Exact Emulation of an Output Queueing Switch by a Combined Input Output Queueing Switch

Ion Stoica, Hui Zhang
Carnegie Mellon University
Pittsburgh, PA 15213
e-mail: {istoica,hzhang}@cs.cmu.edu

Abstract—

Combined input output queueing switches (CIOQ) have better scaling properties than output queueing (OQ) switches. However, a CIOQ switch may have lower switch throughput, and more importantly, it is difficult to control delay in a CIOQ switch due to the existence of multiple queueing points. In this paper, we study the following problem, originally formulated and studied by Prabhakar and Mckeown [16]: Can a CIOQ switch be designed to *behave identically* to an OQ switch? In [16], an algorithm was proposed so that a CIOQ switch with an internal speedup of four can behave identically to an OQ switch with FIFO as the output queueing discipline. In this paper, we propose a new switch scheduling algorithm called Joined Preferred Matching (JPM) that improves Prahhakar and Mckeown's results in two aspects. First, with JPM, the internal speedup needed for a CIOQ switch to achieve exact emulation of an OQ switch is only 2 instead of 4. Second, the result applies to OQ switches that employ a general class of output service disciplines, including FIFO and various Fair Queueing algorithms [1]

This result lays the theoretical foundation for designing scalable high-speed CIOQ switches that can provide same throughput and QoS as OQ switches, but require lower speed internal memory.

Keywords: QoS guarantees, Speedup, Output Queueing, Combined Input Output Queueing

This research was sponsored by DARPA under contract numbers N66001-96-C-8528 and N00174-96-K-0002, and by a NSF Career Award under grant number NCR-9624979. Additional support was provided by Intel Corp., MCI, and Sun Microsystems. Views and conclusions contained in this document are those of the authors and should no be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel, MCI, Sun, or the U.S. government.

[1] We note that Chuang *et al* have independently come up with similar results [6].

I. Introduction

Due to its conceptual simplicity, output queueing (OQ) represents a natural way to design an $N \times N$ communication switch. In an OQ switch, when a packet arrives at an input port, it is immediately put into the buffer that resides at the corresponding output port. Since buffering and queueing happen only at output ports in an OQ switch, it is possible to design output queue scheduling algorithms that provide various QoS guarantees [20].

However, OQ has fundamental scaling limitations. Because packets destined for the same output port may arrive simultaneously from many input ports, the output buffer needs to enqueue traffic at a much higher rate than a single port may dequeue it. In the worst case, N (the number of line cards in the switch) packets could arrive in the amount of time a port could send one. This requires that the memory bandwidth and control systems speed to scale as a function of the number of cards in the switch, which places stringent limits on the system size.

In order to reduce cost and simplify implementation, most high performance switches (both research [5], [14] and commercial [7], [8] have chosen architectures employing some form of input buffering. By having buffers at input ports, it is possible to build high performance switches with speedup much smaller than N, where the speedup is defined as the ratio of the line card's bandwidth into/from the switch core to the link speed.

Buffering at the input changes the contention problem inside the switch. While contentions only happen at output links in an output buffered switch, they also happen at input and output cards in an input buffered switch – multiple packets from the same input card may be destined to the different output cards and multiple packets from different input cards may

be destined to the same output card. If the input buffer is FIFO, there is no contention at input cards, but it introduces the problem of Head-of-Line (HOL) blocking [13]: if the packet at the head of the queue is blocked due to contention of the output card, packets that are on the same input card but destined to other contention-free output cards cannot be forwarded. By maintaining at an input card a separate queue for each output card [1], the HOL problem can be eliminated. Additional flexibility can be obtained by having buffering at both input and output cards [5], [7], [8], [14], [17]

Most of the early studies on input queueing (IQ) and CIOQ switches have focused on the throughput achievable by these switches with various speed-ups and under different workloads [2], [4], [10], [12], [15]. While it is important to achieve switch throughput, it is also critical to provide QoS guarantees, either for each individual flow, or for both individual flows and traffic aggregates.

There are several recent studies on how to provide QoS guarantees in CIOQ switches. In [16], Prabhakar and Mckeown considered the following problem: is it possible to construct a CIOQ switch that behaves identically to an OQ switch? They proposed an algorithm, called the the most urgent cell first algorithm (MUCFA), and showed that it can identically emulate an OQ switch that employs FIFO output queueing scheduler under any arrival pattern as long as the speedup is no less than 4. In [3], [19], Charny et al. and Stephens and Zhang studied a similar problem. Their focus was not to emulate the exact behavior of an OQ switch, but to provide QoS guarantees that are comparable to those provided by an OQ switch. Charny et al. assume a switch architecture employing maximal matching algorithms. Stephens and Zhang assume a switch architecture with buffered crossbar that can operate with variable packet sizes.

In this paper, we study the Prabhakar and Mckeown's problem and propose an algorithm called Joined Preferred Matching (JPM) that improves their results in two aspects. First, with JPM, the internal speedup needed for a CIOQ switch to achieve exact emulation of an OQ switch is only 2 instead of 4. Second, the result applies to any OQ switch that employs a *monotonic* and *work-conserving* output scheduling discipline. A scheduling algorithm is monotonic if the arrival of a new packet does not change the relative scheduling order of the packets already enqueued. Many of the widely used scheduling disciplines, such as First-In-First-Out (FIFO) and various Fair Queueing algorithms, are monotonic and work-conserving. We have recently learned that the same problem has been independently addressed by Chuang *et al.* in [6] with an algorithm similar to JPM. In addition, they show that a speedup of $2-1/N$ is both a necessary and sufficient condition for a FIFO OQ switch in which a slot consists of either one or two phases.

Similarly to [16] we assume that only fixed sized packets, also called *cells*, are transfered inside the switch. This assumption is supported by the design of many of today's high speed switches, where the variable length packets are segmented in cells when they arrive and are reassembled before they depart. Also we assume that the time is divided in slots, and during each slot at most one cell can arrive at an input, and at most one cell can depart from an output. Similarly, in a CIOQ switch with speedup S at most S cells can be removed from an input, and at most S cells can arrive at an output. Finally, for convenience we assume that one slot is divided in S phases (or sub-slots), so that during each phase at most one cell can be removed from an input and at most one cell can arrive at an output.

II. Algorithm Description

To each input and output we associate a preference list. Based on these preference lists at the beginning of every phase we perform a stable matching [9] between inputs and outputs. Then, similarly to [16] we use this matching to transfer cells from inputs to outputs.

Next, we define the notions of input and output preference lists, which are used to match inputs with outputs.

Definition 1: The **preference list** of an **input** represents the list of cells at that input ordered in the inverse order of their arrival times.

Definition 2: The **preference list** of an **output** represents the list of cells from all inputs that should be forwarded to that output, ordered by their scheduling times in the corresponding OQ schedule. Ties are broken by the index of the input where the cell is enqueued.

As outlined above, at the start of each phase we perform a *stable* matching. A matching is an one-one correspondence between inputs and outputs. An in-

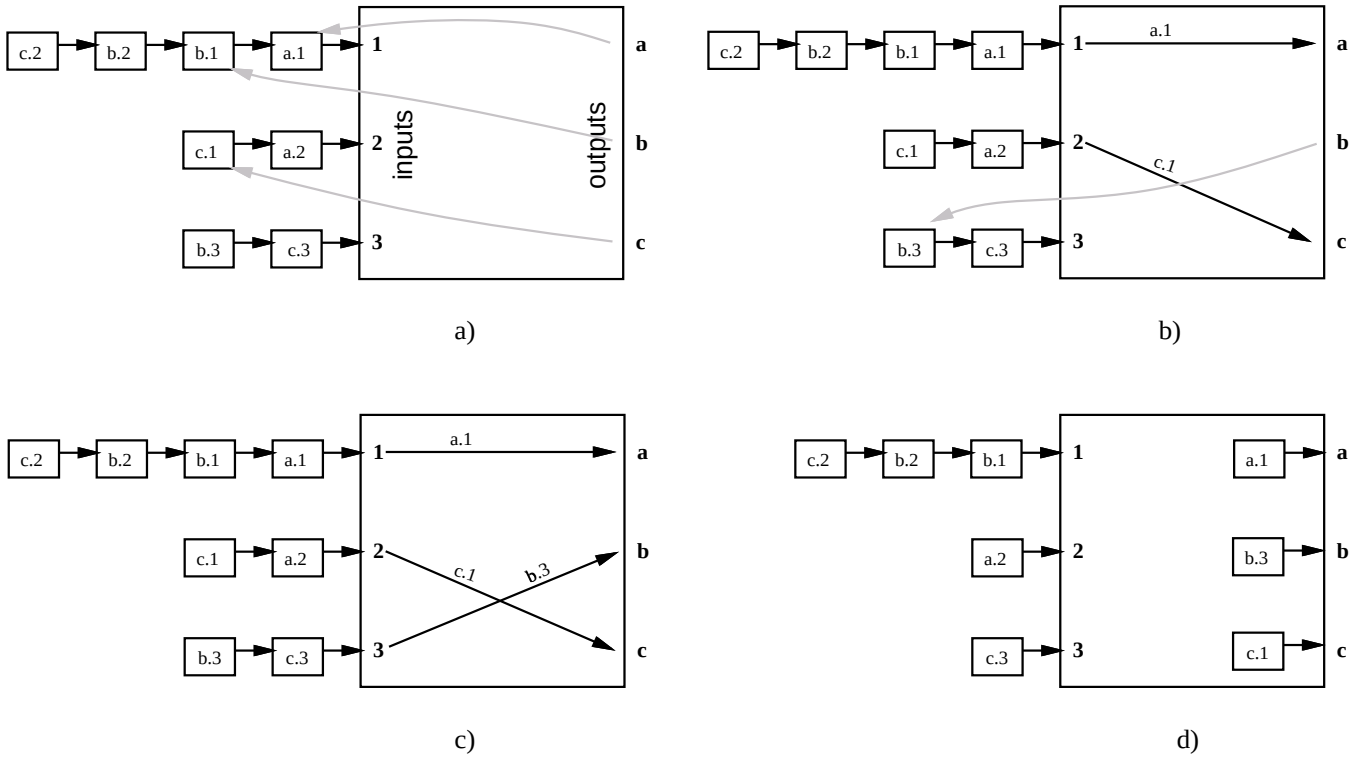

Fig. 2. Example to illustrate the Gale-Shapeley algorithm to compute a stable matching for a 3×3 switch. The letter in each cell denotes the output to which the cell is destined, while the number indicates its order in the output schedule. The light arrows represent the output requests, and the dark arrows represent input-output matching pairs.

while there are unmatched outputs that where not rejected by all inputs **do**

  each unmatched output requests its most preferred cell from an input that has not rejected it yet;

  each input grants the request to the output with the most preferred cell;

Fig. 1. The Gale-Shapley algorithm to compute a stable matching.

put i and output j are said to block a matching M, or to be a *blocking pair* for M, if i and j does not match in M, but both i and j prefer each other to their current match in M. A matching that has no blocking pair is called *stable*; otherwise it is called *instable*. To compute a stable matching we use the Gale-Shapely algorithm [9]. This algorithm is summarized in Figure 1.

Figure 2 illustrates how the matching algorithm works for a 3×3 switch. The letter in each cell denotes the output port where the cell should be forwarded, while the number denotes its order in the preference list of that output. The light arrows indicate the requests made by outputs, while the dark arrows represent the requests granted by inputs. During the first iteration each output asks for its most preferred cell enqueued at the inputs (see Figure 2(a)). In turn, input 2 grants the only request it receives to output c, while input 1 grants the request corresponding to its most preferred cell, i.e., the request issued by output a for cell $a.1$. Thus, after the first iteration outputs a and c are matched to their most preferred cells. During the next iteration, the unmatched output b requests its most preferred cell from input[2] 3 (see Figure 2(b)). As a result, input 3 grants the output b's request and the matching completes. Figure 2(d) shows the switch's state after cells are transferred ac-

[2]Note that since output b was rejected in the first iteration by input 1, it does no longer consider this input in the current iteration.

cording to the resulting matching.

A shown in [11], the stable matching problem has a lower bound of $\Omega(N^2)$, where N is the number of inputs/outputs. By using ranking arrays for expressing the preference of both inputs/outputs it can be shown that the Gale-Shapely algorithm can be implemented in N^2 time [11]. However, in order to construct the ranking arrays efficiently we need to maintain for each input, besides its preference list, a virtual list associated to each output which contains all cells destined to that output, ordered by their schedule times in the corresponding OQ schedule. Furthermore, it can be shown that the average time complexity of the Gale-Shapely algorithm is $O(N \log N)$. Finally, in the context of addressing the same problem Chuang *et al* [6] have shown that it is possible to reduce the matching complexity to N, by carefully considering only a sub-set of cells from the preference lists.

III. Algorithm Analysis

In this section we prove that a CIOQ switch with a speedup $S \geq 2$ operating under JPM behaves identically to an OQ switch that employs a monotonic and work-conserving scheduling discipline (Theorem 1). The main idea behind the proof is to establish a sufficient condition such that Theorem 1 holds (Lemma 2), and then to find an invariant that makes the sufficient condition true (Lemma 3).

As noted in Section I we consider a switch model in which every slot is divided in S phases. During each phase we compute a stable matching which is used to transfer cells from inputs to outputs. We assume that a new cell arrives at the beginning of a slot, before the first phase starts, and that a cell is transmitted at the end of the slot, after all S phases complete. Also, we assume that if time t represents the starting time of a slot, the preference lists do *not* include the cells that eventually arrive in that slot. For simplicity, throughout this section we assume *unit* time slots. Finally, whenever it is clear from the context a slot that starts at time t is simply referred as slot t. We start with two simple definitions.

Definition 3: Let p be a cell at input i that needs to be transferred to output j. Then we define:

- $rank(p, t)$ – the number of cells at output j that are ahead of cell p in the OQ schedule.
- $pos(p, t)$ – the position of cell p at time t in the preference list of input i. (The cell positions in the preference list are assumed to be one-indexed.)

As noted in [16], input and output contention are the *only* reasons for which a cell p is not transferred from its input i to its output j during a phase. Input contention happens when input i chooses to send a cell more preferred than p, while output contention happens when output j receives a cell more preferred than p during that phase. The next result relates the rank and the position of a cell p that is not transferred during a time slot.

Lemma 1: Consider a CIOQ switch operating under JPM with speedup S, and let t_0 be the starting time of an arbitrary slot. Then, for any cell p that has arrived in a previous slot and which is not transferred during slot t_0, we have

$$rank(p, t_0 + 1) - pos(p, t_0 + 1) \geq rank(p, t_0) - post(p, t_0) + S - 2. \quad (1)$$

Proof. Recall that if a cell p is not transferred during a phase this is due to either *input* or *output* contention. If there is input contention this means that a more preferred cell at that input is transferred, and therefore the position of cell p decreases by one. On the other hand, if cell p is not transferred due to output contention, this means that a more preferred cell was transferred to that output, and therefore according to Definition 3 the rank of cell p increases by one. Finally, note that in the particular case when a cell q more preferred than p is transferred from input i to the same output j (i.e., cell q is also more preferred than p by output j) the rank of p increases by one, while its position decreases by one. Thus, in either of these cases, the difference between cell's rank and its position increases by at least one.

Since cell p is not transferred during the entire slot, and since there are S phases during each slot, the difference between the rank of cell p and its position increases by S.

In addition, at most one cell is received by input i, and at most one cell is transmitted by output j during the slot starting at t_0. According to Definition 1 a new cell that eventually arrives at input i will become the most preferred cell of that input, and consequently will increase the position of p by *one*. At the same time, note that since the OQ scheduling discipline is

assumed to be monotonic a new arrival cell will *not* change the rank of p. However, a cell that is eventually transmitted by output j decreases the rank of cell p by *one*, which concludes the proof. □

The next lemma gives a sufficient condition under which a CIOQ switch behaves identically to an OQ switch.

Lemma 2: A CIOQ switch operating under JPM behaves identically to an OQ switch that employs a work-conserving and monotonic scheduling discipline, regardless of the input traffic pattern, if at the *beginning* of every time slot the ranks of all cells[3] at the inputs are greater or equal to 1.

Proof. Assume this is not true, i.e., the ranks of all cells at the inputs are always greater or equal to 1, but there is a cell q that is not scheduled at the same time in the CIOQ switch as in the corresponding OQ switch. Let t_0 be the starting time of the slot in which cell q *should* be transmitted in the OQ switch, and let t be the starting of the slot during which q is actually transmitted in the CIOQ switch. Consider two cases, whether $t < t_0$, or $t > t_0$, and let j be the output to which cell p is destined.

If $t < t_0$, let p be the cell that is transmitted during the slot starting at t by output j in the corresponding OQ switch. Since the output scheduling discipline is work conserving, and since cell p is presented in the system at time t, but it is not transmitted until time $t_0 > t$, we are guaranteed that such cell exists. But then cell p will miss its schedule (because cell q is transmitted instead), which contradicts our assumption that q is the *first* cell that is not scheduled at the same time in the CIOQ switch as in the corresponding OQ switch.

On the other hand, if $t > t_0$, it is easy to see that cell q does not reach output j during the slot starting at t_0. To see why, assume this is not true. Since the output scheduling discipline is assumed to be work-conserving, the only reason for which cell q is not transmitted during slot t_0 is because another more preferred cell p is being transmitted. But this means that p is also late (it should have been sent during a previous slot), which again violates our assumption that q is the first cell to miss its schedule. Thus, at time t_0, cell q will be enqueued at input i, and there is *no* more preferred cell than q at output j. But then according to Definition 3 we have $rank(q, t_0) = 0$, which contradicts the hypothesis, and therefore proves the lemma. □

The following lemma establishes an invariant for the JPM algorithm, invariant which makes the assumption of Lemma 2 true.

Lemma 3: Consider a CIOQ switch with speedup $S \geq 2$ operating under JPM. Let t be the starting time of an arbitrary slot, and let p be an arbitrary cell enqueued at its input. Then, at the beginning of every slot t, we have

$$rank(p, t) \geq pos(p, t). \tag{2}$$

Proof. The proof is by induction on the starting times of the slots during which a cell is enqueued at its input. (Recall, that a cell arriving in slot t_0 is assumed to *not* be enqueued at time t_0.)

Basic Step. Assume cell p, destined to output j, arrives at input i in slot t_0. Further, assume that cell p is not transferred[4] during slot t_0. Since p is the most preferred cell of input i during slot t_0, it follows that the only reason for which p is not transferred in this slot is because there is contention its output j (i.e., a more preferred cell is transferred to that output instead) during every phase of slot t_0. Since there are S such phases and during the entire slot at most one cell is transmitted from output j, it follows that at t_0+1, output j will hold at least $S-1$ cells which are more preferred than p. This yields

$$rank(p, t_0 + 1) \geq S - 1 \geq 1. \tag{3}$$

Since by definition $pos(p, t_0+1) = 1$, the proof for the basic step follows.

Induction Step. Assume (2) is true at time t for any cell p enqueued at an input. Then according to Lemma 1, if cell p is not transferred in slot t, we have

$$\begin{aligned} rank(p, t+1) - pos(p, t+1) \quad &\geq \\ rank(p, t) - pos(p, t) &\geq 0, \end{aligned} \tag{4}$$

[3] Recall that this does not include the cell that is eventually received during the current slot.

[4] Otherwise, since cell p is not enqueued at its input the proof is trivially true.

which concludes the proof. □

Since, by Definition 3, for any cell p which is enqueued at the beginning of slot t we have $pos(p,t) \geq 1$, from Lemmas 2 and 3 follows the main result of the paper.

Theorem 1: Under an arbitrary traffic pattern, a CIOQ switch with speedup $S \geq 2$ operating under JPM behaves identically to an OQ switch that employs a monotonic and work-conserving scheduling discipline.

The above results gives a sufficient condition for emulation an OQ switch by using a CIOQ switch. An equally important question is whether this condition is also necessary. In other words: Is there any algorithm that can emulate an OQ switch by using a CIOQ switch with speedup smaller than two ? In our model where each slot consists of an integer number of phases during which only fixed size packets are sent, the answer is no. This is because in such a model the speedup is an integer and it can be easily shown that a speedup of one is not enough. To see this consider a 2×2 switch where during the first slot each input receives a cell for output 1, and during the second slot input 2 receives a cell for output 2. In addition, assume that when two cells arrive during the same slot for the same output, ties are broken by the input index. In our case this means that the cell that arrives at input 1 during the first slot should depart first from output 1. Then it is easy to see that having a speedup of one is not sufficient. This is because during the first slot only the cell from input 1 is transferred to output 1. This leaves input 2 with two cells to be transferred during the same slot, which is impossible with the speedup of one. Moreover, it can be shown that even in a more general model where a phase duration is a fraction of a slot, and where a cell can be transferred during a phase only if it is already enqueued at the beginning of that phase, a speedup of two is an asymptotic lower bound [18]. More precisely, in [6] it is shown that in a switch model in which a slot can consists of either one or two phases, a speedup of $2 - 1/N$ is a necessary condition.

IV. Summary

We have proposed a scheduling algorithm called JPM that allows a combined input output queueing (CIOQ) switch to emulate exactly an output queueing (OQ) switch. We extend previous results in two aspects. First, we reduce the internal speedup required from 4 to 2. Second and more important, while previous work tries to emulate OQ switches that use FIFO as output scheduling discipline, our algorithm emulates OQ switches that use a general class of output scheduling algorithms, including various Fair Queueing algorithms. This enables CIOQ switches not only to provide the same throughput, but also to support the same QoS guarantees as OQ switches, while requiring a much lower speed of the internal memory.

V. Acknowledgement

We are grateful to Shang-Tse Chuang, Ashish Goel, Nick McKeown and Balaji Prabhakar for pointing out an error in a previous version of our algorithm. Also, we would like to thank Donpaul Stephens for many helpful discussions that greatly enhanced our understanding of the problem and the quality of our paper.

References

[1] T. Anderson, S. Owicki, J. Saxe, and C. Thacker. High speed switch scheduling for local area networks. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, MA, October 1992.

[2] C-Y. Chang, A. J. Paultraj, and T. Kailath. A Broadcast Packet Switch Architecture with Input and Output Queueing. In *Proceedings of Globecom '94*, pages 448–452, 1994.

[3] A. Charny, P. Krishna, N. Patel, and R. Simcoe. Algorithms for Providing Bandwidth and Delay Guarantees in Input-Buffered Crossbar with Speedup. In *IWQoS'98*, Napa, CA, May 1998.

[4] J.S. Chen and T. E. Stern. Througput analysis optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch. *IEEE Journal on Selected Areas in Communication*, 9(3):439–449, September 1991.

[5] F.M. Chiussi, Y. Xia, and V.P. Kumar. Backpressure in shared-memory-based atm switches under multiplexed bursty sources. In *Proceedings of IEEE INFOCOM'96*, pages 830–843, San Francisco, CA, March 1996.

[6] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching Output Queueing with a Combined Input Output Queued Switch, April 1998. Technical Report, CSL-TR-98-758.

[7] Ascend Communications. GRF family of switches. www.ascend.com.

[8] Digital Equipment Corporation. GIGAswitch. www.networks.digital.com.

[9] D. Gale and L. S. Shapley. College Admissions and the Stability Marriage. *Mathematical Monthly*, 69:9–15, 1962.

[10] A. L. Gupta and N. D. Georganas. Analysis of a Packet Switch with Input and Output Queueing. In *Proceedings of IEEE INFOCOM'91*, pages 694–700, Bal Harbour, FL, April 1991.

[11] D. Gusfield and R. W. Irving. The Stable Mariage Problem: Structure and Algorithms, MIT Press. 1989.

[12] I. Iliadis and W. E. Denzel. Performance of Packet Switches with Input and Output Queueing. In *Proceedings of ICC '90*, pages 747–753, Atlanta, GA, April 1990.

[13] M. J. Karol, M. G. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, 35(12):1347–1356, December 1987.

[14] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz. The Tiny Tera: A Packet Switch Core. *IEEE Micro*, pages 26–33, January 1997.

[15] Y. Oie, M. Murata, and H. Miyahara. Effects of Speedup in Nonblocking Packet Switch. In *Proceedings of ICC '89*, pages 410–414, Boston, MA, June 1989.

[16] B. Prabhakar and N. McKeown. On the Speedup Required for Combined Input and Output Queued Switching. *manuscript*, 1997.

[17] R. Simcoe and T. Pei. Perspectives on ATM Switch Architecture and the Influence of Traffic Patterns on Switch Design. *Computer Communication Review*, 25(2):93–105, April 1995.

[18] D. C. Stephens, February 1998. Personal communication.

[19] D.C. Stephens and H. Zhang. Implementing distributed packet fair queueing in a scalable switch architecture. In *INFOCOM'98*, pages 282–290, San Francisco, CA, March 1998.

[20] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10):1374–1399, October 1995.