Multicast Support for a Storage Area Network Switch

Multicast Support for a Storage Area Network Switch

Andrea Bianco, Paolo Giaccone, Enrico Maria Giraudo,
Fabio Neri, Enrico Schiattarella
Dipartimento di Elettronica - Politecnico di Torino - Italy
e-mails: {bianco, giaccone, giraudo, neri, schiattarella@mail.tlc.polito.it}

***Abstract*— Efficient support of multicast traffic in Storage Area Networks (SANs) enables applications such as remote data replication and distributed multimedia systems, in which a server must access concurrently multiple storage devices or, conversely, multiple servers must access data on a single device. In this paper we extend an innovative switching architecture, proposed in a previous paper, to support multicast traffic. We describe the most important aspects, focusing in particular on the mechanisms that permit to achieve lossless behavior. We then use simulation to analyze system performance and the impact of such mechanisms under various traffic patterns. Although the work is inspired by a specific switch architecture, results have a more general flavor and permit to highlight interesting trends in flow controlled architectures.**

I. Introduction

The deployment of large data-centers, comprising hundreds or thousands of servers has imposed a reorganization of storage resources and stimulated an evolution of I/O interfaces. Dedicated bus connections between servers and storage resources have been replaced by high-speed packet-switched networks, known as Storage Area Networks (SANs). Storage networking overcomes the performance, scalability, reliability and management problems posed by the traditional Directly Attached Storage (DAS) paradigm and enables consolidation and virtualization of storage resources.

Most SANs are implemented using Fibre Channel technology [1], [2], specifically designed for the interconnection of computing peripherals and able to satisfy the demanding requirements of storage traffic. Fibre Channel standards prescribe loss-free operation and provide buffer-to-buffer and end-to-end flow-control mechanisms that allow a node to control the rate at which it receives frames. SAN switches can use such mechanisms to regulate incoming frames, but in general are not allowed to drop or reorder frames.

Multicast support in SANs enables critical applications such as disaster recovery, in which a server stores multiple copies of the same data at geographically distant sites (similar to RAID-1 mode), and distributed multimedia systems [3], in which multiple servers access data (typically video streams) stored in a central repository and deliver it to their local pool of users [4].

In this paper we present a switch architecture designed for Fibre Channel SANs, and study its performance under multicast traffic. The architecture was previously introduced in [5]: it employs centralized arbitration, backpressure and buffer management techniques to achieve lossless behavior while isolating congesting flows from non-congesting ones. Another highly-distinctive feature of the switch is its fully *asynchronous* design, that provides significant advantages in terms of scalability, cost and simplicity [6]. Traditional input-queued switches operate in a *synchronous* fashion: time is divided in intervals of equal size called *timeslots* and modules (line-cards, fabric, scheduler) have a common time reference. Variable-size packets are segmented into fixed-size data units called *cells*, transferred through the switching fabric within a timeslot and reassembled at the output line-cards. In an asynchronous switch, on the contrary, the line-cards and the fabric run on independent clock domains. It is not necessary to distribute a global clock signal (a task that can be especially problematic when the system is distributed over multiple racks) and to synchronize transmission through the switching fabric. Variable-length packets can be supported natively, without the need for segmentation and reassembly buffers. Moreover, fabric arbitration is simplified because output contentions can be solved independently, without employing complex centralized scheduling algorithms. The price to pay for these advantages is additional buffering in the fabric and a small speed-up to mitigate the effects of Head-of-the-Line blocking [7].

The paper is organized as follows: in Section II we summarize previous work on multicast scheduling, in Section III we recall the switch architecture and extend it to support multicast traffic; in Section IV we study system performance under various traffic patterns, highlighting important effects of flow-control on performance; finally, we conclude in Section V.

II. Previous Work

Multicast scheduling in packet-switches is an important topic that has been researched extensively. In [8]–[10] the authors study the problem from a theoretical point of view, trying to analyze performance and characterize the optimal scheduling policies under various architectural and traffic assumptions. Practical scheduling algorithms, with various trade-offs between performance, complexity and fairness have been proposed in [11]. These works, however, are not applicable to our architecture, because they are all aimed to synchronous switches.

In [12] the authors develop a multicast scheduling algorithm for a switch employing a buffered crossbar, a fabric

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

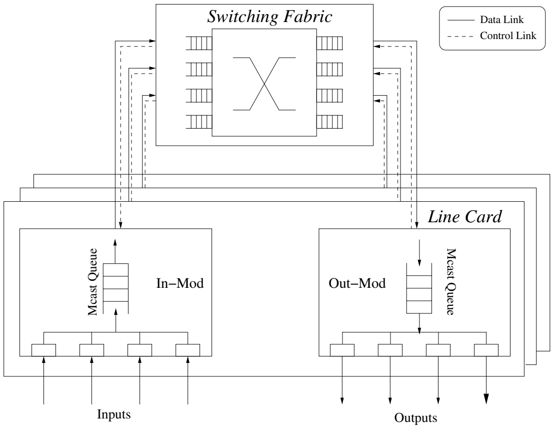

Fig. 1. Reference architecture

particularly suitable for asynchronous switches [6]. However, they still assume synchronous operations, so their reference architecture does not provide all the advantages highlighted before.

To the best of our knowledge, this is the first work to address the performance of an asynchronous switch under multicast traffic.

III. System Architecture

The switch is composed by a number of line-cards interconnected by a switching fabric (Figure 1). Line-cards receive packets on input ports, store them in buffers and multiplex them on the uplinks towards the switching fabric. The switching fabric transfers packets from the inputs to the outputs and transmits them on the downlinks, to the line-cards hosting the destination output ports. The bandwidth of the uplink and the downlink is equal to the sum of the bandwidths of the input/output ports on a line-card, i.e. links are not oversubscribed nor constitute a bottleneck.

To minimize the load on the uplink, a single copy of each multicast packet is transmitted from the line-card to the switching fabric, and replication to multiple line-cards is performed inside the switching fabric. Conversely, to reduce the load on the downlink, a single copy of the packet is sent from the switching fabric to each line-card, and replication to multiple output ports is performed independently on each line-card.

A. Line-cards

Line-cards contain two separate buffering stages for multicast packets entering and exiting the switch, named respectively *In-module* and *Out-module*.

The In-module contains a memory where multicast packets entering input ports are stored. The memory is organized as a single FIFO queue. The total queue capacity is statically partitioned among the input ports. An input port can enqueue new packets only if space is available in its share of the queue, so no input port can hog the whole buffer. Each position in the queue is dimensioned for a maximum size packet and, if a smaller packet is enqueued, the residual part of that memory portion remains unusable. This choice potentially results in inefficient use of the In-module memory, but allows the buffer management policy to be implemented very easily, using just one counter for each input port. The inefficiencies in general are not an issue, because line-cards can host moderately large amounts of memory.

The Out-module stores multicast packets received from the switching fabric in a buffer organized as a single FIFO queue. This very fast memory is accessed “per-byte”, hence the number of available positions depends on the size of enqueued packets. When a packet reaches the head of the Out-module queue, it is replicated to all output ports it is destined to, and dequeued.

B. Switching fabric

The switching fabric consists of a crossbar with a small single high-speed FIFO queue at each input and output. The crossbar has a moderate internal speed-up ($K = 3$) to mitigate the effects of Head-of-the-line (HOL) blocking; hence, fabric output queues are larger than fabric input queues to sustain temporary overload conditions. Both input and output queues are accessed per-byte to maximize space efficiency.

Each fabric output has a *fabric arbiter* that controls access from fabric inputs. When an input wants to be connected to an output, it sends a request to the corresponding fabric arbiter. In case multiple inputs request the same output, the fabric arbiter resolves the contention according to a round-robin policy.

The crossbar is equipped with internal multicasting capability, meaning that it can replicate a packet to multiple outputs at the same time with no extra cost. By using this feature it is possible to reduce packet delays and fabric input queues occupancy; however, doing so requires multiple outputs to be free at the same time. Waiting to gain access to all the intended outputs before transmitting a packet can be counterproductive, because it forces outputs that have already granted access to stay idle while the others become free.

To exploit the benefits of crossbar replication without compromising efficient usage of output ports, fabric inputs transmit packets in two phases:

1) the input requests all the outputs included in the fanout set of the packet at the head of the fabric queue and sends it “in a single shot” to all the outputs that grant immediately;
2) afterwards, the input individually sends individual copies of the packet to the remaining outputs as soon as they grant access.

C. Flow-Control mechanisms

To support lossless delivery, the switch must be endowed with internal flow-control mechanisms that regulate access to switch buffers and prevent overflow. The simplest form of flow-control is *backpressure*. When the occupancy of a buffer reaches a certain threshold, backpressure blocks packet transmission from previous buffering stages. When occupancy decreases, the signal is deactivated and transmission can restart. In case of persistent congestion, all the buffers in the data path eventually fill-up and the backpressure signal propagates back to the source(s).

In our system, four backpressure signals are available:

1) from the Out-module to the fabric output queues;
2) internally to the fabric, from the fabric output queues to the fabric input queues;
3) from the fabric input queues to the In-module;
4) from the In-module to the input ports.

Backpressure is sufficient by itself to prevent packet losses. Its main drawback is that it is not selective, i.e. it blocks all flows, even those which are not responsible for congestion. In [5] we have illustrated the benefits achieved by controlling individually unicast flows with centralized arbitration. The same result cannot be obtained for multicast, because the number of possible flows traversing the switch grows exponentially (rather than quadratically) with the number of ports N. This implies that no switch resource can be assigned per-flow. In particular, both on the ingress and egress side of line-cards packets are stored in a single FIFO queue, regardless of their fanout set. As a consequence, an internal flow-control mechanism, capable of providing differentiated services to multicast flows and isolate congesting ones is not available.

IV. Performance study

We have developed a discrete-event simulator modeling the architecture described in the previous section. We study system performance under multicast traffic when backpressure mechanisms are enabled and disabled. When backpressure is off, whenever a packet cannot be enqueued it is simply discarded. This behavior is not realistic in a SAN environment, but the results that we obtain under this assumption are useful to understand the impact of backpressure on system dynamics.

A. Simulation settings

The simulated system is a 16×16 switch with 4 line-cards (N_{LC}) hosting 4 input/output ports (P_{LC}) each. Each port runs at 10 Gbps, hence the speed of the up/downlink is 40 Gb/s and the aggregate bandwidth is 160 Gbps. The In-module shared buffer is able to contain 8000 maximum-size packets, corresponding to 16 MB of memory, while the size of the Out-module queue is 320 KB. Finally, the size of fabric input and output queues is 10 KB and 20 KB (per port) respectively.

In our experiments we assume that output ports absorb traffic at line-rate, i.e. they do not receive blocking signal from downstream devices (end-nodes or other switches).

B. Traffic model

In all experiments, three packet size distributions have been considered: minimum size (80 bytes) only, maximum size (2000 bytes) only, and uniform between 80 and 2000 bytes, with 40 bytes increments. Each active source emits a packet with probability ρ_{in}, $0 \leq \rho_{in} \leq 1$ and with probability $1 - \rho_{in}$ remains idle for a period with the same distribution of the packet duration, which can be fixed (minimum or maximum size packets only) or variable (packet size uniformly distributed).

If the backpressure signal from the In-module is active, generation of new packets is blocked. As soon as the backpressure signal is deactivated, the source can start generating again. Thus, the effective average input load generated by a source is $\tilde{\rho_{in}} \leq \rho_{in}$. We neglect the fact that when a source is blocked by backpressure, packets *generated* but not *emitted* accumulate. Taking it into account would introduce a perturbation of the input load and complicate throughput analysis, especially in overload conditions.

The average offered load to an output port is ρ_{out} and it is equal to the sum of the ρ_{in} of input ports transmitting to that output times the probability of selecting that output. If $\rho_{out} > 1$, traffic is not admissible and the output port is overloaded. As we have assumed that output ports can always drain packets at line rate, the replication of a multicast packet to multiple output ports on the same line-card does not have any effect on throughput. Thus, in all the traffic patterns we have selected, the destination ports in the fanout set always reside on different line-cards. If the fanout of a packet is F, then it must be replicated to F line-cards.

C. Broadcast traffic scenario

We first present results obtained in a “broadcast” scenario, in which all input ports are active and transmit packets to four output ports on four different line-cards. As each output port receives packets from 4 input ports, $\rho_{out} = 4 \times \rho_{in}$ and traffic is admissible if $\rho_{in} \leq 0.25$. By setting $\rho_{in} > 0.25$ we can generate non-admissible traffic load and observe how the system behaves in overloading conditions.

Throughput vs. offered output load curves are shown in Figure 2. On the top of the graphs the corresponding input load is also indicated.

When backpressure is enabled, throughput closely matches the offered load and saturates to 100% for $\rho_{in} = 0.25$. When backpressure is not used, on the contrary, the system starts experiencing losses when the offered output load grows beyond 90%. This is due to the fact that the burstiness of the traffic entering the fabric can cause the fabric output queues to temporarily saturate even in under-load conditions. Packets that reach a full fabric output queue are simply discarded and throughput decreases. In overload conditions throughput slowly grows to 100%, as packets in excess compensate for those that are discarded. Note that the phenomenon is more evident for maximum-size packets, when the queues can host fewer packets, but it is present also when packet size is variable.

D. “Residue” traffic pattern

We now consider traffic patterns known to be particularly critical for input-queued switches [10]. These patterns are composed by packets that have a small fanout, yet generate a high number of output contentions. It is thus possible to impose a high packet injection rate without violating the admissibility condition (thanks to the small fanout) and, at the same time, stress the switching fabric (due to the high

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

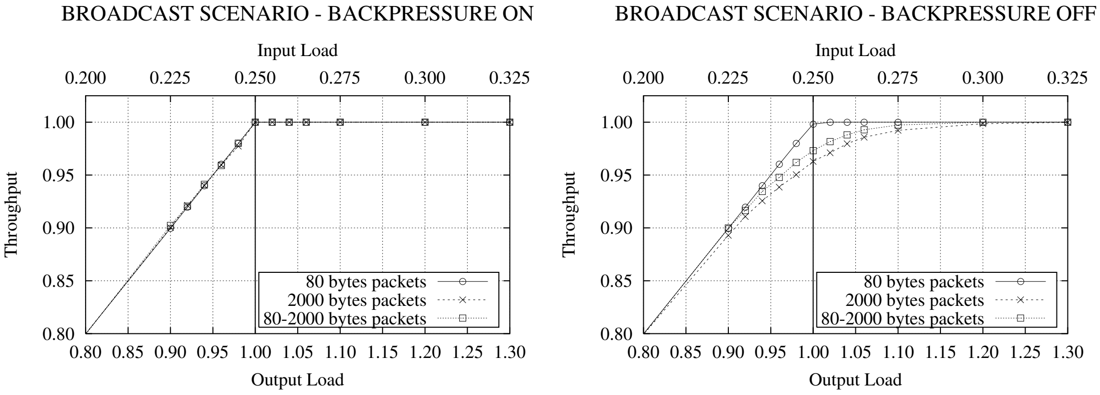

Fig. 2. Throughput vs. offered load (broadcast scenario)

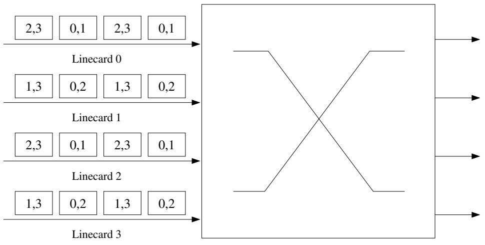

Fig. 3. "Residue" multicast traffic pattern with fanout 2, from the fabric point of view. Numbers in boxes represent destination line-card.

Input Load	Output Load	Throughput	Average service	Xbar Queue	
				In's	Out's
0.40	0.80	0.800	1.471	4.3 %	16.9 %
0.45	0.90	0.899	1.374	7.1 %	29.6 %
0.47	0.94	0.940	1.266	17.4 %	42.8 %
0.49	0.98	0.973	1.041	86.6 %	64.9 %
0.50	1.00	0.973	1.040	86.6 %	64.9 %
0.55	1.10	0.950	1.059	86.9 %	60.8 %
0.60	1.20	0.916	1.067	87.3 %	56.0 %

TABLE I
PERFORMANCE RESULTS WITH "RESIDUE" PATTERN (FANOUT 2, BACKPRESSURE ON, 2000 BYTES PACKETS)

number of contentions). As fabric speed-up may not be sufficient to accommodate all contending packets, some of them receive partial service, i.e. a *residue* is left at the fabric input queue. For this reason, we name this kind of traffic patterns "Residue".

The first pattern we consider, seen from the fabric point of view, is depicted in Figure 3. Packets coming from a line-card always contend with at least two packets from other line-cards. For instance, packets coming from $LC\,0$ always have one conflict with packets coming from $LC\,1$ and $LC\,3$ and one conflict on average with packets coming from $LC\,2$. Each line-card has four active input ports and each output port is loaded by two inputs, so traffic is admissible if $\rho_{in} \leq 0.5$.

Figure 4 shows throughput vs. offered load curves when backpressure mechanisms are set ON or OFF. When backpressure mechanisms are OFF, system performance is similar to that obtained in the previous scenario, both in under-load and in overload conditions. When backpressure mechanisms are ON, on the contrary, significant differences can be observed. System throughput is close to ideal when the output offered load is less than ~ 0.96, but at that point it stops growing and actually starts decreasing; then, as it approaches its maximum value ($\rho_{out} = 2.0$) it increases again. Throughput loss is especially evident when maximum size packets are used but, is significant also when packet size is uniformly distributed. With minimum size packets, on the contrary, no loss is experienced.

Table I reports fabric queues occupancy when maximum size packets are used. We can see that input queues occupancy grows rapidly as ρ_{out} approaches 0.96 and saturates to $\sim 87\%$. Output queues occupancy, on the contrary, reaches its maximum at 0.96 and steadily decreases afterward. To understand this behavior, we must focus on what happens at the In-modules. If the average fabric input queues occupancy is high, In-modules are subject to backpressure very often, and they fill up as well. When the In-module memory is almost full, backpressure towards the sources is activated. As the In-module memory is statically partitioned, each source enters and exits backpressure individually; in particular, sources that recently have generated more aggressively enter backpressure earlier. When a source experiences backpressure, the process of packets entering the line-card changes. Consider, for instance, $LC\,0$: port (0,0) and (0,2) generate only packets destined to line-cards $\{0, 1\}$, whereas ports (0,1) and (0,3) generate only packets destined to line-cards $\{2, 3\}$. If all ports on the line-card are active, on average half of the enqueued packets are destined to line-cards $\{0, 1\}$ and the other half to line-cards $\{2, 3\}$. Besides, they are roughly alternated, because all sources generate uncorrelated packets. On the contrary, if a $\{0, 1\}$ source is blocked, more packets destined to line-cards $\{2, 3\}$ are enqueued than packets aimed at line-cards $\{0, 1\}$. It can even happen that both of $\{0, 1\}$ sources are backpressured at the same time and a long burst of $\{2, 3\}$ packets enter the In-module and ultimately reach the fabric input queues. Figure 5 shows a trace of maximum-size packets entering the In-module queue on $LC\,0$ at low and high load over 50000 timeslots. The high load graph displays ~ 330 packets, and the bursts are approximately 30 packets long.

Burstiness naturally leads to performance penalties. Con-

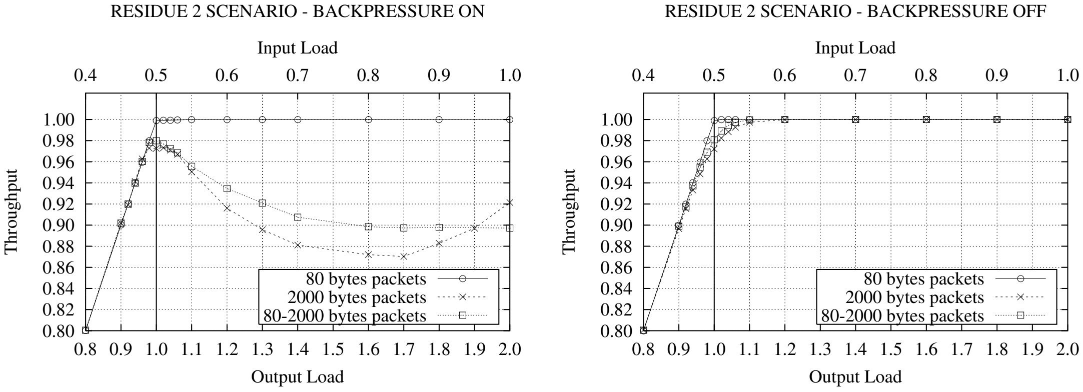

Fig. 4. Throughout vs. offered load (Residue pattern, fanout 2)

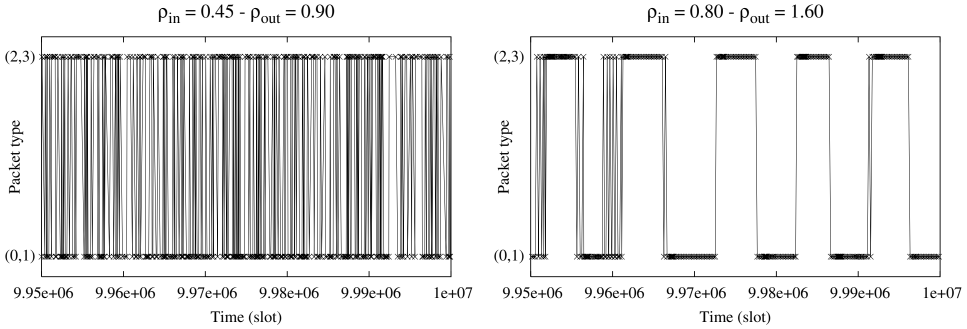

Fig. 5. Time traces of maximum-size packets entering the fabric input 0, at low load (left) and high load (right)

flicting bursts in the fabric input queues prevent efficient usage of the crossbar switching capacity. Some line-card may not receive packets for long periods, despite the fact that many packets destined to them are present in the queue. This is a form of head-of-the-line blocking due to the usage of a single queue for multicast traffic. Notice that this phenomenon is self-sustaining: a source that enters backpressure may remain blocked for a long time if large bursts of packets from different sources are present ahead in the In-module queue. The hysteresis mechanism used to activate and deactivate backpressure towards sources further facilitates this phenomenon: a blocked port cannot transmit until a minimum number of packets belonging to it are removed from the In-module queue. When small packets are used, the In-module queue is drained much faster *in terms of number of packets per unit of time*, so sources remain blocked for shorter periods of time and long bursts do not form.

Finally, as ρ_{in} approaches 1, sources tend to synchronize (because they all generate equal size packets almost back-to-back), so they enter and exit backpressure at the same time and burst length decreases. A corresponding throughput increase is visible in Figure 4 for $\rho_{out} \geq 1.7$. If variable size packets are used, sources do not synchronize and no throughput improvement is observed.

To evaluate system performance under the Residue traffic pattern but without the induced burstiness, all sources on a line-card generate with equal probability both kind of packets. For example, all sources transmitting from $LC\,0$ generate with equal probability packets destined to line cards $\{0, 1\}$ and packets destined to line cards $\{2, 3\}$. With this “modified” pattern we make sure that bursts do not form regardless of how many sources are experiencing backpressure at any time. In this scenario, throughput still reaches its maximum value for $\rho_{out} \simeq 0.96$, but it does not decrease afterward. Correspondingly, fabric output queues occupancy grows up to 63% and remains at that level for $\rho_{out} > 0.96$. Switch performance is satisfactory: despite the hardness of the traffic pattern, maximum throughput loss is 5% for 2000 bytes packets and 4% for variable size packets.

The same set of experiments, performed with a “Residue 3” traffic pattern, in which flows have fanout 3 and each packet has at least two conflicts with packets coming from any other line-card, confirm our analysis. We still observe throughput loss in the overload region, however it is much less evident, because whenever a packet is served by the fabric, it feeds at least three output queues. With the “modified” traffic pattern, when all sources on a line-card generate with equal probability all the packets, the phenomenon disappears.

E. *Uniform traffic pattern*

In this section we analyze system performance under uniform traffic, i.e. when the fanout set of each packet is chosen randomly and independently over the set of all possible fanout sets. As noted in Section IV-B, the replication of a packet to

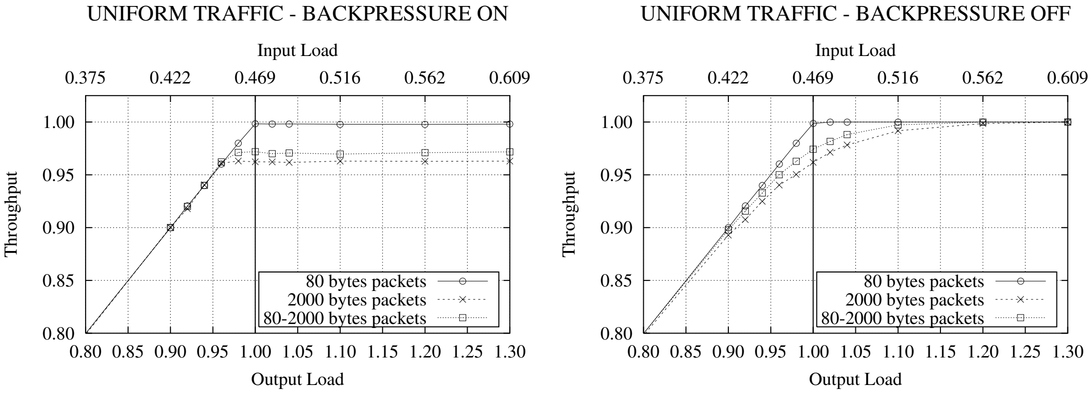

Fig. 6. Throughout vs. offered load (uniform traffic)

multiple ports on the same line-card does not affect system performance. Hence, we simplify the pattern by forcing each packet to be destined to at most one port on each line-card.

As in our configuration $N_{LC} = 4$, every port on a line-card can generate a total of $2^{N_{LC}} - 1 = 15$ packets. Of these, 4 have fanout 1 (unicast), 6 have fanout 2, 4 have fanout 3 and 1 has fanout 4 (broadcast). This corresponds to an average fanout of $32/15$ and traffic is admissible for $0 \leq \rho_{in} \leq 15/32 = 0.46875$.

Figure 6 shows throughput vs. offered load when backpressure mechanisms are enabled and disabled.

When backpressure mechanisms are OFF, performance is similar to that observed in previous traffic scenarios. The system experiences losses as the offered load grows beyond 90%, causing throughput reduction of a few percentage points for maximum- and variable-size packets; however, in the overload region, packets in excess compensate for the losses and throughput grows to 100% for any packet size distribution.

When backpressure is enabled, throughput tracks offered load up to $\rho_{out} = 0.96$ and then saturates to 96% for 2 Kbytes packets and 97% for variable size packets; in the overloading region no reductions are observed, because packets are uncorrelated and backpressure does not originate bursts. Throughput saturation is due to a new form of HOL blocking caused by backpressure: a packet destined to one or more fabric outputs whose queues are full is blocked until space becomes available. All the other packets in the input queue cannot be served (because the queue is FIFO) and some fabric outputs may remain idle even if packets destined to them are enqueued. The available speed-up in the switching fabric addresses HOL blocking caused by output contention but does not help when HOL blocking is caused by backpressure, as previously discussed in [5]. Throughput reduction is more marked when maximum-size packets are used, because the output queues can host a smaller number of packets and so activate backpressure more frequently.

V. Conclusions and future works

We have presented an innovative switching architecture specifically designed for SANs and we have studied its performance by means of simulation. Results show that the system achieves satisfactory performance under various multicast traffic patterns and for various distributions of packet sizes. The backpressure mechanisms used to achieve lossless behavior do not degrade system throughput, except under a particularly challenging traffic pattern and in overloading conditions.

The results of this paper confirm that this architecture is suitable for today data-centers, where high-performance and scalable storage switches are required. In the future we plan to further improve it, investigating viable solutions to provide per-flow control of multicast traffic, employing a reasonable number of queues and an implementable arbiter.

References

[1] ANSI INCITS 373-2003, “Fibre Channel - Framing and Signaling (FC-FS)”, rev. 1.90, April 2003.

[2] R. W. Kembel and R. Cummings, “Fibre Channel: A Comprehensive Introduction”, Northwest Learning Associates, Tucson, AZ, 2000

[3] V. O. K. Li and W. Liao, “Distributed Multimedia Systems”, Proc. IEEE, vol. 85, No. 7, pp 1063-1108, July 1997.

[4] S.-H. Gary Chan and F. Tobagi, *Distributed Servers Architecture for Networked Video Services*, IEEE/ACM Trans. Networking, Vol. 9, No. 2, pp. 125-136, April 2001.

[5] A. Bianco, P. Giaccone, E. M. Giraudo, F. Neri, E. Schiattarella, *Performance Analysis of Storage Area Network Switches*, IEEE Workshop on High Performance Switching and Routing (HPSR 2005), Hong Kong, May 2005.

[6] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, N. Chrysos, *Variable packet size buffered crossbar (CICQ) switches*, IEEE International Conference on Communications (ICC 2004), Paris, France, June 2004.

[7] M. Karol, M. Hluchyj, S. Morgan, “Input versus output queueing on a space division switch”, *IEEE Trans. Commun.*,

[8] J. Hui, T. Renner, “Queueing analysis for multicast packet switching”, *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 723-731, Feb./Mar./Apr. 1994.

[9] Z. Liu, R. Righter, “Scheduling multicast input-queued switches”, *J. Scheduling*, vol. 2, pp. 99-114, 1999.

[10] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, “Multicast Traffic in Input-Queued Switches: Optimal Scheduling and Maximum Throughput”, *IEEE/ACM Trans. Networking*, Vol. 11, No. 3, pp. 465-477, June 2003.

[11] B. Prabhakar, N. McKeown, R. Ahuja, “Multicast scheduling for input-queued switches”, *IEEE J. Sel. Areas Commun.*, vol. 15, no. 5, pp. 855-866, Jun. 1997.

[12] L. Mhamdi and M. Hamdi, *Scheduling Multicast Traffic in Internally Buffered Crossbar Switches*, IEEE International Conference on Communications, ICC’04. Paris, June 2004.