Fast scheduler solutions to the problem of priorities for polarized data traffic

Gérard Damm, John Blanton, Prasad Golla, Dominique Verchère, and Mei Yang

Alcatel USA Research and Innovation Department
1201 E. Campbell Rd, Richardson, Texas 75081, USA
{Gerard.Damm, John.Blanton, Prasad.Golla, Dominique.Verchere, Mei.Yang}@usa.alcatel.com

Abstract

Three fast scheduling algorithms are proposed to handle efficiently polarized traffic in an advanced architecture for multiple-server optical IP packet routers. These schedulers must be capable of transferring several fixed-sized packets per port per time slot rather than one cell per time slot. An optical burst packet (OBP) contains a number of cells, obtained either from a network protocol or from segmentation. To optimize the resource utilization, it is important to maximize the filling ratio of these OBPs, but also important to adjust the right time-out to trigger the transfer of non-full packets. Moreover, the hardware implementation must be feasible and cost-effective. The solutions presented in this paper are detailed and compared by means of simulations. The first of them, MultiSLIP, is a generalization of iSLIP to the multiple-server architecture. The second one, PDRR, is a generalization of DRR (Dual Round Robin) to multiple-server architecture with priorities. The third one, FR-DRR (Flexible Ring DRR), features a new arbitration scheme based on a circular list of pending requests.

I. Introduction

The switch fabric model considered in this paper is a set of *N* input ports are connected to a switching device such as an optical matrix (OM), which is connected to *N* output ports. The input memory is organized in VOQs (Virtual Output Queues) so as to avoid HOL (Head Of Line) blocking [1]. In the classical architecture, each port has one server, which allows the transfer of one cell per port per matrix cycle (or time slot). The scheduling problem comprises picking a requesting VOQ in each input port at each time slot while optimizing a set of different properties such as throughput maximization, fairness, non-starvation, and hardware feasibility [2].

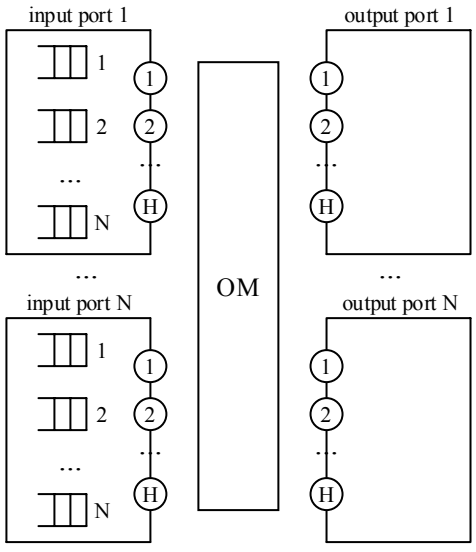

Figure 1. Multiple-server architecture

New advances in hardware systems have made it possible to have more than one server per port, so as to match the high capacity of the OM. This change imposes the design of innovative scheduling algorithms. In the multiple-server architecture, the scheduler can pick several OBPs (Optical Burst Packets) from different VOQs in each input port, and can also transfer several OBPs from one VOQ, with all the possible combinations. Any input server can take an OBP from any VOQ in its port and send it to any output server through the OM. Figure 1 represents the architecture with *N* ports and *H* servers per port.

The OBP is the data unit transferred by a server each time slot. It contains ECPs (Elementary Composite Packets), which in turn contain a number *K* of cells. This composition mechanism is usually referred to as *burstification* [3,4]. Figure 2 shows the structure of an OBP. For the purpose of architecture comparison, the capacity of a port (in bit/s) is supposed to be independent from *H*. Therefore, an OBP contains *N/H* ECPs (*H* is supposed to be a factor of *N*) so that each port can always send at most *N* ECPs at each time slot, whatever *H*. Intuitively, the finest granularity and highest flexibility should be obtained when *H=N*, i.e. when an OBP contains only one ECP. This result is confirmed by the simulations. In this paper, we will consider architectures in which *N*=16 and *H*∈ {2,4,8,16}.

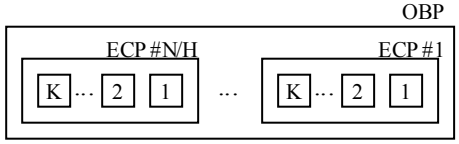

Figure 2. Optical Burst Packet structure

The specification of a core router includes a time-out limit *L*, which guarantees that, with a given probability, the transit time of all incoming data is smaller than or equal to *L* seconds. When there is uneven (polarized) data traffic into the different input queues, it will eventually become necessary to trigger the transfer of non-full OBPs through the matrix in order to clear data cells before they grow stale. This reduces the throughput of the switch, since non-full OBPs represent unused capacity. Due to space limitation, we keep out of the scope of this paper all the functionalities related to the ingress and egress stages, including reassembly of IP packets previously segmented into cells.

To meet the conflicting requirements of maximizing the throughput of the switch matrix and respecting the time limit, we propose three scheduling algorithms, described in section II. We compare them by simulation in Section III. Measured by the average data cell transit time, the performance of each scheme is assessed for a number of server and time slot configurations under different degrees of traffic load and polarization. Section IV presents a list of hardware implementation issues and section V concludes the study.

II. Description of the Schedulers

2.1. Computation of Grants

MultiSLIP

The first idea to deal with multiple-server architecture is to generalize the single server iSLIP algorithm [5]. This solution,

MultiSLIP, is described in detail in [6]. Basically, it consists in associating an RRA (Round Robin Arbiter) with each output server for the Grant phase of the RGA (Request-Grant-Accept) strategy. Each RRA makes its own pick according to its current state and the propagated requests. The Accept phase uses only one RRA per port to sequentially select among the issued grants. Just like in iSLIP, an output arbiter is updated only if and only if its grant has been accepted in the first iteration. Note that the choice between a port arbiter that makes sequential selections and server arbiters that make selections in parallel involves many issues, such as hardware implementation, execution time, selection validations, etc.

PDRR

The second solution, PDRR (Prioritized Dual Round Robin), generalizes DRR [7] to the multiple-server case. It also introduces the use of priorities associated with each scheduling request from the input VOQs. DRR uses the RG (Request-Grant) strategy. The version of PDRR presented here makes use of port RRA both at the input and the output sides. The requirement to service data packets within the specified time limit is satisfied by assigning the highest priority to queues containing incipiently stale data. The need to make maximum use of available switch matrix capacity is also satisfied by assigning an intermediate priority to requests for full packets when there are no incipiently stale data cells. The lowest priority is given to the non-full packets.

The handling of priorities is performed as described hereafter. In the Request phase, the input port RRAs sequentially pick among the highest priority requests, up to the number of available (not yet matched) servers in their port. If a priority level is exhausted, the next lower level is considered, and so on. In the output port RRAs, the same sequential pick procedure is applied to make selections (grants) among the issued requests.

FR-DRR

The flexible ring dual round robin (FR-DRR) scheme places the burden of resolving the data priority conflicts entirely on the scheduler. In this scheme, the VOQs do not explicitly request service but only report the most recent additions of data cells to the queues. A specific arbiter is introduced to manage those differential requests: the Circular Queue Arbiter (CQA). It consists of a linked list with a pointer. An element in a CQA represents a request from a VOQ for the transfer of one OBP. The pointer represents the currently favored request. It moves to the next position after each use, and comes back to the head of the queue at the end. New requests are always inserted at the end regardless of the pointer position. If the selection is validated, the request is removed from the list. This is a way to procure a proportional service to the requesting VOQs: the more pending requests a VOQ has, the more likely it will get serviced.

In FR-DRR, CQAs are used in the input ports for the Request phase, whereas the Grant phase in the output ports uses server RRAs. It would not make sense to use CQAs in the Grant phase, since there is no corresponding persistent state, due to the arbitration performed each cycle for issuing the requests.

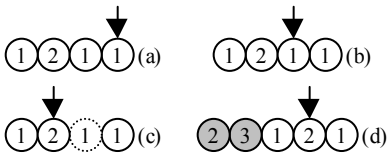

Figure 3. Circular queue

Figure 3 shows an example of a circular queue state evolution. In state (a), VOQ 1 has 3 pending requests and VOQ 2 has one pending request. The pointer is currently set to the head of the queue, on a request from VOQ 1. The CQA will therefore select it. Let us assume it is not granted. In the next iteration, the pointer is set to the next request from VOQ 1, as shown in state (b). Let us assume that this time, it is granted. In this case, VOQ 1 benefited over VOQ 2 from its larger number of pending requests. The request is removed and the pointer is moved to the next position, as shown in state (c). Note that the VOQ is still emptied in a FIFO manner, regardless of the selected request. State (d) represents the insertion of two additional requests, one request from VOQ 3 and one request from VOQ 2.

2.2. Generation of Requests

All these schedulers are designed to process requests and issue corresponding grants. However, the generation of a request is an independent problem and has an important impact on the efficiency of the whole scheduling.

MultiSLIP

It is especially true for MultiSLIP, since it has no implicit information regarding the age of the data or the filling ratio. We have presented in [8] a discussion on the generation of requests policy for VOQ-state unaware schedulers such as MultiSLIP. The simplest one, called *greedy* policy, consists in sending requests whenever at least one cell is in the queue. Comparatively, it provides the best throughput for uniform traffic, but is very limited for even slightly polarized traffic. The *thrifty* policy is better and consists in holding data until either a packet is full or the time limit is near. This policy performs rather well to handle polarized traffic but fails to sluice non-full packets in temporary low traffic situations. The best one, called *conservative* policy, improves the thrifty policy by examining the amount of requests issued at port level and comparing it to the number of servers. The possible remaining capacity is not fully requested, since the performance would otherwise drop back to the greedy policy level for polarized traffic. To avoid requesting one third of the remaining servers empirically proved to be efficient.

PDRR

As for PDRR, the policy for generating requests is simplified, since the priority concept maps naturally to the VOQ state of emergency. For the results presented in this paper, we have considered 5 priority levels (the higher the number, the higher the priority):

Priority	Request
4	Time-out (L) - 2 time slots
3	Time-out (L) - 10 time slots
2	full and VOQ length > 10
1	full and VOQ length <= 10
0	non-full packets

Figure 4. PDRR priority levels

The highest priority is for packets (full or not full) that will time out in less than 2 time slots. The next level is for packets (full or not full) that will time out in less than 10 time slots. Level 2 is for full packets in long VOQs (more than 10 packets). Level 1 is for full packets in short queues. Finally, the lowest level is for the non-urgent non-full packets.

FR-DRR

The generation of requests for FR-DRR addresses the throughput maximization requirement by making new requests for full packets. The time-out specification is enforced by issuing requests for packets that will grow stale in less than 10 slots. Incidentally, the queue length in a port tends to be balanced thanks to the proportional service, since a queue has a probability to be

serviced in proportion to its number of pending requests in the flexible ring. A drawback of this policy is that the low traffic conditions are not taken advantage of to evacuate non-full packets. However, any extra cells that might have arrived in a queue between the request and the grant are used to fill the OBPs as much as possible. As a consequence of this padding out, the number of grants received can be greater than the number of requests issued. A simulation program, as well as a hardware implementation, should occasionally expect grants for empty VOQs. The corresponding lost bandwidth due to these wasted grants is marginal, since it applies only to emergency requests.

III. Simulation Results

3.1. Traffic model

We used a polarized traffic pattern characterized by unbalanced traffic in the VOQs, while the global traffic is still balanced (so as to keep the same maximal throughput). For a given external load ρ, the arrival process for an input port consists of N Bernoulli trials with probability ρ every K^{th} of a slot, but the destination VOQ is not uniformly distributed. Instead, the proportions of traffic received by each VOQ form a geometric progression with a factor q: each queue receives q times more traffic than the previous one. With normalized proportions, the lowest traffic proportion is $(q-1)/(q^N-1)$. The proportions are rotated from one input port to another so that the output ports receive the same average traffic. This is the definition of a geometrically polarized traffic. We will consider polarization factor values q ranging from 1.0 to 2.0. The value 1.0 is really a uniform traffic, where all the proportions are equal to $1/N$.

3.2. Analysis of the simulations

The length of the time slot determines K, the number of cells per ECP. The maximum number of allowed iterations was either 4 or unlimited. We present here only the most significant results, with 16 cells per ECP and 4 iterations allowed per scheduling.

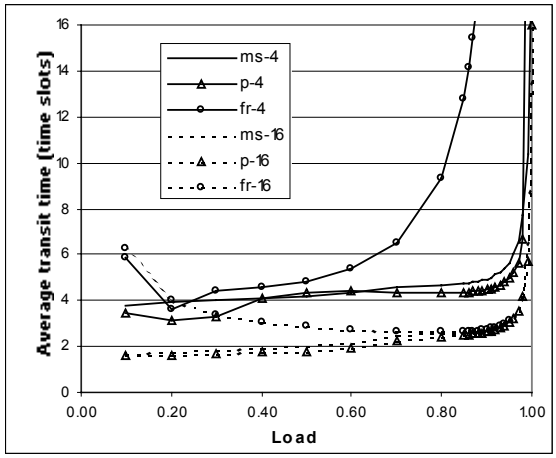

Figure 5. Average transit time (number of slots) as a function of the load for uniform traffic

Figure 5 shows the performance of the three schedulers in uniform traffic, for two configurations: 4 servers (full lines) and 16 servers (dotted lines). In this figure, the following abbreviations are used: *ms* for MultiSLIP, *p* for PDRR, and *fr* for FR-DRR. With only 4 servers, PDRR is the best, closely followed by MultiSLIP. FR-DRR has interesting performance only up to 60% load, and then degrades. Given that a realistic time-out limit L corresponds to about 100 slots, FR-DRR still reaches 95% throughput, whereas PDRR and MultiSLIP reach 99%. When 16 servers are available per port, the 3 schedulers converge to the same performance at high loads, but as expected, FR-DRR does not handle low traffic as well as the other two (also with 4 servers). Also, FR-DRR saturates earlier, near 95%. So for a uniform traffic, FR-DRR does not perform as well as PDRR and MultiSLIP, but is still acceptable.

We have run simulations for polarization factors ranging from 1.25 to 2.0, and present the most polarized case in Figure 6. Here again, we compare the average transit ime for configurations with 4 servers and with 16 servers. The rightmost point in a curve is the last measurement before saturation (last successful simulation as the load goes up). For instance, FR-DRR can only sustain up to 70% of traffic with 4 servers, while MultiSLIP approaches 90% and PDRR reaches 98%. With 16 servers, FR-DRR is weaker than the other two at low loads, but performs well at high loads. MultiSLIP starts to saturate just after 90%. PDRR has a larger average transit time, but reaches slightly higher loads than FR-DRR (99% versus 96%). In all cases, the transit times are acceptable, so the critical performance indicator is the maximum admissible throughput (saturation point).

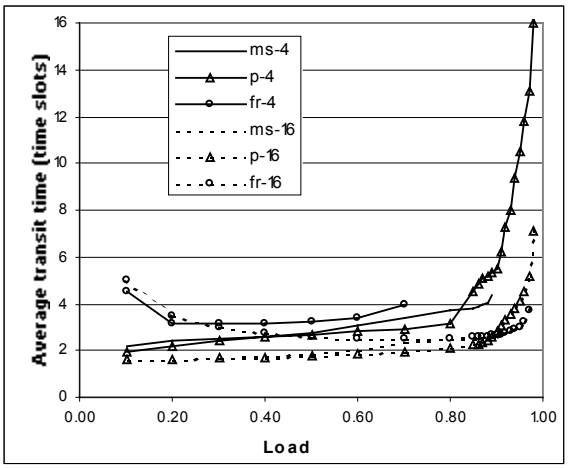

Figure 6. Average transit time (number of slots) as a function of the load for polarized traffic (factor 2.0)

Therefore, PDRR proves to be the best combination, since it maintains the highest load even under highly polarized traffic and low number of servers. It also handles low traffic well and provides an acceptable average transit time. Its request generation is straightforward and the implementation makes use of simple round robin arbiters, as described in the next section. However, it requires the management of priorities in the computation of grants.

MultiSLIP is the second best of this test. Its maximum load is a little limited under polarized traffic and its generation of requests requires to sum port requests and to compare the total to the number of available servers. However, its implementation can be designed fairly easily based on the experience of iSLIP.

Finally, FR-DRR is based on a new arbitration idea and provides globally acceptable results, except for the maximum throughput in polarized traffic with few servers. Also, the management of low traffic could be improved by other means.

IV. Hardware Implementation Issues

An important point for designing fast scheduling algorithms is simple hardware implementation possibilities. The implementation of our three proposed scheduling algorithms can be based on the architecture shown in Figure 7, assuming there is only one arbiter at each input/output port (i.e. port arbiters, not server arbiters). 2N

arbiters are used for this architecture. An N^2-byte wide vector is needed to record request status of VOQs. In order to find the maximum match, multiple iterations may be required to achieve final decisions for both RG and RGA.

For RG strategy, the request arbiters select multiple requests among the contending VOQs and pass them to grant arbiters, where grant decisions are made. If there still are available servers or ungranted requests, the previous grant decisions are fed back to request arbiters and another iteration is started, provided the maximum number is not reached. Otherwise, the final decisions are sent to configure the switch matrix and set the input/output ports and their servers. Similarly, for RGA, the grant arbiters grant multiple requests among requests from all VOQs and pass them to accept arbiters, where accept decisions are made. If there still are available servers or ungranted requests, the previous accept decisions are fed back to grant arbiters to possibly start another iteration. Otherwise, the final decisions are sent out. Therefore, each request/grant arbiter should maintain a matrix of requests and update it before each iteration. The decision register should maintain a matrix of server status and update it after each iteration.

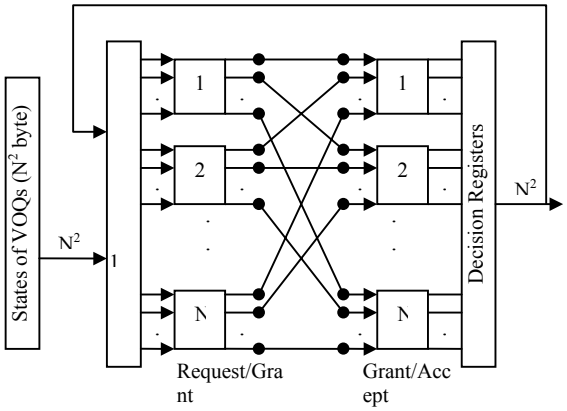

Figure 7. Block diagram of the switch scheduler

The design of arbiters varies for the different scheduling algorithms. For MultiSlip and PDRR, round-robin arbiters are employed, which can be implemented by programmable priority encoders [5]. Each arbiter should have its own register for accepts "a_i"and grants "g_i". In PDRR, the priority of each request must be filed in the request vector. For FR-DRR, round robin arbiters are used at the grant side; at the request side, a linked list based on RAM is proposed to be an ideal structure for a flexible ring.

V. Conclusion

We have presented three scheduler solutions that are able to handle a multiple-server architecture. This kind of architecture is likely to be more and more widely used for optical burst packet networks, since the speed of optical components increases faster than the speed of electronic components. Parallel data access in the electronic components will be required to fully exploit the capacity of the new optical matrices. We extended two concepts that already existed in the classical single-server architecture (iSLIP and DRR) and proposed a new arbitration scheme, the Flexible Ring, based on a linked list. Because of the aggregation of cells into OBPs, an additional issue is the bandwidth loss due to non-full OBPs. An appropriate request generation procedure has to be proposed so as to maximize the filling ratio while respecting the time-out specification of the router. We proposed a different request generation procedure for each scheduler. In all cases, the idea is to give priority to urgent packets, then to full packets, then to the others. In the switch fabric, pathological situations can be modeled with geometrically polarized traffic.

We have compared the performance by simulation of MultiSLIP (multiple-server iSLIP), PDRR (Prioritized Dual Round Robin) and FR-DRR (Flexible Ring Dual Round Robin) with different architecture configurations and different polarization factors. All three scheduler solutions prove to be good candidates, but FR-DRR would need some improvement in the case of high polarization and small number of servers. The best solution is PDRR, both in terms of theoretical performance and practical hardware implementation. It is an iterative scheduling algorithm, based on the RG strategy, with independent port round robin arbiters that make sequential selections for each unmatched server within each iteration. It reaches very high throughputs even under highly polarized traffic and small number of servers, and maintains a low average transit time.

Further work

We are expanding further PDRR so as to deal with the QoS issues related to the next generation Internet. Also, we are investigating alternative (and implementable) arbitration schemes that are well adapted to the new multiple-server architectures. They involve both the design of atomic arbiters and the interaction scheme between a group of arbiters.

Acknowledgements

We would like to thank Francesco Masetti, leader of the Core Routing project in the Research and Innovation department of Alcatel, for his support to this paper and our study.

References

[1] M. Hluchyj and M. Karol, "Queueing in High-Performance Packet Switching", IEEE Journal on Selected Areas in Communications, Vol. 6, No.9, December 1988, pp. 1587-1597.
[2] N. McKeown, "Fast Switched Backplane for a Gigabit Switched router", White Paper, http://www.cisco.com/.
[3] Y. Xiong, H. C. Cankaya, M. Vandenhoute, "Performance of Optical Routers in Burst-Switched WDM Networks", Proceedings of Networld+Interop 2000 Conference, May 2000.
[4] F. Masetti et al., "The Perspective of Optical Packet Switching in IP-Dominant Backbone and Metropolitan Netwoks", IEEE Communications Magazine, March 2001.
[5] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches", IEEE/ACM Transactions on Networking, Vol.7, No.2, April 1992.
[6] J. Blanton, H. Badt, G. Damm, and P. Golla, "Iterative Scheduling Algorithms for Optical Packet Switches", unpublished, submitted to IEEE Communications Letters.
[7] J. Chao, "Saturn: A Terabit Packet Switch Using Dual Round-Robin", IEEE Communications Magazine, December 2000, pp. 78-84.
[8] J. Blanton, H. Badt, G. Damm, and P. Golla, "Impact of Polarized Traffic on Scheduling Algorithms for High Speed Optical Switches", ITCom 2001, Denver, August 2001.