



# Packet dispatching using module matching in the modified MSM Clos-network switch

Janusz Kleban<sup>1</sup> 

Published online: 9 March 2017

© The Author(s) 2017. This article is published with open access at Springerlink.com

**Abstract** A modified memory-space-memory (MSM) Clos-network switch, called MCNS, with a module-level matching packet dispatching scheme, is presented in the paper. The MCNS is a modification of the MSM Clos-network switch proposed to simplify the packet dispatching scheme. In this paper, we evaluate the combinatorial properties of the MCNS, as well as a new module-level matching packet dispatching algorithm. We also show how this algorithm can be implemented in an field-programmable gate array chip. Selected simulation results obtained for the MCNS are compared with the results obtained for the MSM Clos-network switch using other module-level matching algorithms.

**Keywords** Clos network · Dispatching algorithm · Packet switching · Packet scheduling

## 1 Introduction

Research on the architecture of large-capacity packet switching nodes for the future Internet is still in progress. The performance of very fast switches/routers may be limited by switching fabrics that are not fast enough to switch incoming packets with a speed equal to the line rate. Low-end or middle-size switches/routers may employ a crossbar switch (a one-stage switching fabric), but high-end routers need more sophisticated multi-stage switching fabrics. The most popular multi-stage switching fabric was proposed by Clos and is known as the Clos network or the Clos switching fabric

[1]. Clos networks are scalable and can be composed according to required combinatorial properties.

In general, non-blocking switching networks, where every combination of connections between inputs and outputs may be established, can be divided into four categories [2]: strict-sense non-blocking (SSNB), wide-sense non-blocking (WSNB), rearrangeable (RRNB) and repackable (RPNB) non-blocking. In SSNB networks, no call is blocked at any time. WSNB networks are able to connect any idle input and any idle output, but a special path searching algorithm must be used. RRNB networks can also establish required connections, but a rearrangement of some existing connections to other connecting paths may be needed to change the network state in order to unblock a blocked call. In this case, rearrangements are used when a new request arrives and is blocked. In RPNB networks, it is also possible to connect an idle input-output pair, but a suitable control algorithm must be used to assign connecting paths for new calls, and a repacking algorithm to re-route some of the connections in progress. Call repackings are executed immediately after one of the existing calls is terminated and are intended to reach a network state where no blocking can occur. So, repacking process is executed in order to pack the existing calls more efficiently and prevent a switching network from entering blocking states before a new call arrives, in contrast to rearrangements which are started after a new call is blocked. A basic investigation of the properties of SSNB and RRNB networks was performed by Clos [1] and Beneš [3] respectively. WSNB networks were analyzed by Beneš [3,4]. Repackings, as well as the first repacking algorithms, have been proposed by Ackroyd [5]. The class of repackable networks was defined by Jajszczyk and Jekel [6]. Generally speaking, RPNB networks contain less hardware than SSNB and WSNB, but more hardware than RRNB networks, which

✉ Janusz Kleban  
janusz.kleban@put.poznan.pl

<sup>1</sup> Faculty of Electronics and Communications, Poznan University of Technology, Polanka 3, 60-965 Poznan, Poland

can be built using a theoretical minimum number of cross-points [3].

The RRNB fabrics may be used in carrier routing systems, where fixed-length packets called cells are transmitted in time slots from the ingress to the egress side of the switching fabric [7]. Since a new set of connecting paths may be set up for each time slot, the RRNB fabric is enough to satisfy all connection requests. To alleviate the complexity of packet dispatching algorithms, switching fabrics use buffers.

Among different architectures of packet switching fabrics, the MSM Clos-network switch is well investigated. The first and the third stages of this switching fabric may be composed of shared memory modules or cross-point queuing (CQ) crossbars [8]. To avoid the head-of-line (HOL) blocking phenomenon, virtual output queuing (VOQ) [7] is implemented. The problems of internal blocking and output port contention are solved by arbiters and packet dispatching algorithms. Switching fabrics with fast arbitration schemes for large-capacity packet switching nodes are still under investigation and seem to be a serious research challenge.

This paper is devoted to the MCNS, which is a modification of the MSM Clos-network switch and seems to be an interesting solution for future packet switching nodes. The remaining part of the paper is organized as follows. In Sect. 2 we discuss related works and explain our motivation for the research. We introduce the MCNS in Sect. 3. The following sections deal with the evaluation of the combinatorial properties of the investigated switching fabric (Sect. 4), the packet dispatching scheme (Sect. 5), hardware implementation of the module matching algorithm (Sect. 6), and performance evaluation (Sect. 7). Section 8 concludes the work.

## 2 Related works and motivation

Most of the packet dispatching schemes developed for the three-stage Clos-network switches insist on using the request-grant-accept handshaking routine, based on the effect of the desynchronization of arbitration pointers [9, 10]. In this routine, if cells are waiting for transmission in queues, input arbiters send requests to corresponding output port arbiters. Next, output port arbiters send grants to selected input port arbiters, and finally, after the selection process, input port arbiters send *accept* signals to output port arbiters. This mechanism, with many iterations, is used to match VOQs to output ports in input modules, and then to match central module output ports to input module output ports. A lot of request-grant-accept messages may cause congestion among these messages, and extra on-chip memory is needed to solve this problem.

Module-level matching algorithms for MSM Clos-network switches were proposed and evaluated in [11–13]. The algorithms proposed in [11] and [12] employ two phases: input

and output module matching, and then port matching. The request-grant-accept scheme with many iterations is used to select cells to be transmitted to output modules. We focus mainly on the algorithms presented in [13], where a central arbiter maintains cell counters, and we intend to compare selected results obtained for the MCNS with the results published in this paper.

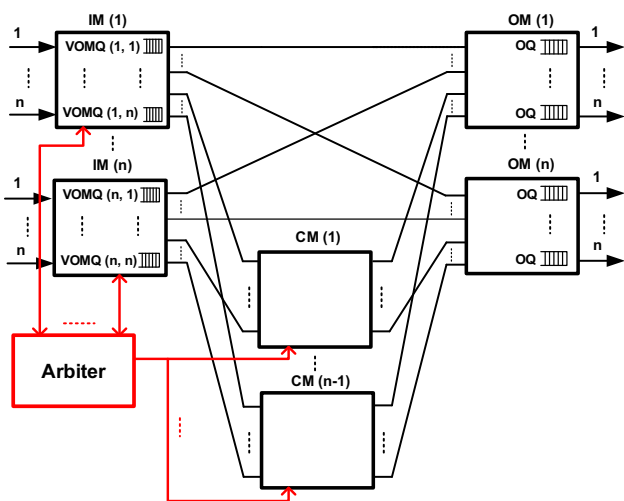
The main goal of our work is to propose a switching fabric architecture with a packet dispatching algorithm that performs very well under uniform, as well as non-uniform traffic distribution patterns, and avoids the request-grant-accept routine with many iterations. In this paper, we suggest using the MCNS with a central arbiter and a module-level matching scheme. Central arbiters were not proposed for multi-stage switching fabrics due to the scalability problem and considerable processing power needed to solve contention problems. The arbitration process proposed by us is very simple and can be implemented using logic gates. The implementation of the proposed module matching algorithm in an FPGA chip is described in Sect. 6.

## 3 The modified MSM Clos-network switch (MCNS)

The three-stage Clos switching fabric is denoted by  $C(n, m, r)$ , where  $n$  represents the number of input ports in each ingress switch, and the number of output ports in each egress switch,  $r$  represents the number of ingress and egress switches of capacity  $n \times m$  and  $m \times n$ , respectively, and  $m$  represents the number of second stage switches of capacity  $r \times r$ . The total capacity of this switching fabric is  $N \times N$ , where  $N = nr$ . This fabric is SSNB if  $m \geq 2n - 1$  and RRNB if  $m \geq n$ .

The MCNS is shown in Fig. 1 [14]. In this paper we focus on the architecture with  $n$  input modules (IMs),  $n - 1$  central modules (CMs) and  $n$  output modules (OMs). Each IM has  $n$  inputs and each OM has  $n$  outputs. The total capacity of this switching fabric is  $N \times N$ , where  $N = n^2$ . Similarly to the Clos networks we denote the MCNS as  $C_M(n, n - 1, n)$ .

The  $C_M(n, n - 1, n)$  is created by connecting bufferless CMs to a two-stage buffered switching fabric, i.e., buffers are placed in IMs and OMs. Buffers in IMs are organized according to the requirements of the virtual output module queues (VOMQs) technique. It means that incoming cells are sorted by the OM numbers and stored in  $VOMQ(i, j)$ , where  $i$  is the number of  $IM(i)$ ,  $1 \leq i \leq n$ , and  $j$  is the number of  $OM(j)$ ,  $1 \leq j \leq n$ . The shared memory modules, used in the first stage, require a memory speed up. This problem may be alleviated in the future by CQ switches, where it is possible to organize VOQs or VOMQs in cross-point buffers. These kinds of chips are not on the market yet, but the switches are considered in [15, 16]. Currently, the speed up problem may



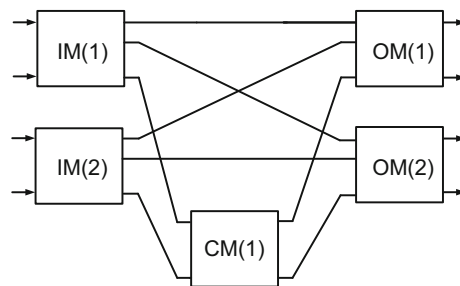
**Fig. 1** The modified Clos-network switch (MCNS); the same information is sent from the arbitrator to each CM

be definitely avoided using CQ switches in the third stage, where output queues (OQs) are located.

The MCNS allows one cell to be sent from each IM to any OM using a direct link, and no arbitration process is needed. These direct connections are almost sufficient to serve uniformly distributed traffic. However, for non-uniform traffic distribution patterns, CMs are necessary to rapidly unload VOMQs during one time slot. The module-level matching scheme for the MCNS is described in Sect. 5. The architecture of the MCNS is very flexible and the number of needed CMs may be adjusted to the traffic distribution patterns - it may vary between 1 and  $n - 1$ . This means that it is possible to switch off some CMs if the latency of cells is acceptable, and switch them on if cells are waiting too long in the input buffers. This approach takes into account the requirements of a new research area called energy aware or green energy networks.

### 4 Combinatorial properties of MCNS

The combinatorial properties of the MCNS are given by Theorems 1 and 2. In Theorem 1, we show that for  $n = 2$ , the  $C_M(n, n - 1, n)$  is SSNB, however, for  $n \geq 3$ , it is RRNB (Theorem 2), like the MSM Clos-network switch  $C(n, n, n)$  used as the reference network for comparison of obtained results. We use the MSM Clos-network switch as the reference model because both networks can have the same organization of buffers located in the first and the third stages, and both are RRNB. Moreover, contrary to other Clos-network switches like space-memory-memory (SMM) or memory-memory-memory (MMM), where buffers are used in the second and the third stages or in all stages respectively, cell sequence is preserved due to a bufferless second stage.



**Fig. 2** The MCNS of capacity  $4 \times 4$

Out-of-sequence forwarding requires a special re-sequencing mechanism at the outputs, which needs additional memory and may influence packet processing time.

**Theorem 1** *The  $C_M(n, n - 1, n)$  is SSNB for  $n = 2$ .*

*Proof* To prove the strict sense nonblocking property of the MCNS for  $n = 2$ , we consider the worst case state, similarly to the evaluation of space division Clos' networks. The MCNS for  $n = 2$  ( $N = 4$ ) is shown in Fig. 2.

Let us consider connections from  $IM(1)$ . Assume that only the direct link from  $IM(1)$  to  $OM(1)$  is busy. It means that  $IM(1)$  still has one idle input port. If a new connection is destined for  $OM(2)$ , it may be established using the direct link between  $IM(1)$  and  $OM(2)$ . If a new connection is destined for  $OM(1)$ , the only connecting path leads through  $CM(1)$ . The link between  $CM(1)$  and  $OM(1)$  is not busy, because one output port of  $OM(1)$  is still idle, so there is no connection between  $IM(2)$  and  $OM(1)$ . Therefore, it is possible to establish the requested connection from  $IM(1)$  to  $OM(1)$ . Analogically, it is possible to show that it is always possible to set up connections from  $IM(2)$  to  $OM(1)$  and  $OM(2)$  using direct links, or two connections from  $IM(2)$  to  $OM(1)$  or  $OM(2)$  using one direct link and a second one leading through  $CM(1)$ .  $\square$

It is necessary to emphasize that proposed MCNS for  $n = 2$  is SSNB and has two switches fewer than the SSNB Clos network. The latter one uses three  $2 \times 2$  switches in the second stage to be SSNB, and has 36 cross-points, whereas the MCNS uses only one switch in the second stage and has 28 cross-points. Generally speaking, the number of cross-points in the MCNS is higher than in the RRNB Clos network, but lower than in the SSNB Clos network.

**Theorem 2** *The  $C_M(n, n - 1, n)$  is RRNB for  $n \geq 3$ .*

*Proof* Necessity is obvious. To set up connections from  $n$  inputs in any IM, we need at least  $n$  switches in the second stage. In the MCNS, the functionality of one CM is replaced with direct connections from IMs to OMs. Actually, these direct connections substitute one CM, so in the second stage, the MCNS has  $n - 1$  switches, i.e., one switch less than

the RRNB Clos network. The expansion ratio in IMs and OMs is the same as in the SSNB Clos networks, because in each IM there are  $n$  outputs connected directly to OMs, and  $n - 1$  outputs connected to CMs. In each OM,  $n$  inputs are connected to IMs, and additionally  $n - 1$  inputs are connected to  $n - 1$  CMs.

Sufficiency can be proved analogically to the RRNB Clos network, using Hall's theorem on distinct representatives.  $\square$

## 5 Packet dispatching scheme for the MCNS

Within the MCNS, in each time slot, one cell may be sent from any IM to any OM using direct connections. The CMs are used when in  $IM(i)$  the queue of cells destined for  $OM(j)$  increases and it is necessary to decrease the number of cells waiting for transmission. To resolve contention problems related to matching  $IM(i) - OM(j)$  pairs, we propose to use a central arbiter. Two types of requests may be sent to the central arbiter: high or low priority. The high priority request is sent only if the number of cells waiting to be sent to  $OM(j)$  is equal or greater than  $kn$ , where the value of  $k$  may be adjusted to particular needs. The low priority request is sent when the number of cells is equal or greater than  $n$  and smaller than  $kn$ . It is obvious that the high priority requests are processed by the central arbiter before the low priority requests. To establish the sequence of requests to be processed, the round-robin selection is used separately among the high and low priority requests. Each request consists of  $n + 1$  bits, where the least significant bit (LSB) is the priority bit, and successive bits are used to point out the requested OMs. For example, for  $n = 8$ , the request [000010100] means that it is the low priority request, and the IM has more than  $n$  and less than  $kn$  cells waiting to be sent to  $OM(2)$  and  $OM(4)$ . Since the packet dispatching algorithm employs module-level matching, the connection pattern in each CM will be the same. To match IMs and OMs the central arbiter uses a binary buffers load matrix as the main component. The rows and columns of the matrix represent IMs and OMs, respectively. If the value of matrix element  $x[i, j]$  is 1, it means that  $IM(i)$  can send  $n$  cells to  $OM(j)$ , one through the direct connection and  $n - 1$  through CMs. The central arbiter resolves contention problems by accepting the only 1 in column  $i$  and in row  $j$ . The requests that lose the competition are rejected.

In detail, the packet dispatching algorithm works as follows:

- **Step 1 (each IM)** If there is any  $VOMQ(i, j)$  having at least  $n$  cells for  $OM(j)$ , the high or low priority request is sent to the central arbiter.
- **Step 2** Using round-robin selection, the central arbiter loads the high priority requests at first, and next, the low

priority requests into the buffers load matrix. The priority bits are not loaded into the matrix, so now the LSB of each request points out  $OM(1)$  and the most significant bit (MSB) points out  $OM(n)$ . Then, the central arbiter goes through the matrix from the LSB to the MSB and from the top to the bottom, and chooses the first not masked 1 in each row. When this 1 is selected, successive bits in the row (till the MSB) and in the column (till the bottom) are masked, and they are not considered when the rows below are examined. As a result, the only 1 is selected in each row and each column, and it is clear which IM–OM pairs are allowed to use CMs to move cells. Next, the round-robin pointers for the high and low priority requests are set to new values.

- **Step 3** The central arbiter sends the matching decisions to IMs, and the IM–OM matching patterns to CMs for connecting path assignment.
- **Step 4 (each IM)** Cells are selected to be sent to OMs through direct connections between IMs and OMs. If the request is accepted by the central arbiter,  $n - 1$  cells are sent through CMs according to the arbiter's decision.

The OM with the lowest number, pointed out in the request loaded into the top row of the matrix, will definitely be granted. In the request loaded into the second row another OM will be granted, and so on.

The proposed module matching scheme is more sophisticated and produces better performance results than the SDRUB algorithm published in [14], where only one OM is pointed out in the request. The new module matching algorithm was prepared, with special attention to its implementation on logic gates in an FPGA chip. Contrary to the SDRUB scheme, in the new algorithm, it is possible to point out all OMs to which the number of waiting cells is higher than  $n$ . Moreover, high priority requests are used to point out OMs for which the VOMQs are the longest, and these requests are processed firstly. It should be also emphasized that it is possible to redefine the high and low priority requests, so that, e.g., the low level requests may be sent when IM has fewer than  $n$  cells waiting for transmission to  $OM(j)$ . This redefinition does not influence the operation of the central arbiter. A comparison of the average cell delay, obtained for both algorithms under Hot-spot traffic, is shown in Sect. 7. In the same chart, selected results obtained for the MSM Clos-network switch and published in [13] are also shown. The MSM Clos-network switch investigated in [13] uses a centralized scheduler to maintain  $n$  cell counters for each IM, one for each VOMQ. Module-level matching is performed using the following schemes: maximum weight matching (MWM), APSARA or DISQUO. Additionally, after the module-level matching phase, static dispatching or dynamic cell dispatching schemes are used in [13] to improve the delay performance of the module-

level matching scheme. To reduce the wiring complexity, a grouped dynamic cell dispatching scheme is used.

The MWM, APSARA and DISQUO algorithms adopted the idea of bipartite graph matching algorithms to solve the problem of module-level matching. It is well known that in traditional input-queued (IQ) switches (crossbar switches) the switch scheduling problem is similar to a matching problem in an  $N \times N$  weighted bipartite graph. The weight of the edge connecting input  $i$  to output  $j$  may be, e.g., queue-lengths or the age of cells. To adopt these kinds of schemes to the MSM Clos-network switch, it is necessary to map each IM and OM to input and output ports, respectively, whereas each VOMQ represents a VOQ in traditional IQ switches. The MWM algorithm finds the matching with the highest weight out of all  $N!$  possible matchings [21]. The complexity of this algorithm adopted to module-level matching is reduced to  $N^{1.5}$ , and is still high for large-scale switches. The APSARA algorithm [17] tries to speed up the scheduling process by searching in parallel for the highest weight from reduced space matchings. For example, by swapping any two connections from the matching used in the previous time slot, the APSARA algorithm can find a new matching. It chooses the matching with the highest weight among these swapped matchings. The DISQUO (DIStributed QUeue input-Output) algorithm was proposed for combined-input-output-queued (CIOQ) switches [18]. It was adjusted to work with IQ switches in [19]. The basic DISQUO algorithm uses the Hamiltonian Walk to find a matching; next, it compares it with the existing matching, and decides either to release the current connection or to establish a new one. The modified version of DISQUO uses time-division multiplexing instead of the Hamiltonian Walk and a simple heuristic matching algorithm to improve the delay performance. The

hybrid approach is called StablePlus. In [13] it is adopted to the module-level matching scheme.

### 6 Hardware implementation of the module matching algorithm

The proposed algorithm employs only mechanisms that can be easily implemented on logic gates in an FPGA chip. The general idea of this implementation is shown in Fig. 3.

The main component of the central arbiter is the buffers load matrix. The implementation of this matrix in an FPGA chip is similar to the implementation of matrices for the controller of  $\log_2(N, 0, p)$  networks described in [20]. The requests sent by IMs are reordered according to the priority status and round-robin selection, and then they are processed by a combinational function implemented on logic gates in an FPGA chip. The combinational function starts to work from the LSB of the first request in the buffers load matrix. When the first not masked 1 in the row is found, other 1s in this row and the related column will be masked and changed into 0 (hatched elements of the matrix shown in Fig. 3). The high priority requests are processed firstly. The low priority requests are processed in the second place. In each set of requests the processing order depends on round-robin selection. As a result, the buffers load matrix with the only 1 in each row and column is obtained.

The operation of the buffers load matrix was tested using the Xilinx-Kintex-7 FPGA module with -3 speed grade (the highest performance). The experiments show that only 4 ns and 4.8 ns is needed for a matrix of size  $32 \times 32$  ( $N = 1024$ ), and  $64 \times 64$  ( $N = 4096$ ), respectively, to obtain the final results. After this time, the decisions may be sent to IMs and

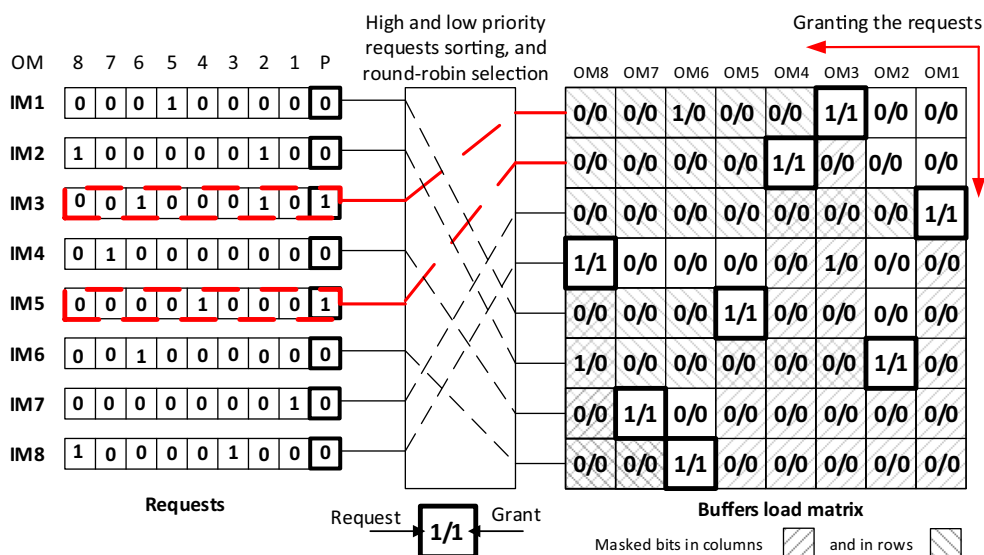


Fig. 3 A general idea of the proposed module matching scheme implementation in FPGA chip

CMs. For the speed of 10 Gb/s and 64 B of cell size the time slot is 50 ns. These experiments show that the buffers load matrix is not a bottleneck of this solution. Other limitations of the scalability of the MCNS, like the number of I/O pins and transmission speed at one pin, depend on the type of FPGA chip available on the market. For example, the Xilinx-Virtex-7 FPGA chip (XC7V2000T) provides 1200 I/O pins with data rates up to 1.866 Gb/s, and 96 transceiver circuits may be used with a speed up to 28.05 Gb/s. This number of I/O pins is sufficient even to serve the MCNS of a very big size, e.g.,  $n = 128$  ( $N = 16384$ ). Let us evaluate the time that is needed to collect requests from IMs. For  $n = 128$  it is necessary to transmit 128 sets of 129 bits as IM requests. We may use 1024 I/O pins with the data rate of 1.866 Gb/s - 8 I/O pins for parallel transmission from each IM. Less than 10 ns is needed to collect all requests. Because the connection pattern is the same in each CM, it is possible to send it to all CMs simultaneously using a bus. The connection pattern is a set of CM output port numbers (OM numbers in the binary system) to which consecutive input ports of CM should be connected. The first number means that the first input port of CM should be connected to this output port of CM, e.g., if the first number is 5, it means that the first input port should be connected to the fifth output port. For  $n = 128$ , it will be only 896 bits ( $128 \times 7$  bits = 896 bits). Using 128 I/O pins with the data rate of 1.866 Gb/s for parallel transmission, we need less than 1 ns to transfer the OM numbers to CMs.

## 7 Performance evaluation

The performance of the MCNS was evaluated using event-driven computer simulation. To conduct the simulation experiments, we have used self-developed software. The MCNS of size  $64 \times 64$  with  $n = 8$ , and  $k = 2$ , was tested under uniform and non-uniform traffic distribution patterns. In each simulation experiment, we considered a wide range of traffic loads per input port, from  $p = 0.05$  to  $p = 1$ , with the step 0.05. To reach a stable state of the MCNS, we used the starting phase comprised 50,000 time slots. The 95 confidence intervals calculated after t-student distribution for five series with 200,000 cycles are not shown in the figures because they are at least one order lower than the mean value of the simulation results.

Two packet arrival models were considered in the simulation experiments: the Bernoulli arrival process and the bursty traffic model. The following traffic distribution models were used in simulation: (1) uniform:  $p_{ij} = p/N$ ; (2) Chang's traffic:  $p_{ij} = 0$  for  $i = j$ , and  $p_{ij} = p/(N - 1)$  otherwise; (3) diagonal:  $p_{ij} = 2p/3$  for  $i = j$  and  $p_{ij} = p/3$  for  $j = (i + 1) \bmod N$ ; (4) Hot-spot:  $p_{ij} = p/2$  for  $i = j$ , and  $p/2(N - 1)$  for  $i \neq j$ , where  $p_{ij}$  is the probability that a cell arriving at input  $i$  is directed to output  $j$ . In our simulation

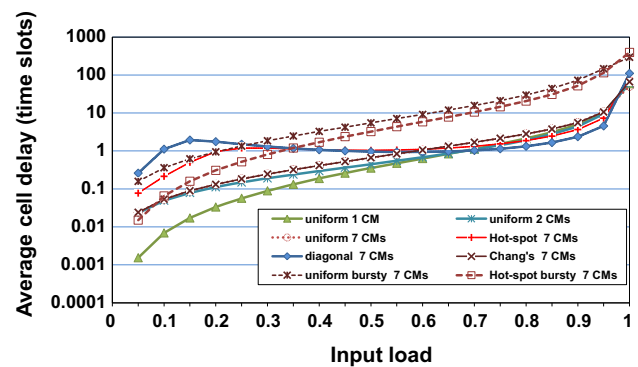


Fig. 4 Average cell delay behind the third stage of the MCNS

experiments, we used the same or similar traffic distribution patterns as other authors, in order to compare our results with other published results. We assume that the size of the buffers is unlimited.

Different kinds of performance measures were evaluated, mainly: throughput (as was defined in [21]), average cell delay, the VOMQs and output buffers size. Selected results obtained under 100% throughput for the average cell delay are shown in Fig. 4. In this figure, we also show the results obtained for uniform traffic and only one and two CMs, in order to show that, if the input traffic is close to uniform, it is possible to reduce the number of CMs by switching them off, and to achieve good results. We can see that for heavy input loads ( $p > 0.6$ ), the average cell delay is almost the same for all kinds of investigated traffic distribution patterns and the Bernoulli arrival model. Moreover, the average cell delay is shorter than 10 for  $p < 0.95$  regardless of the traffic distribution pattern. We can observe differences in average cell delay for  $p < 0.6$ . The best results were obtained for uniform traffic when only one CM was used. When  $n - 1$  CMs are used, cells have to wait longer for transmission through the CMs to the OMs, and the average delay is a little longer. Due to this mechanism, the average cell delay for diagonal traffic increases for  $p < 0.2$  to the value of 2, and then is reduced to 1. For bursty traffic, the average cell delay is longer than for the Bernoulli traffic, but it is less than 100 when  $p < 0.95$ .

In Fig. 5, the average delay of cells behind the first stage of the MCNS is shown. The average cell delay for uniform as well as non-uniform traffic distribution patterns is below 1 for a wide range of input loads and the greatest value of the average cell delay is less than 2. Compared to Fig. 4, we can state that cell delay is caused mainly by the output buffers located in OMs. The module matching scheme unloads VOMQs, and moves cells from IMs to OMs faster than the output links can transmit cells outside the system. This observation is confirmed in Fig. 6, where a comparison between the average VOMQs size and the output queues size is shown. For heavy input loads ( $p > 0.75$ ), the aver-

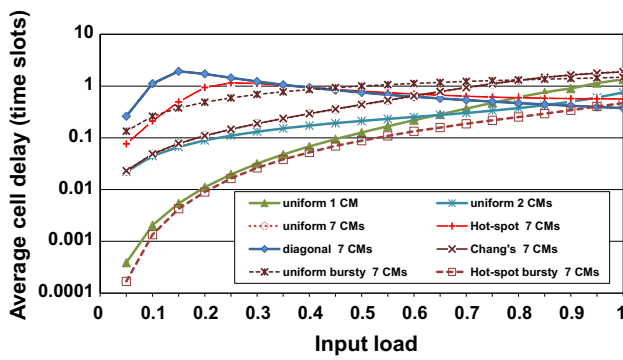


Fig. 5 Average cell delay behind the first stage of the MCNS

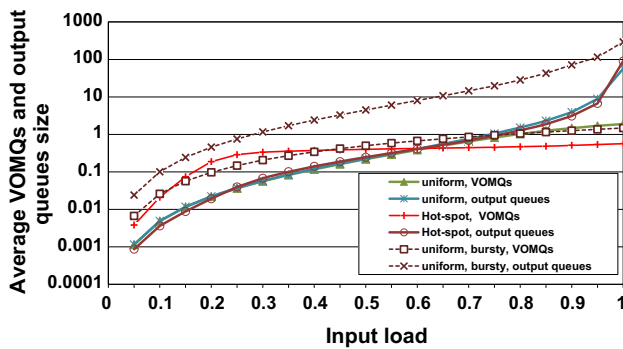


Fig. 6 Average size of VOMQs and output queues

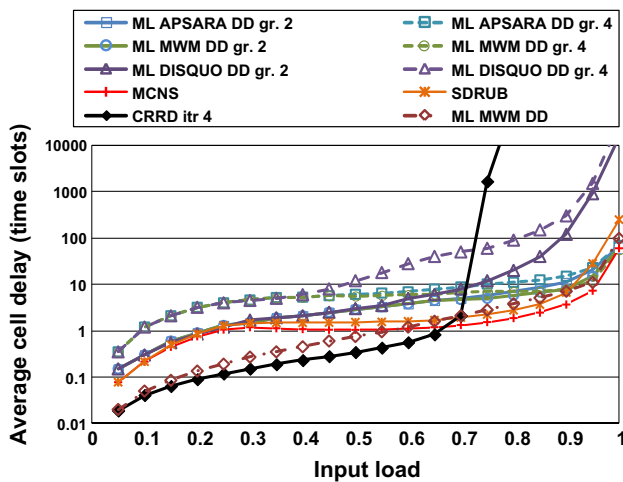


Fig. 7 Comparison of average cell delay under Hot-spot traffic obtained for the MSM Clos-network switch and MCNS using selected matching algorithms

age size of the output queues increases and the average size of VOMQs decreases. For the bursty traffic the average size of the output queues is greater than the average size of the VOMQs for the whole range of  $p$ .

Figure 7 presents a comparison of the results obtained under Hot-spot traffic for module-level matching schemes ML MWM DD, ML APSARA DD, and ML DISQUO DD

with grouped IM output links (gr. 2 and gr. 4), published in [13], with the results obtained for the MCNS, SDRUB, pure ML MWM DD, and CRRD (concurrent round-robin dispatching) [9]. It is possible to see that the ML DISQUO DD algorithm performs poorly, but the MCNS produces the shortest average cell delay using a simpler packet dispatching scheme. Slightly better performance than that provided by the MCNS for Hot-Spot traffic and input load  $p < 0.55$  can be achieved under the ML MWM DD scheme, but for  $p > 0.55$ , better performance is offered by the MCNS. The ML MWM DD scheme seems to be unfeasible in real equipment due to time and wiring complexity. It may be used for reference purposes. Figure 7 also shows the results received for the CRRD with 4 iterations. The CRRD is the basic packet dispatching algorithm proposed for the MSM Clos-network switch and performs well under uniform traffic. This algorithm uses the request-grant-accept handshaking routine, and the idea of its operation is completely different than that of module-level matching schemes. It performs quite well for input load lower than ( $p < 0.7$ ), but for a higher input load, the average cell delay rises very fast. This means that the MSM Clos-network switch under this scheme can achieve only about 70% throughput for Hot-Spot traffic.

### 8 Conclusions

In this paper, we evaluated the combinatorial properties of the MCNS, presented a module matching packet dispatching scheme based on central arbitration, discussed basic issues related to hardware implementation of the central arbiter, and showed selected simulation results. The MCNS can efficiently handle both uniform and non-uniform traffic distribution patterns. According to our experience, the proposed central arbitration scheme is not a bottleneck in scaling up the MCNS to a larger switching capacity.

The matching algorithm was implemented in hardware, using the Xilings-Kintex-7 FPGA chip, to check the response time of the central arbiter. The implementation proved that the IM-OM matching can be completed during 4 ns and 4.8 ns for a matrix of size  $32 \times 32$  ( $N = 1024$ ), and  $64 \times 64$  ( $N = 4096$ ), respectively. The time needed to collect all requests from IMs and distribute matching decisions to OM is also relatively short—about 11 ns. The obtained results encourage us to state that the proposed solution can serve a switching fabric of high-capacity. Moreover, the algorithm is simple in comparison to other module-level matching schemes, based on bipartite graph matching algorithms and/or request-grant-accept handshaking routine, and can be implemented using logic gates. The simplicity of the scheme does not worsen the algorithm performance. On the contrary, this performance, in terms of the average cell delay, is even better than for other

scheduling algorithms proposed for traditional MSM Clos-network switches. The proposed algorithm is very flexible; e.g., it is possible to send requests for any number of waiting cells if necessary. Without any changes in the operation of the central arbiter, we may redefine the high and low priority requests and also switch selected CMs off and on. The latter feature may be very useful for green networking in the future.

The MCNS needs more investigation concerning scalability problems. To decrease the delay, caused by collecting  $n$  cells to be sent using CMs when  $n$  is large, we propose to redefine the high and low priority requests. For example, high priority requests may be sent when the IM has more than  $n$  cells to be sent to  $OM(j)$ , and low priority requests—when the IM has fewer than  $n$  but more than 1 cells. It is also possible to employ multi-plane architectures to scale up the MCNS, but it needs a new arbitration scheme. The next very important problem is a control algorithm that can switch CMs off and on, to save energy without any degradation of performance for transported packets. We also plan to employ CQ switches in the first and the third stages of the MCNS to completely eliminate the speed-up issue.

**Acknowledgements** The work described in this paper was financed from the funds of the Ministry of Science and Higher Education for the year 2016 under Grant 08/82/DSPB/8214. The author would especially like to thank Adam Łuczak and Marek Michalski for helpful discussions on hardware implementation of the module matching algorithm.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Clos, C. (1953). A study of non-blocking switching networks. *The Bell System Technical Journal*, 32, 406–424.
- Kabacinski, W. (2005). *Nonblocking electronic and photonic switching fabrics*. Berlin: Springer.
- Beneš, V. E. (1965). *Mathematical theory of connecting networks and telephone traffic*. New York: Academic Press.
- Beneš, V. E. (1973). Semilattice characterization of nonblocking networks. *The Bell System Technical Journal*, 52, 697–706.
- Ackroyd, M. H. (1979). Call repacking in connecting networks. *IEEE Transactions on Communications*, COM-27(3), 589–591.
- Jajszczyk, A., & Jekel, G. (1993). A new concept-repackable networks. *IEEE Transactions on Communications*, 41(8), 1232–1237.
- Chao, H. J., & Liu, B. (2007). *High performance switches and routers*. New Jersey, USA: Wiley.
- Yoshigoe, K. (2011). The crosspoint-queued switches with virtual crosspoint queueing. In *Proceedings of the 5th International Conference on Signal Processing and Communication Systems, ICSPCS 2011* (pp. 277–281).
- Oki, E., Jing, Z., Rojas-Cessa, R., & Chao, H. J. (2002). Concurrent round-robin-based dispatching schemes for clos-network switches. *IEEE/ACM Transactions on Networking*, 10(6), 830–844.
- Rojas-Cessa, R., Oki, E., & Chao, H. J. (2010). Maximum and maximal weight matching dispatching schemes for MSM clos-network packet switches. *IEICE Transactions on Communications*, E93-B(2), 297–304.
- Rojas-Cessa, R., & Lin, Ch. B. (2006). Scalable two-stage clos-network switch and module-first matching. In *Proceedings of HPSR 2006*, Poznan, Poland (pp. 303–308).
- Lin, Ch B, & Rojas-Cessa, R. (2007). Module matching schemes for input-queued clos-network packet switches. *IEEE Communications Letters*, 11(2), 194–196.
- Xia, Y., & Chao, H. J. (2012). Module-level matching algorithms for MSM Clos-network switches. In *Proceedings of HPSR 2012*, Belgrade, Serbia (pp. 36–43).
- Kleban, J., Sobieraj, M., & Weclewski, S. (2007). The modified MSM clos switching fabric with efficient packet dispatching scheme. In *Proceedings of HPSR 2007*, New York, USA (pp. 241–246).
- Dong, Z., Rojas-Cessa, R., & Oki, E. (2011). Memory-memory clos-network packet switches with in-sequence service. In *Proceedings of HPSR 2011*, Cartagena, Spain (pp. 121–125).
- Dong, Z., Rojas-Cessa, R., & Oki, E. (2011). Buffered clos-network packet switch with per-output flow queues. *IEEE Electronics Letters*, 47(1), 32–34.
- Giaccone, P., Prabhakar, B., & Shah D. (2002). Towards simple, high-performance schedulers for high-aggregate bandwidth switches. In *Proceedings of INFOCOM 2002* (Vol. 3, pp. 1160–1169).
- Ye, S., Shen, Y., & Panwar, S. (2010). DISQUO: A distributed 100% throughput algorithm for a buffered crossbar switch. In *Proceedings of HPSR 2010*, Dallas, USA (pp. 75–81).
- Xia, Y., & Chao, H. J. (2011). StablePlus: A practical 100% throughput scheduling for input-queued switches. In *Proceedings of HPSR 2011*, Cartagena, Spain (pp. 16–23).
- Kabacinski, W., & Michalski, M. (2013). The control algorithm and the FPGA controller for non-interruptive rearrangeable  $\log_2(N, 0, p)$  switching networks. In *Proceedings of ICC 2013*, Budapest, Hungary.
- McKeown, N., Mekkittikul, A., Anantharam, V. V., & Walrand, J., (1999). Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47(8), 1260–1267.



**Janusz Kleban** an assistant professor at the Poznan University of Technology (PUT), Poland. He received the MSc and PhD degrees in telecommunications from PUT in 1982 and 1990, respectively. Since 1983 he has been working at the Institute of Electronics and Telecommunications at PUT, and now he is with the Chair of Communication and Computer Networks of the Faculty of Electronics and Telecommunications. His scientific interests cover packet dispatching algorithms for single and multistage switching fabrics, photonic broadband switch architectures, optical switching systems, control algorithms for lightwave networks, and Network on Chip (NoC). He has participated in many research projects for the telecom industry, as well as in the EU 6th and 7th Framework projects. He is the author or co-author of over 100 technical papers, chapters in books, and many unpublished reports. He served as a reviewer for IEEE Communications



Letters, Computer Communications, International Journal of Parallel, Emergent and Distributed Systems, IEEE Global Telecommunications Conference (GLOBECOM), IEEE International Conference on Com-

munications (ICC), IEEE Conference on High Performance Switching and Routing (HPSR), Interconnection Network Architectures On-Chip, Multi-Chip (INA-OCMC) Workshop.