Analysis and Simulation of a Parallel Packet Switch for Satellite On-board Switching

ANALYSIS AND SIMULATION OF A PARALLEL PACKET SWITCH FOR SATELLITE ON-BOARD SWITCHING

Guido Albertengo – Politecnico di Torino – Electronics Department
Corso Duca degli Abruzzi 24 – 10129 Torino – Italy
Phone: +390112276604 – Fax: +390115644099 – E-mail: guido.albertengo@polito.it
Diego Borsetti – Politecnico di Torino – Electronics Department
Corso Duca degli Abruzzi 24 – 10129 Torino – Italy
Phone: +390115644000 – Fax: +390115644099 – E-mail: diego.borsetti@polito.it

Abstract

In this paper we consider a packet switching system composed of *X* parallel switching planes operating independently and at a speed lower than the input lines. Arriving traffic is segmented into fixed length cells, then each cell is sent to one of the *X* planes, where it is switched to the correct output port and finally recombined with the other cells, coming from other planes, to reconstruct the original packet. This architecture, originally proposed by Iyer and McKeown [1], is referred to as a Parallel Packet Switch (PPS) and allows to design a switching fabric operating at a fraction of the line rate *R*. A PPS, with planes operating at rate *r*, must have at least *k=R/r* planes to avoid systematic packet losses. In [1] it was proved that a PPS can emulate the behavior of an Output Queue Switch (OQS) with the same buffering capabilities and the same number of ports. However, the centralized scheduling algorithm required to achieve this result can not be easily implemented in hardware, due to its complexity.
In this paper we propose a Redundant Parallel Packet Switch (RePPS), i.e. a PPS with more than *k* planes, with a distributed scheduling algorithm, and multiplexing/demultiplexing stages without coordination buffers, which is a fair trade-off between performance and complexity. In particular we show that the minimum number $n = X - k$ of redundant planes required to emulate an OQS with FIFO policy under any incoming traffic type is $n = k^2-2k+1$. The distributed scheduling algorithm, which is the key component of the proposed switch, is presented and its performance, analyzed thru simulation, is discussed for a realistic fabric with a limited number of redundant planes. The results so far obtained suggest a possible application of this architecture for satellite on-board packet switches.

Introduction

Satellite On-Board switch design has always been a tough technical and technological challenge due to the limitation in speed, complexity and power consumption faced by the designers. Another critical issue of these systems is their reliability and thus their expected lifetime: a terrestrial packet switch can be repaired within a few hours from its failure, an on-board switch should operate for several years (usually some 12-15 years are required) automatically replacing failed parts.
The classical approach to cope with these requirements is to use highly reliable components and duplicate the switching fabric: the resulting cost/performance ratio is very high, when compared to the same figure of a terrestrial switch, but is still reasonable when compared to the overall cost of a satellite. However, the increase in performance required to these switches, combined with the pressure from the operators to reduce the satellite cost, could make the current design rules out-of-date very soon.
In our lab, we are studying non conventional architectures that combine good performance with low cost. One of them is the so called Parallel Packet Switch (PPS), originally proposed by Iyer and McKeown [1] and later studied by several scientists [2][3][4] for terrestrial applications. This fabric, originally devised to overcome the gap between the data rates on input and output optical fiber links and the speed of the electronic switching circuits, was very soon abandoned due to the complexity of its control. However, for satellite on-board switches, where reliability (and therefore life expectancy) is a key factor, the PPS, or another fabric derived from it, could be effectively used to implement a reliable on-board switching system, providing that a simpler control scheme could be devised. This is actually the subject of this paper, which is organized as follows: Section 1 describes the original PPS fabric and the Redundant Parallel Packet Switch (RePPS) fabric we propose. In this section are also introduced the notations used throughout this paper. In Section 2, the multiplexing algorithm required to recombine cells into packets is briefly described, while in Section 3 the more complex

demultiplexing algorithm is presented. These two algorithms together implement the control scheme of the switch. Section 4 reports some performance figures of the proposed fabric, obtained by simulation. Finally, Section 5 summarizes the results so far achieved and concludes the paper.

1 The Switching Fabric

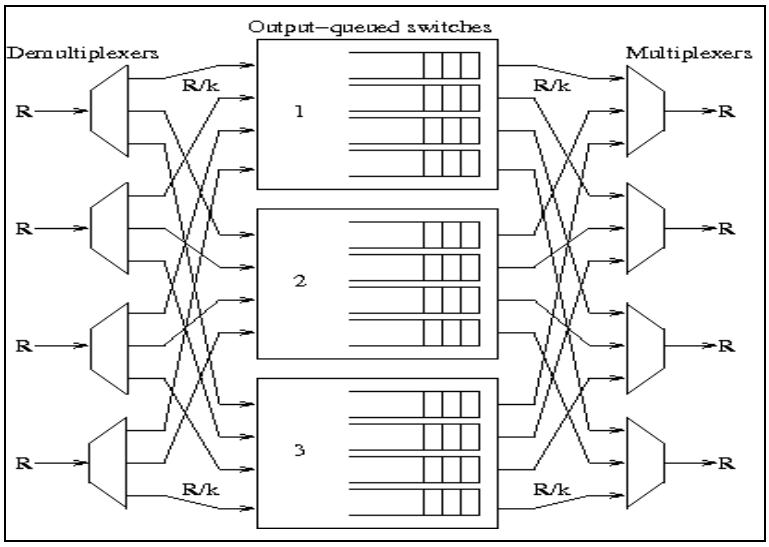

Figure 1 - The Parallel Packet Switch structure (k=3)

The PPS, whose structure is shown in Figure 1 is composed of three stages: the Central Stage (CS) made of *X* identical Output Queue Switches (OQS) operating independently and in parallel; the Input Demultiplexers (IDX) which split incoming packets into fixed size units and distribute them among the available CS planes; and the Output Multiplexers (OMX) which get the cells from the planes and reconstruct the packets. It is worth to notice that this fabric is similar to the well known three stage Clos switch, a strictly non-blocking circuit switch. This similarity has been exploited to derive some constraints to be fulfilled to make the behavior of a PPS identical to an OQS with FIFO output policy.
Always referring to Figure 1, it is worth to recall the following notations, which will be used throughout this paper:

- *N* number of inputs and outputs of the switch
- *X* number of planes in the Central Stage of the switch
- *R* external link data rate
- *r* internal link data rate
- *k=R/r* ratio between the external and internal rate

Notice that both multiplexers and demultiplexers are assumed to be without buffering capabilities, since this functionality is provided by the switching planes in the CS, and that from the outside a PPS looks like a *NxN* fabric operating at rate *R*.
Let us now introduce some definitions used in this paper:

Time-slot T_S: the time taken to transmit or receive a fixed length cell of size C_S at a link rate *R:*

$$T_S = C_S / R$$

Internal time-slot T_{SI}: the time taken to transmit or receive a fixed length cell of size C_S at rate *r=R/k:*

$$T_{SI} = C_S / r = k\, T_S$$

Shadow switch: a OQS with output FIFO policy, with the same number of input and output ports as the PPS. It is known that this fabric offer the best achievable performance [5]. It will be used as a reference in simulation runs, where the same traffic sent to the PPS under exam is also fed to the shadow switch.

Relative queueing delay *Q*: the additional latency of a cell going thru the PPS with respect to the latency same cell going thru the shadow switch. It can easily be expressed as:

$$Q = T_{PPS} - T_{oq} - k\, T_S$$

Where T_{PPS} is the latency thru the PPS and T_{OQ} is the latency thru the OQ shadow switch. Notice that the reduction in speed in the Central Stage, that adds a fixed input-to-output delay in the PSS, is accounted in this formula by the last term.

Emulate: a PPS is said to emulate his shadow OQ switch if, under identical input traffic, iff

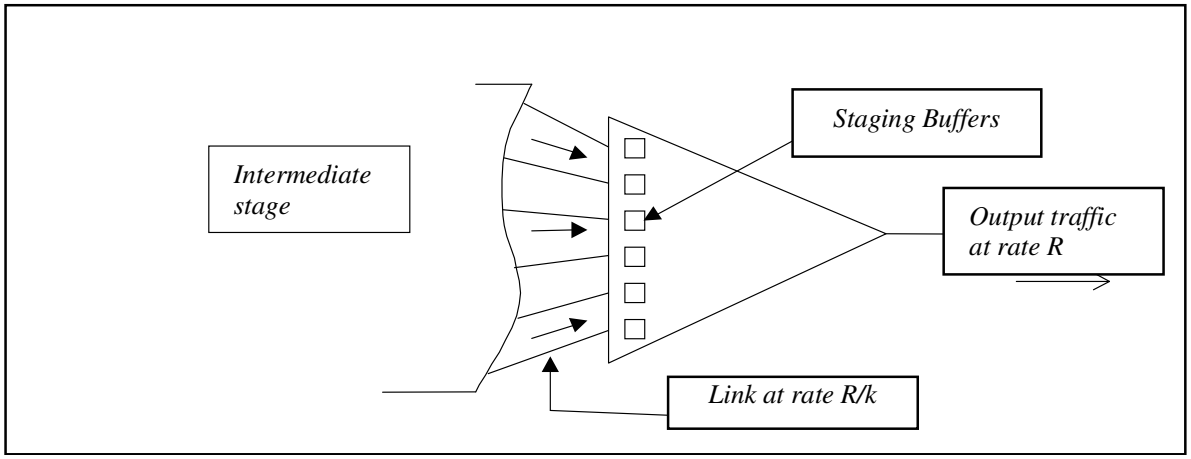

Figure 2 - The multiplexer structure

$$T_{PPS} = T_{OQ} + k\ T_S \text{ that is } Q=0$$

Notice that a PPS to emulate his shadow switch should be work-conserving and should use the same queuing policy of his shadow switch (in our case this policy is FIFO).

Redundancy degree *n*: the number of additional switching planes in the CS of a PPS above the minimum required not to loose packets due to bandwidth bottlenecks. This lower limit is $k=R/r$, so that the internal available bandwidth is no less than the external available bandwidth (this is obviously a necessary but not sufficient condition not to loose packets in the PPS).The *n* additional planes in the CS of a RePPS provide some excess bandwidth inside the fabric, which in the literature is often referred to as the switch internal space speed-up. It is worth to recall that an OQS with unlimited buffers, which has an intrinsic speed-up equal to *N*, is always lossless, i.e. it does not loose any incoming packet under arbitrary input traffic [5]. Moreover, it has also been demonstrated that a Combined Input Output Queuing (CIOQ) switch with speed-up equal to 2 perfectly emulates a OQS [6]. The RePPS architecture combines input buffering (demultiplexers), output buffering (multiplexers) and internal queuing (the switching planes in the CS). Notice that the *n* redundant planes actually provide a spatial speed-up to be exploited to increase both the switch performance and its reliability.

Allowable input link set *AIL(i,t)*: every demultiplexer is constrained to send a cell to a specific CS plane at most once every *k* time slot; since the internal links operate *k* times slower than the external input links. The allowable input link set *AIL(i,t)* is the set of planes to which a demultiplexer D_i can start sending a cell at time slot *t*.

Departure time *DT(t,i,j)*: the departure time of a packet passing thru generic queuing system is defined as the time in which the packet, after being queued, leaves the system. If the system is work conserving, it is possible to find out the departure time of each incoming packet only knowing the number of packets already buffered in the queues and the scheduling policy. Hence, the *DT(t,i,j)* of a cell arriving to input *i* of the PPS at time slot *t* is the time when the cell leaves the CS plane where it has been stored and is transmitted to the output port *j* thru multiplexer M_j. It is worth to recall that throughout this paper the scheduling policy is always FIFO.

Allowable output link set *AOL(j,DT(t,i,j))*: as explained for input demultiplexing stage, every output multiplexer is constrained to start the transmission of a cell from a plane once every *k* time slot; so that for a cell arriving at time *t*, *AOL(j,DT(t,i,j))* is the set of CS planes available to start sending a cell to multiplexer M_j at time *DT(t,i,j)*, i.e. when this cell will be ready to leave the plane where it was queued.

2 The Multiplexing Algorithm

The structure of the last stage of the PPS, i.e. the multiplexer, is shown in Figure 2. Notice that no buffers are needed, but very small size staging buffers for rate conversion between the internal link at rate *r* and the external link at rate *R*.

As already explained, when a packet arrives at the PPS, it is first divided in cells, then spread over the CS planes and finally switched to the correct output port. The multiplexer must be able to recombine the packets, cell by cell, taking into account the correct sequence order of each cell. In order to simplify the reconstruction of the original packet from its cells, a constraint is imposed on the demultiplexing stage: every incoming packet should be spread, cell by cell, in a round robin fashion. This means that if the first cell of the original packet is sent to plane *p*, the next cell will be sent to plane *p + 1*, and so on till the last cell. In Figure 3 this spreading technique is depicted.
With this constraint, the output multiplexer, while sending the last cell of a packet, must find out the plane, in the current *AOL* set, where the first cell (i.e. the *head cell*) of the next packet was stored. From now on, it will be able to get each subsequent cell of that packet and schedule their transmission in a very simple way. In order to satisfy the FIFO policy, the multiplexer shall choice, in the *AOL* set, the *head-cell* with the lowest arrival time. Note that no communication between multiplexing and demultiplexing stage is needed and, furthermore, each multiplexer can work independently and in parallel.
In the following of this paper we will assume that the multiplexing stage works as described above and we will concentrate on the demultiplexing algorithm.

3 The Demultiplexing Algorithm

First of all, it should be noticed that the first PPS stage, i.e. the demultiplexer, does not need any coordination buffer but very small staging memories to perform rate conversion. When a packet arrives at an input port, the associated demultiplexer selects an available plane in the CS to store the head cell of the incoming packet; subsequent cells will be simply stored in subsequent CS planes in a round robin fashion. Hence, choosing the plane where the head cell has to be stored in, is the only challenge faced by this switching element. As we will see in Section 4, the performance of the whole system strictly depends on this operation. Notice that the CS plane must be selected in the current *AIL* set.
Apart from selecting a proper set of planes to store the packet, another non trivial problem is to emulate, according to the definition given in Section 1, a OQS with FIFO policy. Furthermore, to keep the system complexity low, the algorithm controlling the behavior of the IDX stage should also be distributed and not centralized.
To address these issues it is useful to find out a sufficient condition on the number *X* of CS planes needed to emulate a FIFO OQ switch with a PPS. From the theorems in [1], we know that if a PPS guarantees that each cell is allocated to plane *l*, such that $l \in AIL(t,i)$ and $l \in AOL(j,IDT(t,i,j,))$, then the switch is work conserving. Now, in a *PPS* operating according to the round robin plane allocation described in Section 2, it is possible to prove that iff $X \geq k^2-k+1$ then $AIL(t,i) \cap AOL(j,IDT(t,i,j,)) \neq \varnothing$.

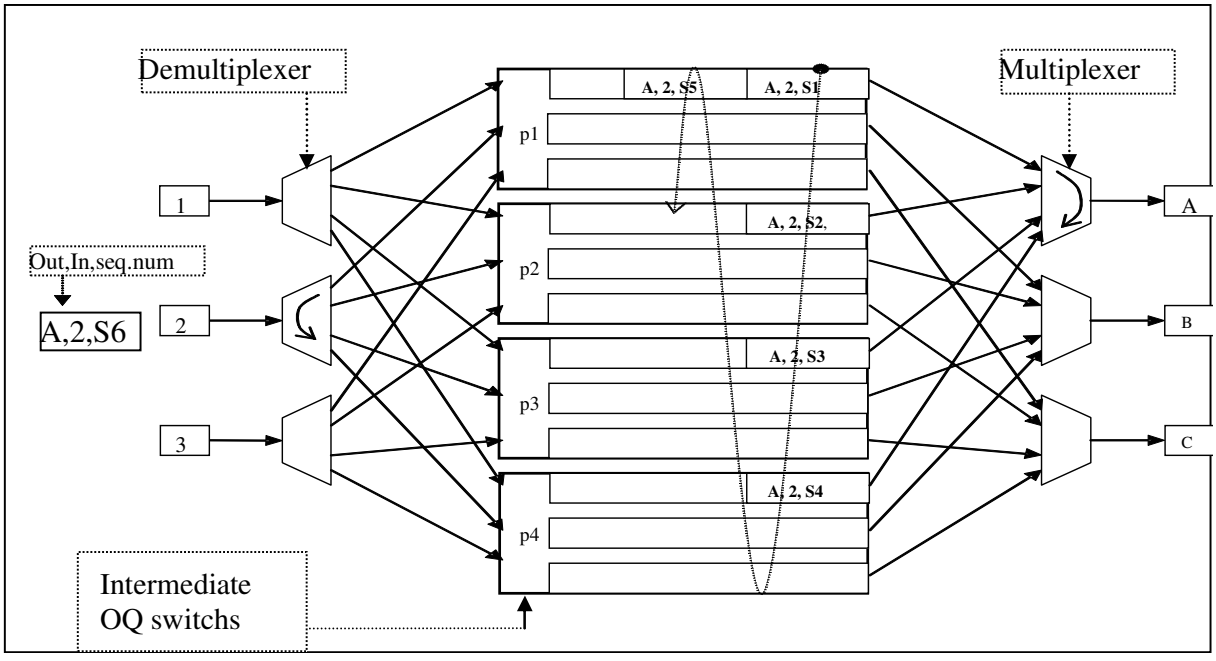

Figure 3 – Cell distribution in CS planes

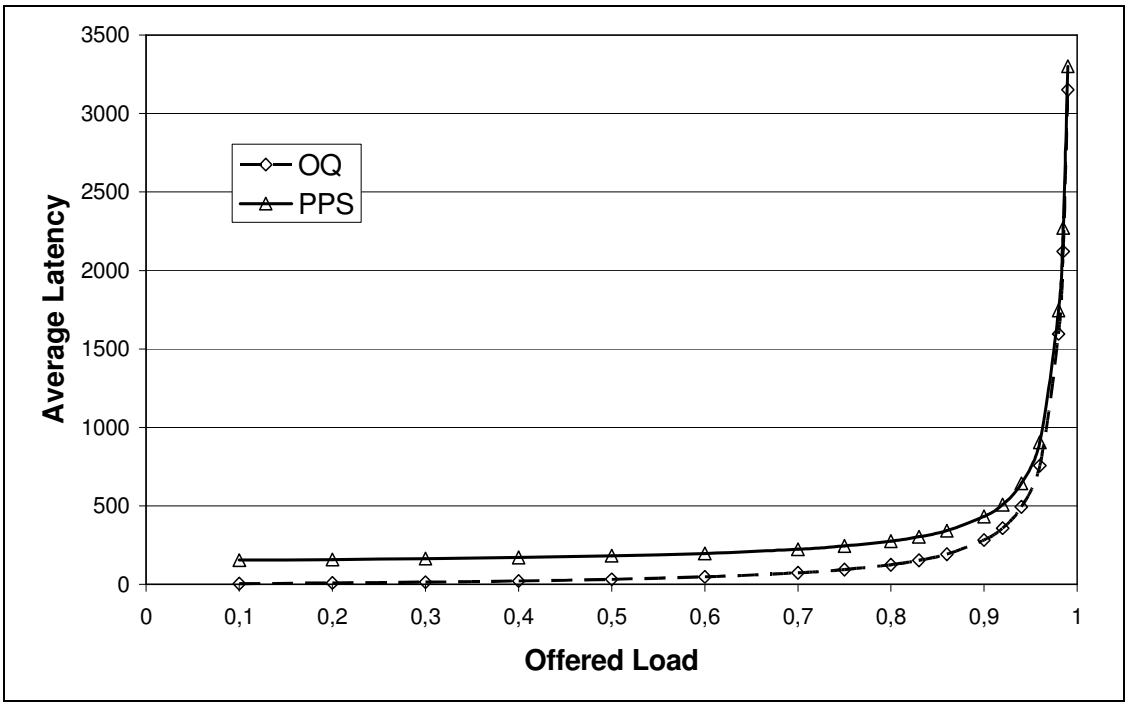

Figure 4 - Comparison between PPS and OQ average latencies

Due to the limited space in this paper the proof is omitted, but the interested reader can find it in [7].
This means that if we have a PPS with $X \geq k^2-k+1$, it is always possible to design a demultiplexing algorithm able to find out for each incoming cell a plane l such that $l \in AIL(t,i) \cap AOL(j,IDT(t,i,j,))$.
The problem here is that IDX easily knows the $AIL(t,i)$ set, since it directly controls the slow speed transmission lines towards the CS planes, but it does not directly know the $AOL(j,IDT(t,i,j,))$ set. Its knowledge requires that all demultiplexers share information about the incoming traffic and use the same CS plane selection algorithm. From a practical point of view this means that each time a head cell is sent to a CS plane, this information, along with the total number of subsequent cells following this one, should be sent to any IDX.
Until now, we have considered the most general case in which incoming packets may have variable length in term of number of cells and we have taken into account also very small packets made of a single cell. Now we will introduce a lower bound on packet dimension such that any packet is at least composed of k cells, it can be proved that iff $X \geq 2k-1$ then $AIL(t,i) \cap AOL(j,IDT(t,i,j,)) \neq \varnothing$ (see [7] again for the proof). Since the cell size can be made very short (ideally one bit, more realistically some octets), this constraint can be easily satisfied even for short TCP/IP packets. Therefore, in all the simulation runs described in the following of this paper we will always consider input traffic composed of packets that satisfy this constraint.

4 The Simulation of the Switching Fabric

To evaluate the performance of the proposed switching architecture we built a simulation program in C language, designing it according to the principles of object programming. This simulator is composed of five modules: the traffic generators, the demultiplexers, the central stage, the multiplexers and the measurement units. Different scenarios, i.e. different combinations of traffic generators and PPS control algorithms, were easily implemented, just building the simulation program with the appropriate modules. The shadow switch, to be used to compare the performance of our architecture against the perfect FIFO switch, was easily implemented using the same object implementing each plane of the Central Stage. The simulator can be instructed to operate as a real switch, i.e. with limited buffering capabilities, or as an ideal switch, i.e. with infinite buffer size. The simulator was validated using the results in [1] and in [5]. The confidence interval for all simulation runs was set to 95%. This means that the simulation stops only when the requested accuracy has been reached.
Using this tool, a large number of simulations were run. Here, for the sake of conciseness, we just present a few of them. In all simulation runs, the traffic generators were homogeneous and uniform, i.e. all generators were identical and the packet destinations were randomly selected. The packet length distribution and the packet interarrival time was either Poisson or geometric.

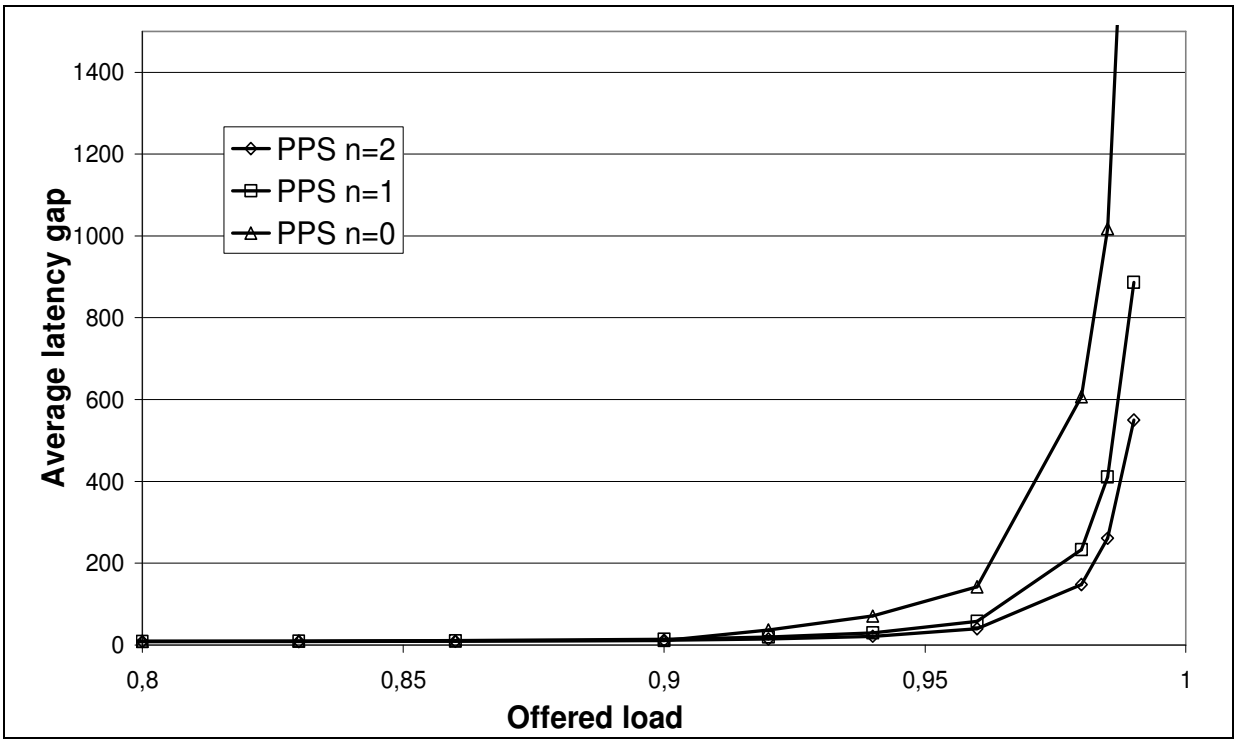

Figure 5 – Average latency gap versus offered load

A first result we obtained was the confirmation that when the number of cells per packet is greater than *k* and the number of CS planes is at least *2k – 1*, the PPS fully emulates an OQS with FIFO policy. This is shown in Figure 4, where we can see that the latency gap *Q* does not depend from the offered load. Another important result was the confirmation that the traffic generator characteristics do not affect this performance figure, which is another indication of the correctness of the simulation.

After this first phase, we investigated the performance of the PPS when one or more CS planes are out-of-service. The parameter we observed was either the latency or the packet loss probability. When a plane is disabled, the algorithm we have described in last section looses its efficiency, since the conditions under which the intersection between *AIL(t,i)* and *AOL(j,IDT(t,i,j,))* is always not empty are no more satisfied. Thus, an optimal CS plane for storing the head cell could not be available, as well as the subsequent planes for storing the subsequent cells. In this situation, the demultiplexing algorithm cannot guarantee a perfect emulation of an OQ switch; nevertheless, it can still provide a *best effort* service choosing the central switching plane which minimizes the amount of time when the multiplexer would not be work-conserving.

In Figure 5 the simulation results for a fabric with *N=16*, *k=4* and *X=4,5,6* (i.e. with *n=0,1,2*, respectively) are shown. This fabric is always unable to emulate a OQS with FIFO policy, since to do this the number of plane would have been, at least, $X = 2k - 1 = 7$. It is easy to see that just adding a single plane to the minimum required (*X=k+1*, i.e. *n=1*) the switch performance dramatically increases. At the same time, when the offered traffic is lower than 0,8 E (80% of the available bandwidth) the additional planes do not significantly affect the switch performance and the average latency gap is kT_{SI}, as expected. More additional planes improve the performance but with decreasing gain as the number of planes increases.

This result is quite significant, since this implies that a PPS can still be working with satisfactory performance even after several CS planes are broken. Moreover, the failure of a CS plane softly affects the switch performance: for any n>2 the performance is almost identical. This also suggests to design on-board switches where only a couple of the additional CS planes are operating. As an example the switch could have n=k additional CS planes, but only two of them will be active at any given moment. Only when a CS plane fails, another one is activated to replace the broken one.

In order to evaluate performances of PPS in a real scenario, we also run some simulations using limited size buffers. In this latter case, we focused our attention on load balancing and, consequently, on the packet loss probability. It is trivial to see that the packet loss probability is minimized if all CS planes are uniformly loaded. Therefore, each element in the first stage, i.e. each demultiplexer should

balance the traffic sent to each CS plane. It is easy to see that this optimal local policy is globally optimal in the long period.
To check this, we run several simulations with the storage parameters shown in Table 1. The switch parameters were *N=16*, *k=4* and *X=7* (i.e. with *n=3* additional planes).

Internal time slot [sec]	Cell length [bit]	Single layer capacity	System capacity	Scenario
0.8 μs	24 [bit]	6144[Kb]	30720[Kb]	A
0.8 μs	24 [bit]	12288 [Kb]	61440[Kb]	B

Table 1 – Storage capacity in simulations with limited buffering capabilities

The results coming from these runs are shown in Figure 6, where the buffer occupancy level of each CS plane is plot for several traffic loads. Our distribution algorithm performs well, balancing the load on each CS plane very effectively, albeit not perfectly. This behavior is as expected, since the selection of the CS plane is only based on local information, and not on global information.
Another important parameter for the evaluation of the switch performance is the packet loss probability. Notice that here we actually compute the fraction of packets lost in the switch, and a packet is lost when at least one of the cells it was split in is lost. In Figure 7, the packet loss probability of the PPS and the optimal OQ switch are shown. As expected, the OQ switch performs better than the PPS, but the shape of both curves is very similar. This means that the PPS performance degradation as the incoming traffic increases is no worse than the degradation of the optimal OQ switch.

5 Conclusions

In this paper we studied a Redundant Parallel Packet Switch and presented a scheduling algorithm operating in a distributed manner, but still able to achieve a good performance. We called this switching architecture RePPS. We reported that, under given redundancy conditions, our system is able to emulate the performances of an OQS, and we validated these results by simulation. An important result is that the latency gap between the RePPS switch and the optimal OQS are fixed and do not depend on incoming traffic characteristics. Hence, a RePPS operating at a speed lower than the input line rate, performs similarly to an OQS.
This architecture, currently non interesting for terrestrial applications, is on the contrary well suited for satellite on-board packet switching systems, where reliability and dependability are of capital importance. The scheduling algorithm proposed in this paper is able to automatically detect out-of-service switching planes and it can keep on working without unnecessarily loosing packets.

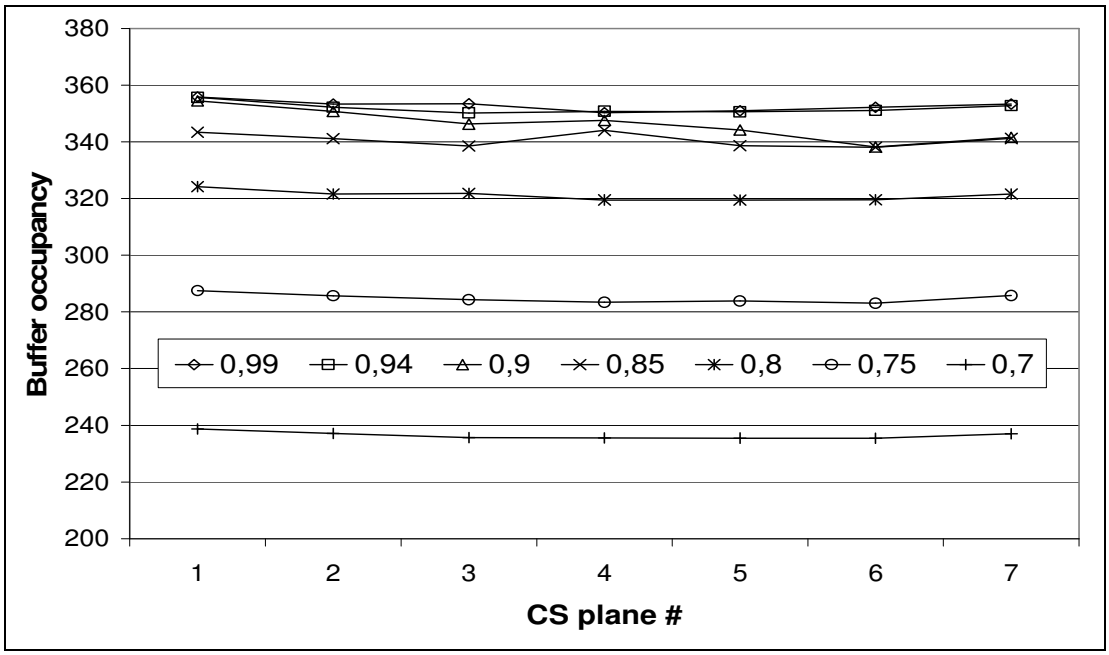

Figure 6 -Buffer occupancy in CS stages

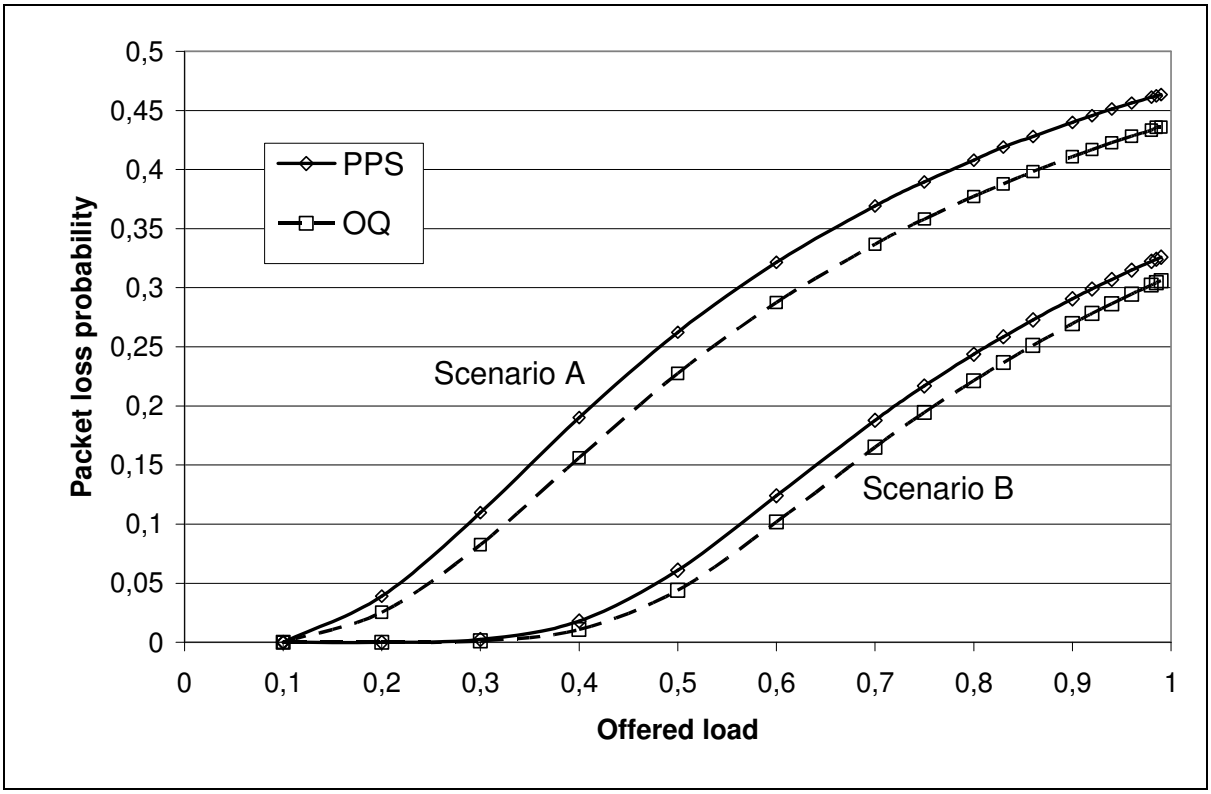

Figure 7 – Packet loss probability versus offered load in switches with finite buffer

Simulation was widely used to asses the performance of our system when the redundancy degree that guarantees full OQS emulation is no longer satisfied, as it could be real case of on-board switches. It was shown that, also in this situation, the RePPS can provide good performance for medium-high offered load, also when the number of redundant switching planes is very low. This preliminary work clearly points out that, instead of fully duplicating the switching fabric, it could also be possible to implement a satellite on-board switch which is less complex, but equally reliable.

6 References

[1] S. Iyer, N. McKeown, “Analysis of a parallel packet switch architecture”, IEEE/ACM Transaction on Networking (2003) 314-324.

[2] A.Aslam, K. Christiansen, “A parallel packet switch with multiplexors containing virtual input queues”, Computer Communications 27 (2004) 1248-1263.

[3] S.Mneimneh, V.Sharma, “Switching Using Parallel Input-Output Queued Switches with no Speedup”, IEEE/ACM Transaction on networking (2002) 653-665.

[4] D. A. Khotimsky, S. Krishnan, “Stability Analysis of. a Parallel Packet Switch with Bufferless Input Demultiplexors”, IEEE International conference, Jun 2001.

[5] Mark J. Karol, Michael G. Hluchyj, Samuel P. Morgan, “input Versus Output Queueing on a Space-Division Packet Switch” IEEE Transaction on Communication, vol COM-35, no. 12, December 1987.

[6] S.-T. Chuang, A. Goel, N. McKeown, B. Prabhakar, "Matching output queueing with a combined input output queued switch", IEEE Journal on Selected Areas in Communications, 17(6):1030-1039, 1999

[7] http://www.lipar.polito.it:8080/lipar/content/e300/e462/e730/Tesi_versionecompleta_ita.pdf (in Italian)