Bandwidth Reservations by Maximal Matching Algorithms

Aleksandra Smiljanić, *Member, IEEE*

***Abstract*—A maximal matching algorithm switches packets through a cross-bar with the speed-up of two without blocking them. Namely, traffic will go through the cross-bar controlled by a maximal matching algorithm if its outputs are not overloaded. Consequently, bandwidth reservations with delay guarantees are simple to provide. We propose a protocol for distributed bandwidth reservations, where users check the communication availability among themselves. It will be also shown that maximal matching algorithms cannot utilize full cross-bar capacity for some particular traffic patterns.**

***Index Terms*—Cross-bar fabric, maximal matching algorithms, packet switches, scheduling.**

I. Introduction

It has been recognized that packet switches with input buffers are attractive because of their scalability [1]. Packets are stored in input buffers at the line bit-rate, and transferred to the designated output ports through a cross-bar fabric. Packets are split into cells of the same length, and, then transferred through the fabric. In each time slot, a cross-bar fabric can transfer at most one cell from any input, and at most one cell to any output.

An incoming packet is stored in a virtual output queue (VOQ) of an input buffer according to its destination port. The central arbiter of a switch gets the information about status of input–output queues, and determines which inputs and outputs will be connected in each time slot. This information feeding the central arbiter can be represented by a bipartite graph. In a bipartite graph nodes are divided into two groups and edges exist only between nodes in different groups. In our case, nodes of one group correspond to inputs, and nodes of the other group correspond to outputs. An edge exists between two nodes if there are packets from the input to the output in question. A scheduling algorithm performed by the central arbiter should find a subgraph of this graph such that the degree of each node is at most one. Then, input–output pairs will be connected if there are edges between corresponding nodes.

A maximal matching algorithm has been defined in graph theory and numerous networking papers on scheduling algorithms for packet switches with input buffers.

Definition: A maximal matching algorithm finds a maximal subgraph of a bipartite graph in which a node degree is at most one. Here the maximal subgraph (called maximal matching) is the one to which no edge can be added while keeping a node degree to be at most one.

In our case, if there is a packet from some input to some output, a maximal matching algorithm will schedule either the input or the output in question. If we assume the opposite (that there is a packet from some input to some output and neither the input nor the output are scheduled), a bipartite subgraph that corresponds to this schedule would not be maximal because an edge can be added between the nodes corresponding to these input and output, and their degrees become one (and are at most one).

The parallel iterative matching (PIM) algorithm proposed in [1] was shown to find a maximal matching between inputs and outputs after $\log_2 N$ iterations on average. The SLIP algorithm is derived from the PIM algorithm, and is simpler to implement [6]. The SLIP algorithm also finds a maximal matching after $\log_2 N$ iterations on average, because the proof of this property in [1] applies equally to SLIP. Note that SLIP is implemented in the Cisco Gigabit Switched Router (GSR). A two-dimensional wavefront arbiter (WFA) has been proposed in [11]. Modules on a two-dimensional grid get requests from corresponding input–output queues and send acknowledgments to them. In each arbitration slot, a module gets the information if its either input or output have been selected by previous modules, and if they were not, the module sends a grant to the queue in question. WFA obviously finds a maximal matching between inputs and outputs and can be implemented by pipelining. We have proposed a scalable maximal matching algorithm, named sequential greedy scheduling (SGS) in [10]. In SGS, inputs sequentially choose outputs one after another. It finds a maximal matching after N iterations, but processing can be arbitrarily relaxed by using pipelining. Namely, SGS can be implemented in a distributed fashion so that each input processor communicates with two adjacent processors. Similar pipelining has been proposed in [5], however, there, inputs can schedule only a specified number of heading packets in their FIFO queues, and such an algorithm is not maximal matching. We believe that the one-dimensional structure of SGS is simpler for implementation than the two-dimensional structure of WFA. In addition, the SGS arbiter could be conveniently placed on different line cards, where only adjacent line-cards would communicate.

If its cross-bar has the speedup of two and is run by a maximal matching algorithm, the packet switch is stable for admissable traffic patterns that obey the strong law of large numbers [3], [4]. In this paper, we derive admission control and policing mechanisms that provide rate and delay guarantees through a cross-bar fabric with the speedup of two which is run by a

Manuscript received July 30, 2004. The associate editor coordinating the review of this letter and approving it for publication was Dr. M. Ma.

The author is with AT&T Research Labs, Middletown, NJ 07748 USA (e-mail: aleks@research.att.com).

Digital Object Identifier 10.1109/LCOMM.2004.825721

maximal matching algorithm. We also evaluate actual fabric capacities that can be reserved by exemplary maximal matching algorithms.

II. Switch Capacity That can be Reserved

It has been proven in [9], [10] that SGS can reserve a half of the fabric capacity due to the fact that it is the maximal matching algorithm. We repeat this theorem here for the sake of completeness. Afterwards, a protocol for distributed bandwidth reservations for this switch architecture will be proposed.

Lemma 1: A maximal matching protocol ensures a_{ij} time slots per frame lasting F time slots to input–output pair (i,j), $1 \leq i, j \leq N$, (N is the number of ports) if the following condition holds:

$$\sum_m a_{im} + \sum_m a_{mj} - a_{ij} \leq F. \tag{1}$$

Proof: Let assume that in any frame, the protocol passes only packets that have arrived in the previous frame. Observe time slots within a frame in which either input i or output j are connected, but not to each other. In each of these time slots, sum $s_{ij} = \sum_{m \neq j} b_{im} + \sum_{m \neq i} b_{mj}$ is greater than 0, and then it is decremented by at least 1. Here b_{im} is the number of packets that have arrived to input i for output m in the previous frame, but have not been transferred through the switch yet. Sum s_{ij} is the largest at the beginning of a frame and from (1), it fulfills

$$s_{ij} = \sum_{m \neq j} a_{im} + \sum_{m \neq i} a_{mj} \leq F - a_{ij}.$$

As a conclusion, in at least a_{ij} time slots per frame, neither input i is connected to some output other than j, nor is the output j connected to some input other than i. In these time slots, input i reserves output j if there are packets in queue (i,j), i.e., $b_{ij} > 0$, by the definition of a maximal matching algorithm. In summary, if condition (1) is fulfilled then a_{ij} time slots per frame are guaranteed to input–output pair (i,j). □

Lemma 2: A maximal matching algorithm ensures a_{ij} time slots per frame to input–output pair (i,j), $1 \leq i, j \leq N$, if the following condition holds:

$$\sum_m a_{im} \leq \frac{F+1}{2}, \quad \sum_m a_{mj} \leq \frac{F+1}{2}. \tag{2}$$

Proof: This proof was presented in [8] and directly follows from Lemma 1. □

Theorem 1: A maximal matching algorithm ensures p_{ij} of the line bit-rate to input–output pair (i,j), $1 \leq i, j \leq N$, if the following condition holds:

$$\sum_m p_{im} \leq \frac{1}{2}, \quad \sum_m p_{mj} \leq \frac{1}{2}. \tag{3}$$

Proof: Condition (3) implies (2), so Theorem 1 follows from Lemma 2. □

Note that in both claims and proofs we only substituted "the SGS algorithm" by "a maximal matching algorithm", otherwise they are identical. This is because all proofs in [9] use the sole property of SGS that it is a maximal matching algorithm.

It follows from Theorem 1 that a cross-bar fabric with the speed-up of two, and run by any maximal matching algorithm will pass packets without blocking them. Namely, as long as packets bound for outputs do not overload them over some time period, they will get through the fabric. Bandwidth reservations become very simple in this architecture. If input–output pair (i,j) requests a new portion of bandwidth, Δp_{ij}, it is accepted if

$$\sum_m p_{im} + \Delta p_{ij} \leq \frac{s}{2}, \quad \sum_m p_{mj} + \Delta p_{ij} \leq \frac{s}{2}, \tag{4}$$

where s is the cross-bar speed-up. If input i sends a_{ij} packets per frame to output j for all $1 \leq i, j \leq N$, any arriving packet will be transmitted in the next frame and experience the delay of at most two frames. In this way, both bandwidth and delay guarantees can be provided to users.

Admission control as well as policing can be done both in a distributed or a centralized fashion. In [8] we described centralized admission control and policing. The central arbiter stores reserved bandwidths of output ports, and decides if the new request can be granted. A counter, c_{ij} is associated to each input–output pair, and it is loaded at the beginning of each frame to the negotiated number of credits per frame $c_{ij} = a_{ij}$. Whenever some queue is served, its counter is decremented by 1, and the arbiter considers queues for service as long as their counters are positive.

Alternatively, admission control and policing can be done in a distributed manner. In this case the sum of bit-rates of all users attached to some port should be smaller than the port bandwidth. Then, a user sourcing some information checks if its destination has capacity to receive the information, and if so, it transmits data. Policing can be distributed as well, but with some caution. Each source stores a counter for each of its destinations which is set once per frame to the negotiated number of packets, e.g., $c_{sd} = a_{sd}$ for source s and destination d. Similarly as in the centralized case, a source sends packets as long as its counter is positive, and decrements the counter whenever a packet is sent. In the most straightforward implementation, frame boundaries at all switch ports are the same. Otherwise, more than the negotiated number of packets may arrive to the switch per frame, and some of the traffic could be blocked.

If the frames at different ports are not synchronized, the correct switch operation can be accomplished in the following way. Access routers/switches attached to the high-capacity switch port, would delineate frames by designated packets. One extra bit per packet, FB, is set at the port to denote its frame, and is toggled in each frame. In some frame the switch arbiter will schedule only packets received before this frame with FB equal to the specified switch bit, SB. SB toggles in each frame as well. Our proposal is illustrated in Fig. 1. The upper axis in Fig. 1(a) shows the switch frame boundaries, while the lower axes in Fig. 1(b) and (c) show the port frame boundaries. At the beginning of each switch frame, SB toggles, and at the beginning of each port frame, FB toggles, as shown. So, only packets with $\mathrm{FB} = \mathrm{SB} = 0$ that have arrived before the switch frame $k+2$ in Fig. 1(a) will be scheduled in the switch frame $k+2$, and these are packets of the upper port frame $m+1$ in Fig. 1(b). Similarly, packets of the port frame $m+2$ will be scheduled in the switch frame $k+3$, etc. In Fig. 1(b), the port is synchronized properly, while in Fig. 1(c), it is not. Namely, packets arriving

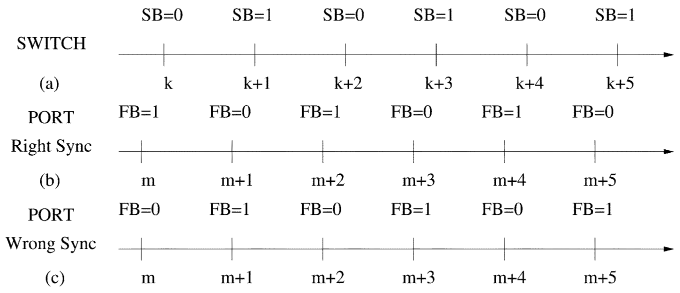

Fig. 1. Synchronization of the packet scheduling.

at the end of the port frame m and packets arriving at the beginning of the port frame $m+2$ are eligible for scheduling in the switch frame $k+3$. So, the number of packets bound for some output that will be scheduled in frame $k+3$ might exceed negotiations, and would be blocked. So, SB and FB's have to be properly synchronized: an arbiter sets $\mathrm{FB}=1-\mathrm{SB}$ if the switch frame boundary preceded the previous port frame boundary (delineation packet), or $\mathrm{FB}=\mathrm{SB}$ otherwise, where FB is the frame bit of the first packet arriving as the synchronization process started. The traffic will get through a cross-bar fabric with the speed-up of two, because only the negotiated number of packets are scheduled in each frame.

III. Examples of Bandwidth Reservations

Here, we evaluate the efficiency of bandwidth reservations provided by SLIP with N iterations and SGS. SLIP performing N iterations is maximal matching. In the frame synchronized version of these algorithms, only packets that have arrived in the previous frame are scheduled in the current frame. The packets that remained unscheduled are dropped, because the frame duration is chosen to meet the end-to-end delay budget, while the longer delay would be unacceptable. The algorithm efficiency is a normalized throughput for which the packet loss is below $5\cdot 10^{-2}$. In our simulations $N\in\{16,32\}$.

It has been shown in [7] that the efficiency of SLIP drops for nonuniform traffic pattern. Namely, queues start to build up for the traffic pattern:

$$p_{ij}=p\cdot\begin{cases}\frac{0.5}{N}+0.5 & j=i,\ 1\le i\le N\\ \frac{0.5}{N} & j\ne i,\ 1\le i,\ j\le N\end{cases} \tag{5}$$

and the efficiency (normalized throughput) of $p=80\%$. This result has been confirmed when packet arrivals form a Bernoulli random process, however, in our proposed synchronized case, the efficiency does not drop.

Another example in which the SLIP efficiency drops has been described in [2]. It has been shown that if queue pointers get wrongly synchronized, the efficiency of SLIP drops to $p=67\%$ for the traffic pattern:

$$p_{11}=p_{12}=p_{22}=p_{23}=p_{33}=p_{31}=\frac{p}{2}. \tag{6}$$

We obtained the same efficiency drop for the frame synchronized version of the algorithm, and for the more general traffic

$$p_{ij}=p\cdot\begin{cases}0.5, & 1\le i\le n,\ j\in\{i,i+1\bmod n\}\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

where values of input and output pointers $i_k, o_k, 1\le k\le n$ are initially:

$$\begin{aligned}i_k&=k-\mathrm{div}(k+1,3),\\ o_k&=k-1+\mathrm{div}(k-1,3)\end{aligned} \tag{8}$$

and $\mathrm{div}(x,y)$ is an indicator equal to 1 if x is divisible by y. Any combination of these traffic patterns would also cause the efficiency drop.

Efficiency of the SGS algorithm does not drop for traffic patterns (5), (7). However, we found that its utilization drops for the traffic pattern:

$$p_{ij}=p\cdot\begin{cases}0.5 & 1\le i\le \frac{N}{2},\ j\in\left\{i,i+\frac{N}{2}\right\},\\ \frac{1}{N} & \frac{N}{2}<i\le N,\ 1\le j\le N\end{cases} \tag{9}$$

to the value of $p=76\%$.

We conclude that SLIP cannot reserve more than 67%, while SGS cannot reserve more than 76% of port capacities, because the traffic might be blocked when port loads exceed these values.

IV. Conclusion

A maximal matching algorithm can reserve a half of the cross-bar capacity. Proposed simple algorithm would allow distributed bandwidth reservations without a strict switch synchronization with the rest of the network. In addition, when users perform presented policing, the delay guarantees are provided in a packet switch with input buffers run by a maximal matching algorithm. Rate and delay guarantees are essential for a support of the delay sensitive traffic such are voice and video through packet switches. Since scalable maximal matching algorithms have been designed, delay sensitive traffic can be supported in high-capacity packet switches.

References

[1] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Computer Syst.*, vol. 11, no. 4, pp. 319–352, Nov. 1993.
[2] C. S. Chang, D. S. Lee, and Y. S. Jou, "Load balanced Birkhoff-von Neumann switches," in *IEEE Conf. on High Performance Switching and Routing 2001*, pp. 276–280.
[3] G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *IEEE INFOCOM 2000*, pp. 556–564.
[4] E. Leonardi, M. Mellia, A. Marsan, and F. Neri, "Stability of maximal matching scheduling in input-queued cell switches," in *Proc. IEEE Int. Conf. on Communications 2000*, pp. 1758–1763.
[5] H. Matsunaga and H. Uematsu, "A 1.5 Gb/s 8×8 cross-connect switch using a time reserved algorithm," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1308–1439, Oct. 1991.
[6] N. McKeown *et al.*, "The tiny tera: A packet switch core," *IEEE Micro*, vol. 17, no. 1, pp. 26–33, Jan./Feb. 1997.
[7] R. Rojass-Cessa, E. Oki, and H. J. Chao, "CIXB-k combined input-crosspoint-output buffered packet switch," in *Proc. GLOBECOM 2001*, vol. 4, pp. 2654–2660.
[8] A. Smiljanić, R. Fan, and G. Ramamurthy, "RRGS-round-robin greedy scheduling for electronic/optical terabit switches," in *Proc. GLOBECOM*, Dec. 1999, pp. 1244–1250.
[9] A. Smiljanić, "Flexible bandwidth allocation in terabit packet switches," in *Proc. IEEE Conf. on High Performance Switching and Routing*, June 2000, pp. 233–241.
[10] ——, "Flexible bandwidth allocation in high-capacity packet switches," *IEEE/ACM Trans. Networking*, vol. 10, pp. 287–293, Apr. 2002.
[11] Y. Tamir and H. C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 13–27, 1993.