A Belief-Propagation Approach for Multicast Scheduling in Input-Queued Switches

A Belief-Propagation Approach for Multicast Scheduling in Input-Queued Switches

Paolo Giaccone
Dipartimento di Elettronica
Politecnico di Torino

Marco Pretti
CNR - Consiglio Nazionale delle Ricerche
Istituto dei Sistemi Complessi - Politecnico di Torino

***Abstract*—Scheduling multicast traffic in input-queued switches requires solving a hard combinatorial optimization problem in a very short time. This task advocates the design of algorithms that are simple to implement and efficient in terms of performance. We propose a new scheduling algorithm, based on message passing and inspired by the belief propagation paradigm, meant to approximate the provably optimal scheduling policy. Our main finding is that our algorithm outperforms other centralized greedy scheduling policies, achieving a better tradeoff between complexity and performance.**

I. Introduction

In the last decade, input-queued (IQ) switches have been the reference switching architecture for the design of high speed routers in the Internet [1]. Furthermore, at a much smaller spatial scale, they are widely employed to switch data flits in Network-on-Chips [2]. The main reason is that IQ switches offer a convenient tradeoff between computational complexity and memory speed. Indeed, input queues run at a speed equal to the line rate, so that the performance bottleneck due to the limited memory access time is minimized. In IQ switches, a scheduling algorithm must choose the packets to be transferred from input to output ports while satisfying the switching fabric constraints, which allow at most a single transfer from each input port and to each output port. Note that the same packet can be transferred to any subset of output ports, given that the switching contraints are satisfied. Solving the scheduling problem for unicast traffic requires to compute a maximum weight matching in a bipartite graph, and this model represents the prototype for a large class of resource allocation problems in telecommunication networks. In the same way, we expect that the relevance of the multicast scheduling problem, addressed in this paper, goes beyond the classical framework of IQ switches.

Unicast traffic has been the predominant traffic in the Internet for a long time, but, nowadays, new applications have been arising, based on multicast traffic, in which packets are sent to a set of destinations, rather than a single one. Examples of such applications are IP video broadcasting, P2P networks and financial networks supporting high-speed trading. So far, the support of multicast traffic in IQ switches has been very naïve and inefficient, since based on simple "patches" to unicast scheduling algorithms.

In this work we specifically address the problem of scheduling multicast packets in an IQ switch. We propose a new scheduling algorithm, inspired by the belief propagation (BP) paradigm, and designed to approximate a provably throughput-optimal scheduling policy.

The paper is organized as follows. In Sec. II we define the multicast scheduling problem in IQ switches and the related optimal policy [3]. In Sec. III we introduce the BP approach, and describe the new algorithm. In Sec. IV we compare the performance of the algorithm with other centralized greedy algorithms for multicast scheduling. Conclusions are drawn in Sec. V.

II. Multicast traffic in input queued switches

We consider an IQ switch of size $N \times M$ (Fig. 1), where $N = |I|$ and $M = |O|$, with I and O denoting the sets of input and output ports, respectively. A *fanout set* is defined as a subset of output ports, so that a multicast packet can be described by the fanout set of its output destinations. The adopted queueing architecture is MC-VOQ [3], i.e., one *logical* queue is present for each possible fanout set and each input port. This architecture is optimal, because it avoids the head-of-line blocking problem. Combined with optimal queueing, we consider a throughput-optimal scheduling policy for multicast traffic. Such a policy is called "fanout-splitting" [3], since it allows for partial packet transmissions: A packet can be sent to just a subset of its destination ports, leaving some residual destinations for future transmissions. In this case, the packet is re-enqueued to the queue corresponding to the residual fanout set. Such a behavior may introduce out-of-sequence packet transmissions, whose impact can be controlled and mitigated by the techniques discussed in [4].

Let S denote the set of all possible fanout sets, whose cardinality is $|S| = 2^M$, where we have artificially added the null $\emptyset$ fanout set just for notational reasons. As one logical queue is present for each nonempty fanout set, the set of

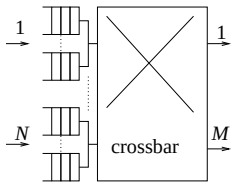

Fig. 1. IQ switch with MC-VOQ supporting multicast traffic.

all possible logical queues can be represented by S. We now define a set of $N \times (2^M - 1)$ matrices:

- Let $Y(t) = [y_{iq}(t)]$, with $i \in I$ and $q \in S$, be the *queue length matrix* at timeslot t; we assume that, by definition, the queue corresponding to the null fanout is always empty: $y_{i\emptyset}(t) = 0$.
- Let $A(t) = [a_{iq}(t)]$ be the *arrival matrix* at timeslot t: $a_{iq}(t) = 1$ if a new arrival occurs at input i for queue q, and $a_{iq}(t) = 0$ otherwise.
- Let $D(t) = [d_{iq}(t)]$ be the *service matrix* at timeslot t: $d_{iq}(t) = 1$ if queue q is served at input i, $d_{iq}(t) = -1$ if a packet is re-enqueued to q (at input i), and $d_{iq}(t) = 0$ otherwise.

The queue length evolution can be described by the usual relation:

$$Y(t+1) = Y(t) + A(t) - D(t) \tag{1}$$

with $t \in \mathbb{N}$.

A traffic scenario, described by a stochastic matrix process $A(t)$, is said to be *admissible* if $\mathbb{E}\{A(t)\}$ does not overload any input nor any output port [3]. As shown in [3], an output-queued switch obtains 100% throughput under any admissible multicast traffic, whereas an IQ switch does not.

The optimal scheduling decision $\hat{D}(t)$ for the service matrix at timeslot t, capable of maximizing the achievable throughput under any admissible multicast traffic pattern[1] was proposed in [3] and can be formalized as a constrained maximization problem for the linear functional

$$w(D, Y(t)) = \sum_{i \in I} \sum_{q \in S} d_{iq} y_{iq}(t) \tag{2}$$

In the following section, we will introduce a new notation more suitable to formalize this problem in terms of belief propagation. Furthermore, we will discuss the meaning of (2) when commenting (5).

The *feasibility constraints* for $D(t)$, imposed by the switching hardware, have also been described formally in [3] and will be also discussed in the next section. In conclusion, the resulting combinatorial optimization problem is NP-hard, so that only approximate algorithms are viable to solve this problem.

III. Belief propagation approach

A preliminary observation is that a service matrix D can be equivalently represented by N pairs of fanout sets $\sigma_i, \tau_i \in S$, one for each input $i \in I$:

$$D \quad \Leftrightarrow \quad [\sigma_i, \tau_i]_{i \in I}$$

Note that such N pairs depends on D, but we prefer to neglect it in the notation for the sake of simplicity; furthermore, since the algorithm has to run at each timeslot, we neglect also the time index t.

As previously mentioned, a nonempty fanout set identifies a queue. In particular, we shall assume that σ_i represents the served queue, τ_i the subset of outputs to which the packet is actually transmitted (*transmission* fanout set), and $\sigma_i \setminus \tau_i$ the queue in which the packet is in case re-enqueued (*residual* fanout set). Since the residual fanout set is, by construction, a subset of the original transmission fanout set, it holds:

$$\sigma_i \supseteq \tau_i \qquad \forall i \in I \tag{3}$$

Moreover, in order to complete the feasibility constraints, we must avoid conflicting packets at each output, namely, we have to impose that no more than one packet is transferred to each output port and hence:

$$\sum_{i \in I} \chi\{\tau_i \ni j\} \leq 1 \qquad \forall j \in O \tag{4}$$

where $\chi\{\cdot\}$ denotes a characteristic function, equal to 1 if the condition denoted by the argument is verified (i.e., input i transmits to output j), and 0 otherwise.

To clarify the notation, let us report here a toy example. Consider a 2×3 switch with a packet at input 1 destined to outputs $1, 2$, and a packet at input 2 destined to outputs $1, 2, 3$. Assume that the scheduler, adopting a fanout-splitting discipline, sends completely the first packet to its destinations, whereas it sends just one copy of the second packet to output 3. Thus, the second packet is re-enqueued to the queue corresponding to the residual fanout set $\{1, 2\}$. This scheduling decision can be described by $\sigma_1 = \tau_1 = \{1, 2\}$, whereas $\sigma_2 = \{1, 2, 3\}$ and $\tau_2 = \{3\}$.

It is possible to rewrite function (2) to minimize as a function of the fanout set variables σ_i, τ_i in the following form

$$w\left([\sigma_i, \tau_i]_{i \in I}, Y\right) = \sum_{i \in I} \left(y_{i\sigma_i} - y_{i(\sigma_i \setminus \tau_i)}\right) \tag{5}$$

Equation (5) clearly shows that the "weight" function w represents the difference between the length of all the served queues ($y_{i\sigma_i}$) and the length of all the queues where the residual fanout sets are re-enqueued ($y_{i(\sigma_i \setminus \tau_i)}$). This is an extension of the well known max-pressure policy [5], in which the weight of serving one queue is computed as the local queue length minus the (downstream) queue length where the packet is sent. In our case, the downstream queue corresponds to the queue where the residual fanout set is re-enqueued.

Now, it is important to observe that the "matching" constraints (4) involve variables τ_i associated to different inputs, whereas, for a given input i, the variable σ_i is only coupled to the corresponding τ_i, by the condition (3). As a consequence, the optimal σ_i for a given choice $\tau_i = \tau$, which we shall denote as $\hat{\sigma}_{i\tau}$, can be determined by a local optimization procedure at each input i, namely

$$\hat{\sigma}_{i\tau} = \arg \max_{\sigma \in S \,|\, \sigma \supseteq \tau} \left\{y_{i\sigma} - y_{i(\sigma \setminus \tau)}\right\} \tag{6}$$

Defining also the optimized "local" weights

$$w_{i\tau} \triangleq \max_{\sigma \in S \,|\, \sigma \supseteq \tau} \left\{y_{i\sigma} - y_{i(\sigma \setminus \tau)}\right\} \tag{7}$$

the original problem is reduced to a (constrained) optimization over the sole τ_i variables (transmission fanout sets). The

[1] More precisely, the traffic is generated according to a generalization of the standard Bernoulli i.i.d. model defined for unicast.

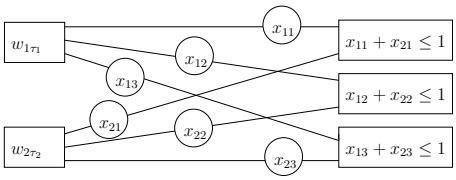

Fig. 2. Factor graph. Circles and rectangles denote variable and function nodes, respectively. In the toy example of the text, we would have $x_{11} = x_{12} = x_{23} = 1$ and $x_{13} = x_{21} = x_{22} = 0$.

optimal values can be written as

$$[\hat{\tau}_i]_{i\in I} = \arg \check{\max}_{[\tau_i]_{i\in I}} \sum_{i\in I} w_{i\tau_i} \tag{8}$$

where the check mark recalls that the optimization is constrained by (4).

We can conveniently describe the constrained optimization problem in terms of a *factor graph* [6]. The latter is in general a bipartite graph, whose two species of nodes (called *variable nodes* and *function nodes*) are associated respectively to the decision variables and to the couplings among them. An arc between a function node and a variable node means that the corresponding variable is involved in the corresponding coupling. In our problem, a convenient set of decision variables is that of "characteristic" variables $x_{ij} \triangleq \chi\{\tau_i \ni j\} \in \{0, 1\}$, appearing in the matching contraints (4), which completely specify any transmission fanout set as $\tau_i = \{j \in O \,|\, x_{ij} = 1\}$. In terms of these variables, we can identify two different kinds of couplings, namely, the local weights $w_{i\tau_i}$, appearing in (8), and the contraints (4) themselves. Each local weight is associated to an input i and involves variables $x_{i1}, \ldots, x_{iM}$, whereas each matching constraint is associated to an output j and involves variables $x_{1j}, \ldots, x_{Nj}$. In conclusion, the factor graph associated to our problem is of the type sketched in Fig. 2.

As shown in [6], given the formulation of a combinatorial optimization problem in terms of a factor graph, the construction of BP equations is conceptually a well-established issue, even though non-trivial manipulations are often required to put them in a conveniently simple form. Due to limited space, here we give only the final equations, and we shall report the full derivation elsewhere. Let us only mention the fact that, even though we generically speak of BP, our algorithm is indeed of the *min-sum* type [6], which can be regarded as a special case of BP, specifically suited for computing MAP (maximum a posteriori probability) estimates. We can define the *beliefs*, associated to each transmission fanout set variable τ_i, as

$$m_{i\tau} = w_{i\tau} - \sum_{j\in\tau} b_{j\to i} \tag{9}$$

These quantities represent, apart from an irrelevant additive constant, an estimate of the weight that can be obtained by choosing a specific value $\tau_i = \tau$. Moreover, the "backward" *messages* $b_{j\to i}$ are an estimate of the weight degradation due to possible conflicts generated at output j by the choice $x_{ij} = 1$,

DEC-BPn (input: $[y_{i\sigma}]_{i\in I,\sigma\in S}$; output: $[\sigma_i, \tau_i]_{i\in I}$)
0. **for** $i \in I$ and $\tau \in S$, **compute** $w_{i\tau}$ and $\hat{\sigma}_{i\tau}$ by (7) and (6)
1. **set** $\tilde{I} = I$ and $\tilde{O} = O$
2. **while** $\tilde{I} \neq \emptyset$
3. **for** $i \in \tilde{I}$ and $j \in \tilde{O}$, **set** $b_{j\to i} = 0$
4. **repeat** n times
for $i \in \tilde{I}$ and $j \in \tilde{O}$, **compute** $f_{i\to j}$ by (11) and (9)
for $i \in \tilde{I}$ and $j \in \tilde{O}$, **compute** $b_{j\to i}$ by (10)
5. **for** $i \in \tilde{I}$ and $\tau \in S \,|\, \tau \subseteq \tilde{O}$, **compute** $m_{i\tau}$ by (9)
6. **choose** $i \in \tilde{I}$ and $\tau \in S \,|\, \tau \subseteq \tilde{O}$ that maximize $m_{i\tau}$
7. **if** $m_{i\tau} = 0$, **set** $\tau = \emptyset$
8. **set** $\tau_i = \tau$ and $\sigma_i = \hat{\sigma}_{i\tau}$
9. **set** $\tilde{I} = \tilde{I}\backslash i$ and $\tilde{O} = \tilde{O}\backslash\tau_i$

Fig. 3. DEC-BPn algorithm (decimation with n BP iterations).

i.e., $j \in \tau_i$ (transmission from i to j). These messages are defined by suitable self-consistency equations, namely

$$b_{j\to i} = \max_{i'\in I\backslash i} f_{i'\to j} \tag{10}$$

$$f_{i\to j} = \max\left\{0\,,\ \max_{\tau\in S\,|\,\tau\ni j} m_{i\tau} + b_{j\to i} - \max_{\tau\in S\,|\,\tau\not\ni j} m_{i\tau}\right\} \tag{11}$$

where the "forward" messages $f_{i\to j}$ can be finally regarded as an estimate of the weight gain that can be obtained by the single choice $x_{ij} = 1$ (rather than 0). The solution of these self-consistency equations by iterative refinement is an instance of BP, as it involves message passing from input to output ports (forward messages) and vice versa (backward messages). It is a well known fact that BP likely converges, if the underlying factor graph is treelike (equations are exact if the graph is rigorously a tree). In our case, the factor graph is densely connected, and, consistently, we find several instances of the problem in which BP does not converge. Because of this problem, it is not possible to use directly the beliefs (9) to fix the decision variables, since this may lead to unsatisfied constraints. This is why we have resorted to use BP with a fixed number of iterations, in conjunction with a simple decimation algorithm, which at each step fixes a given variable $\tau_i = \tau$ with the maximum belief $m_{i\tau}$, simplifies the equations to be compatible with the choice taken, and then reruns BP. The resulting algorithm may be described by the pseudocode reported in Fig. 3.

The algorithm takes as input the queue length matrix Y at timeslot t and returns the scheduling decision, in terms of the fanout set variables σ_i, τ_i for each input i. Step 0 performs the local optimization procedure, defined by equations (6) and (7). The "lists" $\tilde{I}$ and $\tilde{O}$ of "unreserved" inputs and outputs, respectively, are initialized at step 1, assuming that all the ports are initially available. Step 2 begins the decimation loop, which continues until every input has taken a decision, i.e., as far as $\tilde{I}$ is not empty. Steps 3–5 represent three different phases of BP, namely, initialization of backward messages, computation of forward messages as a function of backward messages and vice versa (for a number of iterations), and

GR-LQF (input: $[y_{i\sigma}]_{i\in I,\sigma\in S}$; output: $[\sigma_i,\tau_i]_{i\in I}$)
1. **set** $\tilde{I}=I$, $\tilde{O}=O$, and $\sigma_i=\tau_i=\emptyset$ **for** $i\in I$
2. **while** $\tilde{I}\neq\emptyset$
6. **choose** $i\in\tilde{I}$ and $\sigma\in S\,|\,\sigma\cap\tilde{O}\neq\emptyset$ that maximize $y_{i\sigma}$
7. **if** not found or $y_{i\sigma}=0$, **break**
8. **set** $\sigma_i=\sigma$ and $\tau_i=\sigma\cap\tilde{O}$
9. **set** $\tilde{I}=\tilde{I}\backslash i$ and $\tilde{O}=\tilde{O}\backslash\tau_i$

GR-RND (input: $[y_{i\sigma}]_{i\in I,\sigma\in S}$; output: $[\sigma_i,\tau_i]_{i\in I}$)
1. **set** $\tilde{I}=I$, $\tilde{O}=O$, and $\sigma_i=\tau_i=\emptyset$ **for** $i\in I$
2. **while** $\tilde{I}\neq\emptyset$
6. **choose** random $i\in\tilde{I}$ and $\sigma\in S\,|\,\sigma\cap\tilde{O}\neq\emptyset$ such that $y_{i\sigma}>0$
7. **if** not found, **break**
8. **set** $\sigma_i=\sigma$ and $\tau_i=\sigma\cap\tilde{O}$
9. **set** $\tilde{I}=\tilde{I}\backslash i$ and $\tilde{O}=\tilde{O}\backslash\tau_i$

Fig. 4. Greedy algorithms GR-LQF (longest queue first) and GR-RND (randomly chosen queue).

computation of beliefs (as a function of backward messages). Step 6 chooses an input i and a transmission fanout set τ, such that i is available and τ contains only available outputs, maximizing the belief $m_{i\tau}$ (since the maximum may be not unique, a random choice among the equivalent maxima turns out to improve the scheduling fairness). Step 7 states that, if the maximum belief found is zero, the algorithm assigns a null transmission fanout set (which corresponds to a vanishing belief as well). The transmission fanout set at input i and the corresponding optimal queue to be served are fixed at step 8. Step 9 updates the lists of available inputs and outputs. Finally, it is understood that, when the decimation loop is over, the current values of the fanout set variables are returned.

IV. Performance evaluation

In this section, we compare the performance of the DEC-BPn algorithm with other centralized scheduling algorithms, designed to support multicast traffic, under different traffic conditions. These algorithms are based on a greedy approach, with two slightly different strategies, described by the pseudocodes in Fig. 4. Note that the overall structure of both algorithms is similar to that of DEC-BPn, even though the steps typical of BP (0 and 3–5) are missing. The characterizing step is in fact only 6: GR-LQF tries to serve the longest queue, whereas GR-RND chooses a random queue, provided, in both cases, that the corresponding fanout set includes some available outputs.

The input traffic is generated according to a Bernoulli i.i.d. arrival process, in which ρ is the average input load (i.e., the probability that a packet arrives at an input port during a timeslot). The corresponding fanout set is chosen at random in a possible set of candidate ones, as described below. Traffic admissibility implies $\rho\leq\rho_{\max}$, where $\rho_{\max}=\frac{M}{Nf}$ is the maximum admissible input load and f is the average *fanout* (i.e., the average cardinality of the fanout set). We consider two different families of candidate fanout sets. The first one is referred to as *uniform traffic* and derived from [7]: The fanout set of each packet is chosen at random among all possible 2^M-1 ones. For this case, it can be shown that

$$f=\frac{M\,2^{M-1}}{2^M-1}\quad\text{and}\quad\rho_{\max}=\frac{2^M-1}{N\,2^{M-1}}\qquad(12)$$

Note that, since $\rho_{\max}\approx\frac{2}{N}$, the input load becomes very small for large switches, which does not allow to create critical (even though admissible) traffic patterns. This observation motivates the other traffic family, which has been devised in such a way to keep $\rho_{\max}$ independent of the switch size. The latter family is referred to as *concentrated traffic* and corresponds to the worst-case traffic model presented in [3]. Such a model has been designed to create extensive contention among inputs and was crucial in [3] to show the intrinsic throughput limitations of IQ switches under multicast traffic. Without going into the details of their construction, in Table I we describe three different concentrated traffic scenarios (denoted as "Conc-1", "Conc-2", and "Conc-3"), reporting the corresponding $\rho_{\max}$ and the list of all fanout sets for each input.

TABLE I
FANOUT SETS FOR EACH CONCENTRATED TRAFFIC SCENARIO.

Traffic	$\rho_{\max}$	Input 1	Input 2	Input 3
Conc-1 2×4	1.00	$\{1,2\}$ $\{3,4\}$	$\{1,3\}$ $\{2,4\}$	not used
Conc-2 2×4	0.67	$\{1,2,3\}$ $\{2,3,4\}$	$\{1,2,4\}$ $\{1,3,4\}$	not used
Conc-3 3×12	1.00	$\{1,2,3,4\}$ $\{5,6,7,8\}$	$\{1,5,9,10\}$ $\{2,6,11,12\}$	$\{3,7,9,11\}$ $\{4,8,10,12\}$

In order to compare the algorithms, we have evaluated both the throughput and the average delay. Throughput is evaluated in terms of maximum sustainable load at the outputs; this is a value between 0 and 1, representing the maximum fraction of timeslots exploited to transmit a packet at the outputs. Even though the traffic is admissible, the throughput may be less than unity, even for the optimal scheduling algorithm, because of the aforementioned intrinsic throughput limitations [3]. The delay is evaluated as the average time interval between the timeslot when a packet enters the switch and the one when its copies leave the switch; the average is evaluated taking into account all the copies.

A. Numerical results

Let us start by comparing the performances under uniform traffic. Table II shows the maximum achievable throughput under three uniform scenarios. When considering the symmetric traffic scenario (second column), all the algorithms behave exactly in the same way and achieve maximum throughput. This is due to the low input load (always less than $\rho_{\max}$, to be admissible), which does not generate "critical" loading conditions. Conversely, when concentrating the traffic on few inputs (4 and 2), performances are different, and, in both scenarios, DEC-BP outperforms the other two centralized greedy approaches. Let us note that increasing the number

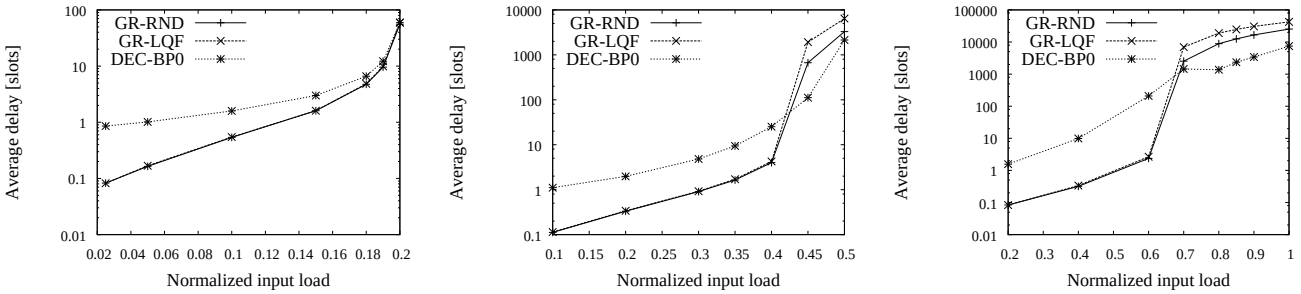

Fig. 5. Average delay under uniform traffic: (a) 10×10, (b) 4×10, (c) 2×10.

TABLE II
MAXIMUM THROUGHPUT UNDER UNIFORM TRAFFIC.

Traffic	Uniform 10×10	Uniform 4×10	Uniform 2×10
$\rho_{\max}$	0.20	0.50	1.00
Algorithm	Throughput		
GR-LQF	1.00	0.86	0.64
GR-RND	1.00	0.92	0.75
DEC-BP0	1.00	0.98	0.95
DEC-BP1	1.00	0.98	0.95
DEC-BP2	1.00	0.97	0.95
DEC-BP4	1.00	0.97	0.95

TABLE III
MAXIMUM THROUGHPUT UNDER CONCENTRATED TRAFFIC.

Traffic	Conc-1 2×4	Conc-2 2×4	Conc-3 3×12
$\rho_{\max}$	1.0	0.67	1.0
Algorithm	Throughput		
OPTIMAL	0.75	0.99	-
GR-LQF	0.70	0.91	0.63
GR-RND	0.69	0.89	0.61
DEC-BP0	0.75	0.97	0.66
DEC-BP1	0.75	0.97	0.66
DEC-BP2	0.75	0.97	0.66
DEC-BP4	0.75	0.98	0.66

of iterations n in DEC-BPn seems not to improve the performance at all. A deeper investigation about the reasons underlying this rather unexpected result is beyond the scope of this short paper. Here, we simply suggest to promote DEC-BP0 as the best candidate algorithm for multicast scheduling.

Fig. 5 shows the average delays under the three uniform scenarios. Here, we do not report the curves for DEC-BPn for $n \geq 1$, as they turn out to be fully overlapped with that of DEC-BP0. Fig. 5(a) shows that, under symmetric traffic, all the algorithms achieve maximum throughput (for $\rho = \rho_{\max}$) but the delay in the low load regime is worst for DEC-BP0. This effect is to be ascribed to the peculiar form of the weight function (recall the explanation of (5)), which does not equalize and minimize the queue sizes, and trades higher delays at low load with higher throughput. Note however that the higher delays experienced by DEC-BP0 are negligible in absolute terms. The other two uniform scenarios highlight some relevant differences among the three algorithms, especially in terms of throughput. Table II shows that DEC-BP0 always outperforms both greedy approaches, with a gain between 6% and up to 48%. In terms of delays, Fig.s 5(b) and 5(c) display a behavior similar to Fig. 5(a) for low load, but different when the load is higher, due to the different maximum throughput. Let us finally note that, when the traffic is no longer sustainable, the delays are still finite because of the finiteness of the queues and/or of the simulation; otherwise they would have grown to infinity.

Let us now consider the concentrated traffic scenarios. Table III displays the achievable throughput for all the policies considered so far, with the addition of the OPTIMAL algorithm, which simply maximizes the weight function (5) by an exhaustive search. OPTIMAL has not been run for a number of outputs larger than 4, due to the large computational effort required. Our results show that, even in this case, the effect of the number of iterations in DEC-BPn is negligible, which again promotes DEC-BP0 as the best scheduling algorithm. In both Conc-1 and Conc-2 traffic scenarios, DEC-BP0 achieves the optimal throughput. As in the previous scenarios, DEC-BP0 outperforms the other greedy approaches, with throughput gains between 5% and 10%.

In terms of delays, all the curves in Fig. 6 exhibit a quite similar behavior, coherent with the maximum achievable throughput. Furthermore, we can observe an interesting property of the optimal policy: In the low load regime, the delay is much larger than for the other policies. As already observed when discussing uniform traffic, we argue that such an effect is just due to the metric based on the weights in (5), which does not minimize and equalize the queue lengths. The latter observation corroborates our previous argument, namely, that the higher delays experienced by DEC-BP0 are mainly due to the fact that this algorithm approximates the optimal policy.

V. CONCLUSION

We have proposed a new algorithm, DEC-BPn, aimed at approximating the optimal scheduling policy for the transmission of multicast packets in IQ switches. Our algorithm has two main advantages. First, the proposed message-passing approach is amenable to an efficient hardware implementation.

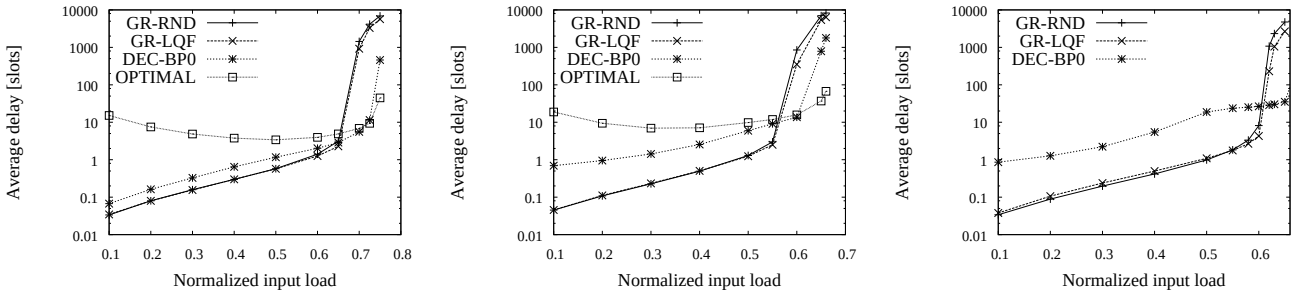

Fig. 6. Average delay under concentrated traffic: (a) Conc-1, (b) Conc-2, (c) Conc-3.

Second, we have shown that it outperforms other greedy approaches, even when the number of iterations n is zero or very small. These encouraging findings allow us to conclude that our approach provides a very convenient tradeoff between implementation complexity and performances.

ACKNOWLEDGEMENTS

The research leading to these results has been funded by the European Union Seventh Framework Programme under the FET project “STAMINA”.

REFERENCES

[1] H. Chao and B. Liu, *High performance switches and routers*. Wiley-IEEE Press, 2007.

[2] W. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks,” in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684–689.

[3] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, “Multicast traffic in input-queued switches: optimal scheduling and maximum throughput,” *IEEE/ACM Transaction on Networking*, vol. 11, no. 3, pp. 465–477, 2003.

[4] S. Sarkar and L. Tassiulas, “A framework for routing and congestion control for multicast information flows,” *IEEE Transactions on Information Theory*, vol. 48, no. 10, pp. 2690–2708, 2002.

[5] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[6] F. Kschischang, B. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[7] B. Prabhakar, N. McKeown, and R. Ahuja, “Multicast scheduling for input-queued switches,” *Selected Areas in Communications, IEEE Journal on*, vol. 15, no. 5, pp. 855–866, 1997.