Paper

Packet switch architecture with multiple output queueing

Grzegorz Danilewicz, Mariusz Głąbowski, Wojciech Kabaciński, and Janusz Kleban

Abstract— In this paper the new packet switch architecture with multiple output queuing (MOQ) is proposed. In this architecture the nonblocking switch fabric, which has the capacity of $N \times N^2$, and output buffers arranged into N separate queues for each output, are applied. Each of N queues in one output port stores packets directed to this output only from one input. Both switch fabric and buffers can operate at the same speed as input and output ports. This solution does not need any speedup in the switch fabric as well as arbitration logic for taking decisions which packets from inputs will be transferred to outputs. Two possible switch fabric structures are considered: the centralized structure with the switch fabric located on one or several separate boards, and distributed structure with the switch fabric distributed over line cards. Buffer arrangements as separate queues with independent write pointers or as a memory bank with one pointer are also discussed. The mean cell delay and cell loss probability as performance measures for the proposed switch architecture are evaluated and compared with performance of OQ architecture and VOQ architecture. The hardware complexity of OQ, VOQ and presented MOQ are also compared. We conclude that hardware complexity of proposed switch is very similar to VOQ switch but its performance is comparable to OQ switch.

Keywords— high-speed packet switching, output queueing, buffer, switch fabric, switching node, multicast.

1. Introduction

The transmission capacity of optical fibers has caused a tremendous increase in data transmission speed. The development of broadband access technologies resulted in the need for next generation routers with high-speed interfaces and large switching capacity. One of constrains that limits the switching capacity is the speed of memories used for buffering packets to resolve contention resolution in packet switches. Buffers can be placed on inputs, outputs, inputs and outputs, and/or within the switch fabric. Depending on the buffer placement respective switches are called input queued (IQ), output queued (OQ), combined input and output queued (CIOQ) and combined input and crosspoint queued (CICQ) [1].

In the OQ strategy all incoming cells (i.e., fixed-length packets) are allowed to arrive at the output port and are stored in queues located at each outlet of switching elements. The cells destined for the same output port simultaneously do not face a contention problem because they are queued in the buffer at the outlet. To avoid the cell loss the system must be able to write N cells in the queue during one cell time, N is the total number of inlets of the switch. No arbiter is required because all the cells can be switched to respective output queue. The cells in the output queue are served using FIFO discipline to maintain the integrity of the cell sequence. In OQ switches the best performance (100% throughput, low mean time delay) is achieved, but every output port must be able to accept a cell from every input port simultaneously or at least within a single time slot (a time slot is the duration of a cell). If more cells will request access to a particular output port than the switch fabric output buffer can support, the excess cells must be discarded. An output buffered switch can be more complex than an input buffered switch because the switch fabric and output buffers must effectively operate at a much higher speed than that of each port to reduce the probability of cell loss. The bandwidth required inside the switching fabric is proportional to both the number of ports N and the line rate. This speed is necessary when all inputs simultaneously transfer a cell to the same output port. Such case is called "hot spot" and often occurs when a popular server is connected to a single switch port. The internal speedup factor is inherent to pure output buffering, and is the main reason of difficulties in implementing switches with output buffering. It is no longer possible to find RAMs with sufficiently fast access time taking into account an increasing line rate. Since the output buffer needs to store N cells in each time slot, its speed limits the switch size.

The IQ packet switches have the internal operation speed equal to (or slightly higher) than the input/output line speed, but the throughput is limited to 58.6% under uniform traffic and Bernoulli packet arrivals because of head-of-line (HOL) blocking phenomena [2]. This problem can be solved by selecting queued cells other than the HOL cell for transmission, but it is difficult to implement such queueing discipline in hardware. Another solution is to use speedup, i.e., the switch's internal links speed is greater than inputs/outputs speed. However, this also requires a buffer memory speed faster than a link speed. To increase the throughput of IQ switches space parallelism is also used in the switch fabric, i.e., more than one input port of the switch can transmit simultaneously [3].

One of the proposed solution for IQ switches, which is recently widely considered in papers, is a virtual output queuing (VOQ) [4, 5]. In this solution an input buffer in each input port is divided into N parallel queues, each storing packets directed to different output port. When a new cell arrives at the input port, it is stored in the destined queue and waits for transmission through a switch fabric. In this architecture, the memory speed remains compati-

ble with the line rate, but a good matching algorithm between inputs and outputs is needed so that it can achieve high throughput and low latency. The performance of the switch can be improved when the internal switch fabric operates a few times faster than the line rate, but faster memories are also needed in this case. Different scheduling algorithms for VOQ switches were considered in the literature [5–9], most of them achieve 100% throughput under uniform traffic, but the throughput is usually reduced under non-uniform traffic. The arbitration scheme should be realized slot by slot, therefore the arbiter's speed also limits the capacity of the switch.

In this paper we propose a new switch architecture which uses multiple output queuing (MOQ). In this architecture buffers are located at output ports and are divided into N separate queues. Each of N queues in one output port stores packets from one input port. We assume, that fixed-length switching technology is used, i.e., variable-length packets are segmented into fixed-length packets, called time slots or cells, at inputs and reassembled at the outputs. We will use terms cell and packet interchangeably further on. In the proposed architecture at most one packet is to be written to the one output queue in one time slot. Therefore, the memory speed is equal to the line speed, but the performance of the switch is very similar to those of OQ switch.

The rest of the paper is organized as follows. In Section 2 the general switch architecture is proposed. The possible switch fabric structures are described. Centralized and distributed switch fabrics structures and possible buffer arrangements are considered. In the next section performance evaluation of the proposed switch architecture using simulation is done. Then some comparison between implementation complexity of the proposed switch architecture and a VOQ switch are given, followed by conclusions.

2. The switch architecture

2.1. General architecture

In this paper we propose the new switch architecture which uses output queueing. To reduce the memory speed an output buffer at each output port is divided into N separate queues. Each queue stores packets directed to the output only form one input. In this way this architecture is similar to the VOQ switch, but multiple buffers are located at output ports not at input ports. We will call this architecture the multiple output queueing switch. The general architecture of the switch is shown in Fig. 1. The switch consists of N input ports, N output ports and the switch fabric. Input and output ports can be implemented on separated ingress and egress cards, as it is shown in Fig. 1, or they may be placed on one line card, as it will be shown latter. Each ingress card is connected to the switch fabric by one line, while N outputs from the switch fabric are connected to one egress card. At the output port buffer memory is divided into N separate queues. Each queue stores packets directed from one input port. The output queue denoted by $OQ_{j,i}$ at the output port j stores packets directed to this output port from input i. At the given time slot each input port can send at most one packet and each output port can receive up to N packets, each from different input ports. Therefore, these packets can be simultaneously written to N queues.

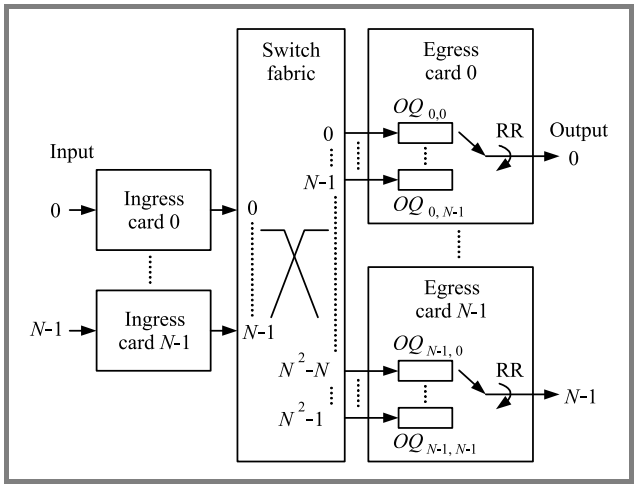


Fig. 1. The switch architecture with multiple output queueing.

The main advantage of these architecture is that it can operate at the same speed as input and output ports, and the lack of arbitration logic, which decides which packets from inputs will be transferred through the switch fabric to output ports (this arbitration mechanism is needed in VOQ switches). However, since we have N queues in each output port, it is necessary to use an output arbiter, which chooses a packet to be sent to the output line. We propose to use round-robin scheme, which is widely used because of its fairness. The buffer management algorithms will be discussed in more details later on.

2.2. The switching fabric architecture

We will now consider some possible switch fabric implementations. The switch fabric in the proposed switch should have a capacity of $N \times N^2$ and should be nonblocking at the packet level. It should be noted that there is no need to support full connectivity in the switch fabric. Any input should only have a possibility to send packets to N different switch fabric's outputs, each of these N outputs should be connected to the different output port. In general, input i, $0 \leq i \leq N-1$ should be able to transfer a packet to the switch fabric output $jN+i$, $0 \leq j \leq N-1$, when this packet is directed to output port j. The packet will be then stored in $OQ_{j,i}$.

The switch fabric can be organized either in the centralized mode or the distributed mode. In the first case the switch fabric constitutes one module produced on one board (or several boards). This architecture is shown in Fig. 2. We assume here that one input port and one output port

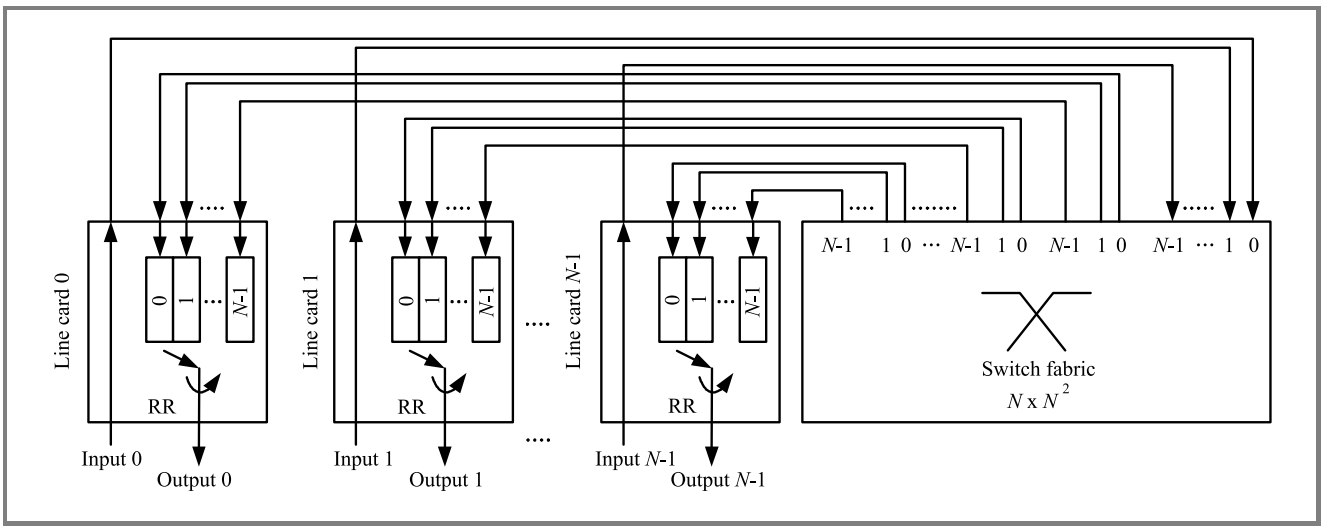


Fig. 2. The switch architecture with the centralized switch fabric.

are arrange on one line card. Buffers are placed at output ports. The switch fabric may be realized for instance using the tree architecture, or may be based on the crossbar architecture as it is shown in Figs. 3 and 4, respectively. The stacked-banyan switch fabric proposed in [10] can also be used. In all these solutions $N^2 + N$ lines is needed for connecting input and output ports to the switch fabric. Each line card is connected by means of $N+1$ lines to the switch fabric. For switches of greater capacity number of connectors will limit the switch size. The capacity of the switch may be increased by using fiber connections with wavelength multiplexing. The other solution is to combine output buffers within the switch fabric. In this case line cards will be connected with the switch fabric by two lines, but the switch fabric will require more boards with buffer memories.

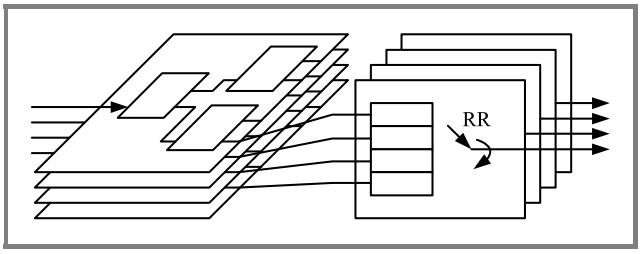


Fig. 3. The switch fabric using the tree architecture.

In the distributed mode the switch fabric is distributed over line cards (or ingress/egress cards). In this case each line card comprises also a segment (or a part) of the switch fabric. The capacity of such segment is $1 \times N$, and for each line card there is N outgoing lines to connect outputs of the $1 \times N$ switch fabric to buffers located on the same and other line cards, and N incoming lines to N output queues (see Fig. 5). The switch fabric based on the tree architecture can be decentralized by putting each $1 \times N$ segment on one line card (compare Fig. 3). The crossbar architecture can be also decentralized in such a way that each row of the crossbar switch fabric (which corresponds to one input – see Fig. 4) is placed on one line card.

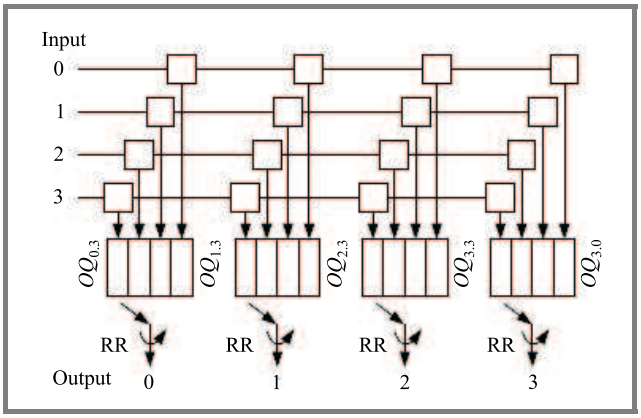


Fig. 4. The switch fabric based on the crossbar architecture.

The drawback of the decentralized architecture described above is the number of outputs from line cards. We need $2N$ lines (N incoming and N outgoing) for each line card. This number may be reduced by putting the switch fabric on the output side of the line card, as it is shown in Fig. 6. Connection lines between cards work as busses and arriving packets are broadcast from inputs to all outputs. Address filters AF at each line card determine whether respective packets are destined to the output. Cells directed to the given output are passed through the address filters to the output queues. The advantage of this architecture is the reduced number of connecting lines between line cards, which is now equal to N. The number of address filters is N^2, and they should operate with the line speed. The speed of connecting lines between line cards is also equal to the line speed.

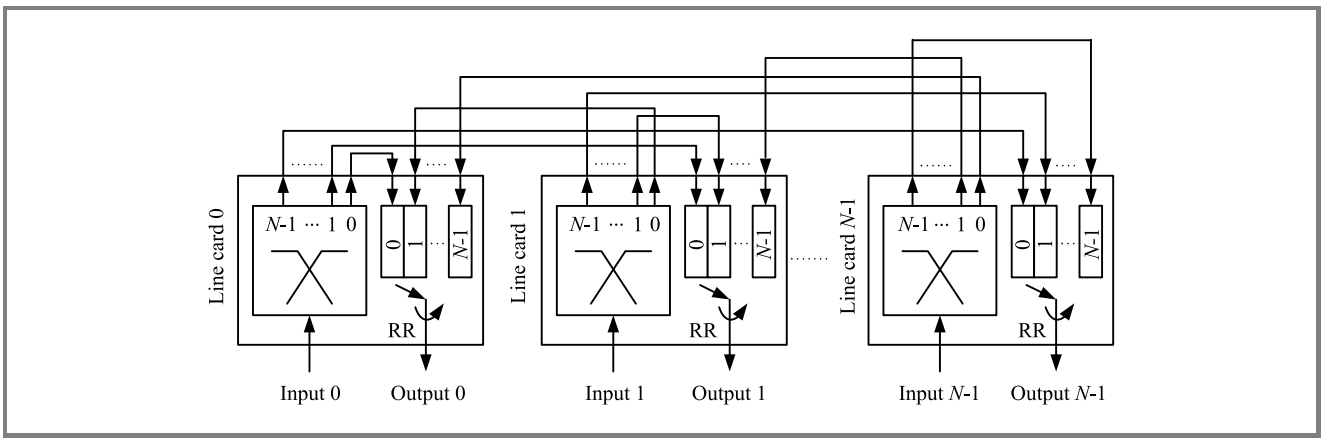


Fig. 5. The switch architecture with the decentralized switch fabric – version 1.

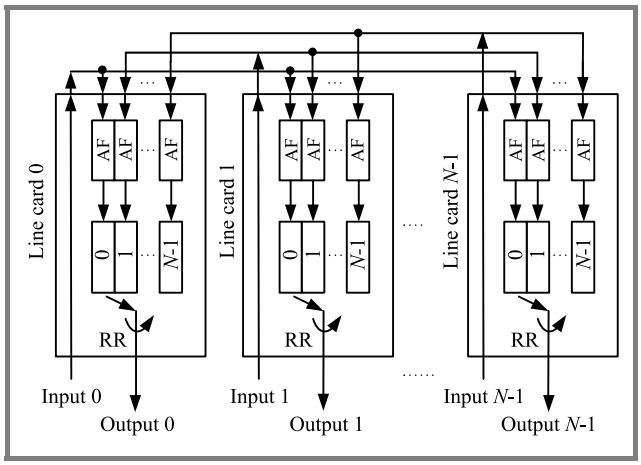


Fig. 6. The switch architecture with the decentralized switch fabric – version 2.

2.3. Buffer arrangements

Buffers in output ports are arranged into N separate queues. When N packets from N input ports are directed to one output port in the same time slot, each packet is written to the different queues. Therefore, the memory speed is the same as the line speed. Buffers may be arranged as separate queues with independent write pointers or as a memory bank with one pointer which points the same memory cells in each queue. Packets from N queues in each output port are read out using the round-robin (RR) algorithm. When independent write pointers are used, the round-robin pointer, denoted by RR, is moved to the queue next to those read out in the previous time slot. When packets are written to the same position of the buffers (one write pointer is used), the operation of RR is modified in such a way, that when all packets from the same position (i.e., which were simultaneously written to the buffers) are already read out, the RR is set back to 0. The operation of these two arrangements will be described by means of the following example.

In the first case the separate pointer is assign to each queue. This pointer, denoted by $MP_{j,i}$, points the end of queue $OQ_{j,i}$, where the next incoming packet to output j from input i will be written to. The example for output x is shown in Fig. 7. It is assumed that all queues are empty at the beginning of the first time slot. Pointers are shown by arrows which shows the state of the pointers at the end of respective time slots. In the first time slot two packets (numbered 1 and 2) from inputs 0 and 1 arrive to the considered output x. The round-robin pointer is set to 0 (the HOL packet from $OQ_{x,0}$ has the highest priority). Since buffer $OQ_{x,0}$ is empty, the packet from input 0 is immediately directed to the output, the RR pointer is set to 1, and packet 2 is stored in $OQ_{x,1}$. The state of RR at the end of the time slot is shown in Fig. 7. The pointer of $OQ_{x,1}$ is moved to the next memory cell. In the next time slot packets from inputs 0, 1 and 3 arrive (numbered 3, 4, and 5, respectively). They are stored in respective queues, while packet 2 from $OQ_{x,1}$ is sent out. During the third time slot packets 6, 7 and 8 arrive from inputs 1, 2, and 3, respectively. Since RR is now set to 2 and buffer $OQ_{x,2}$ is empty, packet 7 is sent directly to the output, while packets 6 and 8 are stored in $OQ_{x,1}$ and $OQ_{x,3}$. In the next time slot packet 5 will be sent out from $OQ_{x,3}$. In this example packet 7 is sent before packet 5, but these packets arrive to the considered output from different inputs. The sequence of packets from the same input port is preserve.

In the second case there is one pointer for all queues. This pointer, denoted by MP_j, points to the memory cells in all queues of output j, where the next incoming packets will be written to. The example is shown in Fig. 8. In the first time slot two packets (numbered 1 and 2) from inputs 0 and 1 arrive to the considered output x. The round-robin pointer is set to 0 (the HOL packet from $OQ_{x,0}$ has the highest priority). Since buffer $OQ_{x,0}$ is empty, the packet from input 0 is immediately directed to the output, packet 2 is stored in $OQ_{x,1}$, the MP_x is moved to the next memory cells in all queues (shown by arrows in Fig. 8), and the RR pointer is set to 1 (here also the state of RR is

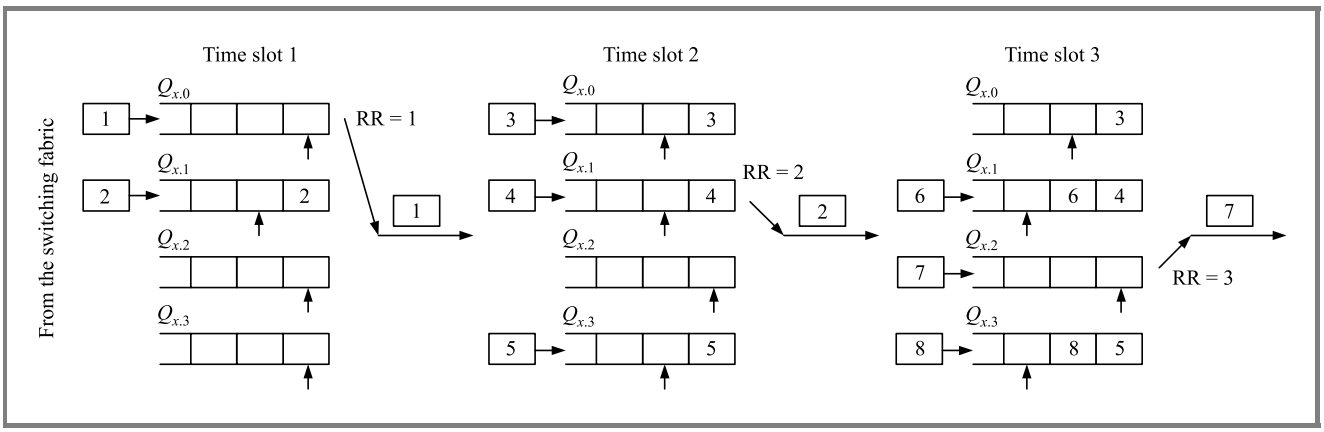


Fig. 7. The example of buffer operation with separate pointers.

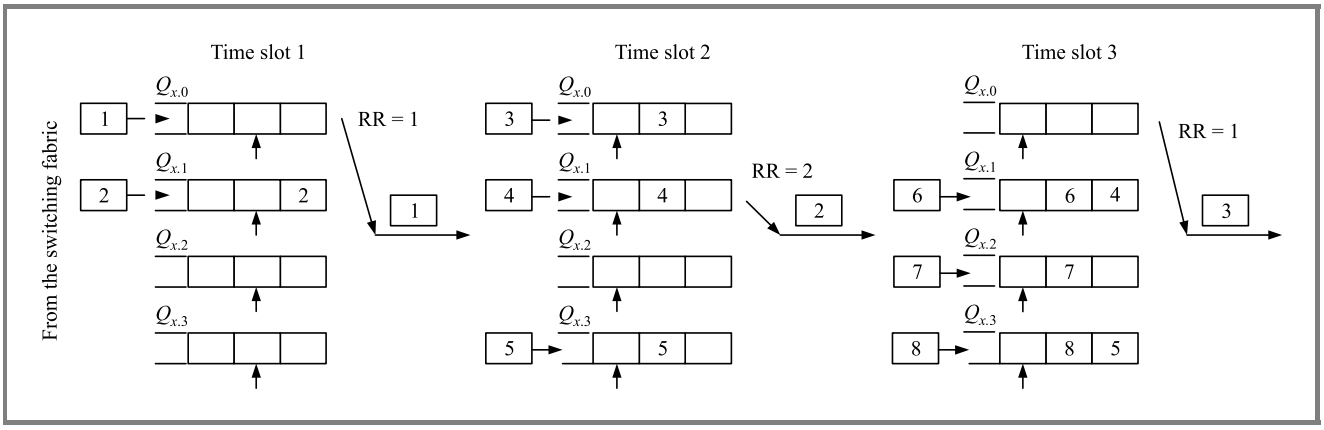


Fig. 8. The example of buffer operation with one pointer.

shown at the end of the time slot). In the next time slot packets from inputs 0, 1 and 3 arrive (numbered 3, 4, and 5, respectively). They are stored in the second memory cell of respective queues, while packet 2 from $OQ_{x,1}$ is sent out. After this packet is read out, there is no any packet in the first memory cell in all queues. Therefore, the next cells in the queues are moved to the HOL position, and the RR is set to 0. During the third time slot packets 6, 7 and 8 arrive from inputs 1, 2, and 3, respectively. Since RR is now set to 0, packet 3 from $OQ_{x,0}$ is sent to the output, while new packets are written to the buffer. In the next three time slots packets 4, 5, and 6 will be sent out from $OQ_{x,1}$, $OQ_{x,3}$, and $OQ_{x,1}$, respectively.

In this second approach all packets which arrive to the given output are written in the same position of each buffer. So we can use only such positions where all memory cells are empty. When in the given time slot less than N packets arrive to the output, some memory cells will be empty and they could not be used to store packets until all packet in the same position of all buffers are read out. Therefore, the memory is not used as efficiently as in the first approach. In the next section only the performance of this first approach will be evaluated.

3. Performance evaluation

In order to evaluate performance measures for the proposed MOQ switch architecture, the corresponding simulation researches have been conducted. The researches have been carried out for the switch with a size of $N \times N$ ($N = 8$) for the following values of traffic load for an input port: $p = 0.6; 0.7; 0.8; 0.9$. We have assumed that offered traffic is uniformly distributed for N outputs (uniform traffic). We have further assumed that the service time of each cell is deterministic and equal to one.

The results of the simulations are shown in the charts (Figs. 9 and 10) in the form of marks with 95% confidence intervals that have been calculated after the t–student distribution for the five series with 10,000,000 time slots. For each of the points of simulation the value of the confidence interval is at least one order lower than the mean value of the results of the simulation. In many cases the value of the confidence interval is lower than the height of the sign used to indicate the value of the simulation experiment.

We have evaluated two performance measures for switch architectures, i.e., mean waiting time (mean cell delay) and cell loss probability (CLP). The obtained performance

measures of MOQ architecture have been compared with performance of OQ architecture [6] and VOQ architecture (algorithm iSLIP with one and four iterations) [7].

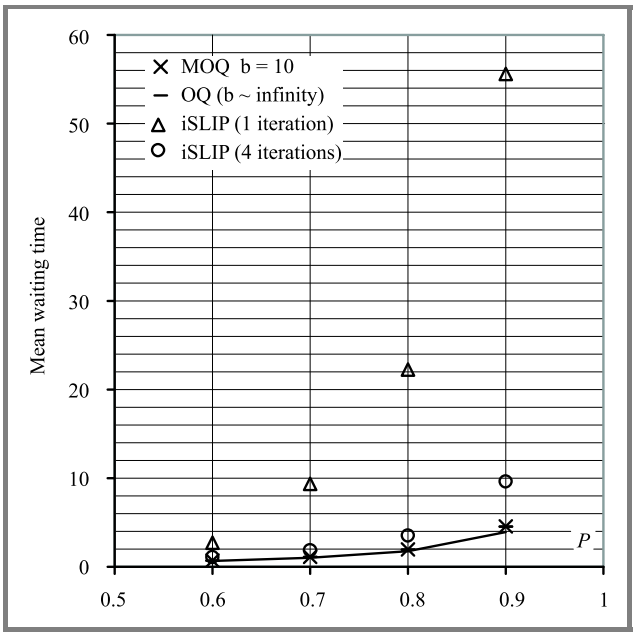


Fig. 9. Mean waiting time (in time slots), $N = 8$.

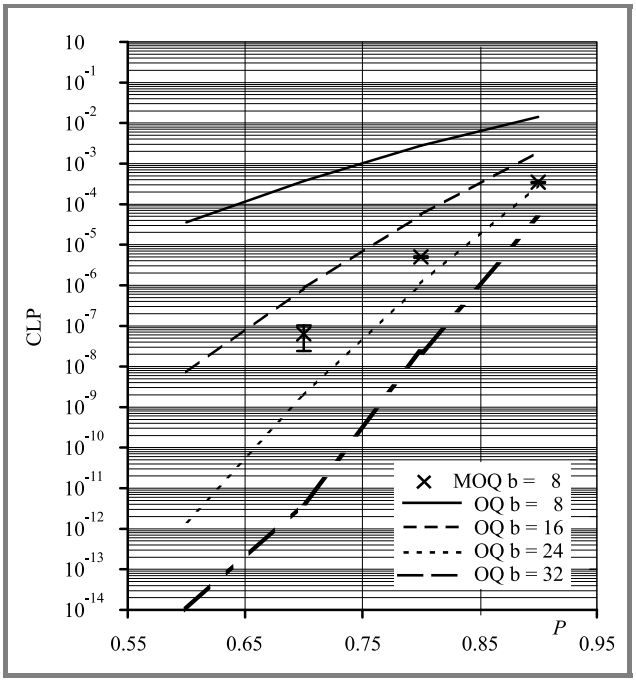


Fig. 10. Cell loss probability, $N = 8$.

Figure 9 plots the mean waiting time (in time slots) of the MOQ switch as a function of input load p. The presented results have been obtained for the switch in which the size of each of $N = 8$ output buffers (of the tagged output port) was limited to $b = 10$. The adopted buffer size assures – for each value of traffic load – stable values of mean waiting times (the application of larger buffers do not lead to increase in values of waiting time). The simulation results enabled us to compare the MWT values in the proposed MOQ switch architecture with the results obtained for OQ architecture. For OQ switch we have assumed that the buffer size is large enough to get stable values of MWT parameter. We can notice that both architectures are comparable. This phenomenon results from similar characteristics of both FIFO discipline for single queue and round robin discipline for cyclic-service set of queues. Additionally, Fig. 9 shows the performance of iSLIP algorithm in virtual output queuing architecture. It is evident from the presented results that – regardless of the number of iterations in iSLIP algorithm – the VOQ switch architecture is charcterised by higher values of MWT than the proposed MOQ switch architecture.

Another important performance measure for packet switches is the cell loss probability. Figure 10 compares the results of CLP obtained for MOQ switch with the results calculated for the switch with output queuing. It is intuitively clear, that the proposed switch architecture requires greater total number of memory cells (N buffers for each output port) in order to keep the same value of CLP parameter as in the case of switches with single output queue for each output port.

4. Comparison

In the previous section MWT and CLP in MOQ, OQ and VOQ switches were compared. Now we compare a hardware complexity of these architectures. This comparison is summarized in Table 1. The MOQ switch uses the same number of buffers as VOQ switch and of the same speed as the line rate. The OQ switch comprise only N buffers, but they have to be N times faster than the line speed.

Table 1
The hardware complexity of different buffering strategies

Parameters	OQ	VOQ	MOQ
Number of buffers	N	N^2	N^2
Memory speed (in line speed)	N	1	1
Switch fabric capacity	$N \times N$	$N \times N$	$N \times N^2$
Switch fabric speed	N	1	1
Switch fabric hardware	N^2	N^2	N^2
Number of schedulers	–	$2N$	N
Wiring complexity	N	N^2	N^2

The switch fabric speed in MOQ is also the same as line speed, and the same is true for VOQ switch, provided that no speed-up is used to increase the performance of the VOQ switch. In OQ switch the switch fabric is N times faster. However, in MOQ architecture the switch fabric has the capacity of $N \times N^2$ instead of $N \times N$. But this greater capacity does not result in greater hardware complexity, since MOQ switch require the same number of switching elements (when crossbar architecture is considered) as OQ or VOQ switches.

The important issue is the packet scheduling mechanism and wiring complexity. The OQ switches do not need packet schedulers and the wiring complexity is $O(N)$. On the other hand, VOQ switches need $2N$ schedulers when iterative maximal matching algorithm is used (one scheduler in each input port and one in each output port). These schedulers are to be connected between themselves so the wiring complexity is $O(N^2)$. The similar complexity is needed when one centralized scheduler is used, since each VOQ has to be connected with the scheduler to send request signal when it has a HOL packet. The MOQ architecture has the same wiring complexity but lines are used to connect MOQs with the switch fabric, instead of connecting VOQs and schedulers. The MOQ switches need also N schedulers, one for each output, but there is no need to connect schedulers between themselves. The MOQs of the given output are to be connected to the scheduler of this output and this is done inside the output port (a line card or an egress card).

Comparing the hardware complexity and the performance of the switches we can say, that the MOQ architecture is attractive and worth considering in constructing high-speed and high-capacity switches. The hardware complexity is very similar to VOQ switches but the performance of the MOQ architecture is much better, at least when uniform traffic is considered. Simulation results shows, that the performance of the MOQ switch is very similar to the OQ switch.

5. Conclusions

We have proposed the new packet switch architecture which uses multiple output queueing. This architecture looks attractive for constructing high-speed packet switches. The hardware complexity of this architecture is very similar to VOQ switch but its performance is comparable to OQ switch. This paper contains the first considerations and the first results we obtained for this architecture. The architecture is also very promising since it can naturally support multicast traffic. Further research are needed to evaluate the performance of the MOQ switch under other traffic types (non-uniform, hot-spot), the buffer length evaluation, as well as the practical buffer implementation in either separate chip or in the switch fabric.

References

[1] K. Yoshigoe and K. J. Christensen, "An evolution to crossbar switches with virtual ouptut queuing and buffered cross points", *IEEE Network*, vol. 17, no. 5, pp. 48–56, 2003.

[2] M. K. Karol, M. Hluchyj, and S. Morgan, "Input versus output queuing on a space-division packet switch", *IEEE Trans. Commun.*, vol. 35, pp. 1347–1356, 1987.

[3] J. Xie and Ch.-T. Lea, "Speedup and buffer division in input/output queuing ATM switches", *IEEE Trans. Commun.*, vol. 51, no. 7, pp. 1195–1203, 2003.

[4] Y. Tamir and G. Frazier, "High performance multi-queue buffers for VLSI communications switches", in *Proc. Comput. Archit.*, Honolulu, Hawaii, United States, 1988, pp. 343–354.

[5] T. Anderson *et al.*, "High-speed switch scheduling for local-area networks", *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319–352, 1993.

[6] H. J. Chao, C. H. Lam, and E. Oki, *Broadband Packet Switching Technologies: A Practical Guide to ATM Switches in IP Routers.* New York: Wiley, 2001.

[7] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in input-queued switches", *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260–1267, 1999.

[8] H. J. Chao, "Saturn: a terabit packet switch using dual round-robin", *IEEE Commun. Mag.*, vol. 38, no. 12, pp. 78–84, 2002.

[9] E. Oki, R. Rojas-Cessa, and H. J. Chao, "A pipeline-based approach for maximal-sized matching scheduling in input-buffered switches", *IEEE Commun. Lett.*, vol. 5, no. 6, pp. 263–265, 2001.

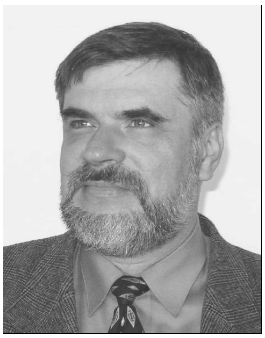
[10] B. Kraimeche, "Design and analysis of the stacked-banyan ATM siwtch fabric", *Comput. Netw.*, vol. 32, no. 2, pp. 171–184, 2000.

Grzegorz Danilewicz was born in Poznań, Poland, in 1968. He received the M.Sc. and Ph.D. degree in telecommunication from the Poznań University of Technology (PUT), Poland, in 1993 and 2001, respectively. Since 1993 he has been working in the Institute of Electronics, Poznań University of Technology, where he currently is an Assistant Professor. His scientific interests cover photonic broadband switching systems with special regard to the realization of multicast connections in such systems. He is a member of the IEEE Communication Society. He has published 25 papers.

e-mail: gdanilew@et.put.poznan.pl
Institute of Electronics and Telecommunications
Poznań University of Technology
Piotrowo st 3A
60-965 Poznań, Poland

Mariusz Głąbowski was born in Turek, Poland, in 1973. He received the M.Sc. and Ph.D. degrees in telecommunication from the Poznań University of Technology (PUT), Poland, in 1997 and 2001, respectively. Since 1997 he has been working in the Institute of Electronics and Telecommunications, Poznań University of Technology, where he currently is an Assistant Professor. He is engaged in research and teaching in the area of performance analysis and modelling of multiservice networks and switching systems. He has published papers.

e-mail: mglabows@et.put.poznan.pl
Institute of Electronics and Telecommunications
Poznań University of Technology
Piotrowo st 3A
60-965 Poznań, Poland



Wojciech Kabaciński received the M.Sc., Ph.D., and D.Sc. degrees in communication from Poznań University of Technology (PUT), Poland, in 1983, 1988 and 1999, respectively. Since 1983 he has been working in the Institute of Electronics and Telecommunications, Poznań University of Technology, where he currently is an Associate Professor. His scientific interests cover broadband switching networks and photonic switching. He has published three books, 78 papers and has 10 patents. Prof. Kabaciński is a member of the IEEE Communication Society and the Association of Polish Electrical Engineers.
e-mail: kabacins@et.put.poznan.pl
Institute of Electronics and Telecommunications
Poznań University of Technology
Piotrowo st 3A
60-965 Poznań, Poland

Janusz Kleban was born in Pobiedziska, Poland. He received the M.Sc. and Ph.D. degrees in telecommunications from the Poznań University of Technology (PUT) in 1982 and 1990, respectively. From August 1982 to November 1983 he was with Computer Centre for Building Industry in Poznań where he worked on data transmission systems. He has been with Institute of Electronics and Telecommunications at PUT, where he currently is an Assistant Professor, since December 1983. He is involved in research and teaching in the areas of computer networks, switching networks, broadband networks and various aspects of networking. He is author and co-author of many publications and unpublished reports.
e-mail: jkleban@et.put.poznan.pl
Institute of Electronics and Telecommunications
Poznań University of Technology
Piotrowo st 3A
60-965 Poznań, Poland