Complexity Analysis and Performance Evaluation of a Two-Step Scheduler for Modular Optical Packet Switches

(Invited Paper)

P. G. Raponi, N. Andriolli, P. Castoldi
Scuola Superiore Sant'Anna, Pisa, Italy
Email: raponi@sssup.it

A. Bianchi
Ericsson, Pisa, Italy

***Abstract*—In this paper we evaluate the complexity and the performance of two different schedulers for high-speed optical switches with a large number of ports. We compare the classical approach with a two-step scheduling framework recently proposed for multi-card optical switches in which firstly transmitters on each card are assigned a different wavelength, then for every wavelength a maximal matching is found among all cards.**

We demonstrate that the two-step scheduler is characterized by a lower complexity and it is thus more scalable than the classical approach. Then we analyze the performance of the proposed framework under bursty traffic and the benefits introduced by the use of transfer speedup, a technique which allows to send more than one packet in the same time slot, while issuing a single scheduling decision.

Results obtained by simulations on a wide range of cards and wavelengths, show that the two-step scheduling framework achieves a performance comparable to the classical approach at high loads. Moreover the introduction of a small transfer speedup is shown to significantly reduce the average cell latency.

I. Introduction

Pushed by the advances of the new generations of microprocessors, computer systems performance significantly improved over the last years. Data centers and server farms have increased the amount of data they can store and process [1], [2]. A steady growth has affected also the global Internet traffic: to cope with this, commercial router capacity per rack has then doubled every eighteen months [3].

Within these high-end servers and routers a dedicated interconnection network is in charge of providing the necessary communications capability among all system stations. However, currently exploited electronic interconnection networks are approaching their physical limitations, mainly in terms of number of backplane interconnections and power density [4]. Indeed each server/router generation needs more power than the previous one, making the packaging into a single rack of equipment more and more difficult. To mitigate the power density issue, the currently exploited solution is to split the whole system into several racks. Nevertheless the overall required power and the footprint are increased, and novel issues related to inter-rack wiring and communications must be solved. Furthermore the traditional switch design implies an high initial investment due to the high cost of the electronic switching fabrics used nowadays [5].

The introduction of optics within high-end servers and routers has the potential to solve many of these issues, helping to scale to higher capacities [2], to reduce the power consumption, the footprint, as well as the initial cost for partially equipped systems. To this aim, switch modularity can be a key feature to guarantee the switch scalability, i.e., the ability to cope with increasing traffic in a cost-effective way. On the contrary, in this paper we exploit a modular optical switch where a passive optical backplane connects a variable number of line cards [6]. This allows to keep the connecting device as simple as possible, while pushing the intelligence toward the cards. Each line card hosts a number of ports, each one composed of a fast tunable transmitter and a fixed receiver. A two-step routing is then performed, based on space routing to reach the destination card, and on wavelength routing to reach the destination port on the card. With this purpose, a two-step scheduling framework is adopted, capable of seamlessly adapting to switch architectural constraints: first, transmitters on each card are assigned a different wavelength, then for every wavelength a matching is found among all cards. To further improve switch performance, the transfer speedup technique is exploited, which allows to send more than one packet within a time slot, while issuing a single scheduling decision.

The aim of this paper will be to demonstrate that the proposed two-step scheduler is computationally simpler than the classical approach. Then the scheduler behavior in the modular optical switch will be analyzed under bursty traffic, assessing also the benefits of transfer speedup. Finally, the cell latency experienced in switches with different configurations of cards and wavelengths will be investigated.

II. Architecture

The considered reference architecture is the modular optical switch based on space and wavelength routing shown in Fig. 1. The architecture is scalable since a variable number of cards (or modules) can be installed in the switch, each one with a fixed number of ports, i.e., transceivers.

This work was supported, in part, by Ericsson through a grant to Scuola Superiore Sant'Anna.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7268
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7268

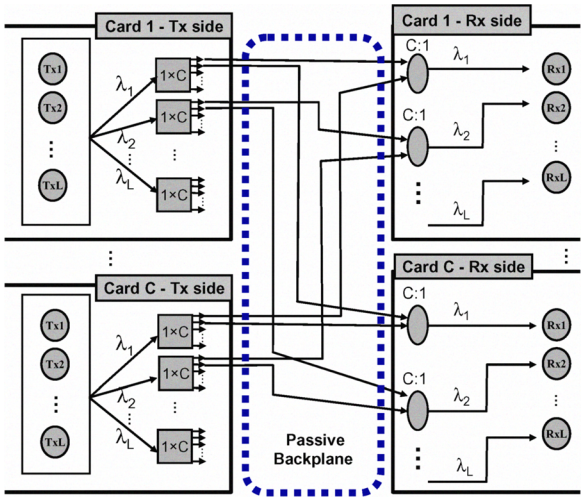

Fig. 1. Modular optical switch architecture.

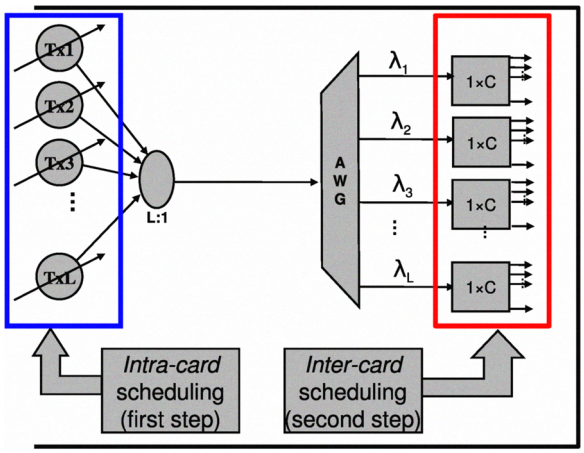

Fig. 2. Scheduling steps.

The switch is composed of up to C independent cards with L ports each (the cards are shown Fig 1 divided into a transmitter and a receiver side for ease of understanding). The transmitter side of each card is composed of L fast tunable transmitters generating a comb with up to L wavelengths. The comb is then demultiplexed and each wavelength is directed to a $1 \times C$ space switch, as shown in Fig. 2. Each transmitter is input-buffered and the buffer is partitioned into $C \cdot L$ virtual output queues (VOQs). The receiver side of each card has L fixed receivers, each of them preceded by a $C:1$ coupler.

Proper connectivity between all card transmitter and receiver sides is ensured by a passive optical backplane. In the considered architecture, the outputs of the first $1 \times C$ switch of each card (switching the wavelength λ_1) are connected with the first $C:1$ coupler of each card (collecting all outputs on λ_1), and so on, as shown in Fig. 1.

The architecture is synchronous and fixed length packets (i.e., cells) are forwarded at each packet time.

III. Scheduling

The scheduler is in charge of assigning a packet to every transmitter in order to optimize the switch performance while complying with the architectural constraints.

We model our switch as a bipartite graph (S, D, E); S is the set of source nodes (i.e., tunable transmitters) and D the set of destination nodes (i.e., fixed receivers), with $|S| = |D| = C \cdot L$. We denote with $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}$ the set of used wavelengths. Every edge $e \in E$ goes from some $s \in S$ to some $d \in D$. Each source or destination node is identified by a couple $s = (c_s, i)$ and $d = (c_d, j) = (c_d, \lambda_j)$, where c_s and c_d are the transmitter and receiver card indexes respectively, i and j the transmitter and the receiver indexes respectively. The destination side has fixed receivers, hence a biunique relationship holds between the index of the receiver in a card and its wavelength. E represents the possible set of transmissions between S and D.

The aim of the scheduler is to find a matching in a subset of edges $X \subset E$ according to the following constraints given by the switch architecture (with the notation $\overline{x}$ we denote that x is given): at the transmitter side, $\lambda_i \neq \lambda_j$ with $\lambda_i, \lambda_j \in \Lambda$ for edges exiting from the same card; at the receiver side, $\lambda_i \neq \lambda_j$ with $\lambda_i, \lambda_j \in \Lambda$ for links reaching the same card. In other words, the transmission constraints can be abstracted as follows:

1) $\forall s = (\overline{c}, i) \in S$ with $i = 1, 2, \ldots, L$, X contains one or zero edge connecting to $d = (c, \overline{\lambda}) \in D$ with $c = 1, 2, \ldots, C$,
2) $\forall d = (\overline{c}, \lambda_j) \in D$ with $\lambda_j \in \Lambda$, X contains one or zero edges connecting from $s \in S$.

Our approach is based on a two-step scheduling framework, which:

- satisfies the architectural constraints in a sequential manner, firstly assigning the wavelength to use to each tunable transmitter and then performing a best matching on a set of bipartite graphs;
- fits the modular design in a way that the introduction of further cards does not significantly influence the computational complexity of neither the first nor the second step.

The two steps in which the scheduling problem can be divided, namely the *intra*-card scheduling aimed at setting the transmitters and the *inter*-card scheduling aimed at setting the $1 \times C$ switches, are highlighted in Fig. 2.

The first step (i.e., intra-card), takes charge of the wavelength assignment in a single card so that $\forall s = (\overline{c}, i) \in S$ with $i = 1, 2, \ldots, L$; $(\overline{c}, i) \Rightarrow (\overline{c}, \lambda_k)$, with $\lambda_k \in \Lambda$. The algorithm is independently run in each card, based only on the queue status in each transmitter, thus no global or inter-card information is required. At the end of the first step every transmitter has been assigned a wavelength complying with the first constraint explained above. Then the second constraint can be accomplished in the second (i.e. inter-card) step: since the destination ports on each card are identified by the wavelength, each $s \in S$ can only be assigned an edge

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7268
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7268

toward a single $d \in D$ for each card. Therefore the matching problem is now reduced to find the best matching in L bipartite graphs composed of up to C nodes.

It is worth noting that no assumption is made on the specific algorithms to be adopted in each step of the scheduling framework: they could be either a customization of existing schemes or brand-new ones, specifically designed according to the architecture specifications. Since all intra-card scheduling are independently performed on each card, and wavelength-by-wavelength inter-card scheduling can be parallelized as well, this framework allows to keep the computational complexity tractable.

We now discuss more in detail the two steps in which the scheduler is divided.

A. First Step

The first step aims at selecting the most suitable wavelength for each transmitter on every card, as shown in Fig. 2.

Each transmitter has $C \cdot L$ VOQs, each of them collecting the cells to be delivered to the destination (c, λ), with $c = 1, 2, \ldots, C$ and $\lambda \in \Lambda$. For sake of representation, we consider them organized in a hierarchical fashion, first according to their destination wavelength and then according to their destination card, as depicted in Fig. 3. Each transmitter keeps L counters $q(\lambda)$, i.e., one for each wavelength λ. Then the algorithm sorts the L^2 counters present on the card (L for each transmitter), chooses the largest one, and sets the relative transmitter to the corresponding wavelength, and so on until all the L wavelengths have been assigned to a different transmitter.

Different values can be assigned to each counter, corresponding to different wavelength assignment metrics, e.g. the Queue Length (QL) metric or the Cell Age (CA) metric. With the QL metric $q(\overline{\lambda})$ of each transmitter stores the total number of cells waiting to be transmitted on wavelength $\overline{\lambda}$, i.e. toward the destinations $(c, \overline{\lambda})$, with $c = 1, 2, \ldots, C$. QL thus prioritizes the queues according to their length. On the other hand, with the CA metric $q(\overline{\lambda})$ of each transmitter stores the age (i.e., the number of time slots spent in the queue) of the oldest cell waiting to be transmitted on wavelength $\overline{\lambda}$. CA therefore gives priority to the queue with the oldest cell.

It is worth noting that the first step is independently run in each card, because no inter-card communication is required.

B. Second Step

The second step aims at setting the $1 \times C$ switches by finding the best destination card for each transmitter on every card.

The transmitters have been assigned a wavelength in the former step, thus the scheduling can be done simultaneously wavelength by wavelength, since no information except for the status of the queues is needed. For each wavelength $\overline{\lambda}$, a C–node bipartite graph must be populated, composed of the C transmitters (one for every card) tuned on $\overline{\lambda}$ and of the C receivers (one for every card) fixed on $\overline{\lambda}$. Therefore the second step reduces to finding the best matching on a set of

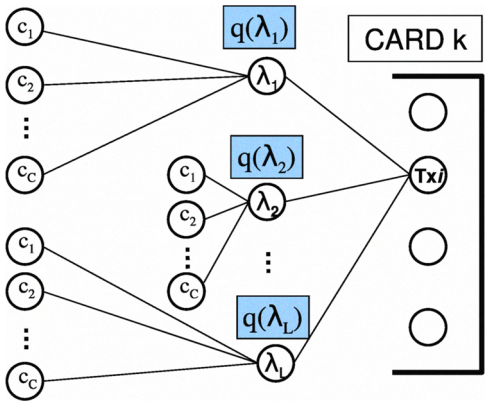

Fig. 3. Queue hierarchical organization.

L independent bipartite graphs. Several scheduling algorithms present in literature can be used to solve this task.

It has been proved that by using the optimal maximum weight matching (MWM) algorithm, 100% throughput can be achieved for independent, identically distributed arrivals [4], [7]. However MWM complexity is too high for practical implementations. A broad class of algorithms, called maximal matching algorithms, approximate MWM: they lead to a maximal match by incrementally adding connections without removing previously made connections. In a maximal match, if a non-empty input is not connected to any output, then all the destination outputs of the cells waiting in that input must be matched with some other input [4]. Many scheduling algorithms for input-queued switches have been proposed, with the common goal of offering scalability together with low delay characteristics, such as PIM [4], DRRM [8], iSLIP [9], exploiting multiple iterations to converge on a maximal matching. However such maximal matching algorithms, though simple, achieve a limited performance under bursty arrivals. To counteract this issue, transfer speedup can be applied to the two-step scheduling framework, as described in the following section.

C. Transfer speedup

A switch with a speedup of 1 is said to allow at most one cell from each input to reach the destination output during one time slot. Typically in the literature it is assumed that if a switch has a speedup of s, s scheduling decisions can be issued during each time slot and correspondingly s transmissions of cells may occur from input queues to output ports. We shall define such speedup as *scheduling speedup* [10], [11], [12].

Scheduling speedup however is not a scalable method to face the increasing traffic, because as line rates increase and thus the duration of packets decreases, the scheduling time becomes a limiting factor. To solve this issue, as first stated in [11], it is possible to define another kind of speedup, namely the *transfer speedup* t. With the transfer speedup just one scheduling decision is issued during each time slot, while the rate at which cells can be transferred from input buffers to output ports is multiplied by a factor of t compared to the

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7268
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7268

external cell arrival rate. In other words, during each time slot the switch is set once and a train of t cells is transmitted on each established input-output connection.

It is clear that whenever applying a speedup greater than one, either scheduling or transfer speedup, buffering is required at the output ports, since more than one cell may arrive during a single time slot. Therefore the simple input queued (IQ) switch must evolve into a more complicate switching architecture commonly referred to as combined input-and-output-queued (CIOQ) [13]. Nevertheless the physical implementation of the transfer speedup in the modular optical packet switch in Fig. 1 requires only minor architectural modifications and is much simpler than in an electronic fabric, because of the transparency of optics to data rates. Indeed the only required change is a multiplication by a factor of t of the transmitter modulation rate and of the receiver bandwidth. Moreover the electronic bottleneck, which may arise for increasing t in terms of modulator and photodiode bandwidth and output buffer writing speed, can be overcome by applying well-known optical multiplexing techniques, such as optical time division multiplexing (OTDM).

IV. Complexity

The choice of a particular scheduler over another in the design of a switch depends on a number of factors. First of all, it must be simple to implement. Indeed complexity requires generally more chip area and power, and a computation time that spans over multiple cell slots. In addition, the scheduler should provide high throughput even under bursty traffic, since real network traffic is highly correlated from cell to cell.

In this section we will compare the implementation complexity of the two-step scheduling framework with the classical single-step approach. First, an asymptotic complexity analysis will be performed, and then a detailed comparison of the worst case scenario will be carried out.

A. Asymptotic complexity

In the first step of the proposed scheduling framework the complexity of any chosen metric is given by the complexity of sorting a vector of L^2 elements, i.e., $O(L^2 logL^2) = O(L^2 logL)$. The second step complexity depends on the chosen bipartite graph matching algorithm. Since the graph size is at most equals the number of cards C, if *iSLIP* (or other maximal matching iterative algorithm) is selected, the complexity becomes $O(C^2 logC)$ [14], [15]. However some iterative algorithms, such as iSLIP, can be run on parallel processors, reducing the complexity to $O(C logC)$ [14], [15]. In both steps, the algorithms are run simultaneously, in the first step card by card, and in the second step wavelength by wavelength. Thus the total complexity, supposing to fully exploit parallelization, is in the order of:

$$O(L^2 logL) + O(C logC) \tag{1}$$

For comparison, we now suppose to apply a classical single-step scheduling algorithm on a switch with the same total number of input/output ports, i.e., LC. Since the graph size is now LC, the complexity using a maximal matching iterative algorithm, such as iSLIP, is:

$$O(LC\, logLC) \tag{2}$$

It can be verified that the asymptotic complexity (i.e., as L and C increase) of the classical approach in Eq. 2 is greater than the one of the two-step scheduling framework in Eq. 1. This complexity however gives information only on the asymptotic behavior and thus it cannot be used to directly compare the two approaches in a limited domain of cards and wavelengths. For this purpose a comparison of the exact number of operations performed in the worst case is required, as described in the following section.

B. Worst case analysis

In order to precisely compare the complexity of the two approaches in a worst case scenario, an accurate estimation of the total number of operations is computed. It is worth noting that this choice provides the highest degree of generality and allows a fair comparison between the two approaches. However the proposed metric does not take into account the complexity reduction attained by exploiting parallel implementation of specific sorting or scheduling algorithms. For what concerns the first step of the proposed framework, the problem of determining the exact number of operations is non trivial for a number of reasons. Multiple implementations of several sorting algorithms can be used, with different average and worst case complexities. We decided to use as a term of comparison the sorting algorithm named *Introsort* [16], as it is used in SGI Standard Template Library (STL), it is not optimized for any particular type of data, nor designed to be parallelized, and it is well studied in the literature. The total number of operations in the worst case is [16], [17]:

$$30.39\, N\, logN + 352.5\, N - 14.0$$

Regarding the second step of the proposed framework and the classical approach, the total number of operations done by iSLIP (for a N–graph) is $3\, N^2\, logN$ since in each iteration 3 steps (namely request, grant and accept) are performed throughout all the nodes [18], [19].

Thus, the total number of operations needed with the proposed two step framework is:

$$Z_1(L,C) = \underbrace{30.39\, L^2\, logL^2 + 352.5\, L^2 - 14.0}_{1^{st} step} + \underbrace{+\, 3\, C^2\, logC}_{2^{nd} step}$$

while with the classical single-step approach (i.e., LC–node bipartite graph matching using iSLIP) is:

$$Z_2(L,C) = 3\, L^2 C^2\, logLC$$

We can then evaluate the worst case complexity difference $Z(L,C)$, between the classical approach and the two-step framework:

$$Z(L,C) = Z_2(L,C) - Z_1(L,C) \tag{3}$$

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7268
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7268

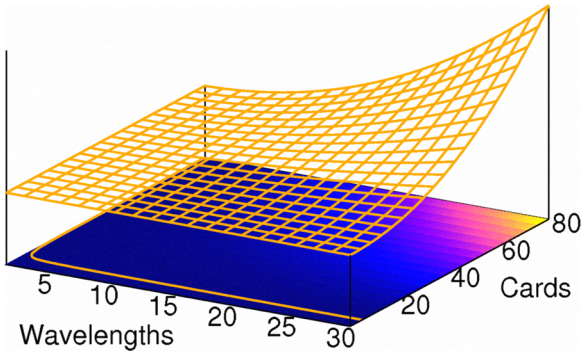

Fig. 4. 3D plot of Eq. 3. The solid line in the bottom plane shows where $Z_1(L,C) = Z_2(L,C)$.

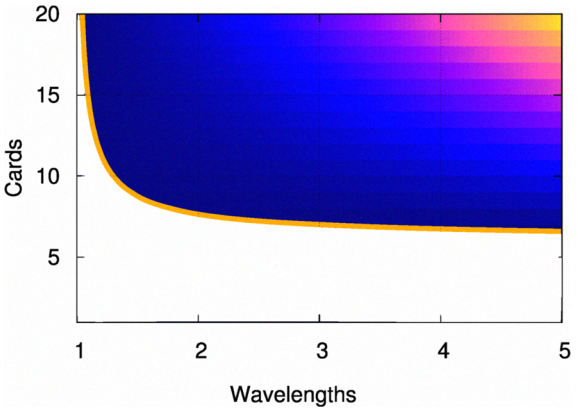

Fig. 5. Zoom plot of the region close to the origin. The white region represents where $Z_1(L,C) > Z_2(L,C)$.

whose 3D plot is shown in Fig. 4. It is worth noting that the function $Z(L,C)$ is defined only for $L,C \in \mathbb{N}$, but it is shown in Fig. 4 and Fig. 5 for real values of L and C for ease of understanding. As already pointed out, asymptotically $(C,L \rightarrow \infty)$ the complexity of $Z_2(L,C)$ is greater than $Z_1(L,C)$.

In Fig. 5 the region close to the origin (i.e., for low values of L,C) is detailed. The plot highlights with a solid line the curve corresponding to $Z(L,C) = 0$. The white region denotes where $Z_2(L,C) < Z_1(L,C)$, or otherwise said, the (L,C) values for which the classical single-step approach is computationally more efficient. From this figure we can notice that just few cards are sufficient to make the two-step framework preferable over the classical approach for any $L > 1$.

V. Simulated Performance

In this section the latency performance of the modular optical switch is evaluated through simulations: four configurations of cards $C = \{32, 64\}$ and ports/wavelengths per card $L = \{10, 20\}$ have been tested. Each VOQ can store 1000 cells, dropping the exceeding ones.

The chosen first step metric is QL, which usually suits any second step scheduling decision since it tends to select wavelengths with many queued packets. The chosen second step algorithm is *iSLIP*, because of its reduced complexity and good performance even with a small number of iterations.

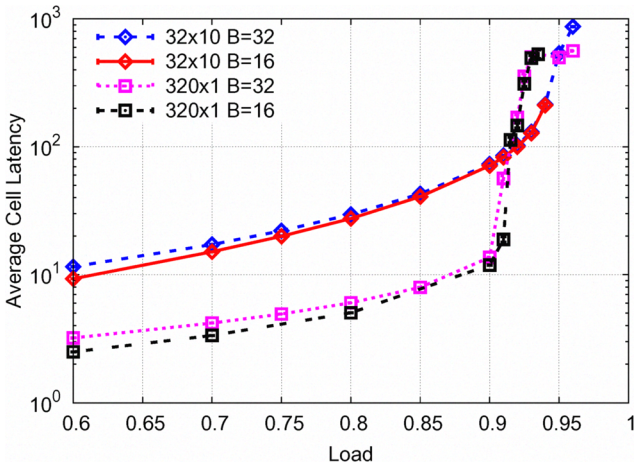

Fig. 6. Latency comparison of two switches with 320 input/output ports.

The bursty traffic is modeled as an ON-OFF Markov chain [20]: cells continuously arrive at input ports during geometrically distributed ON periods, whose average duration B is either 16 or 32 cells. If p and q denote the probabilities that the Markov chain remains in the ON and OFF state respectively, it is possible to relate p and q with the mean burst length B and the mean arrival time λ through the following:

$$B = \frac{1}{1-p}$$

and

$$\lambda = \frac{1-q}{2-q-p}$$

Following the common use notation, we denote the service rate with μ and the load with $\rho = \lambda/\mu$. We will consider from now on the load normalized to $\mu = 1$. So given $B = \{16, 32\}$ and the conditions $p, q < 1$, it is possible to achieve a load up to 0.941 and 0.969 respectively. The simulations are run until the confidence interval of the average cell latency is below 5% at 95% confidence level. The 95% confidence intervals, estimated by using replication method [21], are shown in all the figures, but may be too small to be visible.

Fig. 6 shows the average cell latency versus the offered load for two switches having a total number of 320 input/output ports, comparing the classical single-step approach with the proposed two-step framework, where the 320 input ports are divided into 32 cards with 10 wavelengths each. It must be considered in the comparison that while the total number of ports is the same, the 320×1 switch exploiting classical approach has looser architectural constraints, i.e., the wavelength assignment is avoided. Thus an inherent difference in comparing the mean latency difference must be considered. It is thus expected that the classical approach experiences lower delay for light loads, due to the impact of the first step, which introduces an additional delay depending on the number of wavelengths used. However, as the load increases the classical

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7268
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7268

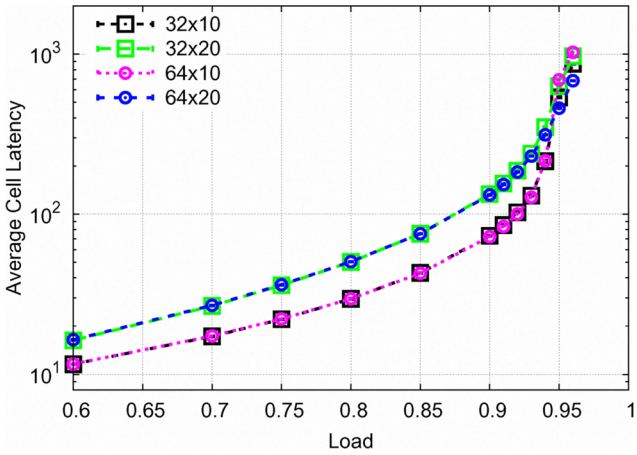

Fig. 7. Average cell latency versus load for different configurations with $C = \{32, 64\}$ and $L = \{10, 20\}$ and $B = 32$.

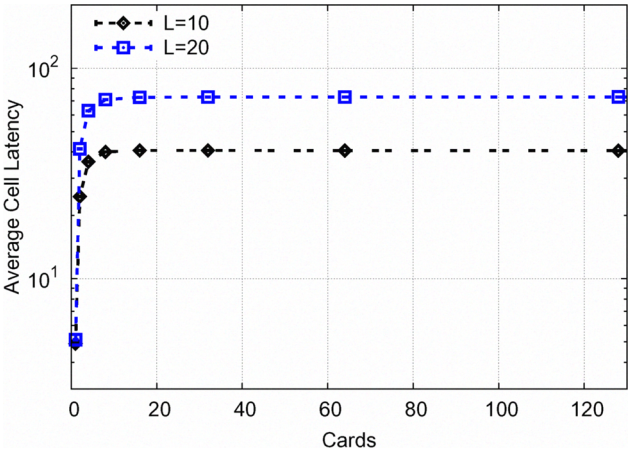

Fig. 9. Average cell latency versus the number of cards with load 0.85 and $B = 16$.

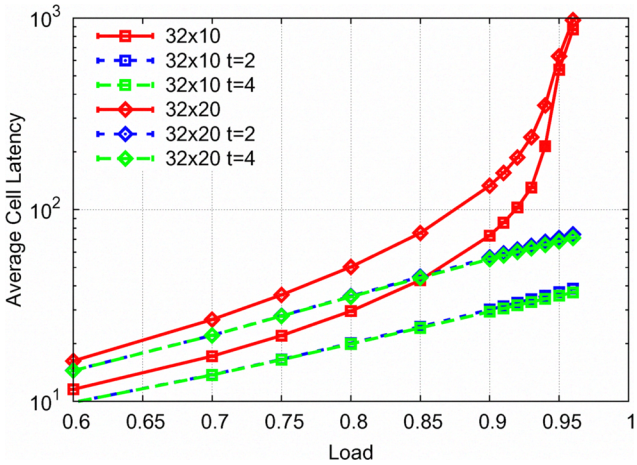

Fig. 8. Cell latency versus load with the introduction of transfer speedup t.

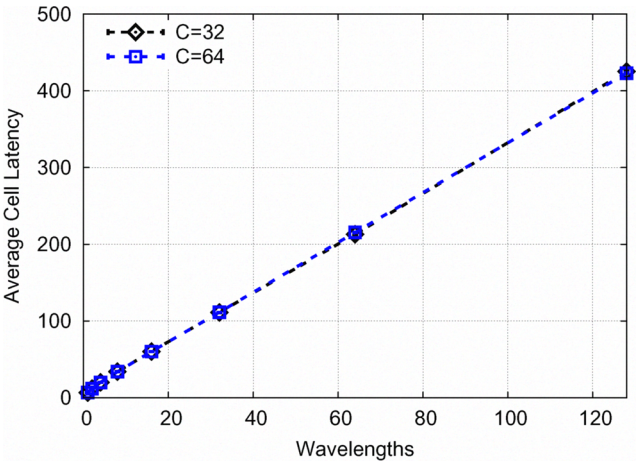

Fig. 10. Average cell latency versus the number of wavelengths per card with load 0.85 and $B = 16$.

approach experiences a steeper growth in latency and an early saturation, leading to higher delays for loads greater than 0.91. Furthermore the mean burst length B does not significantly affect performance: a slightly longer latency is obtained with $B = 32$ compared to $B = 16$, since the traffic is more bursty.

In Fig. 7 different switch configurations are compared. We can notice that in the two-step framework, latency is mainly affected by the number of wavelengths used, rather than the number of cards or the total number of input/output ports. Indeed the 32×20 configuration experiences higher delays than the 64×10 one (with the same total number of ports) and a comparable delay with the 64×20 configuration.

With the introduction of a small transfer speedup t, as explained in previous section, it is possible to dramatically improve the performance in term of average latency, at the expense of a small additional implementation cost. Fig. 8 shows the results for a configuration with 32 cards, with 10 or 20 wavelengths each with bursty cell arrivals with $B = 32$. No noticeable difference exists between a transfer speedup of 2 and 4. This behavior shows that just the minimum amount of transfer speedup, i.e. 2, is sufficient to improve performance.

In Fig. 9 it is shown how the average cell latency is affected by the number of cards for a fixed load of 0.85 and a mean burst length of 16. As expected, the latency does not depend very much on the number of cards, as pointed out also in [22]. This result is valid for loads lower than the saturation limit, i.e. typically $0.96 \div 0.97$, and for a number of iterations that makes the algorithm approximate a maximal match. The delay introduced by the first step is reflected in the gap between the two curves.

Fig. 10 shows the latency in function of the number of wavelengths per card, while keeping the offered load fixed at 0.85. The mean burst length is 16. The curves for $C = 32$ and $C = 64$ are superposed, as in Fig. 7. It is possible to notice a linear dependence between the number of wavelengths and latency. Indeed, under uniform traffic distribution and high load, a given input is assigned a given wavelength in average every L time slots, hence the proportionality.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7268
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7268

VI. Conclusions

In this work we showed that the proposed two-step scheduling framework for modular optical packet switches is computationally more efficient than the classical approach for a wide range of configurations. In particular, a worst case analysis has been carried out to determine the domain of cards and wavelengths for which the two-step scheduler is beneficial. Then, we investigated the latency performance of several switch configurations, showing how the latency can be improved with the application of the smallest transfer speedup. Finally we found out that the latency grows linearly as a function of the number of wavelengths, and is constant as a function of the number of cards.

References

[1] E. Desurvire, "Capacity demand and technology challenges for lightwave systems in the next two decades," *IEEE J. of Lightwave Technology*, vol. 24, no. 12, pp. 4697–4710, Dec. 2006.

[2] A. F. Benner, M. Ignatowski, J. A. Kash, D. M. Kuchta, and M. B. Ritter, "Exploitation of optical interconnects in future server architectures," *IBM J. Res. Dev.*, vol. 49, no. 4/5, pp. 755–775, 2005.

[3] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics," in *Proc. SIGCOMM '03*. ACM, 2003, pp. 189–200.

[4] H. J. Chao and B. Liu, *High Performance Switches and Routers*. Wiley-IEEE Press, 2007.

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

[6] P. G. Raponi, N. Andriolli, P. Castoldi, and A. Bianchi, "Multi-card wavelength scheduling in modular optical packet switches," in *OFC/NFOEC 2009*, Mar. 2009, paper OML1, pp. 1–3.

[7] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. on Communications*, vol. 47, no. 8, pp. 1260–1267, Aug. 1999.

[8] Y. Li, S. Panwar, and H. Chao, "On the performance of a dual round-robin switch," in *Proc. of IEEE INFOCOM 2001*, vol. 3, Apr. 2001, pp. 1688–1697.

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

[10] I. Elhanany and D. Sadot, "DISA: a robust scheduling algorithm for scalable crosspoint-based switch fabrics," *IEEE J. on Selected Areas in Commun.*, vol. 21, no. 4, pp. 535–545, May 2003.

[11] X. Li and I. Elhanany, "Stability of a frame-based maximal weight matching algorithm with transfer speedup," *IEEE Commun. Letters*, vol. 9, no. 10, pp. 942–944, Oct. 2005.

[12] ——, "Stability of a frame-based oldest-cell-first maximal weight matching algorithm," *IEEE Trans. on Communications*, vol. 56, no. 1, pp. 21–26, Jan. 2008.

[13] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. of IEEE INFOCOM 2000*, vol. 2, Mar. 2000, pp. 556–564.

[14] E. Leonardi, F. Neri, and B. Yener, "Algorithms for virtual output queued switching," in *Proc. of GLOBECOM '99*, vol. 2, 1999, pp. 1203–1210.

[15] M. A. Marsan, A. Bianco, E. Leonardi, and L. Milia, "RPA: a flexible scheduling algorithm for input buffered switches," *IEEE Trans. on Communications*, vol. 47, no. 12, pp. 1921–1933, Dec 1999.

[16] D. R. Musser, "Introspective sorting and selection algorithms," *Software: Practice and Experience*, vol. 27, no. 8, pp. 983–993, 1997.

[17] D. R. Musser, J. Moser, K. Ross, and W. Su, "Introsort." [Online]. Available: http://www.cs.rpi.edu/∼musser/gp/algorithm-concepts/introsort-screen.pdf

[18] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, and E. Filippi, "On the behavior of input queuing switch architectures," in *European Transactions on Telecommunications*, vol. 10, no. 2, 1999, pp. 111–124.

[19] M. A. Marsan, A. Bianco, E. Filippi, P. Giaccone, E. Leonardi, and F. Neri, "A comparison of input queuing cell switch architectures," in *IEEE 3rd Int. Workshop on Broadband Switching Systems (BSS'99)*, 1999.

[20] I. Elhanany, M. Kahane, and D. Sadot, "On uniformly distributed ON/OFF arrivals in virtual output queued switches with geometric service times," in *Proc. of ICC '03*, vol. 1, May 2003, pp. 173–177.

[21] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[22] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, UC Berkeley, May 1995.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7268
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7268