POLITECNICO DI TORINO
Repository ISTITUZIONALE

Crosstalk-Preventing Scheduling in AWG-Based Cell Switches

Original
Crosstalk-Preventing Scheduling in AWG-Based Cell Switches / Bianco, Andrea; Hay, David; Neri, Fabio. - STAMPA. - (2009), pp. 1-7. (Intervento presentato al convegno IEEE GLOBECOM 2009 (Optical Networks and System Symposium) tenutosi a Honolulu, HI, USA nel December 2009) [10.1109/GLOCOM.2009.5425775].

Availability:
This version is available at: 11583/2353600 since:

Publisher:
IEEE

Published
DOI:10.1109/GLOCOM.2009.5425775

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

29 March 2024

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Crosstalk limiting schedulers in AWG-based optical switches

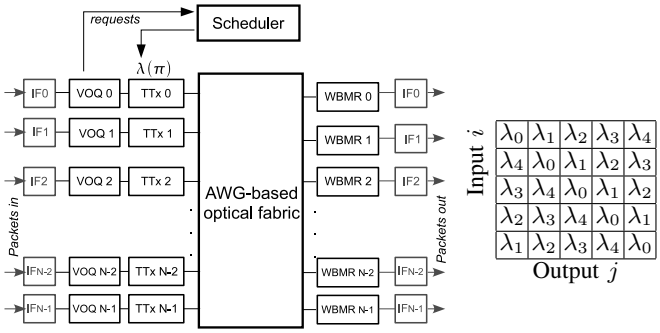
A. Bianco*, D. Cuda**, G. Gavilanes Castillo*, F. Neri*, M. Rodelgo Lacruz†,
F. J. González Castaño†‡, C. López Bravo‡, M. Salvat§
* Dip. di Elettronica, Politecnico di Torino, Italy, Email: {andrea.bianco, guido.gavilanescastillo, fabio.neri}@polito.it
† Gradiant, Spain, Email: mrodelgo@gradiant.org
‡ Universidade de Vigo, Spain, Email: javier.clbravo@det.uvigo.es
§ Universitat Politécnica de Catalunya, Spain, Email: maria.salvat@estudiant.upc.edu
** IEIIT - National Research Center, Italy, Email: davide.cuda@polito.it

***Abstract*—Optical switching fabrics are gaining interest as in multi Terabit switching devices; the advantages over their electronic equivalents are mainly their information density and power consumption. Arrayed Waveguide Gratings (AWGs) are promising optical devices proposed by the academic and industrial community to build optical switching fabrics. Because of their wavelength routing property, AWGs allow wavelength reuse over different ports introducing in-band crosstalk which strongly limits scalability of AWG-based backplanes. However, this effect can be mitigated or even completely avoided by means of proper scheduling algorithms. In this paper, we present several modified scheduling algorithms which limit the effect of coherent crosstalk in AWG-based switching fabrics and achieve good performance in terms of throughput and delay.**

I. Introduction

To cope with the continuous growth of the Internet traffic, increasing at a pace faster than the Moore's law, the urgency of deploying multi Terabit packet switches becomes evident. Current electronic technologies may not be able to support the realization of such high-end switching devices because they are approaching their physical limits, especially considering the maximum length that electronic signals can span before requiring regeneration and their increasing power requirements with the bitrate.

On the other hand, optics exhibit a switching complexity which is almost independent of the bit rate, negligible constraints on the length of internal interconnections, good scalability and lower power requirements [1]. One of the most promising approaches to optical switching fabrics is to use a passive wavelength routing device connecting tunable transmitters and fixed receivers. AWGs have been successful in the commercial deployment of Wavelength-Division Multiplexing (WDM) transmission systems. Their use was proposed in [2] in multi Terabit switching devices exploiting the wavelength dimension to perform switching operations. In addition, AWGs are relatively simple passive devices whose insertion losses depend weakly on the port count [3]. However, the shared optical media constituting AWGs strongly limit their scalability; internally, all light flows interfere on a concave slab waveguide to be coupled to the output waveguides; introducing in-band crosstalk when the same wavelength is reused over several inputs simultaneously. As a consequence, additional power is required to correctly receive signals at outputs. The impact of crosstalk in signal degradation has been analyzed in [4], posing a limit on the maximum size of a practical AWG device around 15 ports in the worst case (when all inputs are carrying traffic in the same wavelength).



(a) Switch architecture (b) λ-matrix for a 5×5 AWG

Fig. 1. Reference architecture.

However, the simultaneous use of the same wavelength can be avoided by properly scheduling packets, controlling the number of ports over which the same wavelength is reused. In [5] and [6], different solutions to control the concurrent usage of the same wavelength were proposed. Those solutions, even though optimal, are highly complex and require specialized hardware implementations.

In this paper we modify well-known scheduling algorithms to respect the wavelength constraint. These algorithms ensure good throughput performance with low complexity, thus offering a practical alternative for scheduling packets through AWG-based optical backplanes.

II. Reference Optical Architecture

The optical switching fabric architecture we are considering is shown in Fig. 1(a). It consists of an AWG connecting N tunable transmitters (TTx) and N fixed receivers. AWGs exploit the wavelength dimension to perform the switching operation. A data packet at an input port is forwarded to an output port depending on the input wavelength and port: at each input port, different wavelengths can be used to reach different output ports. As a consequence, at each output port,

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

information is received from different inputs with different wavelengths. Thus, if enough transceivers are available, an $N \times N$ AWG can be simultaneously traversed by N^2 independent data flows, one for each input/output pair, leading to a full mesh connectivity. However, the use of a single transceiver per port, limit to N the maximum number of data flowing through the switch at the same time. The specific wavelengths used to route information through an AWG depend on the design of the device, but commercial AWGs typically follow the standard ITU grid (with 100 GHz or 50 GHz spacing). Although different wavelength assignments are possible, we assume, with no loss of generality, that *i)* the $N \times N$ AWG operates with a set of N wavelengths $\Lambda = \{\lambda_0, \lambda_1, \ldots, \lambda_{N-1}\}$, and *ii)* at input i, information is delivered to output j (with $i, j = \{0, 1, \ldots, N-1\}$) using wavelength λ_w, with

$$w = (j - i) \bmod N \tag{1}$$

being w the wavelength channel number. As an example, the resulting input-output matrix defining the wavelength routing function of a 5×5 AWG, named λ-*matrix* is described in Fig. 1(b). This cyclic behavior is typical of the interferometric nature of the AWG, whose routing function is replicated over the wavelength axis with a period called Free Spectral Range (FSR). Our assumptions on the AWG behavior imply that only N wavelengths are necessary for the N^2 connection permutations over N input and N output ports. This architecture uses a single wavelength per port, derived from the single transceiver architecture, but given that cells can be received on any wavelength, since they can come from any port, receivers must be wide band and operate in burst mode (WBMR). The optical fabric does not include any active switching element: packet switching is actually controlled by the tunable transmitters (implemented with fast tunable lasers) and exploits the wavelength routing property of the AWG-based optical fabric.

III. Scheduler Framework

The switch is assumed to be synchronous and line cards process packets electronically by implementing input queues and other packet operations at the line card bitrate; this keeps electronic complexity within the single port-speed limit. The scheduler uses Virtual Output Queuing (VOQ) at inputs to avoid the Head of the Line blocking problem. Indeed, VOQs store cells at each input port in N separate FIFO queues, according to their destination port. At each time slot, a scheduler controls the Input Queue (IQ) switch transferring fixed size cells from at the most N inputs to N outputs taking up to one scheduling decision per time slot, since no speedup is available. At each scheduling decision a matching is defined such that at most one cell is transferred from each input port and to each output port. Thus, each scheduling decision is a input-output *permutation* $\pi = [\pi[0], \pi[1], \ldots, \pi[N-1]]$, with $\pi[i]$ denoting the output port which input i is transmitting to. If an optical switching fabric based on AWG is used to forward packets, the scheduler must provide both the input-output permutation π and its wavelength assignment $\lambda(\pi)$ (evaluated according to Eq. (1)). In the following, we say that a certain input-output permutation is k-*legal* if no wavelength is used more that k times according to the given wavelength assignment. Hence, k represents the maximum number of times a single wavelength can be used at different input ports during the same time slot.

The problem of constraining wavelength reuse to control crosstalk in AWG-based switching fabric has been addressed and solved in [5], [6]. It is shown that uniform traffic patterns can be scheduled using only 1-legal permutations with no speedup for switches with an odd number of ports and with $1 + 1/N$ speedup for switches with an even number of ports. Those solutions are highly complex, making them not practical. We focus on modifications of iterative maximal size matching algorithms [7], [8], [9] which usually ensure good performance and are simple enough to be implemented in hardware. In order to cope with the wavelength constraint, the proposed schedulers introduce a λ-*phase* that ensures that the selected permutations are k-legal, i.e., that each wavelength can be used to transmit packets from inputs to outputs at the most k times during the same time slot. As Fig. 1(b) shows, wavelengths on the different anti-diagonals of the λ-matrix are all different. Thus, a good scheduler have to select input-output pairs belonging to the anti-diagonals with high probability. We refer to an *anti-diagonal* as a set of N elements in an $N \times N$ matrix, such that no two elements are in the same row or column and they are all different. For instance, in Fig. 1(b) the vector $\{(1,5),(2,4),(3,3),(4,2),(5,1)\}$ corresponds to the main anti-diagonal and $\{(1,1),(2,4),(3,2),(4,5),(5,3)\}$ is a generalized anti-diagonal.

A. 2DRR, iSLIP and RDSRR

We consider variations of the well-known two-dimensional round-robin (2DRR) [7], of the iSLIP [8] and of the Rotating Double Static Round-Robin (RDSRR) [9] schedulers. We select these algorithms since they achieve good performance and can be easily modified to cope with the wavelength constraint.

In the algorithms we propose each input (output) is identified by a pointer, or arbiter, p_i^I (p_j^O), with $0 \le i \le N-1$ ($0 \le j \le N-1$). In addition, wavelengths are identified by λ-pointers (or λ-arbiter) p_w^W and by λ-counters c_w^W, with w being the wavelength identifier ($0 \le w \le N-1$). We briefly recall the original version of these three algorithms.

1) 2DRR: At each time slot, 2DRR scans the request matrix sweeping through a precomputed set of N generalized diagonals. The state of input VOQs is described by the request matrix, which is a matrix whose number of rows is equal to the number of inputs and the number of columns is equal to the number of outputs, and whose elements (i, j) equal 1 if input i has at least one packet occupying the VOQ associated with output j. A generalized diagonal is a set of N elements in an $N \times N$ matrix, such that no two elements are in the same row or column. Thus, at each time slot, 2DRR analyzes the whole request matrix in N iterations, each iteration being associated with one of the N precomputed diagonals. At each

iteration, 2DRR selects input-output pairs (i, j) which belong to the chosen generalized diagonal and are set to 1, provided that, input i and output j are both free, i.e., they have not been previously selected in any of the former iterations of the current time slot. Note that, to cover entirely the request matrix, 2DRR always iterates N times on each time slot (over the N different generalized diagonals).

2) iSLIP: iSLIP is a matching algorithm which exploits 2 arrays of N round-robin arbiters (one for each input and each output). It is based on three sequential steps which are performed in parallel on each input and output. At the beginning of each time slot, inputs and outputs are unmatched by default. The first iSLIP step is the *Request* phase: each unmatched input sends a request to every output for which it has a cell in the VOQ. The second step is the *Grant* phase (from outputs to inputs): if an unmatched output receives requests, it sends a grant to one of the inputs. Each output chooses the first requesting input found in a fixed round-robin scan of inputs starting from the highest priority element (indicated by the output pointer p_j^O associated with that output). The output notifies each input whether or not its request was granted. The final step is the *Accept* phase: if an input receives a grant, it accepts the one that appears first in a round-robin scan starting from the highest priority element (indicated by the input pointer p_i^I). These three steps are iterated several times to progressively increment the matching size on each iteration. Both input and output pointers indicating the highest priority element of the round-robin schedule are incremented (modulo N) to one location beyond the granted input if, and only if, the grant is accepted during the Accept phase.

3) RDSRR: RDSRR differs from iSLIP because of its pointer updating rule. Indeed, iSLIP performance relies on its ability to desynchronize the arbiters to point to different input-output pairs. The RDSRR scheduler still performs a Request, Grant and Accept phase but, pointers are always updated by one (modulo N) whether there is a grant or not to force full pointer desynchronization.

B. λ-2DRR

The λ-2DRR scheduler follows the same logic of the 2DRR algorithm, but, it exploits the N generalized anti-diagonals to scan over the request matrix. The selection of the anti-diagonals as a covering set of the request matrix implies that all the input-output pairs belonging to the same anti-diagonal can be transmitted using a different wavelength. Thus, the λ-2DRR minimizes the probability of contentions when choosing a wavelength. To ensure fairness, at each time slot, the initial anti-diagonal changes according to a fixed round-robin scheme. At the beginning of each time slot all the λ-counters are initialized to $c_w^W = 0 \ \forall w$ and each time wavelength w is used to match an input to an output c_w^W is incremented by one. The λ-2DRR selects input-output permutations (i, j) which belong to the chosen generalized anti-diagonal, provided that, the element (i, j) of the request matrix is set to 1, input i and output j are both free and their selection does not violate the k-legal constraint, i.e., the usage of wavelength $w = j - i \bmod N$ can be granted if $c_w^W < k$.

C. Centralized-iSLIP

The main gear of the centralized-iSLIP (C-iSLIP) is the λ-*vector*, an array containing the N λ-counters c_w^W. Thus, each of its elements is associated with a specific wavelength and records the number of times that a wavelength has been used during the current time slot. All the elements of the λ-vector are initialized to $c_w^W = 0$ at the beginning of a time slot.

The Request and Accept phases do not change. However, the Grant phase changes as follows: sequentially, each unmatched output j selects, among all the requests, the one coming from the highest priority input i (indicated by p_j^O) and it sends a request to the λ-vector for λ_w, with $w = j - i \bmod N$. The scheduler checks c_w^W and it grants λ_w to the requiring output if $c_w^W < k$, with k being the maximum number a wavelength can be used at each time slot; if λ_w is granted, c_w^W is incremented by one. Note that, the use of this wavelength is still subject to the Accept phase at the inputs. In the C-iSLIP, the Accept messages are sent from inputs to outputs and propagated to the wavelengths. If λ_w can not be used because its granting violates the k-legal constraint (i.e., $c_w^W \geq k$) the λ-vector does not grant the requiring output indicating that the wavelength request is Not ACKnowledged (NACK).

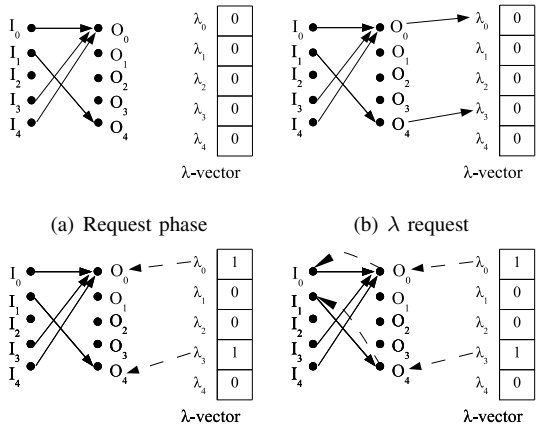
Once an output receives the ACK/NACK for the requested wavelength, the next output of the round-robin scheme is served. To be fair, at each time slot, the first output sending a request to the λ-vector is selected according to a round-robin scheme. After all the outputs have sent a request to the λ-vector, outputs which are still not matched to any wavelength (received a NACK) send another request for the wavelength corresponding to the input with the next highest priority. The Grant phase terminates either when all the outputs are matched to one wavelength, or when there are no requests from inputs left. The algorithm can perform several iterations, repeating the same procedure considering only unmatched inputs and outputs during previous iterations. Fig. 2 shows an example of the C-iSLIP sequence of operations when several inputs require output O_0 which selects input I_0 for transmission.

The main issue of C-iSLIP is that it can not be executed in parallel (all the outputs need to access sequentially the λ-vector). The Distribute-iSLIP and the λ-RDSRR schedulers solve this problem.

D. Distributed-iSLIP

In the Distributed-iSLIP (D-iSLIP) each of the N wavelengths is associated with a λ-pointer (p_w^W) and a λ-counter (c_w^W). Each output j is equipped with an additional pointer q_w^O that manages priorities among the different wavelengths.

The Request and Accept phases are unchanged, whereas the Grant phase is now divided in two steps (see Fig. 3). During the first step of the Grant phase (from outputs to wavelengths), the outputs forward the requests received (during the Request phase) to the corresponding λ-pointer according to Eq. (1). In the second step (from wavelengths to outputs and from



(a) Request phase (b) λ request

(c) $\lambda-$checking and granting (d) Grant phase

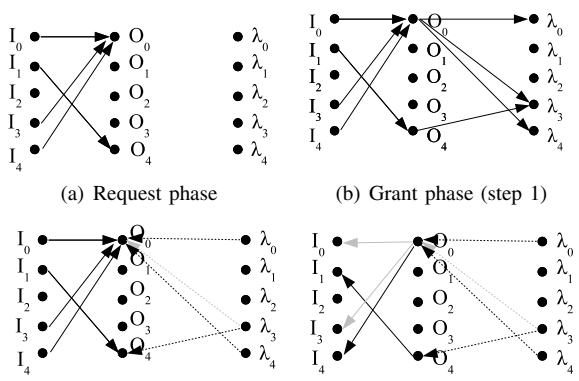
Fig. 2. C-iSLIP phases for $K = 1$ (Accept phase not shown)

outputs to inputs), each output collects the grants or NACKs coming from the λ-pointers. If an output receives a grant from more than one wavelength, it selects the grant corresponding to the first wavelength with the highest priority in the round-robin scheme (indicated by q_w^O). Then, each output sends a grant to the input associated with this wavelength. Pointers p_w^W and q_w^W are incremented (modulo N) one location above the granted wavelength if the grant is accepted during the Accept phase (Accept messages are forwarded by output pointers to wavelength arbiters). The two steps of the Grant phase are interleaved with the λ-phase. During the λ-phase, each pointer p_w^W receives requests forwarded by outputs during the first step of the grant phase. To respect the k-legal constraint, each pointer p_w^W can send up to k grants back to the outputs starting from the one with the highest priority according to the round-robin scheme. Each counter c_w^W tracks the number of times each wavelength is used. For all the other requests, the λ-pointers do not grant the requesting outputs, indicating a NACK. Fig. 3 shows an example of the D-iSLIP phase sequence, without the accept phase. Solid lines indicate requests (from inputs to outputs and from outputs to wavelengths), while dotted-dark (dotted-gray) lines stand for grants (NACKs).

E. λ-RDSRR

The λ-RDSRR algorithm is an adaptation of the Rotating Double Static Round-Robin (RDSRR) scheduler [9] which differs from iSLIP in the pointer updating rule and because unlike the D-iSLIP, λ-RDSRR selects the wavelength at the inputs. Again, each wavelength is associated with a λ-pointer p_w^W and with λ-counter c_w^W, but the wavelength selection is performed at the inputs. Initially, the RDSRR initializes the input and the output pointers to $p_i^I = (-i) \bmod N$ and $p_j^O = (-j) \bmod N$, respectively. λ-pointers are set to $p_w^W = w$. Note that, this initialization corresponds to one of the generalized anti-diagonals and is only one of N possible ways to initialize pointers. Note that each input points to an output that reciprocally points to that same input, so that wavelength assignment is selected to give priority to those pairs.

At each time slot the λ-RDSRR performs five phases.



(a) Request phase (b) Grant phase (step 1)

(c) λ phase (d) Grant phase(step 2)

Fig. 3. D-iSLIP phases for $K = 1$ (Accept phase not shown)

The *Request* and *Grant* phases are identical to iSLIP (except that pointer updates are independent of assignations) and the *Accept* phase is postponed to the λ-phase, which is composed by the following two steps. The first step is the *Wavelength-Request* step: if an input receives one or more grant, it selects the one which appears next in a fixed round-robin schedule, starting from the input pointer p_i^I, and requests the associated transmission wavelength λ_w to the corresponding wavelength pointer according to Eq. (1). The second step of the λ-phase is the *Wavelength-Grant* step; according to the k value, each wavelength pointer p_w^W grants the first $k - c_w^W$ requests following the fixed round-robin schedule starting from the first highest priority input (as indicated by p_w^W). The counter c_w^W is updated to reflect packet assignations, i.e., it is updated only if the input-output permutation has been definitively selected by the scheduler during the Accept phase. The scheduler can run more iterations, at each iteration considering only inputs and outputs which are still free and wavelengths which are still compliant with the k-legal constraint ($c_w^W < k$).

To maintain the initial pointers' scheme which gives priority to input-output pairs on the anti-diagonal, at each time slot p_i^I and p_j^O pointers are incremented (modulo N) while p_w^W pointers are decremented (modulo N) *regardless of the packet assignments*. Furthermore, the search direction is reversed each time slot to improve fairness in case of non uniform traffic [9].

IV. RESULTS

We present here the performance of the proposed algorithms through simulations. Each input is equipped with N queues, one for each output, with a capacity of 10000 cells at each queue. C-iSLIP, D-iSLIP and λ-RDSRR iterates $\log_2(N)$ which usually is enough to ensure that the found matching is maximal. By definition, λ-2DRR iterates N times. In the reported plots N indicates the switch number of ports, k indicates the k-legal constraint and I denotes the number of iterations each algorithm runs at each time slot.

Due to space limitations, we present results for uniform traffic scenario. Let ρ_i be the load at input port i with $0 \leq i \leq N-1$, and ρ_{ij} the traffic that input port i transmits to output port j. In uniform traffic, $\rho_{ij} = \frac{\rho_i}{N}$.

Fig. 4 shows the delays for the different algorithms we propose under uniform traffic conditions when the switching

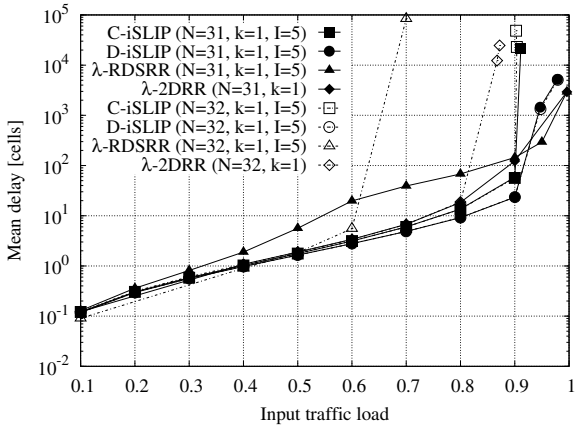


Fig. 4. Delay performance for AWG-based switch with an even number of ports ($N = 32$) and an odd number of ports ($N = 31$)

fabric presents either an even number of ports ($N = 32$ - white-filled markers) or an odd number of ports ($N = 31$ - black-filled markers). For odd N switching fabrics, all the schedulers ensure good performance, the λ-RDSRR shows higher delay because of the fixed pointer updating scheme. The D-iSLIP, the λ-2DRR and the λ-RDSRR ensure almost 100% of throughput, while the delays for C-iSLIP saturate for an input load around 0.9. Performance worsens for AWG-based switching fabrics with an even N and no algorithm is able to achieve 100% of throughput. Again the one performing the worst is the λ-RDSRR followed by the λ-2DRR. Note that, these results agree with the theoretical results presented in [6] where the authors showed that for AWG-based switching fabric with an even N, the maximum achievable throughput is limited to $1 - 1/N$ if no speed up is available.

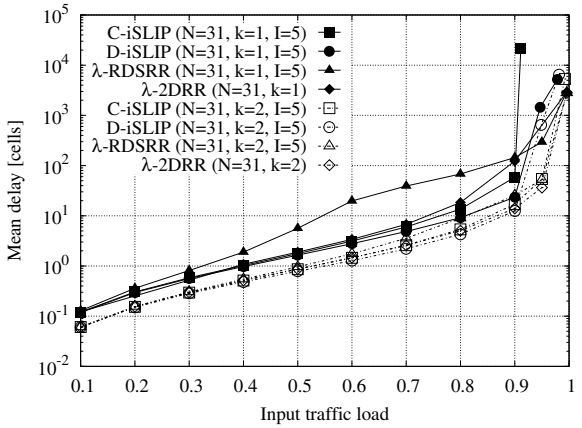


Fig. 5. Delay performance of the different algorithms under uniform traffic when either 1-legal or 2-legal permutations are allowed.

Fig. 5 shows the delay performances for a $N = 31$ ports switch when either 1-legal (black-filled markers) or 2-legal (white-filled markers) permutations are allowed. The relaxation of the k-legal constraint reduces delays since it easier for the scheduler to find an allowed matching. The λ-RDSRR is the algorithms whose performance improves the most passing from $k = 1$ to $k = 2$, since it becomes easier to find input-output permutations which do not belong strictly to the anti-diagonals. Indeed, the λ-RDSRR pointer updating rule is fixed and does not depend on the packets forwarded by the switch. C-iSLIP performance improvement is remarkable too, since it is not able to achieve 100% when $k = 1$, while it ensures 100% of throughput when $k = 2$. The D-iSLIP and the λ-2DRR schedulers proved themselves less sensitive to the k-legal constraint, indeed, only a marginal improvement in the delay can be observed.

V. Conclusions

We presented four heuristics based on maximal size algorithms solutions to control optical backplanes and we evaluated their performance. All the proposed algorithms are simple enough to be easily implemented in hardware and, in particular, the D-iSLIP algorithm achieves the best performance in term of throughput and delay. Thus, even though sub-optimal, the proposed algorithms seem to be good candidates to schedule packets in future optical switching fabrics. Finally, the proposed heuristics allow to drastically overcome the in-band crosstalk limitation to scalability of AWG-based switching fabrics, without worsening performance neither increasing significantly the scheduler complexity; thus, making AWG a viable solution to build future all-optical switching fabrics.

Acknowledgment

Special thanks are due to David Hay for helpful discussions and his valuable insights.

This work was partially supported by the BONE project, a Network of Excellence funded by the European Commission within the 7th Framework Programme, and by the PHOBOS 09TIC014CT grant (Xunta de Galicia, Spain).

References

[1] E. Bonetto, L. Chiaraviglio, D. Cuda, G. Gavilanes, and F. Neri, "Optical technologies can improve the energy efficiency of networks," in *35th European Conference on Optical Communication*, Vienna, Austria, 2009.

[2] J. Gripp, M. Duelk, J. Simsarian, A. Bhardwaj, P. Bernasconi, O. Laznicka, and M. Zirngibl, "Optical switch fabrics for ultra-high-capacity IP routers," *Journal of Lightwave Technology*, vol. 21, no. 11, pp. 2839–2850, Nov 2003.

[3] J. M. Finochietto, R. Gaudino, G. A. Gavilanes, , and F. Neri, "Simple optical fabrics for scalable terabit packet switches," in *IEEE ICC*, Beijing, China, 2008.

[4] H. Takahashi, K. Oda, and H. Toba, "Impact of crosstalk in an arrayed-waveguide multiplexer on $N \times N$ optical interconnection," *Journal of Lightwave Technology*, vol. 14, no. 6, pp. 1097–1105, Jun 1996.

[5] M. Rodelgo-Lacruz, C. López-Bravo, F. J. G.-C. no, and H. J. Chao, "Practical Scalability of wavelength routing switches," in *IEEE ICC*, Dresden, Germany, 2009.

[6] A. Bianco, D. Hay, and F. Neri, "Crosstalk-Preventing scheduling in AWG-Based Cell Switches," in *IEEE Globecom*, Honolulu, Hawaii, USA, 2009.

[7] R. . LaMaire and D. N. Serpanos, "Two-dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Transactions on Networking*, vol. 2, no. 5, pp. 471–482, Oct 1994.

[8] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, Apr 1999.

[9] Y. Jiang and M. Hamdi, "A fully desynchronized round-robin matching scheduler for a VOQ packet switch architecture," in *IEEE workshop on high performance switching and routing*, Paris, France, Jun 2001, pp. 407–412.