QOAG: An Efficient Queueing Policy for Input-buffered Packet Switches

N. H. Liu Kwan L. Yeung

Department of Electronic Engineering
City University of Hong Kong, Tat Chee Avenue, Hong Kong
Fax: +852 27887791 Tel: +852 27887730
{kyeung,nhliu}@ee.cityu.edu.hk

Abstract

An efficient self-adaptive packet queueing policy, called Queueing with Output Address Grouping (QOAG), is proposed for optimizing the performance of an input buffered packet switch. Each input port of the $N \times N$ switch under consideration has Q queues and each queue has B packet buffers, where $1 < Q < N$. Using QOAG, a packet arrived at an input port is assigned to the queue which has some backlog packets with the same output address as that of the new packet. If the output address of the new packet is different from all current buffered packets in all queues, it is assigned to the shortest queue. The performance of QOAG is compared with Odd-Even queueing policy in [1] by simulations. Zipf distribution version II [2] is used to model the non-uniform packet output distributions. We found that for a 16×16 switch with $B = 20$ buffers at each queue and input load $p = 0.7$, the mean packet delays are 58.1 and 91.2 time slots and the mean throughputs are 0.474 and 0.355 for using QOAG and Odd-Even queueing respectively. This represents a 57% cut in mean packet delay and 25% increase in throughput when QOAG is used.

I. Introduction

Asynchronous Transfer Mode (ATM) is an international networking standard designed for cost-effective transfer of multimedia traffic, such as video-on-demand and video conferencing. Switches, as an essential part of an ATM network, deliver incoming packets arriving on a particular input port to the output port associated with the appropriate virtual path. Various ATM switch architectures [3] have been proposed and studied extensively in order to provide high performance packet switching for integrated ATM transport. In this paper we concentrate on input-buffered nonblocking packet switches. The fabric and the memory of an input-buffered switch needs only to run as fast as the line rate. This makes input queueing very attractive for switches with fast lines or with a large number of ports. It has been found [4] that the maximum throughput of an input-buffered packet switch with a single queue per input port is limited to 58.6% under uniformly distributed traffic condition. This is because of the Head-of-Line (HOL) blocking phenomenon. That is a packet can be held up by another packet ahead of it in the same queue and is destined to a different output. To reduce the HOL blocking, switches with multiple queues per input port have been studied.

This work was supported by Research Grant Council Earmarked Grant 9040264.

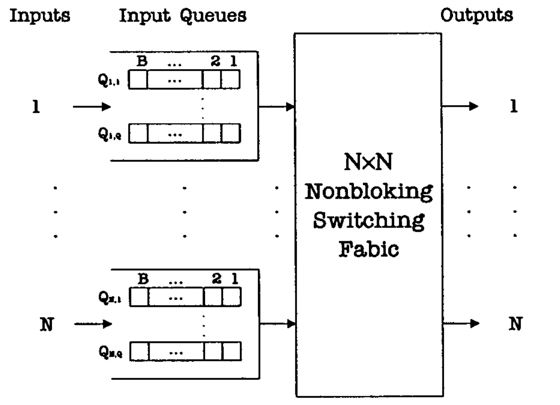

Fig. 1. An $N \times N$ input-buffered nonblocking switch with B buffers for each queue and Q input queues for each input port where $1 < Q < N$.

For an input-buffered packet switch, two design problems are usually addressed: packet queueing policy design and packet scheduling algorithm design. A *packet queueing policy* specifies how packets arrived at an input port are assigned to different input queues. For switches with a single queue per input port, the queueing policy is trivial. We simply place all packets to the same and only queue in a FIFO manner. For switches with a dedicated queue for each output port at each input [5], [6], the queueing policy is also straight-forward. We simply assign packets to each queue according to their output port addresses. For switches with Q queues per input port where $1 < Q < N$ (as shown in Fig. 1), the packet queueing policy design becomes not so straight-forward.

0-7803-6283-7/00/$10.00 © 2000 IEEE

In [1], an Odd-Even queueing policy has been designed.

Unlike packet queueing policy, packet scheduling algorithms have been extensively studied in the literature. A *packet scheduling algorithm* determines how packets at all queues of all input ports are selected for switching to outputs. The scheduling algorithms for switches with single queue per input port have been reported in [7], [8]. For switches with a separate queue for each output, scheduling algorithms that can achieve 100% throughput for both uniform and nonuniform traffic are proposed [9]. The problems with this approach are (a) a large number of input queues (N^2) needs to be maintained, and (b) the complexity of scheduling algorithms is high ($O(N^3)$ or comparable complexity [10]).

In this paper, we focus on packet queueing policy design. An efficient self-adaptive queueing policy, called Queueing with Output Address Grouping (QOAG), is proposed. Using QOAG, a packet arrived at an input port is assigned to the queue which has some packets with the same output address as the new packet. If no packet with the same output address as the new packet can be found in all queues, the new packet is assigned to join the shortest queue. We compare the performance of QOAG with Odd-Even queueing with two queues per input port. Significant performance improvement in throughput and mean packet delay are obtained when the packet output addresses are non-uniformly distributed.

II. Odd-Even Queueing Policy

Consider the Odd-Even switch [1] shown in Fig. 2(a). It has two FIFO queues for each input port, the odd queue for storing packets destined to all odd numbered output ports and and the even queue for storing packets to all even numbered output ports. We call it Odd-E[illegible] queueing policy.

Based on the HOL packets at all queues, the pac scheduling algorithm consists of two arbitration rou in each time slot. During the first round, the HOL pa ets at all even input queues are scheduled. In case c tie, the winning packet is chosen randomly. In the s ond round, the HOL packets at the remaining odd que (where no packet has been scheduled for transmissior the first round at the same port) are scheduled simila Then the successful packets in both rounds are switcl to their respective outputs together. For access fairn the arbitration order is interchanged in every next ti slot.

We can easily generalize the above packet queue policy and packet scheduling algorithm to a switch w Q queues per input port. In this case, the N out ports are partitioned into Q equal portions, one for e input queue. When a packet arrives, it is placed i the queue where its output port belongs to. The pac scheduling now consists of Q contention rounds in e time slot. Arbitration during the first round involves the HOL packets at all first queues of all input ports. In case of a tie, the winning packet is selected randomly. Since each input port can transmit at most one packet in each time slot. In the k-th contention round, arbitration only involves the HOL packets at those k-th queues whose associated input ports have no packet being scheduled for transmission in all previous rounds (i.e. from 1 to $k-1$). Repeat this until all Q contention rounds are finished. Then all packets that have been successfully scheduled are switched to their respective outputs simultaneously. To maintain the fairness among all input queues, the queue serving order is cyclically shifted in each time slot.

In [1], the performance of Odd-Even switch has been compared with switches with a single queue per input port. Significant improvement in throughput and mean packet delay has been obtained. But if the output addresses of packets are not uniformly distributed over all outputs, we can easily show that the performance of the Odd-Even switch decreases dramatically. Fig. 2(a) shows an example of an 8×8 Odd-Even switch, where the output addresses of packets are uniformly distributed only to all *even* numbered outputs. In this case, the odd queues are empty and the Head-of-Line (HOL) blocking remains as serious as in a switch with a single queue per input port. If we adopt a new queueing policy at each input port such that packets destined to outputs 2 and 4 are assigned to the upper queue, and packets destined to outputs 6 and 8 are assigned to the lower queue as shown in Fig. 2(b). Then the HOL blocking can be reduced and the switch throughput can be greatly increased.

It is quite obvious that Odd-Even queueing lacks the flexibility of adapting to traffic changes. We argue that a good packet queueing policy should always maximize the number of HOL packets with ***different*** output addresses in each time slot. This can facilitate the packet [illegible]

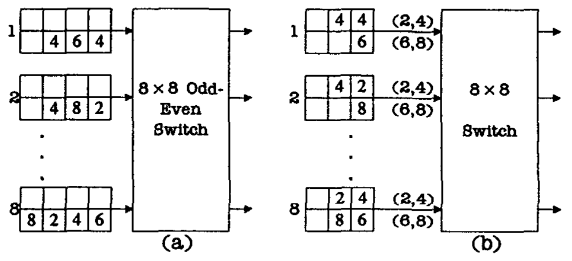

Fig. 2. An 8 × 8 switch two different packet queueing polices. (a)Odd-Even Switch; (b) upper queue for packets with output addresses 2 and 4, lower queue for packets with output addresses 6 and 8.

III. Queueing with Output Address Grouping

Consider the $N \times N$ input-buffered nonblocking switch shown in Fig. 1. In each time slot, at most one packet can arrive at an input port and up to N packets can arrive at the switch. For each input port, let l_i be the length of queue i and c_{ij} be the number of packets at queue i with output address j. If $c_{ij} > 0$, that means there are c_{ij} packets in queue i destined to output j. Otherwise, $c_{ij} = 0$. Using Queueing with Output Address Grouping (QOAG), when a packet with output address k arrives, if $c_{ik} > 0$, assign the new packet to queue i and set $c_{ik} = c_{ik} + 1$; if $c_{ik} = 0$, assign the new packet to the queue with $min\{l_i\}$. If the selected queue is full, the new packet is dropped.

It should be noticed that for packets belong to the same connection, their relative sequence is kept using QOAG. When the input traffic load changes, QOAG can easily adapt itself by distributing the load to queues with shorter queue lengths.

When a packet has been successfully switched, the corresponding values of c_{ij} and l_i at each input port are updated accordingly. To be more specific, if the HOL packet with output address k at queue i of an input port is switched, we set $c_{ik} = c_{ik} - 1$ and $l_i = l_i - 1$.

IV. Performance Evaluations

In this section, we compare the performance of QOAG with the Odd-Even queueing policy in terms of throughput, mean packet delay and packet loss/overflow probability. In our simulations, we adopt the same packet scheduling algorithm as described in Section II for both packet queueing policies. Let the number of packet buffers at each queue be $B = 20$, the number of queues at each input port be $Q = 2$, and the switch size be $N = 16$. Assume packets arriving at each input port follow the same independent Bernoulli process with probability p of having a new packet per time slot. This probability is referred to as the input load. Let the distribution of packet output addresses follow the Zipf distribution version II [2], where the probability to output i is

$$b_i = \frac{i^\theta - (i-1)^\theta}{M^\theta}$$

and

$$\sum_i b_i = 1.$$

The value of θ is set to 0.4 in our simulations. Note that Zipf distribution is typically used to model the nonuniformly distributed traffic patterns. Directly applying Zipf distribution will generate a list of output probabilities in the decreasing order of the output port number. Here we randomly shuffle the positions of the output ports. The resulting packet output access probabilities are shown in Table 1 and are used in our simulations.

Fig. 3 shows the mean packet delay against the input load p. It is very interesting to note that the mean packet delay increases with the input load for $p < 0.3$.

Output address	*Access Probability*
1	0.0476
2	0.0301
3	0.0394
4	0.0254
5	0.0322
6	0.0766
7	0.3299
8	0.0624
9	0.0342
10	0.0430
11	0.0277
12	0.0367
13	0.0265
14	0.1054
15	0.0534
16	0.0290

TABLE I
The distribution of packet output addresses.

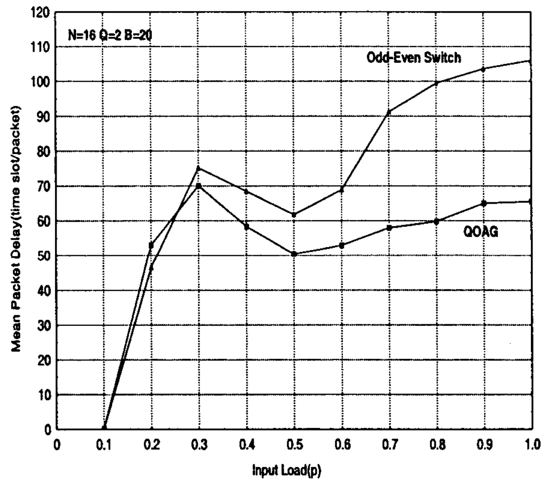

Fig. 3. Mean packet delay vs input load p for a 16×16 switch, where each input queue size is $B = 20$ and the number of queues at each input port is $Q = 2$.

For $0.3 < p < 0.5$, the mean packet delay decreases. For $p > 0.5$, the mean packet delay increases again. This can be explained by the nonuniform distribution of packet output addresses. From Table 1, output port 7 has the highest access probability and is thus the system hotspot. According to QOAG, the incoming packets with the same output address will be assigned to the same queue. For $p < 0.3$, the Head-of-Line (HOL) blocking caused by packets to output 7 is the dominating factor for the mean packet delay to increase. When $p > 0.3$, the queues with packets to output 7 begin to overflow. The packets that can be successfully switched to output 7 experience a *steady/saturated* delay performance. When input load p increases from 0.3 to 0.5, the throughput of the packets destined to outputs other than 7 increases significantly and this results in a overall drop in mean

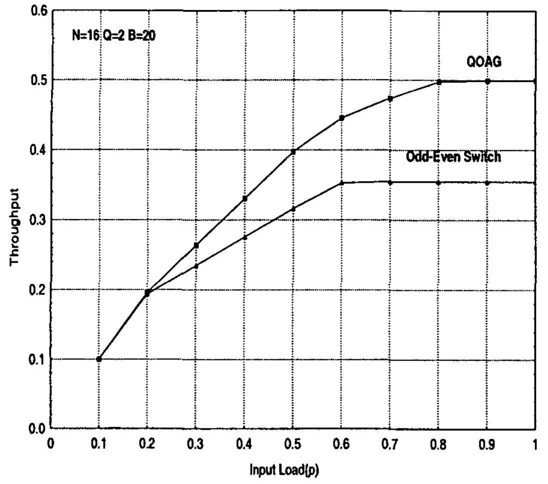

Fig. 4. Throughput vs input load p for a 16×16 switch, where each input queue size is $B = 20$ and the number of queues at each input port is $Q = 2$.

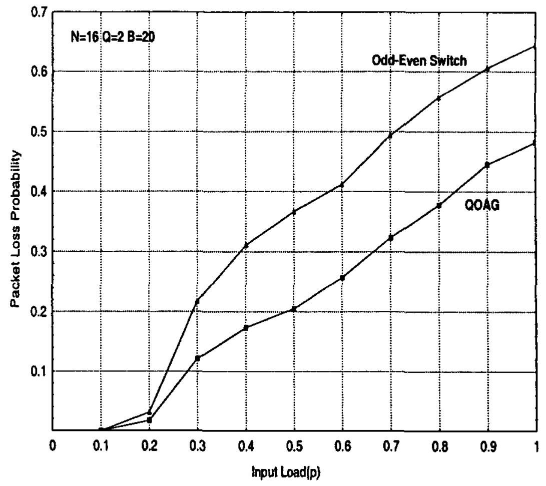

Fig. 5. Packet loss probability vs input load p for a 16×16 switch. Each input queue size is $B = 20$ and the number of queues at each input port is $Q = 2$.

packet delay. For $p > 0.5$, the queues with packets destined to other output addresses begin to build up as well, the HOL blocking increases and causes the overall mean packet delay to increase again. For $p = 0.7$, the mean packet delay is 58.1 time slots using Queueing with Output Address Grouping (QOAG) and 91.2 time slots using Odd-Even queueing policy. A 57% cut in mean packet delay is achieved.

Figs. 4 and 5 show the throughput and packet loss probability versus input load p. For both throughput and packet loss probability, QOAG achieves better performance than the Odd-Even queueing policy. At $p = 0.7$, the throughput is 0.474 for QOAG and 0.355 for Odd-Even queueing, a 33.5% increase in throughput. At the same load, the packet loss probabilities are 0.323 for QOAG and 0.493 for Odd-Even queueing. This gives a 52% cut in packet loss.

As we discussed before, an efficient queueing policy should be able to adapt to traffic variations. We change the distribution of the output addresses at time slot 5000 and we trace the resulting changes in packet loss probability for both QOAG in Fig. 6. The distribution of output addresses after the change is shown in Table 2. The input load p is fixed at 0.8. Each point shown in Fig. 6 is the average packet loss probability for every 5 consecutive time slots. After the output address change at time slot 5000, we can see that QOAG quickly adapts to the change by converging to the previous packet loss probability level.

We also evaluate the performance of QOAG and Odd-Even queueing policy when the buffer size B for each input port is infinite. Fig. 7 shows the mean packet delay against the input load p. At $p = 0.19$, the mean packet delay is 22.5 time slots for QOAG and 37.21 time slots for Odd-Even queueing policy. A 65% cut in mean packet delay is achieved using QOAG. Fig. 9 shows the throughput against the input load p. Again, QOAG outperforms the Odd-Even queueing policy. At input load $p = 0.6$, the throughput is 0.453 for QOAG and 0.355 for Odd-Even queueing policy. A 27% increase in throughput is obtained.

Output address	*Access Probability*
1	0.1054
2	0.0277
3	0.0624
4	0.0290
5	0.0536
6	0.3299
7	0.0430
8	0.0342
9	0.0255
10	0.0305
11	0.0766
12	0.0475
13	0.0322
14	0.0265
15	0.0342
16	0.1054

TABLE II
The distribution of output addresses used after time slot 5000.

V. Conclusions

In this paper, an efficient self-adaptive queueing policy, called Queueing with Output Address Grouping (QOAG), was proposed for input-buffered packet switches. Using QOAG, an incoming packet with the same output address as some packets in the backlog will be assigned to the same input queue as those backlog packets. Otherwise, it is assigned to the shortest queue.

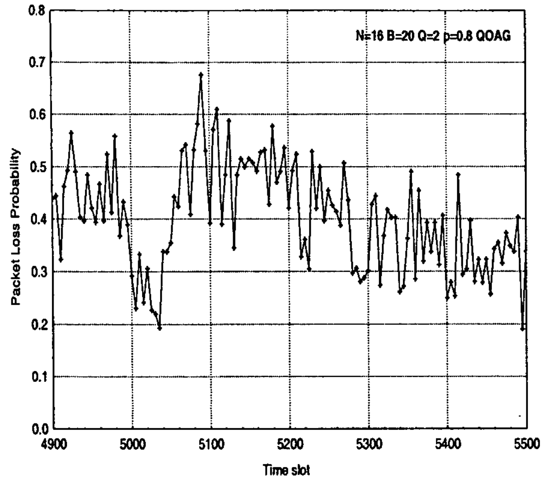

Fig. 6. Packet loss probability from time slots 4900 to 5500 using QOAG for a 16 × 16 switch with p = 0.8, B = 20 and Q = 2. The packet loss probability is taken by averaging over every 5 consecutive time slots.

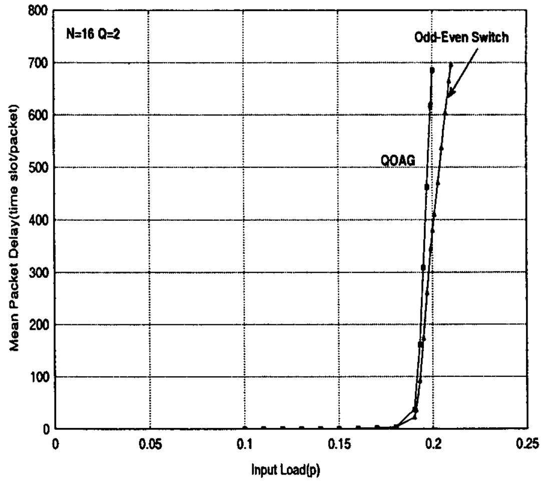

Fig. 7. Mean packet delay vs input load p for a 16×16 switch with infinite buffer at each input queue, and the number of queues at each input port is 2.

The performance of QOAG was compared with Odd-Even queueing policy under nonuniformly distributed packet output addresses. We found that QOAG outperforms the Odd-Even queueing with respect to mean packet delay, throughput and packet loss probability. We also showed that QOAG is adaptive to traffic variations. As a future work, efforts should focus on the the performance of using QOAG for providing quality of service guarantees for different classes of traffic.

REFERENCES

[1] C. Kolias and L. Kleinrock, "The odd-even queueing ATM switch: performance evaluation," *Proceedings of IEEE ICC '96*, pp. 1674-1679 1996.

[2] D. E. Knuth, "The art of computer programming, volume 3," *Second Edition*, Addison-Wesley 1981.

[3] U. Black, *ATM: Foundation For Broadband Networks*. Prentice Hall, 1993.

[4] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queuing on a space division switch," *IEEE Trans. on Commun.*, Vol. 35, pp.1347-1356, Dec. 1987.

[5] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans, on Computer Systems*, pp. 319-352 Nov 1993.

[6] M. Karol, K. Eng, and H. Obara, "Improving the performance of input-queued ATM packet switches," *Proceedings of IEEE INFOCOM '92*, pp. 110-115 1992.

[7] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. on Commun.*, Vol. COM-27, No. 10, pp. 1449-1455, Oct. 1979.

[8] M. Karol and M. Hluchyj, "Queueing in high-performance packet-switching," *IEEE J. Selected Areas Commun.*, vol. 6, pp. 1587-1597 Dec. 1988.

[9] N. McKeown and A. Mekkittikul, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," *Proceedings of IEEE INFOCOM '98*, 1998.

[10] R. K. Ahuja, T. L. Magannti, and J. B. Orlin, "Network Flows: Theory, Algorithms, and Application," *Englewood Cliffs*, NJ: Prentice-Hall 1993.

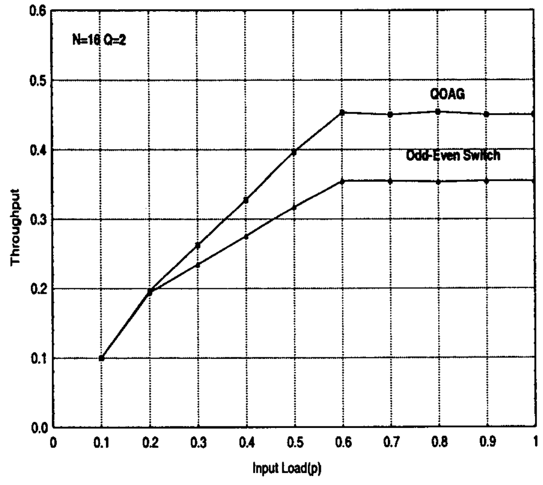

Fig. 8. Throughput vs input load p for a 16×16 switch with infinite buffer at each input queue, and the number of queues at each input port is 2.