Module Matching Schemes for Input-Queued Clos-Network Packet Switches

Chuan-Bi Lin and Roberto Rojas-Cessa

***Abstract*— Current schemes for configuration of input-queued three-stage Clos-network (IQC) switches involve port matching and path routing assignment, in that order. The implementation of a scheduler capable of matching thousands of ports in large-size switches is complex. To decrease the scheduler complexity for such switches (e.g., 1024 ports or more), we propose a configuration scheme for IQC switches that hierarchizes the matching process. In a practical scenario our scheme performs routing first and port matching thereafter. This approach reduces the scheduler size and the configuration complexity of IQC switches. We show that the switching performance of the proposed approach using weight-based and weightless selection schemes is high under uniform and nonuniform traffic.**

***Index Terms*— Clos-network switch, Space-Space-Space switch, matching, input queued, scheduler design.**

I. Introduction

The three-stage Clos-network switches use small switches as modules in each stage to build a switch with a large number of ports and less hardware than that of a single-stage switch of the same size [1]. Each of these modules can be a crossbar switch. Input-queued Clos-network (IQC) switches have queues in the input ports to store cells (variable-length packets are segmented into fixed-length packets, called cells, for internal switching) in case of input or output contention. The configuration of these switches is complex as output contention and path routing need to be resolved for every time slot before the transmission of packets occurs.

Some of the issues that can limit the scalability of these switches are: **a)** the time for configuring all modules before a packet is sent through the switch. This time requires a fast packet scheduler and an efficient exchange of scheduling information among the arbiters. **b)** the number of ports (N), as a large N would require large-size hardware arbiters. For example, a switch with *N=1024*, using a scheduler with an implementation complexity of $O(N^2)$ [2] and a time complexity of $O(\log N)$, would be difficult to build. To the best of our knowledge, implementation of schedulers of $N \leq 64$ has been reported in the literature [2].

One strategy that simplifies the configuration complexity of Clos-network switches is the use of queues in the first- and third-stage modules [3]. In this way, the scheduling of packets becomes a dispatching scheme issue [3]-[6]. However, the queues in the first-stage modules need to work with a speedup of $n + 1$ and those in the third-stage modules need to work with a speedup of $m + 1$, where n is the number of input ports of the first-stage modules, and m is the number of second-stage modules. This makes it complex to build queued Clos-networks switches.

Various matching schemes to configure IQC switches have been proposed [7]-[9]. Many of these schemes solve the configuration process in two phases: port matching first and routing thereafter, as routing uses the results of the port matching phase. For a 1024×1024 switch these schemes require a scheduler able to simultaneously match 1024 input ports to 1024 output ports. However, a scheduler of that size may be complex to implement [2].

In this paper we simplify the scheduler complexity for IQC switches by applying a similar concept of Clos networks, used to reduce the hardware complexity of large switches, to the configuration process. We propose to perform matching between first- and third-stage modules first, and matching between the input and output ports of matched modules afterward. We call this hierarchical approach module-first matching (MoM). We use the longest input queue-occupancy first selection as a weight-based MoM (WMoM) selection to show the switching performance when using this simple configuration approach. We compare the switching performance of WMoM to weightless MoM schemes based on round-robin and random selections. We show that MoM simplifies the configuration of IQC switches. For switches with a large number of ports, say 1024, and *n=m=k=32*, where k is the number of first- and third-stage modules, MoM can use a scheduler size of 32 instead of 1024, and a fast 32×32 scheduler is feasible to implement. We also show that MoM can provide high throughput under several traffic models despite its simplicity.

The remainder of this paper is organized as follows. Section II presents the switch architecture and notations. Section III describes WMoM as an example of the proposed configuration scheme for IQC switches. Section IV discusses implementation details. Section V presents the performance evaluation. Section VI presents our conclusions.

II. IQC Switch Architecture

The IQC switch is a three-stage switch architecture with virtual output queues (VOQs) at the input ports, as shown in Figure 1. We use a terminology similar to that in [3], which is as follows: $IM(i)$: $(i+1)$th input module where $0 \leq i \leq k-1$, $CM(r)$: $(r+1)$th central module where $0 \leq r \leq m-1$, $OM(j)$: $(j+1)$th output module where $0 \leq j \leq k-1$, n: number of input/output ports in each IM/OM, k: number of IMs/OMs, m: number of CMs, $IP(i,g)$: $(g+1)$th input port at $IM(i)$ where $0 \leq g \leq n-1$, $OP(j,h)$: $(h+1)$th output port at $OM(j)$ where $0 \leq h \leq n-1$, and $VOQ(i,g,j,h)$: Virtual output queue at $IP(i,g)$ that stores cells destined to $OP(j,h)$. Each $IP(i,g)$ has $N = n \times k$ VOQs to avoid head-of-line (HOL) blocking.

III. Weight-Based Module-First Matching (WMoM) Scheme

In this section we describe MoM with a weight-based selection scheme as an example. Other selection schemes can be used by following the described process. The MoM scheme uses two classes of schedulers for matching: the module

This work was supported in part by the National Science Foundation under Grant 0435250.

The authors are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102. Roberto Rojas-Cessa is the corresponding author. Email: rrojas@njit.edu.

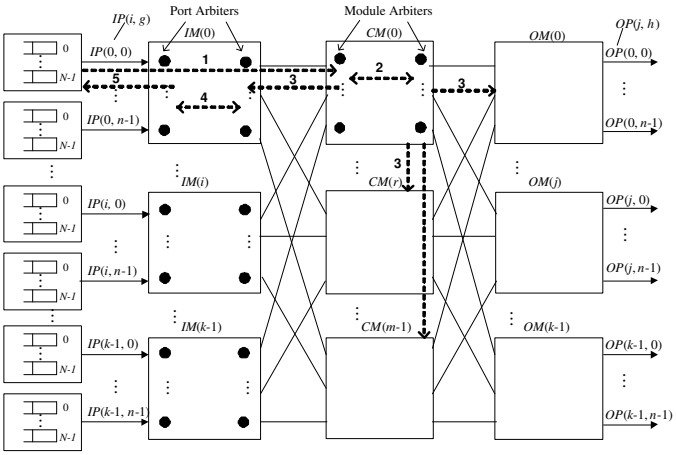

Fig. 1. The input-queued Clos-network switch architecture.

matching scheduler, S_M, which determines the matched *IM(i)-OM(j)* pairs, and the port matching scheduler, S_P, which determines the matched *VOQ(i, g, j, h)-OP(j, h)* pairs after the *IM-OM* pairs are defined. Weight-based MoM (WMoM) uses longest queue-occupancy first as the selection policy, which is similar to the iLQF algorithm [10] for single-stage switches. However, WMoM considers the occupancy of all ports in an IM for module matching.

To determine the weights for the *IM(i)-OM(j)* matching we use a VOQ module counter (VMC) to store the number of cells in $IM(i)$ going to $OM(j)$. A VMC is denoted as $VMC(i, j)$. The $VOQ(i, g, j, h) - OP(j, h)$ matching is performed after module matching. Each of the matching processes follows a request-grant-accept approach. In the general description WMoM performs r iterations of the complete scheme (e.g., module matching is executed r times, where $r \geq 1$), and q iterations for module and port matching (e.g., module matching executes q iterations, where $1 \leq q \leq k$). The following is the description of WMoM:

First iteration of WMoM (*r=1*)

Part 1: Module matching: first iteration

Step 1 (request). Each VMC whose count is larger than zero sends a request to the destined output module arbiter at the S_M. Requests include the number of cells for an output module.

Step 2 (grant). If an unmatched output module arbiter at the S_M receives any requests, it chooses the one with the largest occupancy. Ties are broken arbitrarily.

Step 3 (accept). If an unmatched input module arbiter at the S_M receives one or more grants, it accepts the one with the largest occupancy. Ties are broken arbitrarily.

qth iteration of module matching

Step 1: Each unmatched VMC sends a request to all unmatched output module arbiters at the S_M, as in the first iteration.

Steps 2 and 3: The same procedure is performed as in the first iteration among unmatched VMCs and unmatched output module arbiters.

Part 2: Port matching

After Part 1 is complete, port matching is performed between those ports of the matched IMs and OMs.

First iteration of port matching

Step 1 (Request): Each nonempty VOQ of the matched $IM(i)$ sends a request to each output arbiter in S_P for the matched $OM(j)$ for which it has a queued cell, indicating the number of cells in that VOQ.

Steps 2 (grant) and 3 (accept): The same procedure as in the module matching is performed for matching nonempty VOQs of a matched $IM(i)$ and OPs of a matched $OM(j)$. This matching is perform by input port arbiters and output port arbiters in S_Ps. These output and input arbiters select requests and grants, respectively, with the largest occupancy selection policy. Ties are broken arbitrarily.

qth iteration of port matching

Step 1: All unmatched VOQs in $IM(i)$ at the previous iterations send another request to corresponding unmatched OPs in the matched $OM(j)$ as in Step 1 of the first iteration.

Steps 2 and 3: The same procedure is performed as in the first iteration for matching between unmatched nonempty VOQs and unmatched output ports in the matched *IM(i)-OM(j)* pairs. Count the cumulative number of matched ports per IM and OM at this time slot. The number of matched ports is smaller than or equal to n.

For $r > 1$, the number of matched ports determines the number of central modules that are used to transfer cells from $IM(i)$ to $C(r)$ and from $C(r)$ to $OM(j)$. The selection of modules is performed by selecting those available CMs with the smaller index. For *r=1*, all CM paths are configured by using the module match result, which makes all CMs have the same configuration.

rth iteration of WMoM

Perform Part 1 with those modules that have fewer than n matched ports and whose unmatched ports are non-empty, and Part 2 with the non-empty unmatched ports of the modules matched at the current iteration.

IV. Implementation of MoM

In MoM, a module scheduler performs a $k \times k$ match and a port scheduler performs an $n \times n$ match for the input ports of the matched IM to the output ports of the matched OM. There are n port schedulers, one in each IM, and there is only one module scheduler that can be placed in one of the CMs, where IMs' requests converge, for a distributed implementation. Figure 1 shows the port and module arbiters as small circles in IMs and in a CM, respectively. A centralized implementation can also be considered because of the small size of the schedulers.

The MoM processes with *r=1* (or in an iteration) is as follows: 1) the inputs send a request to the module scheduler, 2) the module scheduler performs module matching with q iterations, 3) the module scheduler sends the grants to port schedulers at IMs, 4) the port schedulers at IMs perform matching with any number of iterations, and 5) the port schedulers send a grant to the input ports. Figure 1 shows these steps with dashed arrows as seen by an input port. The processes are indicated with numbers over the arrows, and the arrows indicate in what direction the information flows. A bidirectional arrow represents an iterative matching process.

V. Performance Evaluation

We modeled three MOM schemes for simulation: WMoM, MoM with round-robin selection, and MoM with random selection to show the performance of weight-based and weightless-based schemes. We considered *r=1* to show the lowest performance of these MoM schemes, and $q = \{1, 8\}$ for a fair comparison of WMoM and the other two schemes. We consider a 256×256 Clos-network switch with *n=m=k=16*. The procedures for the weightless schemes follow the steps described in Section III, except for the selection scheme of

ports and modules. The traffic models considered have destinations with uniform and nonuniform distributions and Bernoulli arrivals. The simulation does not consider the segmentation and re-assembly delays for variable size packets. Simulation results are obtained with a 95% confidence interval and a standard error not greater than 5%.

Figure 2 shows the average cell delay of WMoM under uniform traffic with Bernoulli arrivals. This figure shows that WMoM, as the other schemes, has low throughput with *q=1*. Round-robin delivers the highest throughput with *q=1*, however, of up to 80%. When *q=8* WMoM delivers close to 100% throughput under this traffic model, as the other schemes.

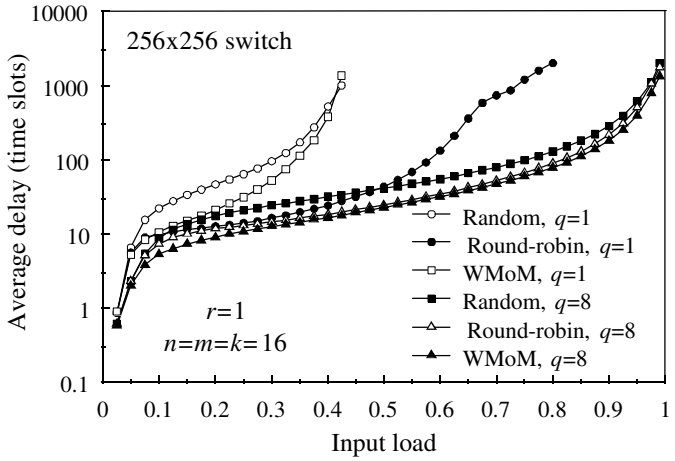

Fig. 2. WMoM in a *n=m=k=16* switch under Bernoulli uniform traffic.

We simulated WMoM under four different nonuniform traffic patterns: unbalanced, Chang's, asymmetric, and diagonal. Because of space limitations, the description of these models are referenced to [11].

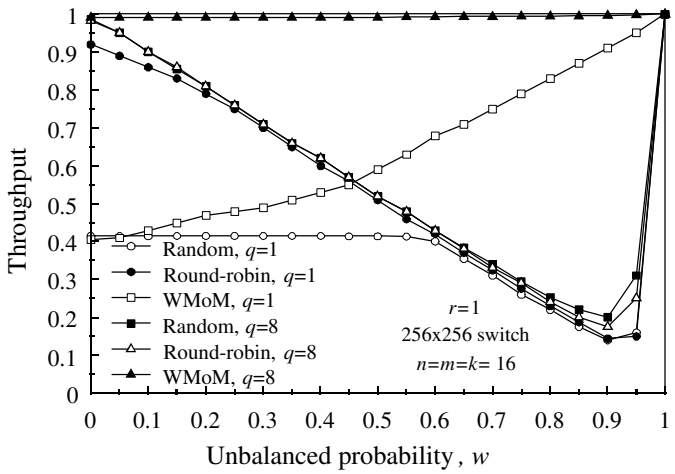

Fig. 3. WMoM in a *n=m=k=16* switch under Bernoulli unbalanced traffic.

Figure 3 shows the throughput performance of WMoM under unbalanced traffic. This figure shows that WMoM delivers 40% throughput, while the other schemes deliver close to 20% throughput (*w=0.9*) with *q=1*. When *q=8*, the throughput of WMoM is close to 100%, while the others remain low. The use of a large q makes WMoM match a larger number of VOQs with high occupancy. The throughputs of the other schemes decrease as w increases because they do not consider the VOQ occupancy in their selection policy, and once modules are matched, the VOQs with large occupancy wait for the following opportunity to send a cell.

Although the graphs are not shown here, we measured the throughput of WMoM with *q=8*, under Chang's, asymmetric, and diagonal traffic models. WMoM delivers close to 100% throughput under Chang's traffic, 91% throughput under asymmetric, and 87% throughput under diagonal. Furthermore, we tested WMoM with larger r values and noted that the switching performance does not increase significantly under these traffic patterns, making *r=1* sufficient in these cases, and therefore, greatly simplifying the configuration of CMs. However, for traffic models with a hot spot distribution, an *r=k* may be necessary. Also, *q=8* is a large number of iterations; however, these are performed in-chip.

VI. Conclusions

In this paper, we proposed a configuration scheme for input-queued Clos-network switches, called MoM, which performs module matching before port matching to reduce the configuration (and matching) complexity. We used a weight-based MoM scheme, called WMoM, based on the selection of the longest VOQ occupancy first to describe MoM and to show the obtainable performance.

The scheduler and configuration complexities for large-size switches can be reduced to $O(N^{1/2})$, where N is the number of ports. This complexity is smaller than any of the schemes previously proposed. For example, a 1024×1024 match by MoM requires parallel and independent 32×32 schedulers while other schemes require 1024×1024 schedulers.

We studied the switching performance of several MoM schemes: round-robin, random, and WMoM in a switch with *N=256*. We showed that WMoM delivers higher performance than the other schemes under uniform and nonuniform traffic models. WMoM, using longest occupancy-queue first, provides 100% throughput under Bernoulli uniform traffic, above 99% throughput under unbalanced and Chang's traffic, 91% under asymmetric traffic, and 87% under diagonal traffic.

References

[1] C. Clos, "A study of nonblocking switching networks," *Bell Syst. Tech. J.*, pp. 406-424, March 1953.

[2] P. Gupta and N. McKeown, "Design and implmentation of a fast crossbar scheduler," *IEEE Hot Interconnects VI*, 8 pages, Aug. 1998.

[3] E. Oki, Z. Jing, R. Rojas-Cessa, and H. J. Chao, "Concurrent roun-robin dispatching scheme for Clos-network switches," *IEEE/ACM Trans. Networking*, Vol. 10, No. 6, pp. 830-844, May 2001.

[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 Commun. Mag.*, pp. 44-53, Dec. 1997.

[5] R. Rojas-Cessa, E. Oki, and H.J. Chao, "Maximum weight matching dispatching scheme in buffered Clos-network packet switches," *IEEE ICC* , Vol. 2, pp. 1075-1079, June 2004.

[6] K. Pun and M. Hamdi, "Static round-robin dispatching schemes for Clos-network switches," *IEEE HPSR*, pp. 239-243, May 2002.

[7] T.T. Lee and S-Y Liew, "Parallel routing algorithm in Benes-Clos networks," *IEEE INFOCOM' 96*, pp. 279-286, 1996.

[8] K. Pun and M. Hamdi, "Distro: A distributed static round-robin scheduling algorithm for bufferless clos-network switches," *IEEE Globecom 2002*, Vol. 3, pp. 2298-2302, 2002.

[9] H.J. Chao, Z. Jing, and S.Y. Liew, "Matching algorithms for three-stage bufferless Clos network switches," *IEEE Commun. Mag.*, Vol. 41, Issue 10, pp. 46-54, Oct. 2003.

[10] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, Vol. 47, No. 8, pp. 1260-1267, Aug. 1999.

[11] R. Rojas-Cessa and C-B. Lin, "Captured-frame eligibility and round-robin matching for input-queue packet switches," *IEEE Commun. Letters*, Vol. 8, Issue 9, pp. 585-587, Sep. 2004.