Distro: A Distributed Static Round-Robin Scheduling Algorithm for Bufferless 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. Since more contention points are introduced in the multistage network, cell buffers are commonly used to resolve the contention. Recently, several scheduling algorithms have been proposed for the buffered Clos-Network switches. These approaches will cause either mis-sequence or memory speedup problem. In this paper, we propose a highly scalable bufferless Clos-network switching architecture. We also propose a distributed sheduling algorithm, *Distro*. It is based on a novel scheduling technique termed *Static Round-Robin* (SRR). Our simulation results demonstrate that our algorithm achieves 100% throughput under uniform traffic.**

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 algorithm have been proposed [8]-[11]. 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.

Since more contention points are introduced in the multistage network, cell buffers are commonly used to resolve the contention. There are basically two approaches. The first one has buffers in the second-stage, such as the WUGS architecture in [3]. 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-sequencing function, which is difficult to implement when the port number increases.

The second type of architecture has no buffers in the second-stage. It uses shared memory modules, in first- and third-stage to aggregate cells. The ATLANTA switch with its Memory/Space/Memory (MSM) architecture constitutes a commercially successful example [4]. This approach is more promising as no mis-sequence problem exists.

The concurrent dispatching (CD) algorithm used in the ATLANTA switch is a random-based scheduling algorithm. 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 iSLIP [9] and DRRM [10]. Similarly, the CRRD, CMSD and SRRD cell dispatching algorithms have been recently proposed for MSM Clos-network switching architecture using simple round-robin arbitration [6] [7].

However, one disadvantage of the MSM architecture is that the input and output stages are both composed of shared-memory modules. This is associating a memory speedup problem. Although the speedup is smaller than that in output-queued switches, it definitely hinders a switch to scale up to a very large port number.

We solve the memory speedup problem by aBuff erless Clos-network switching architecture which contains only crossbars in all stages. All cells are stored in the input port cards, just same as the virtual output queuing structure in the single stage crossbar switches. Since the switching elements are fully distributed by smaller modules, this raises the challenge of how to design the scheduling algorithm in a fully distributed way.

In this paper, we propose a distributed static round-robin scheduling algorithm for Bufferless Clos-network switches, called Distro. This is based on a novel scheduling technique termed Static Round-Robin (SRR). Our simulation results will demonstrate that our algorithm achieves 100% throughput under uniform traffic with comparable delay performance.

The rest of this paper is organized as follows. Section II introduces some background knowledge in the MSM architecture. Section III describes our Distro algorithm in Bufferless Clos-switch architecture. Section IV analyzes its performance. Finally, we conclude this paper in section V.

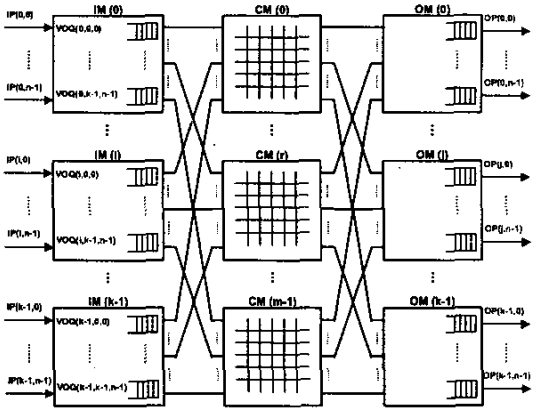

Figure 1. The MSM Switch Model

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

II. THE MSM ARCHITECTURE

A. The MSM Clos-network Switch Model

The MSM switch architecture has been proposed in the ATLANTA switch [4]. As 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 k cells from k 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. A *VOQ Group (i,j)* comprises all VOQs from *IM(i)* to *OM(j)*.

An *IM(i)* has k 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)*.

B. The Concurrent Dispatching (CD)

The distributed architecture of Clos-network implies the presence of multiple contention points. The ATLANTA switch proposed the CD algorithm with highly distributed nature [4]. It works as follows.

In each timeslot, each IM randomly selects up to k 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 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. The Concurrent Master-Slave Dispatching (CMSD)

The concurrent round-robin dispatching (CRRD) scheme has been firstly proposed in [6] to overcome the throughput limitation of the random-natured CD algorithm. The CMSD is an improved version of the CRRD. It employs two sets of arbiters in IM, the master and the slave one, operating in a hierarchal round-robin manner.

Initialization:

Each VOQ(i,j,h) is associated with an arbiter with Pointer_r (i,j,h). Each LI(i,r) is associated with a master arbiter with Pointer_j (i,r), and also associated with a slave arbiter with pointer_h (i,r,j). Each LC(r,j) is associated with an arbiter with Pointer_i (r,j). Set all pointers to 0.

Phase 1: Iteratively Matching within IM:

Step 1: Request. Each VOQ Group sends a request to every output link's master arbiter. At same time, each VOQ(i,j,h) sends a request to every slave arbiter.

Step 2: Grant. Each master arbiter searches one VOQ Group in a round-robin fashion starting from Pointer_j (i,r). At same time, each slave arbiter search one VOQ's request in a round-robin fashion starting from Pointer_h (i,,j,r), it then send the grant to VOQ(i,j,h) only if j has been selected by the master arbiter.

Step 3: Accept. Each VOQ(i,j,h) searches one grant in a round-robin fashion starting from Pointer_r (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 LI(i,r), who was accepted by a VOQ(i,j,h) in Phase 1, sends the request to the CM(r,j).

Step 2: Grant. Each CM(r,j) search one request in a round-robin fashion starting from Pointer_i (r,j).

Finally, the CM(r,j) sends the grant to the selected IM and hence selected VOQ, which sends the head cell in next timeslot. All matched pointers are updated to one position beyond the matched one.

D. The Static Round-Robin Dispatching (SRRD)

The intuition behind the SRRD design is to desynchronize the arbiters' pointers in a static way. It is the same as CMSD except with following changes.

Initialize the pointer by setting Pointer_r (i,j,h) = h, Pointer_h (i,j,r) = r, Pointer_j (i,r) = (i+r) % k, Pointer_i (r,j) = i if (Pointer_j (i,r)==j). In each timeslot, Pointer_j (i,r) & Pointer_i (r,j) are always incremented by one and Pointer_h (i,j,r) & Pointer_r (i,j,h) remain unchanged.

As shown in the Figure 2, with above simple changes, the delay performance of the SRRD algorithm is significantly better than the CMSD. 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.

We also compare the delay performance of the algorithms in the MSM architecture and in a single stage switch under uniform traffic in Figure 2. We used the MSM setting of $n=m=k=8$, which corresponds to a port size of $N=64$ in a single stage switch.

When load is below 0.5, the delay performance of the algorithms in the MSM architecture is larger than those in the single switch, such as PIM, iSLIP, SRR and Output Queued algorithm. But the situation is improved in heavy load region.

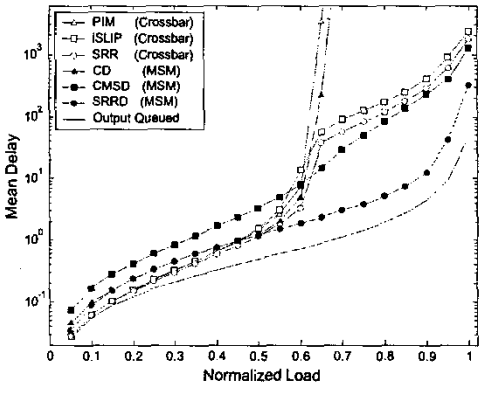

Figure 2. Delay comparison of Crossbar and MSM

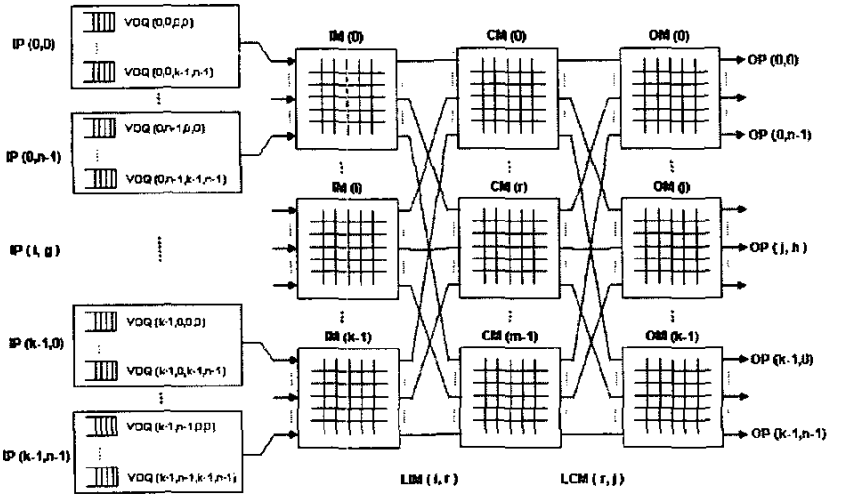

Figure 3. The bufferless Clos-network switch architecture

III. THE BUFFERLESS ARCHITECTURE

A. *The Bufferless Clos-network Switch Model*

One disadvantage of the MSM architecture 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 k in each OM. In output-queued switches, the memory speedup is $N=nxk$. Although the speedup of MSM is smaller than that in output-queued switches, it definitely hinders a switch to scale up to a very large port number.

As depicted in Figure 3, the Bufferless Clos-network architecture is slightly modified from the MSM architecture by replacing all shared memory modules by crossbars. All cells are stored in the input port cards, just same as the virtual output queuing structure in the single stage crossbar switches.

An *IP(i,g)* has *N* virtual output queues, *VOQ(i,g,j,h)*, storing cells that go from *IP(i,g)* to *OP(j,h)* at *OM(j)*. Each virtual output queue can receive at most one cell send at most one cell. A *VOQ Group (i,g,j)* comprises all VOQs from *IP(i,g)* to *OM(j)*.

In this paper, *i* corresponds to an IM, *g* to a specific input port of an IM, j corresponds to an OM, and *h* to a specific output port of an OM.

B. *The Distro Algorithm*

Since the contention points exist in all output links of the IPs, IMs, CMs and OMs, the scheduling in the Bufferless architecture is more challenging than in the MSM architecture. The algorithms for the MSM architecture are based on the request-grant-accept (RGA) handshaking scheme. This approach is difficult to implement when too many contention points exist. Hence our Distro algorithm adopts the request-grant (RG) scheme as proposed by the DRRM algorithm in [10].

Initialization:

Each IP(i,g) is associated with Arbiter_j (i,g) with Pointer_j (i,g). Each VOQ Group in IP(i,g) is associated with Arbiter_h (i,g,j) with Pointer_h (i,g). Each LI(i,r) is associated with Arbiter_g (i,r) with Pointer_g (i,r). Each LC(r,j) is associated with Arbiter_i (r,j) with Pointer_i (r,j). Each OP(j,h) is associated with Arbiter_r (j,h) with Pointer_r (j,h). For all i & g, initialized as follows:

```
j = (g + i) % k; h = i; r = (j - i) % m;
Pointer_j [i][g] = j; Pointer_h [i][g] = h; Pointer_g [i][r] = g;
Pointer_i [r][j] = i; Pointer_r [j][h] = r;
```

Phase 1: Request selection in each IP(i,g):

Each Arbiter_j (i,g) selects an non-empty VOQ Group in a round-robin fashion starting from Pointer_j (i,g). In the same time, each Arbiter_h (i,g,j) selects a non-empty VOQ within VOQ Group (i,g,j) in a round-robin fashion starting from Pointer_h(i,g,j). Then each IP(i,g) sends the request [j,h] to its output link only if j has been selected by the Arbiter_j and h has been selected by the Arbiter_h.

Phase 2: Grant from LI(i,r):

Each LI(i,r) systematically chooses the request [j,h] from IP(i,g) by Arbiter_g (i,r) where Pointer_g (i,r) == g. Then LI(i,r) sends the request to LC(r,j).

Phase 3: Grant from LC(r,j):

If LC(r,j) receives one or more non-empty requests from k LIs, it chooses the request [j,h] in LI(i,r) by Arbiter_i (r,j) in a round-robin fashion starting from Pointer_i (r,j). Then LC(r,j) sends the request to OP(j,h).

Phase 4: Grant from OP(j,h):

If OP(j,h) receives one or more non-empty requests from k LCs, it chooses the request [j,h] in LC(r,j) by Arbiter_r (j,h) in a round-robin fashion starting from Pointer_r (j,h).

Finally, the OP(j,h) notifies the IP(i,g) via the granted path, and the VOQ (i,g,j,h) will sends to OP(j,h) in next timeslot. Pointers are updated as: Pointer_j(i,g)++, Pointer_g(i,r)-- in each timeslot, Pointer_h (i,g)++, Pointer_r (j,h)++ in every k time slots.

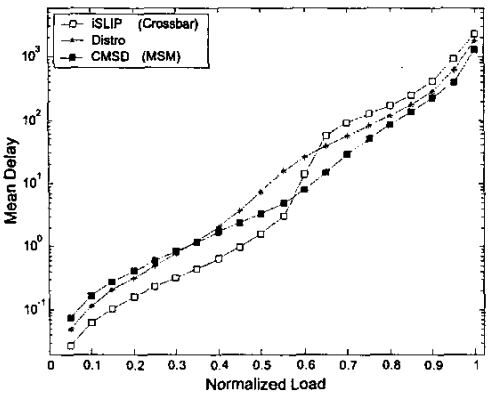

Figure 4.De lay comparison with the Distro

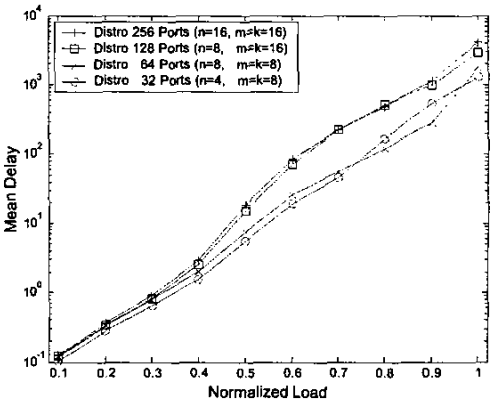

Figure 5. Delay of the Distro with different port size

IV. ANALYSIS OF DISTRO ALGORITHM

A. Delay Performance

We compare the delay performance of our Distro algorithm with other related algorithms in Figure 4. It is clear that Distro achieves 100% throughput under uniform traffic. When load is less than 0.65, the Distro algorithm is worse than the iSLIP in a reasonable scale. This is due to more contention points in the Clos-network switches. However, as the load increases, the desynchronization effect of Distro improves the delay performance. In the heavy load region, the performance of the Distro closely approximates to the performance of the SRR algorithm.

The delay performance of the MSM algorithms is generally large than other algorithms in light load region. Since it used shared-memory modules to resolve the contention for OPs, their delay performances in heavy load region are the best compared with other architectures. But this is compensated by the high memory speedup.

As mentioned in previous section, our bufferless Clos-network architecture is very scalable for the port size. Actually, we have a lot of flexibilities in configurations. We can either scale up the port size by increasing the number of ports n per input/output module, or increasing the number of central modules m. Note that m must larger or equal to n in order to achieve nonblocking property in Clos network.

Figure 5 shows the delay performance by the Distro algorithm with different port size. In general, the mean delay increases with the number of ports. However, if we scale up the port size by increasing m, the mean delay is even smaller in some load regions. This is because a larger number of central modules will decrease the contention.

B. Hardware implementation

The implementation of the Distro schemes is consisting of simple round-robin arbiters, in which priority encoders are adopted as in the iSLIP architecture [9]. For each round-robin arbiter, the hardware complexity is approximately $O(n_{req})$, where n_{req} is the number of requests to be selected by the arbiter. The complexity of all arbiter used in the Distro algorithm is $O(\sqrt{N})$.

In Phase 1, arbiters are interconnected to construct a scheduler in each IP as shown in Figure 6. The state register uses nk bits to record whether a VOQ has cell or not, and k bits to record whether a VOQ Group has non-empty VOQ or not. The Arbiter_j (i,g) selects an non-empty VOQ Group. In the same time, each Arbiter_h (i,g,j) selects a non-empty VOQ within VOQ Group (i,g,j). Then each IP(i,g) sends the request [j,h] to its output link only if j has been selected by the Arbiter_j and h has been selected by the Arbiter_h. The final decision [j,h] is save in the request register.

In static round-robin schemes, updating of the scheduler pointers is independent on the search result. This requires no information transfer during the scheduling and hence the round-robin schedulers are simpler than those used in iSLIP and CMSD. However, the main implementation challenge is not in the individual arbiters, but rather in implementing the arbitration process as a whole. In this paper, we just outline the schematic configuration of schedulers in Figure 3. The detailed implementation can refer to [4].

C. Scheduling Time

In a round-robin arbiter implemented by priority encoders, the time complexity is $O(\log n_{req})$. In Phase 1, the scheduling time complexity for Arbiter_j and Arbiter_h is $\max(\log k, \log n)$. In Phase 2, the matching time complexity is $O(1)$. The scheduling complexity in Phase 3 and Phase 4 is $O(\log k)$ and $O(\log m)$, respectively.

Let α is the constant determined by device technology, β is the transition delay between arbiters. Assume we use the configuration of $n = k = m = \sqrt{N}$. Then the required scheduling time of the Distro algorithm is given by $3\log n + 6\beta$. In contrast, the iSLIP algorithm requires $2\log N + 2\beta$. Hence the scheduling time of the Distro algorithm is highly sensitive to the communication overhead between arbiters.

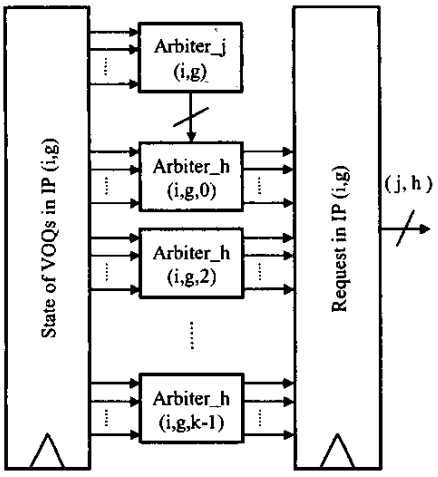

Figure 6. Scheduler in IP

V. CONCLUSION

Single stage switching techniques are inherently limited by their quadratic complexity. 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 traditional MSM arrangement of the CLos-network switches hinder the scalability of a very large port size. In this paper, we propose a distributed static round-robin scheduling algorithm for Bufferless Clos-network switches, called *Distro*. It is based on a novel scheduling technique termed *Static Round-Robin* (SRR). Our simulation results demonstrated that our algorithm achieved 100% throughput under uniform traffic with comparable delay performance with existing algorithms.

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] K. Pun and M. Hamdi, "Static round-robin dispatching schemes for Clos-network switches," *IEEE Workshop on High Performance Switching and Routing*,pp . 239-243, May 2002.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] E. Oki, Z. Jing, R. Rojas-Cessa, and J. Chao, "Concurrent round-robin-based dispatching schemes for clos-network switches," to be appear *in IEEE/ACM Transactions. on Networking*.

[8] 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.

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

[10]Y. Li, S. Panwar and J. Chao, "On the performance of a dual round-robin switch," *IEEE Infocom'2001*, vol. 3, pp. 1688-1697, April 2001.

[11]P. Giaccone, B. Prabhakar and D. Shah "Towards simple, high-performance schedulers for high-aggregate bandwidth switches," *IEEE Infocom'2002*,J une 2002.