Static Round-Robin Dispatching Schemes for Clos-Network Switches[1]

Konghong Pun Mounir Hamdi
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract- **The Clos-network is widely recognized as a scalable architecture for high-performance switches and routers. One of the key challenges in designing a Clos-network switch for a high-speed environment is the design of the dispatching/scheduling so as to be efficient for a wide range of traffic patterns, yet practical to be implemented in hardware. Based on the Static Round-Robin scheduling technique, we propose the SRRD cell dispatching algorithm and its variants for Clos-network switches in this paper. Our algorithms are based on the request-grant-accept (RGA) handshaking scheme, which can be implemented using simple distributed arbiters at the input and output of the Clos-network. The intuition behind our SRRD schemes is to desynchronize the pointers of the arbiters in a static way and by to use a rotating-search technique to improve the performance under non-uniform traffic. Our simulation results will demonstrate that our algorithms achieve the lowest delay and highest throughput among all other related schemes. In addition, their hardware implementations seem to be even simpler than that of related algorithms.**

I. Introduction

Most high-performance Internet backbone routers today are built based on a crossbar switch with a centralized scheduler. Several practical and effective crossbar switches along with the appropriate scheduling have been proposed [7] [8]. However, the complexity of switching hardware and scheduling algorithms usually depends on the square of the number of switch ports. This makes them difficult to scale to a large size in a cost-effective way. As a result, switch architectures based on the three-stage Clos-network are very attractive due to their modularity and scalability. But one of the challenges is how to design a distributed scheduling algorithm.

Clos-network switch architectures can be categorized into two types. The first one has buffers in the second-stage, such as the WUGS architecture in [2]. The function of the buffers is to resolve contention among cells from different first-stage modules. However, cells may be mis-sequenced at the output ports. It requires a re-sequence function, which is difficult to implement when the port speed increases. The second type of architecture has no buffers in the second-stage, such as the ATLANTA switch in [3]. This approach is more promising and will be considered in this paper.

The concurrent dispatching (CD) in the ATLANTA switch is a random-based scheduling algorithm [3] [4]. It can fully distribute traffic evenly to the central modules but the contention cannot be avoided. This is similar to the PIM algorithm for crossbar switches [7]. In particular, the CD algorithm cannot achieve a high throughput unless the internal bandwidth is expanded.

In crossbar switches, round-robin arbitration has been developed to overcome the throughput limitation of the PIM algorithm, such as FIRM and iSLIP [8]. Similarly, the concurrent round-robin dispatching (CRRD) and the concurrent master-slave dispatching (CMSD) schemes have been recently proposed for Clos-network switches in [5] and [6]. They have been shown to achieve 100% throughput under uniform traffic with no buffers and no bandwidth expansion in the second-stage modules.

Recently, a novel scheduling algorithm termed *Static Round-Robin* (SRR) has been proposed for crossbar switches [1]. The SRR has been shown to considerably improve the performance of most round-robin arbiters in crossbar switches using an even simpler hardware implementation. Based on this technique, we propose the *Static Round-Robin Dispatching* (SRRD) algorithm and its variants for Clos-network switches in this paper. The intuition behind the SRR design is to desynchronize the arbiters' pointers in a static way and to use a rotating-search technique to improve the performance under non-uniform traffic.

The rest of this paper is organized as follows. Section II introduces some background knowledge, including the switch model, CD, CRRD and CMSD schemes. Section III describes three SRRD-based scheduling algorithms. Section IV analyzes the performance of the proposed algorithms and their hardware implementation. Finally, we conclude this paper in section V.

II. Background Knowledge

A. Switch Model

The switch architecture used in this paper is based on [3], and is shown in Figure 1. The input and output stages are both composed of shared-memory modules, each with n port interfaces. They are fully interconnected through a central stage that consists of bufferless crossbars of size k x k. In the switch, there are k input modules (IM), m central modules (CM), and k output modules (OM).

An $OM(j)$ has n buffered output ports, $OP(j,h)$. Each output port buffer can receive at most m cells from m central modules and send at most one cell to the output line at one timeslot.

An $IM(i)$ has nk virtual output queues, $VOQ(i,j,h)$, for storing cells that go from $IM(i)$ to $OP(j,h)$ at $OM(j)$. Each virtual output queue can receive at most n cells from n input ports and send one cell to the central module.

An $IM(i)$ has m output links, $LI(i,r)$, connecting to each $CM(r)$. An $CM(r)$ has k output links, $LC(r,j)$, connecting to each $OM(j)$.

[1] This research work was partly supported by a grant from the Hong Kong Research Grant Council under the grant HKUST6181/01E.

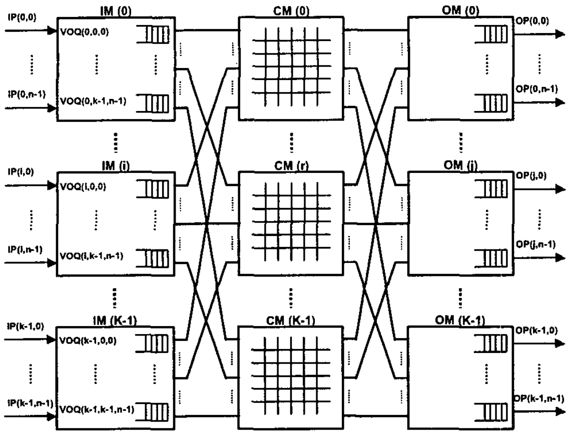

Figure 1. The Clos-network Switch Model

B. *Concurrent Dispatching (CD)*

The distributed architecture implies the presence of multiple contention points in the switch. The ATLANTA switch proposed the CD algorithm with highly distributed nature [3]. It works as follows.

In each timeslot, each IM randomly selects up to *m* VOQs and randomly sends the requests to CMs. If there is more than one request for the same output link in a CM, it grants one request randomly. Finally, the granted VOQs will send to the corresponding OP in the next timeslot.

The original CD algorithm in [3] applies a backpressure mechanism in the dispatching process. We only describe its basic concept and characteristic in this paper. We also assume that the buffer size in IMs and OMs is large enough to avoid cell loss. Hence we can focus the discussion on the properties of the dispatching algorithms.

C. *Concurrent Round-Robin Dispatching (CRRD)*

The CRRD has been proposed in [5] to overcome the throughput limitation of the CD. It is based on the request-grant-accept (RGA) handshaking scheme. In Phase 1, it employs an iterative matching between VOQs and output links in each IMs. Phase 2 then performs contention control for the output links in CMs.

Initialization:

Each VOQ(i,j,h) is associated with an arbiter V(i,j,h) with pointer PV(i,j,h). Each LI(i,r) is associated with an arbiter L(i,r) with pointer PL(i,r). Each LC(r,j) is associated with an arbiter C(r,j) with pointer PC(r,j). VOQs are arranged in an order of VOQ(i,v), where v = hk+j. Set PV(i,j,h) = PL(i,r) = PC(r,j) = 0.

Phase 1: Iteratively Matching within IM:

Step 1: Request. Each un-matched, non-empty VOQ(i,j,h) sends a request to every output link arbiter L(i,r).

Step 2: Grant. Each output link arbiter L(i,r) search one request in a round-robin fashion starting from pointer PL(i,r) and sends the grant to the selected VOQ.

Step 3: Accept. Each VOQ arbiter V(i,j,h) search one grant in a round-robin fashion starting from pointer PV(i,j,h) and sends the accept to the selected output link LI(i,r).

Phase 2: Matching between IM and CM:

Step 1: Request. Each IM output link LI(i,r), who was accepted by a VOQ(i,j,h) in Phase 1, sends the request to the CM output link arbiter C(r,j).

Step 2: Grant. Each CM output link arbiter C(r,j) search one request in a round-robin fashion starting from pointer PC(r,j). It then sends the grant to the selected IM and update the pointer PC(r,j) to one position beyond the grant one.

Step 3: Accept. If the IM receives the grant from the CM, it sends the head cell from the matched VOQs in next timeslot and the matched pointers PL(i,r) and PV(i,j,h) are updated to one position beyond the matched one.

As described above, CRRD uses three sets of round-robin arbiters to resolve the contentions in IMs and CMs. The desynchronization effect of the round-robin pointers in CRRD works exactly as in iSLIP for crossbar switches.

D. *Concurrent Master-Slave Dispatching (CMSD)*

The CMSD is an improved version of the CRRD by reducing the interconnection complexity of dispatching schedulers and hence the dispatching time [6]. CMSD employs two sets of arbiters in IM, the master and the slave one. They operate concurrently in a hierarchal round-robin manner. The CMSD differs from the CRRD only on the iterative matching process in Phase 1.

Initialization:

Each VOQ(i,j,h) is associated with an arbiter V(i,j,h) with pointer PV(i,j,h). Each LI(i,r) is associated with a master arbiter ML(i,r) with pointer PML(i,r), and also associated with a slave arbiter SL(i,j,r) with pointer PSL(i,j,r). Each LC(r,j) is associated with an arbiter C(r,j) with pointer PC(r,j). Set PV(i,j,h) = PML(i,r) = PSL(i,,j,r) = PC(r,j) = 0.

Phase 1: Iteratively Matching within IM:

Step 1: Request. Each un-matched, non-empty VOQ(i,j,h) sends a request to every slave output link arbiter SL(i,j,r). At same time, each VOQ Group G(i,j) that has at least one un-matched, non-empty VOQ sends a request to every master output link arbiter ML(i,r).

Step 2: Grant. Each slave arbiter SL(i,j,r) search one VOQ's request in a round-robin fashion starting from pointer PSL(i,,j,r). At same time, each master arbiter ML(i,r) search one VOQ Group's request in a round-robin fashion starting from pointer PML(i,r). Finally, SL(i,j,r) sends the grant to VOQ(i,j,h) only if j has been selected by ML(i,r).

Step 3: Accept. Each VOQ(i,j,h) search one grant in a round-robin fashion starting from pointer PV(i,j,h) and sends the accept to the selected output link LI(i,r).

Phase 2: Matching between IM and CM:

These Operations are the same as CRRD, except that the pointers PSL(i,j,r) and PML(i,r), instead of PL(i,r), are updated to the one position beyond the matched one.

III. SRRD SCHEMES

A. *Static Round-Robin Dispatching (SRRD)*

The desynchronization effect of pointers in the round-robin arbiters plays an important role in the performance of the switch scheduling algorithms [1]. The more desynchronized the pointers are, the lower is the delay of the switch. As a result, by using the static desynchronized round-robin pointers, we can improve the CMSD scheme with following changes.

Initialization:

These pointers are the same as CMSD, except that their values are set to PV(i,j,h) = h, PSL(i,j,r) = r, PML(i,r) = (i+r) % k, PC(r,j) = i if (PML(i,r)==j).

Phase 1: Iteratively Matching within IM:

These Operations are the same as CMSD.

Phase 2: Matching between IM and CM:

These Operations are the same as CMSD, except that the pointers PML(i,r) & PC(r,j) are always incremented by one (mod k) and PSL(i,j,r) and PV(i,j,h) remain unchanged, no matter there is a match or not.

The novelty of our SRRD is demonstrated by the initial configuration of the round-robin pointers as shown in Figure 2. All pointers are artificially set to be desynchronized to resolve the contention points. This allows the maximum matching from IPs to OPs if all VOQs have a cell available. In other words, SRRD will always achieve 100% throughput under uniform traffic.

Our SRRD will not introduce any starvation in VOQs under any traffic pattern. This is because the pointers *PML(i,r) & PC(r,j)* are incremented during each timeslot and. Hence Each VOQ will be dispatched at least once within a round. Note that *PSL(i,j,h) & PV(i,j,h)* are kept constant. Consequently, all cells in *VOQ(i,j,h)* will be primarily scheduled through *CM(h)*. This will not affect the delay performance or the fairness.

B. *SRRD-r1: Rotating Directed by Time*

The performance of SRRD is expected to be degraded under unbalanced traffic. This is because several arbiters may grant to the same request at the same time. It can be solved by rotating the search directions of the round-robin arbiters [1]. In fact, all conventional arbiters search in clockwise direction. Now, we will allow the round-robin arbiters to search the requests in clockwise direction as well as in anti-clockwise direction.

The SRRD-r1 algorithm is a variant of SRRD such that the arbiters will search the requests in clockwise direction and anti-clockwise direction alternatively, each for one time slot. Practically, we can keep track of time by a 0/1 counter, which will increment by one (mod 2) in each cell time. All counters are initialized to 0 at cell time 0. The SRRD-r1 is the same as the SRRD except in the Step 2 of Phase 1:

If (counter == 0), ML(i,r) search one request in clockwise round-robin fashion.

If (counter == 1), ML(i,r) search one request in anti-clockwise round-robin fashion.

C. *SRRD-r2: Rotating Directed by Module Number*

Although the SRRD-r1 scheme will serve non-empty requests before and after the pointer with an equal chance, it will only grant one of them. This will degrade the system performance when the traffic is non-uniform and the number of ports is large.

We introduce another improved scheme, SRRD-r2. In each time slot, half of the pointers will rotate in clockwise direction, and another half will rotate in anti-clockwise direction, depending on the central module number *r*. The SRRD-r2 is similar to the CRRD-r1 except:

Initialize the counter in ML(i,r) to 0 if r is even, to 1 if r is odd.

IV. ANALYSIS OF SRRD SCHEMES

A. Performance under Balanced Traffic

In this section, we compare the delay performance of the various algorithms. The *relative delay* is obtained by dividing the average total delay by the delay produced by an output-queued switch, which is known as the "ideal"

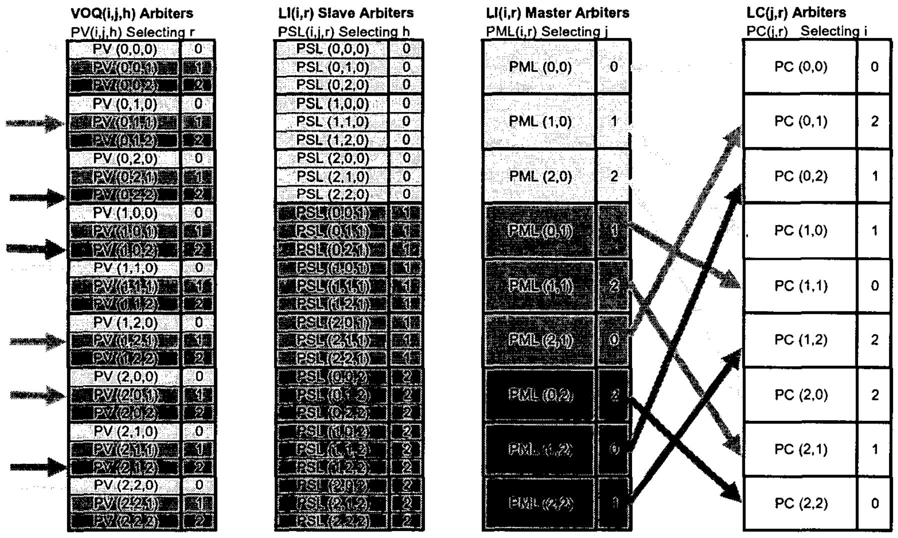

Figure 2. Initial Configuration of SRRD Arbiter Pointers (*n*=*k*=*m*=*3*)

switch and hence is used for reference. We used the Clos-network setting of $m=n=k=8$, which corresponds to a port size of $N=64$.

Figure 3(a) shows the comparison under *Uniform traffic*, in which cell arrival follows a Bernoulli i.i.d. process. Figure 3 (b). shows the comparison under *Bursty traffic*, in which cell arrives in each time slot during the busy period and no cell arrives in the idle period.

The results show that all algorithms achieve 100% throughput under Uniform traffic and Bursty traffic, except the CD algorithm. The delay of the CRRD is lower than the CMSD in the high load region under uniform traffic, but the situation is reversed under Bursty traffic. This is because the master arbiter of CMSD can choose a VOQ group, which affects the matching more seriously.

As expected, the performance of the SRRD-based algorithms is significantly better than the other algorithms. This is due to the full desynchronization of the SRRD pointers. Hence the contentions in the CM and the OM are almost minimized for uniform incoming traffic.

Figure 3 demonstrates that one iteration is sufficient to achieve 100% throughput for all round-robin scheduling algorithms. The number of iterations in Phase 1 just improves the delay performance. But the delay of SRRD almost converges for only two iterations.

B. *Performance under Unbalanced Traffic*

The first type of unbalanced traffic we consider is the *Bi-Diagonal traffic*. In which, input i has cells only for output $j = i$ or $(i+1) \bmod N$. Its load matrix is described as follows, where ρ denotes the normalized load.

$$\rho_{i,j} = \begin{cases} 2\rho/3N & \text{if } j = i \\ \rho/3N & \text{if } j = (i+1) \bmod N \\ 0 & \text{otherwise} \end{cases}$$

This type of traffic is more difficult to schedule compared with uniform traffic. As shown in Figure 4 (a), the basic SRRD has the worst performance even when compared with CRRD or CMSD.

However, both rotating schemes of SRRD can achieve 100% throughput under the Bi-Diagonal traffic. Their delay performances are also very close to the output-queued switches.

We define another type of unbalanced traffic, *Trans-Diagonal traffic* with following load matrix. For each input i, half of the traffic is going to output i, and the remaining traffic is uniformly distributed among other outputs.

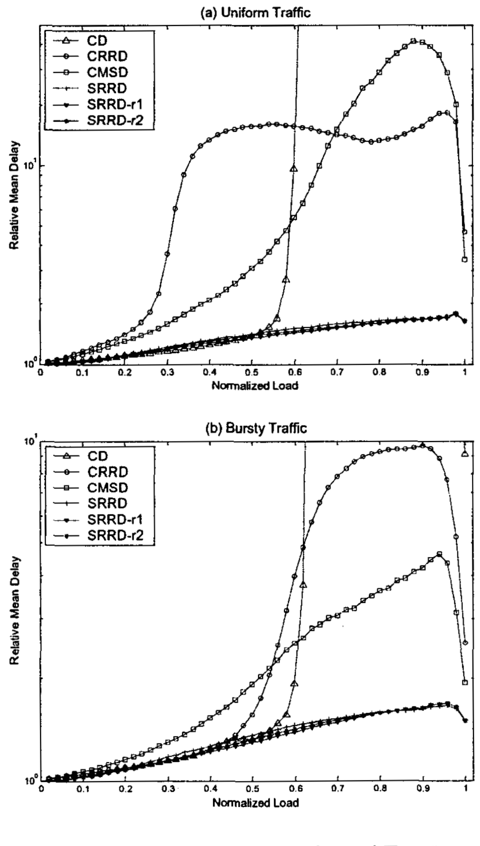

Figure 3. Comparison in Balanced Traffic

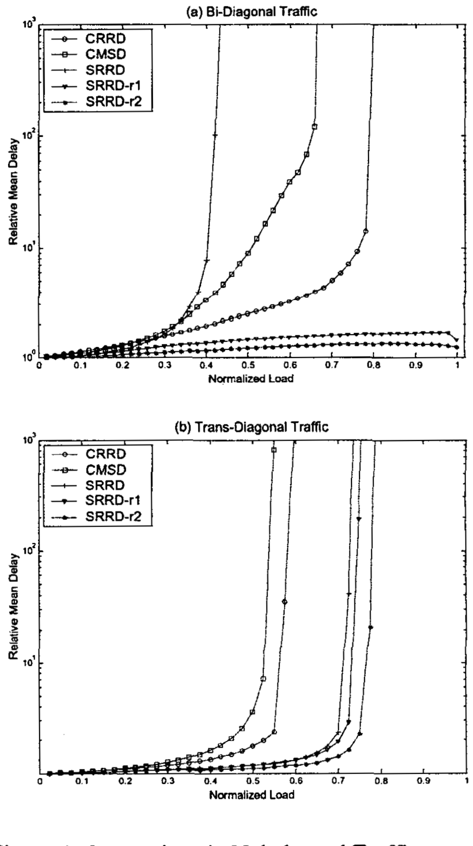

Figure 4. Comparison in Unbalanced Traffic

$$\rho_{i,j} = \begin{cases} \rho/2 & \text{if } j=i \\ \rho/2(N-1) & \text{otherwise} \end{cases}$$

This type of traffic is particularly difficult to schedule in the Clos-network switches. As shown in Figure 4 (b), no algorithm is able to achieve 100% throughput. However, in terms of maximum achievable throughput, our SRRD-based algorithms are still better than the others. In all types of traffic, our second variant, SRRD-r2, achieved the lowest delay and highest throughput.

C. *Hardware Implementation*

Round-robin arbiters are used in the implementation of SRRD-based schemes. In our SRRD-based schemes, the pointers associated with *SL(i,j,r)* and *V(i,j,h)* remain unchanged all the time, and the pointers associated with *ML(i,r)* and *C(r,j)* are always updated by one in each cell time. This requires no information transfer during the scheduling. As a result, our round-robin schedulers are much simpler than those used in iSLIP.

Figure 5 depicted the scheduler architecture in each IM. The state memory uses *nk* bits to record whether a VOQ is empty or non-empty, and *k* bits to record whether a VOQ Group is empty or non-empty. These bit vectors are representing **Step 1**: Request. The master arbiters and slave arbiters implementing **Step 2**: Grant. Each slave arbiter *SL(i,j,r)* search one VOQ's request in a round-robin fashion starting from pointer *PSL(i,,j,r)*. At the same time, each master arbiter *ML(i,r)* searches one VOQ Group's request. Then *SL(i,j,r)* sends the grant to *VOQ(i,j,h)* only if *j* has been selected by *ML(i,r)*. Finally, each *VOQ(i,j,h)* search one grant in a round-robin fashion starting from pointer *PV(i,j,h)*, implementing **Step 3**: Accept. The decision is then saved in the decision register which will be fed back to each master arbiter for the next iteration's arbitration.

One drawback of our switch model in Figure 1 is that the input and output stages are both composed of shared-memory modules. This is associating a memory speedup of *n* in each IM and *m* in each OM. However, these speedup problems can be solved by replacing them with input-queued switches, for example.

V. CONCLUSION

The Clos-network architecture is widely recognized as a very scalable architecture for high-speed switching system. So far, only limited success has been reported in the design of practical distributed scheduling schemes for the Clos-network. The ATLANTA switch is an example of a commercially successful Clos-network switch. But it requires internal bandwidth speedup. Recently, the CRRD and CMSD algorithms have been proposed in order to overcome the throughput limitation and implementation complexity problem.

In this paper, we introduced three improved algorithms based on the Static Round-Robin technique. Under various traffic models, we have shown by simulation that our algorithms achieve the lowest delay and the highest throughput among all other related schemes. In addition, their hardware implementations seem to be even simpler than that of related algorithms.

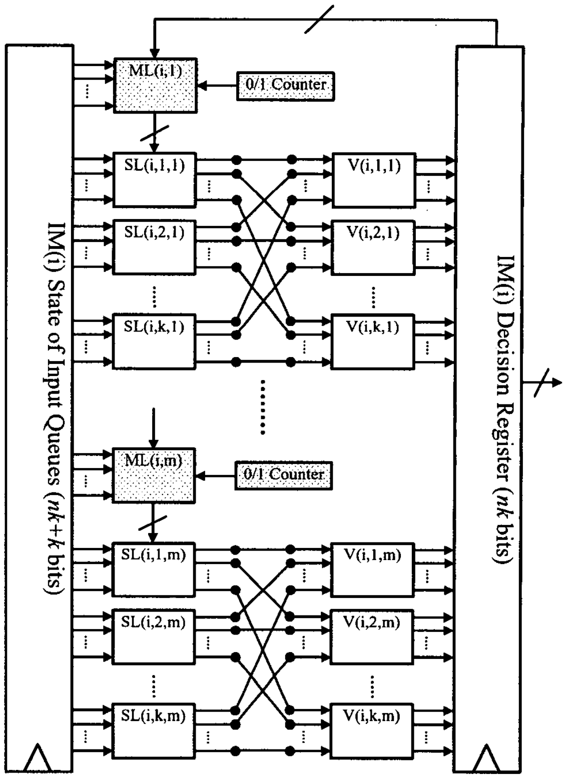

Figure 5. Interconnection of Arbiters in IM(i)

REFERENCES

[1] Y. Jiang and M. Hamdi, "A fully desynchronized round-robin matching scheduler for a VOQ packet switch architecture," *IEEE Workshop on High Performance Switching and Routing*, pp. 407-411, May 2001.

[2] T. Cheney, J.A. Fingerhurt, M. Flucke and J.S. Turner, "Design of a gigabit ATM switch," *Proceedings of IEEE Infocom'97*, vol. 1, pp. 2-11, April 1997.

[3] F.M. Chiussi, J.G. Kneuer and V.P. Kumar, "Low-cost scalable switching solutions for broadband networking: the ATLANTA architecture and chipset," *IEEE Communication Magazine*, vol. 35, no.3, pp. 44-53, December 1997.

[4] F.M. Chiussi, and A. Francini, "A distributed scheduling architecture for scalable packet switches," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2665-2683, December 2000.

[5] E. Oki, Z. Jing, R. Rojas-Cessa, and J. Chao, "Concurrent round-robin dispatching scheme in a clos-network switch," *IEEE International Conference on Communications*, vol. 1, pp. 107-111, June 2001.

[6] E. Oki, Z. Jing, R. Rojas-Cessa, and J. Chao, "Concurrent round-robin-based dispatching schemes for clos-network switches," *unpublished*.

[7] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local ara networks," *ACM Transaction on Computer Systems*, vol. 11, no. 4, pp. 319-352, November 1993.

[8] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188-200, April 1999.