Frame-scheduling for Input-queued Switches with Energy Reconfiguration Costs

Frame-Scheduling for Input-Queued Switches with Energy Reconfiguration Costs

Andrea Bianco, Paolo Giaccone, Marco Ricca
Dipartimento di Elettronica, Politecnico di Torino, Italy

***Abstract*—We consider a slotted input-queued switch with a crossbar-like switching fabric. In each time-slot, a centralized scheduler determines a switching fabric configuration to transfer packets. We consider the energy consumption needed to configure the switching fabric and we assume that the energy depends on the number of modifications in the switching configuration in two consecutive time-slots. We address the problem of scheduling a set of packets to minimize the required energy while preserving high throughput. We reduce the overall problem to the combination of two different optimization problems. We propose a family of algorithms to solve the problem and we discuss their energy-throughput performance.**

I. Introduction

In recent years, power and energy consumption in high speed switches/routers become one of the most critical design problems, mainly due to thermal issues that require complex cooling systems and to increasing energy costs. In this paper, we focus on minimizing the energy consumption in slotted IQ (Input-Queued) switches. IQ are the reference architectures for high-performance packet switches thanks to their good scalability properties: indeed, memory access speed does not increase linearly as a function of link transmission speed and number of switch ports, as in OQ (Output-Queued) architectures. We focus on the issue of controlling packet transfer in IQ switches across the switching fabric to reduce the energy consumption while preserving high throughput.

The relationship between energy consumption, switch configuration and data transfer rate depends strongly on the technology deployed to implement the switching fabric. Traditional electronic switching fabrics are based on CMOS technology. Roughly speaking, in an electronic crossbar, one of the simplest and most widely deployed switching fabric architecture, the output line is connected to the input line through a logic gate. The activation of the logic gate logically corresponds to selecting the proper crosspoint in the crossbar fabric. The energy consumption is mainly a function of the electric charges moved to charge/discharge the input/output lines, for each bit transmission. Thus, the total energy depends on the number of transmitted bits, whereas the required power is a function of the bit rate. This observation shows the poor scalability of electronic switching devices, in terms of energy consumption, when line rates increase.

On the contrary, optical switching fabrics are considered a very promising technology to face energy scalability issues. This is mainly due to the fact that the optical device dynamics are decoupled from the bit rate: the energy consumption is largely independent of the number of transported bits.

In this paper we consider an energy model suitable to describe the energy consumption of generic rate-independent switching fabrics. We focus on the energy spent when the switching fabric changes configuration, which is independent of the data rate, and we assume that the energy deployed depends on the number of input-output connections that are added and/or removed inside the switching fabric. This assumption is suited for some switching technologies such as, e.g., those based on MEMS [1].

The main problem addressed by this work is how to reduce energy consumption by minimizing the number of connections that change inside the switching fabric, whilst preserving high throughput. Intuitively, the approach we investigate is based on changing the switching configuration in a "lazy" way, i.e., trying to keep the switching configuration as similar as possible in consecutive time-slots.

II. Problem Definition

We consider an $N \times N$ synchronous (slotted) IQ switch: fixed-size packets are received and stored at inputs, and input and output ports are assumed to be slot synchronized. Input queues are organized according to the classical Virtual Output Queue (VOQ) architecture, i.e. one separate FIFO queue at each input port for each output port, for a total of N^2 queues in the switch. VOQ_{ij} stores, at input port i, packets that must be routed to output port j.

At each time-slot, a scheduler chooses a switching configuration, i.e. an input/output port interconnection pattern, satisfying the following constraints: at most one packet can be transferred from each input and at most one packet can be transferred to each output in a time-slot. This problem can be modeled as a matching problem in a bipartite graph, in which each left hand side vertex corresponds to an input and each right hand side vertex corresponds to an output. An edge connects input i to output j if the corresponding VOQ_{ij} is not empty. The scheduler computes in each time-slot a matching, i.e. a subset of edges with no vertex in common, corresponding to a feasible switching configuration. A matching can be represented by an $N \times N$ binary matrix $M = [m_{ij}]$, denoted as *matching matrix* in this paper, in which $m_{ij} = 1$ iff input i is connected to output j. In a matching matrix, there is at most one element set to 1 in each row and in each column:

$$\sum_{i=1}^{N} m_{ij} \leq 1 \qquad \sum_{j=1}^{N} m_{ij} \leq 1 \tag{1}$$

978-1-4244-4148-8/09/$25.00 ©2009

The set of all matching matrices satisfying (1) is denoted by $\mathcal{M}$. If matching M is complete, there is exactly one element set to 1 in each row and in each column, and M is said to be a permutation matrix.

A. Energy Model

We try to minimize the energy consumption required to modify the switching configuration in consecutive time-slots. More precisely, if input i was connected to output j in the previous time-slot, and, in the current time-slot, input i becomes connected to output $k, k \neq j$, we consider two energy costs, measured in Joules [J]: e_d is the energy required to delete the connection from input i to output j and e_a is the energy required to add the new connection from input i to output k. The total energy cost of the current switching configuration is obtained as the sum of the energy costs required to delete all connections selected in the previous time-slot and not selected in the current time-slot, plus the energy cost required to add all connections selected in the current time-slot and not selected in the previous one. The actual values of e_d and e_a depend on the considered switching technology; in the case of bi-stable latching MEMS, with forces acting on the micro-mirrors only when mirrors change position, we can reasonably assume $e_d = e_a$, i.e. the same energy cost to remove or to add a connection. We define this quantity as $\hat{e}$: $\hat{e} = e_d = e_a$. In this case, the minimum value of energy required to change the output at which an input is currently connected is $2\hat{e}$. Note that the approach presented in this paper can be easily extended to the general case of $e_d \neq e_a$ or to other values of energy consumptions. In the following, we normalize the energy costs to $\hat{e}$ or, equivalently, set $\hat{e} = 1$.

Define $E\left(M^h, M^k\right)$ as the total amount of energy spent to modify matching $M^h = [m^h_{ij}]$ in matching $M^k = [m^k_{ij}]$. The total amount of energy can be computed by counting the number of edges that are i) removed from M^h and ii) added to M^h, to obtain M^k.

$$E\left(M^h, M^k\right) = \hat{e} \sum_{i=1}^{N} \sum_{j=1}^{N} |m^h_{ij} - m^k_{ij}|$$

By construction, $E\left(M^h, M^k\right) = E\left(M^k, M^h\right)$.

A scheduling frame $\mathcal{F}$ is defined as an ordered sequence of K matchings:

$$\mathcal{F} = (M^k, \phi_k)_{k=1}^{K}, \qquad \text{with } M^k \in \mathcal{M},\ \phi_k \in \mathbb{Z}^+$$

where ϕ_k is the number of consecutive time-slots in which matching M^k is used to configure the switching fabric. We can evaluate the total *energy* cost to configure the switching fabric according to a frame $\mathcal{F}$ as:

$$E(\mathcal{F}) = \sum_{k=1}^{K-1} E(M^k, M^{k+1})$$

Note that $E(\mathcal{F})$ is independent of the values of ϕ_k. The required *power*, measured in [J/slot], can be easily derived as

$$P(\mathcal{F}) = \frac{E(\mathcal{F})}{T}$$

where T is the frame duration in time-slots.

B. Frame Scheduling

We assume that the scheduler operates on a frame basis and not on a time-slot basis. The scheduler samples the state of input queues every T time-slots, T being the frame duration. Let $R = [r_{ij}]$ be an $N \times N$ request matrix, where r_{ij} is the number of fixed-size packets enqueued at input port i and destined to output port j. The maximum row and column sum of R is denoted by F_I:

$$F_I = \max \left\{ \max_{j=1\ldots N} \sum_{i=1}^{N} r_{ij}, \quad \max_{i=1\ldots N} \sum_{j=1}^{N} r_{ij} \right\}$$

The *frame load* is defined as $\rho = F_I/T$ and R is said to be *admissible* when $\rho \leq 1$.

The scheduler computes a frame $\mathcal{F} = (M^k, \phi_k)$ to serve all packets in R, i.e. satisfying

$$\sum_{k=1}^{K} \phi_k M^k \geq R \tag{2}$$

When the frame has been defined, the switching fabric is configured according to the matching sequence in the frame. Note that the three phases of i) samplings the state of the queues, ii) computing the frame and iii) serving the packets can be easily pipelined in subsequent scheduling periods.

Let $F_R = \sum_{k=1}^{K} \phi_k$ be the frame duration, i.e. the total number of slots in $\mathcal{F}$. An admissible request matrix R is said to be *sustainable* according to a scheduling algorithm if in a frame scheduling duration *all* the packets in the request matrix are transferred, i.e. if $F_R \leq T$. Thanks to the Birkhoff-von Neumann theorem [2], a frame satisfying condition (2) has a (minimum) frame duration equal to F_I. In general, $F_R \geq F_I$ and we set $T = F_I$ to evaluate the maximum sustainable load. Let us define the *frame expansion factor* S as the ratio between the actual frame duration and the minimum frame duration: $S = F_R/F_I$. By construction, $S \geq 1$; when $S > 1$, the load of R should be reduced by a factor S to be sustainable in T time-slots. Hence, $1/S$ represents the normalized *maximum sustainable load*.

III. Energy-Aware Frame Scheduling

Our energy-aware frame-scheduling problem can be modeled as a two-objective optimization problem: trying to minimize the energy consumption (due to switching fabric reconfigurations) whilst maximizing the throughput (or, equivalently, minimizing the frame duration F_R). We heuristically solve this two-objective optimization problem of frame definition following a two-step procedure:

- **matching selection**: given the request matrix R, compute an *unordered* frame $\mathcal{F}_u = \{M^k, \phi_k\}_{k=1}^{K}$ such that i) condition (2) is satisfied and ii) F_R is minimized. The objective is to serve all the packets in R while trying to maximize throughput.
- **matching sort**: the final frame $\mathcal{F}$ is computed by ordering $\mathcal{F}_u$ to minimize energy consumption due to switching reconfigurations.

978-1-4244-4148-8/09/$25.00 ©2009

A. Matching Selection

We consider four different algorithms to define the set of matchings composing a frame. The first three are iterative algorithms, exploiting the same generic decomposition algorithm *Gen-DEC*, whose pseudo-code is reported below. At each iteration of *Gen-DEC*, a specific algorithm $\Omega(R)$ computes a matching matrix M on R. Then, the value of the minimum element in R among those selected by the matching matrix M is subtracted from all selected elements in R, and a residual request matrix is obtained. The process iterates until R becomes empty. Since, at each iteration, at least one element (at most N elements) of R becomes zero, N^2 iterations are needed in the worst case to fully schedule R.

Gen-DEC (*Input*: R; *Output*: $\mathcal{F}_u$)
$\mathcal{F}_u = \emptyset, R^1 = R, k = 1$ // initialize
while $R^k \neq 0$ // while R is not completely set to zero
  $M^k = \Omega(R^k)$ // find a matching
  $\phi_k = \min_{1 \leq i,j \leq N}\{m_{ij}^k r_{ij}^k\}$ // value to subtract from R^k
  $R^{k+1} = R^k - \phi_k M^k$ // subtract
  $\mathcal{F}_u = \mathcal{F}_u \cup (M^k, \phi_k)$ // frame update
  $k = k + 1$ // start a new iteration

We considered in this paper four frame decomposition algorithms:

- **BvN**: a *Gen-DEC* based algorithm, equal to the Birkhoff-von Neumann decomposition [2] on R, satisfying condition (2). $\Omega(R)$ is a MSM (Maximum Size Matching) on R, i.e. finds the matching with the highest number of edges corresponding to non-null elements of R. The MSM algorithm complexity is $O(N^{2.5})$. When the sum of all the rows and all the columns of R is constant, the BvN decomposition is optimal, in the sense that it computes the minimum frame duration. Thus, $F_R = F_I$ and $S = 1$. The overall computational complexity is $O(N^{4.5})$.
- **GMax**: a *Gen-DEC* based algorithm. $\Omega(R)$ is a greedy maximum weight matching on R. The algorithm selects the element in R with the maximum value, then it remove the corresponding row and column from R, and repeats the process until all the rows and columns in R are considered. The complexity of each iteration is $O(N^2 \log N)$ (mainly due to sorting the N^2 values in R in the initial step); hence, the overall computational complexity is $O(N^4 \log N)$.
- **GMin**: a *Gen-DEC* based algorithm. $\Omega(R)$ is a greedy minimum weight matching on R. Similarly to GMax, the algorithm chooses the smallest elements in R, then it removes the corresponding row and column from R, and repeats the process until all the rows and columns in R are considered. Similarly to GMax, the complexity is $O(N^4 \log N)$.
- **Diag**: it is the simplest decomposition algorithm we consider, mainly aimed at defining a frame whose matchings lead to a minimum reconfiguration energy. The matching selection is based on a precomputed set of *covering diagonals* $D^k = [d_{ij}^k]$ on R, i.e. N matchings with no elements in common and able to cover all the elements in R. More formally, $d_{ij}^k d_{ij}^h = 0$ for any $h \neq k$, and $\sum_{k=1}^N d_{ij}^k = 1$ for any i, j. The matching duration ϕ_k is chosen equal to the maximum value of the elements in the request matrix selected by D^k. Formally, $\phi_k = \max_{i,j}\{d_{ij}^k r_{ij}\}$. The frame duration is: $F_R = \sum_{k=1}^N \max_{i,j}\{d_{ij}^k r_{ij}\}$. The total number of iterations is N, each of the iterations having a complexity $O(N)$ (the maximum value among N elements of R must be found). Hence, the overall computational complexity is $O(N^2)$.

B. Matching Sort

In this second step of the frame definition algorithm, the matchings determined by the matching selection algorithms in the frame $\mathcal{F}_u$ are ordered to minimize energy consumption due to reconfigurations in consecutive time-slots. One simple way to model this problem is to consider an auxiliary graph. Each matching in $\mathcal{F}_u$ is associated with a vertex, and any pair of vertexes is connected by an edge, thus creating a complete graph by construction. The cost of the edge connecting vertex M^k with M^h is defined as the energy spent to pass from one matching to the other, i.e. $E(M^k, M^h)$. The cost of any path in the graph corresponds to the energy spent following a particular sequence of matchings. The frame sequence $\mathcal{F}$ minimizing the energy consumption can be computed on $\mathcal{F}_u$ by finding the minimum-cost Hamiltonian cycle, also known as the TSP problem, which is NP-complete. However, in our scenario, the edge costs satisfy the triangle inequality, and the problem reduces to a metric TSP [3], which is still NP-complete, but it can be simply approximated.

We consider the following algorithms to sort the matchings in $\mathcal{F}_u$:

- **No-Sort** (NS) leaves the sequence of matching unmodified.
- **Best-Sort** (BS) is a greedy algorithm that finds an approximated minimum cost cycle by visiting all the vertexes: it chooses, at each step, the minimum cost edge towards an unvisited vertex. The initial vertex is chosen at random.
- **Worst-Sort** (WS) is a greedy algorithm that heuristically finds the maximum cost Hamiltonian cycle: it chooses, at each step, the maximum cost edge towards an unvisited vertex, starting from a random vertex. This algorithm permits to define a worse sequence from the energy consumption point of view, and it is useful to highlight the impact of the frame sorting phase.

The above listed algorithms can be freely combined with the matching selection algorithms discussed in the previous section. In the remainder of the paper, we use the notation (matching-selection)-(matching-sort) to denote the particular pair of frame definition algorithms considered in our investigations.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

IV. Performance Results

A. Simulation Scenarios

We compare the performance of the previously presented algorithms for several families of randomly generated request matrices. The first family is denoted as *Average Sum* (AS): the matrix elements r_{ij} are i.i.d. random variables, and satisfy the constraints:

$$E\left[\sum_{i=1}^{N} r_{ij}\right] = E\left[\sum_{j=1}^{N} r_{ij}\right] = \mu N$$

i.e., the sum of each row and column is, on average, equal to a constant μN. Hence, μ represents the average number of packets arrived to each input during T time-slots. Let $\textsc{Geom}(x)$ be a geometric distribution with average x. Among the family of AS request matrices, we consider:

- *Uniform* (Uni-AS): $r_{ij} = \textsc{Geom}(\mu)$. The coefficient of variation of the elements in R $C_v = 1$, i.e. the variance grows with μ.
- *Bimodal* (Bim-AS):

$$r_{ij} = \begin{cases} 0 & \text{with probability } p \\ \textsc{Geom}(\mu) & \text{otherwise} \end{cases} \quad (3)$$

  Since the coefficient of variation is

$$C_v = \sqrt{\frac{(1+p)\mu - 1 + p}{\mu(1-p)}}$$

  we can tune the values of p and μ to obtain a given C_v. For example, setting $p = 0.601$ and $\mu = 100$ gives $C_v \approx 2$.

We also consider the family of Perfect Sum (PS) matrices, whose rows and columns sum exactly to a constant μN:

$$\sum_{i=1}^{N} r_{ij} = \sum_{j=1}^{N} r_{ij} = \mu N$$

Now the elements $\{r_{ij}\}$ are not i.i.d.. PS matrices are an extension to the integer domain of double stochastic matrices, for which the BvN [2] decomposition was originally defined. Among the matrices belonging to the PS family, we consider:

- *Uniform* (Uni-PS): choose a set of μN random permutation matrices $D^k \in \mathcal{M}$ and compute $R = \sum_{k=1}^{\mu N} D^k$. Uni-PS matrices are characterized by elements with low variance, because, for the Central Limit Theorem, $C_v \to 0$, as $N \to \infty$.

B. Performance of the Diag algorithm

The throughput performance of the Diag algorithm can be evaluated analytically for Uni-AS random matrices.

Theorem 1: Let $R = [r_{ij}]$ be a Uni-AS request matrix, with r_{ij} i.i.d. random variables with average μ. To schedule R, the Diag algorithm needs a frame expansion factor S following the law:

$$E[S] = \log N + \gamma$$

where γ is the Euler constant ($\gamma \approx 0.58$).

The proof is omitted for the sake of space.

TABLE I
Energy per packet for Uni-AS request matrices, with $N = 64$

Decomposition alg.	Worst Sort (WS)	No Sort (NS)	Best Sort (BS)
BvN	0.872	0.810	0.741
Diag	0.020	0.020	0.020
GMax	0.425	0.415	0.377
GMin	0.723	0.045	0.049

C. Effect of Matching Sort

We first evaluate the effect of the algorithms sorting the matchings. Table I reports the energy per packet obtained by combining a specific matching selection algorithm with a particular sorting algorithm, for Uni-AS request matrices with $\mu = 100$, in a 64×64 IQ switch. Very similar results were obtained for different switch size and different random request matrices. The Diag algorithm is not affected by the sorting. Indeed, all the matchings are distinct and $2N^2\hat{e}$ is the total energy spent in a frame. Since the total number of packets is, on average $N^2\mu$, the average energy per packet is simply $2N^2\hat{e}/(N^2\mu) = 2\hat{e}/\mu$, independently from R, as shown in the table. This value is the minimum energy achievable by any algorithm under Uni-AS scenario, but it requires a large frame expansion factor S, as shown later.

For the BvN matching selection algorithm, the energy cost decreases significantly when the best sorting (BS) algorithm is adopted, with respect to the scenarios when the no-sorting (NS) or the worst sorting (WS) algorithms are chosen. The large benefit of BS is due to the specific algorithm adopted in BvN, which is based on computing a maximum size matching at each iteration, without considering the energy cost required to move from one matching to the next one. As a consequence, in the remainder of the paper, we consider always the BvN algorithm associated with the BS sorting algorithm, and we denote such version as BvN-BS.

The sorting procedure BS improves also the GMax algorithm performance. In this case, the algorithm without sorting (NS) provides lower energy than the worst case sorting (WS). Indeed, at each iteration the GMax algorithm tries to maximize the total weight of the matching; this induces an implicit sorting which automatically aggregates similar matchings (especially during the first iterations of the matching selection). In the following, we consider only the combination GMax-BS.

In GMin, the effect of sorting is quite negligible. Indeed, the performance of the no-sorting (NS) algorithm are really satisfactory, and much better than those obtained when adopting the worst case (WS) sorting. Although not completely intuitive at a first glance, this effect is due to the particular metric used to compute the matching at each iteration. By subtracting the minimum weight matching M^k from R^k at iteration k, there is a high probability that the new minimum weight matching M^{k+1} shares some (at most, $N-1$) edges with M^k. This correlation induces an efficient self-sorting property, providing an energy efficiency comparable with, and in some situations even better than, the one achieved by the heuristic BS. Thus, GMin can be considered an efficient

978-1-4244-4148-8/09/$25.00 ©2009

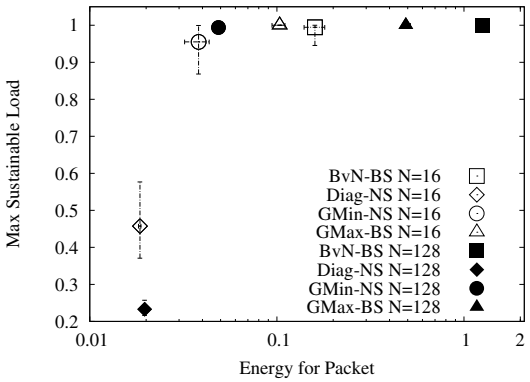

Fig. 1. Throughput and energy tradeoff for Uni-AS traffic.

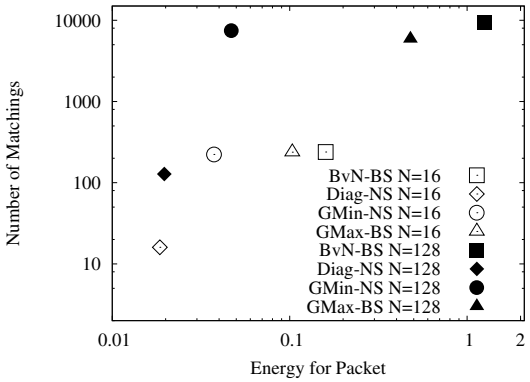

Fig. 2. Tradeoff between the average number of matchings and energy consumption for Uni-AS traffic.

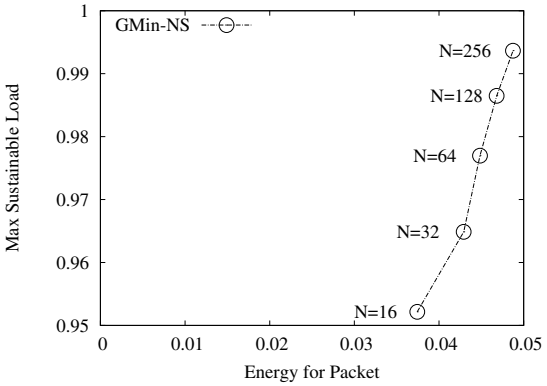

Fig. 3. Throughput and energy tradeoff for the GMin-NS algorithm for Uni-AS traffic.

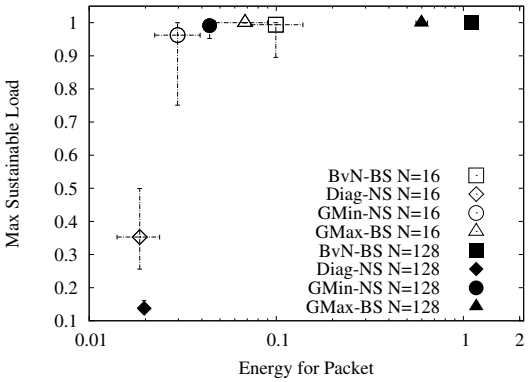

Fig. 4. Throughput and energy tradeoff for Bim-AS scenario.

frame definition algorithm even without any additional sorting algorithm. On the contrary, when running the GMax algorithm, M^k is a (almost) maximum weight matching; as such, there is a very low probability that M^{k+1} shares edges with M^k.

Since the above described observations hold in many scenarios, we consider the following combinations of frame scheduling algorithms in the next sections: BvN-BS and GMax-BS (with sorting procedure), and GMin-NS and Diag-NS (without sorting procedure). These algorithms have very different computational complexities and memory requirements; the sorting procedure itself requires to store the whole frame sequence to sort it. The ranking among algorithms in terms of increasing complexity is: Diag-NS, GMin-NS, GMax-BS and BvN-BS.

D. Energy and Throughput Tradeoff

Simulations are run in a proprietary simulation environment written in C language. The parameter μ, related to the packet arrival rate as previously defined, is set to 100; all simulations results are obtained as an average of 100 simulation runs, each run using a different randomly generated request matrix, to obtain statistically significant simulation results. We mainly report the results for $N = 16$ and $N = 128$; however, similar results hold also for $N = 32$ and $N = 64$, and are not reported here due to lack of space.

In all the reported plots, each point corresponds to the average value (over 100 runs) obtained when running the decomposition algorithms for a combination of a specific algorithm and a given switch size N. Two bars around each point (one horizontal bar and one vertical bar) show the maximum and minimum values obtained considering all 100 runs. When the error-bars are not visible, the results of each run are almost identical to the average value, i.e., a small

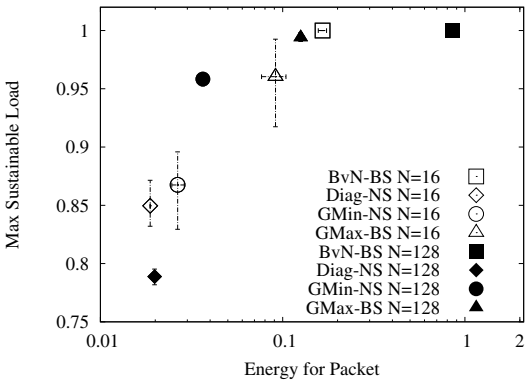

Fig. 5. Throughput and energy tradeoff for Uni-PS scenario.

978-1-4244-4148-8/09/$25.00 ©2009

variance exists when changing the seed to generate the random matrices.

Fig. 1 shows the tradeoff between the maximum sustainable load and the energy per packet obtained by the different algorithms. The plot can be read in the following way: Let us suppose that the switch designer is willing to obtain a minimum sustainable load and a maximum energy consumption per packet. These design constraints define a point (e', ρ') in the graph. All the algorithms whose performance are in the region to the left and above this point (i.e., with energy $\leq e'$ and max sustainable load $\geq \rho'$) satisfy the design constraint.

Only for a small switch size $N = 16$, the algorithms Diag-NS and GMin-NS show some variations in the maximum sustainable load; in all other cases, simulations results show a very small variability when changing the traffic matrix. As expected, the Diag-NS algorithm achieves the minimum energy per packet. Recall the average energy per packet is simply $2\hat{e}/\mu$, corresponding to the values observed in the plots. However, due to the large frame expansion factor required, Diag-NS cannot sustain large loads, as stated in Theorem 1.

The GMin-NS algorithm, despite its relative simplicity, achieves energy consumption levels only 2-3 times larger than Diag-NS, but with a throughput very close to the maximum throughput. BvN-BS and GMax-BS provide almost the same throughput, but at the expenses of large energy consumption level, even if using the matching sort procedure. Furthermore, energy consumption per packet increases a lot for increasing switch size. Note that, in general, GMax-BS is more energy-efficient than BvN-BS, due to the metrics used to compute the matching, as already observed in Sec. IV-C.

Fig. 2 shows the number of distinct matchings computed by each algorithm, for the Uni-AS scenario. The algorithm Diag-NS, by construction, uses only N matchings. All other algorithms use a significantly larger number of matchings and require an energy consumption level larger than one order of magnitude. For $N = 128$, GMin-NS uses the largest number of matchings. Note that roughly $N^2\mu$ packets should be scheduled in a frame, and, to achieve the maximum frame load ($\rho = 1$), each single matching should serve roughly N packets. Hence, there are at most $N\mu = 12800$ different matchings in the frame under Uni-AS. Even if the algorithm GMin-NS uses a large number of matchings (roughly 7500), the total energy cost is small, because the energy cost between any pair of matchings is very small.

In Fig. 3 we focus on the energy-throughput tradeoff obtained by GMin-NS by varying N. It is interesting to observe that, regardless of the switch size, the maximum sustainable load is always significant, and the increase in the energy per packet as a function of N is marginal. Similar observations hold for the Bim-AS scenario, as reported in Fig. 4.

In the case of Uni-PS scenario, Fig. 5 shows that BvN-BS achieves the worst energy performance, even if the maximum sustainable load is always achieved. The best energy result is obtained by Diag-NS, which is much more efficient in terms of throughput under this scenario with respect to Uni-AS. This is mainly due to the variance of the values in the diagonal elements of the request matrix R, because in the Uni-PS scenario this variance is much smaller than in the Uni-AS case: hence, the maximum element on a diagonal is close to the average value and the Diag-NS algorithm performance improves. BvN-BS appears to be very inefficient in terms of energy consumption, when compared with any other algorithm, especially when the switch size grows. GMin-NS is still very efficient in terms of energy, close to Diag-NS, but it can suffer some throughput degradation.

V. Conclusions

We addressed the computation of a minimum-energy frame to schedule a set of packets in a request matrix. We proposed a family of algorithms by decomposing the computation into two different phases: matching selection and matching sort. Throughout a simulation study, we were able to investigate the throughput-energy tradeoff achieved by the different algorithms. We have observed that the energy consumption per packet may vary by almost two order of magnitude depending on the chosen algorithm.

GMin-NS appears to provide the best compromise between sustainable load (always very close to the maximum) and energy consumption (very close to the minimum possible value). Furthermore, the GMin-NS complexity is much lower that the complexities of GMax-BS and BvN-BS, because GMin-NS does not require any ordering among the matchings, being the sorting procedure implicitly included in the matching selection process. BvN-BS, even if it is a well known optimal algorithm for energy-oblivious frame decomposition, is the least efficient among the studied algorithm in terms of energy consumption, further requiring a high computational complexity.

VI. Acknowledgments

This work was partially supported by by the Network of Excellence BONE ("Building the Future Optical Network in Europe"), funded by the European Commission through the 7^{th} Framework Programme.

References

[1] M. C. Wu, O. Solgaard, and J. E. Ford, "Optical mems for lightwave communication," *Lightwave Technology, Journal of*, vol. 24, no. 12, pp. 4433–4454, Dec. 2006.

[2] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von neumann input buffered crossbar switches," *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1614–1623, Mar 2000.

[3] V. V. Vazirani, *Approximation Algorithms*. Springer, March 2004.

[4] B. Towles and W. Dally, "Guaranteed scheduling for switches with configuration overhead," *Networking, IEEE/ACM Transactions on*, vol. 11, no. 5, pp. 835–847, Oct. 2003.

[5] L. Mastroleon, D. O'Neill, B. Yolken, and N. Bambos, "Power aware management of packet switches," *High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on*, pp. 47–53, Aug. 2007.

[6] B. Yolken and N. Bambos, "Power management of packet switches via differentiated delay targets," *Communications, 2008. ICC '08. IEEE International Conference on*, pp. 354–359, May 2008.

[7] B. Yolken, D. Tsamis, and N. Bambos, "Target-based power control for queueing systems with applications to packet switches," *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1–6, Dec. 2008.

978-1-4244-4148-8/09/$25.00 ©2009