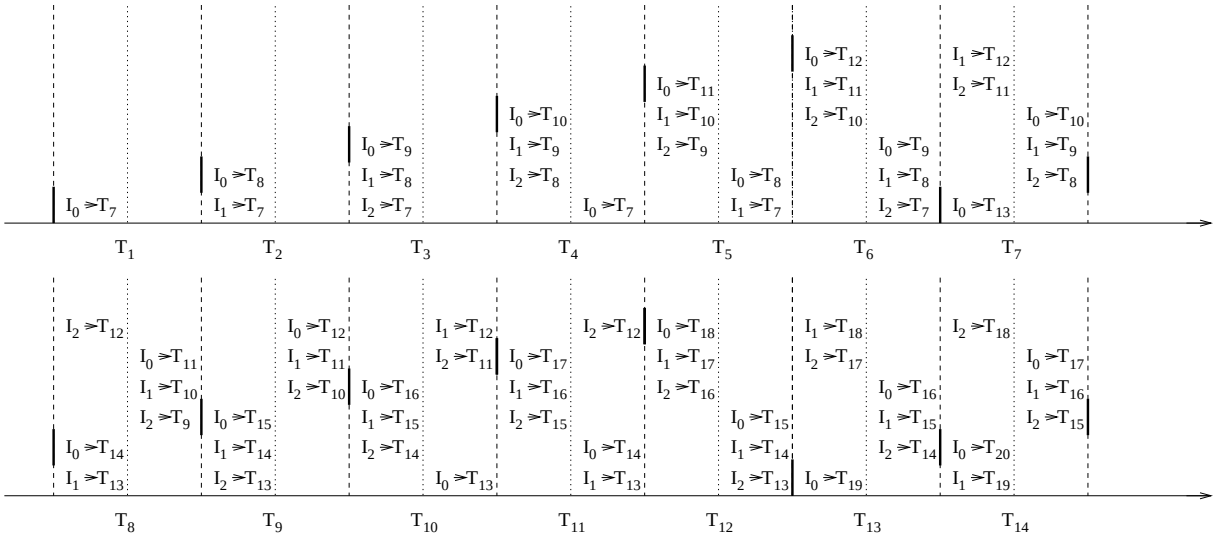


Figure 5: Time diagram for the switch controller which serves additional best-effort traffic. $N = 3$.

Depending on the future time slot for which an input reserves an output, a specified counter of the chosen queue will be decremented by one.

Let us now consider all 1-9 steps of the pipelined WRRGS, including service of the best-effort traffic. In any time slot k, each input chooses outputs for two different time slots in future, $k+N-i$ and $k+2\cdot N-i$ within frames $\lfloor (k+N-i)/F \rfloor$ and $\lfloor (k+2\cdot N-i)/F \rfloor$. First, an input reserves an output with the positive counter for time slot $k+2\cdot N-i$, then, it reserves any output for time slot $k+N-i$. Also, input i resets its counters c_{ij}, $0 \le j \le N-1$, in time slots $lF+1-2\cdot N+i$, where $l \ge \lceil 2\cdot N/F \rceil$. **Figure 5** shows the time diagram for all 1-9 steps of WRRGS applied in a 3×3 switch. For example, in time slot T_7, input I_1 chooses one of the available prioritized outputs for time slot T_{12}, and then it chooses any of the available outputs for time slot T_9. This is because input I_1 uses its first chance to schedule for time slot T_{12} in time slot T_7, and, therefore, it considers only queues with positive counters. On the other side, input I_1 uses the second chance to schedule for time slot T_9 in time slot T_7, and, therefore, it considers all queues for service. It is possible to equalize inputs assuming service of the best-effort traffic as well.

2.3 Protocol Performance

It is essential to determine the portion of the switch capacity that a scheduling algorithm can share among the inputs. More precisely, we want to determine the maximum admissible utilization, p, of any input or output line:

$$\sum_m p_{im} = \frac{1}{F}\sum_m a_{im} \le p,$$
$$\sum_m p_{mj} = \frac{1}{F}\sum_m a_{mj} \le p,$$
$$0 \le i,j \le N-1,$$

which can be guaranteed to the input-output pairs. So, if input-output pair (i,j) requests a new portion of bandwidth, Δp_{ij}, it is accepted if:

$$\sum_m p_{im} + \Delta p_{ij} \le p,$$
$$\sum_m p_{mj} + \Delta p_{ij} \le p,$$

and input-output pair (i,j) is assigned $\Delta a_{ij} = \Delta p_{ij}\cdot F$ new time slots. We will prove that $p = 0.5$ for the WRRGS, due to the fact that the RRGS finds a maximal matching between inputs and outputs.

Theorem: The WRRGS protocol ensures a_{ij} time slots per frame to input-output pair (i,j), $0 \le i,j \le N-1$, if the following condition holds:

$$\sum_m p_{im} \le \frac{1}{2} \Leftrightarrow \sum_m a_{im} \le \frac{F}{2},$$
$$\sum_m p_{mj} \le \frac{1}{2} \Leftrightarrow \sum_m a_{mj} \le \frac{F}{2}. \qquad (1)$$

Proof: We are viewing only prioritized packets, as if WRRGS consists only of steps 1-5. Observe time slots within a frame in which either input i or output j are connected, but they are not connected to each other. In each of these time slots, sum $s_{ij} = \sum_{m\ne j} c_{im} + \sum_{m\ne i} c_{mj}$ is decremented by at least 1. Sum s_{ij} is the largest at the beginning of a frame and from (1), it fulfills:

$$s_{ij} = \sum_{m\ne j} a_{im} + \sum_{m\ne i} a_{mj} \le F - 2a_{ij}. \qquad (2)$$

As a conclusion, in at least $2a_{ij}$ time slots per frame neither input i is connected to some output other than j, nor output j is connected to some input other than i. In at least a_{ij} of these time slots (in which $c_{ij} > 0$), when input i selects an output, it will choose output j if there are packets in queue (i,j). This is because none of the inputs have chosen output j before input i, and input i is not choosing any other output. Therefore, input i will choose output j as supposed by RRGS, and by any other algorithm that finds a maximal matching between inputs and outputs. In summary, if condition (1) is fulfilled then a_{ij} time slots per frame are guaranteed to input-output pair (i,j). □

The above theorem holds for the WPIM as well, considering the fact that PIM finds a maximal matching between inputs and outputs.

Admission control in WRRGS is simple, new Δa_{ij} time slots are assigned to input-

output pair (i, j) if:

$$\sum_m a_{im} + \Delta a_{ij} \leq \frac{F}{2},$$
$$\sum_m a_{mj} + \Delta a_{ij} \leq \frac{F}{2}. \quad (3)$$

Central controller does not have to precompute schedule when a new request is admitted. Only input i has to update the value of $a_{ij} \leftarrow a_{ij} + \Delta a_{ij}$, $0 \leq j \leq N-1$, in order to set the correct counter value $c_{ij} = a_{ij}$ at the beginning of each frame. Consequently, WRRGS can follow fast changes of traffic pattern.

3 Conclusion

We presented very simple way to flexibly share bandwidth in switches with input buffering. The simplicity of the proposed protocol makes it attractive for switching of several Tb/s, assuming the current technology. We have also shown that the proposed WRRGS can share at least 50% of the total switch capacity.

WRRGS has several desirable features. First, WRRGS algorithm can serve traffic with fast varying bandwidth requirements such as variable bit rate (VBR) traffic. For example, bit-rate of video streams changes on a millisecond time-scale. Second, WRRGS requires simple processing: only two round-robin choices are to be performed within one time slot. So, it can switch short cells transmitted at high bit-rates. In addition, a linear structure of the central controller easily scales to accommodate a large number of input and output ports, and provide high switching capacity.

Acknowledgements: I am thankful to G. Ramamurthy and R. Fan at NEC USA Inc. for the problem formulation and useful discussion. I also thank to X. Qiu at AT&T Labs for her interest and comments.

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,* vol. 11, no. 4, November 1993, pp. 319-352.

[2] H. J. Chao, C. H. Lam, and X. Guo, "A fast arbitration scheme for terabit packet switches," *Proceedings of GLOBECOM*, December 1999, pp. 1236-1243.

[3] A. C. Kam, K. Y. Siu, R. A. Barry, and E. A. Swanson, "A cell switching WDM broadcast LAN with bandwidth guarantee and fair access," *IEEE/OSA Journal on Lightwave Technology*, vol. 16, no. 12, December 1998, pp. 2265-2280.

[4] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input vs. output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. COM-35, no. 12, December 1987, pp. 1347-1356.

[5] N. McKeown *et al.*, "The Tiny Tera: a packet switch core," *IEEE Micro*, vol. 17, no. 1, Jan.-Feb. 1997, pp. 26-33.

[6] A. Mekkittikul, and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," *Proceedings of INFOCOM*, March 1998, pp. 792-799.

[7] R. Schoenen, and R. Hying, "Distributed cell scheduling algorithm for virtual-output-queued switches," *Proceedings of GLOBECOM*, December 1999, pp. 1211-1215.

[8] A. Smiljanić, R. Fan, and G. Ramamurthy, "RRGS-Round-Robin Greedy Scheduling for electronic/optical terabit switches," *Proceedings of GLOBECOM*, December 1999, pp. 1244-1250.

[9] D. Stiliadis, and A. Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch," *Proceedings of INFOCOM*, March 1995, pp. 960-968.

[10] T. Weller, and B. Hajek, "Scheduling nonuniform traffic in a packet switching system with small propagation delay," *Proceedings of INFOCOM*, 1994, pp. 1344-1351.