CONFERENCE RECORD *Special Section on the APCC Proceedings*

Quasi-Shared Output Buffer Switch

Seung Yeob Nam[†], Kyu Seek Sohn[†], Dan Keun Sung[†], and Yoon-Young An[††]

Summary
One major drawback of conventional output buffered switches is that the speed of writing cells into output buffer should be N times faster than input link speed. This paper proposes a new output buffer switch that divides one output buffer into several buffers and virtually shares the separated buffers by using a distributor for each output port. These separated buffers can lower the memory speed. Since each input has its own cell path independent of other inputs, cell contentions do not occur at the input ports and no speed-up is needed. The performance of the proposed switch is evaluated in terms of the average cell latency compared with the conventional input buffered switches which use the iSLIP and wrapped wave front arbitration (WWFA) algorithms.

Key words:
ATM switches, Quality of Service (QoS), Output buffer switch, average cell latency.

1.Introduction

Recently, input queuing schemes have been adopted in many high-speed switching systems [1][2] because they can increase the throughput of input-buffered switches from 58.6% to 100% using a virtual output queuing (VOQ) scheme [3][4] for uniform random input traffic.

Even if input buffered switches have virtual output queues at each input port, there remains a problem that a HOL cell belonging to a VOQ may contend with the other HOL cells belonging to different VOQs because even though several HOL cells want to pass through the switch fabric, only one cell from each input can enter the switch fabric during one cell time. This contention is called the input contention. In addition the cells entering the switch fabric from different input ports may be destined to the same output port, but only one cell is permitted to the output port during one cell time. Thus, cell contentions called the output contention occur at the output port of the switch fabric. In general, in order to solve the contentions at the input and output ports and to increase the utilization of the switch, proper arbitration algorithms are used. The demand for the guarantee of quality of service (QoS) gradually increases. In order to support various QoS, the arbitration algorithms require to discriminate different service classes. In input buffered switches cell delay occurs at the input buffer, and the service order of each cell is determined by the designated arbitration algorithm. Thus, in order to support different delay QoS the arbiter should determine the service order considering QoS. If QoS, input/output contentions, and throughput are considered in an arbiter, the complexity of the arbitration algorithm becomes much higher. If this complexity results in long computation time, the algorithm cannot be applied to high speed switching.

Output buffered switches usually use internal speed-up to transfer cells, and thus, no cell has to wait at the input and input contentions do not occur. A scheduling algorithm at an output buffer solves output contentions totally independent of other output buffers. Distribution and independence makes it easier to implement the scheduling algorithm supporting QoS.

Taking both benefits of input buffered switches and output buffered switches into account, we propose a new switch architecture whose operation is logically similar to output buffered switches with lower speed-up. The performance of the proposed switch is evaluated compared with the conventional input buffered switches which use iSLIP [5] and wrapped wave front arbitration (WWFA) [6] algorithms yielding 100% throughput.

2. Quasi-Shared Output Buffer Type Switch Architecture

A. Basic Architecture of Quasi-Shared Output Buffer Type Switch

Fig 1. shows an N x M switch architecture which operates like an output buffered switch without internal speed-up. Each input port has a dedicated routing block, and each output port has a scheduler. There are several output buffers storing cells according to their input port before the scheduler, and there are filters passing the designated cells to the corresponding buffers. Each routing block is connected to the corresponding filters via bus.

The specific operation is described as follows: If a cell arrives at an input port, the routing block of the input port receives the cell, looks into the cell header to determine the destined output port, and attaches a routing tag to the cell.

Manuscript revised March 31, 2001.
[†] The authors are with EECS, KAIST, Guseong-dong, Yuseong-gu, Deajon, 305-701, Korea
[††] The author is with ETRI, Gajeong-dong, Yuseong-gu, Deajon, 305-350, Korea

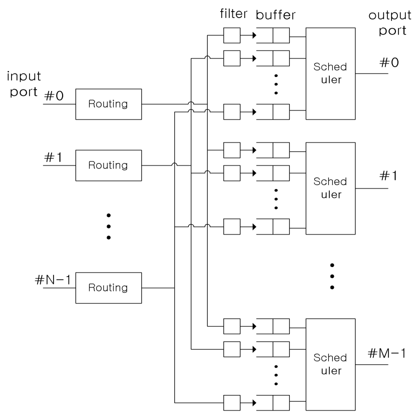

Fig. 1. Distributed Buffer Type Switch Architecture

The routing tag consists of a *M*-bit bitmap and priority or class field. The *i*-th lowest bit of the bitmap is 1 when the cell is destined to the *i*-th output port and 0, otherwise. The priority or class field represents the service class or priority class of the connection of the cell. After the routing tag is attached to the cell at the routing block, the cell is sent toward the corresponding output buffer of every output port. The filter examines the routing tag and passes the cell which is destined to the output to the connected output buffer. The output port has the same number of output buffers as the number of input ports and the *i*-th output buffer receives cells from only the *i*-th input port.

There is a unique bus connecting the *i*-th input port with the *i*-th output buffers of all output ports for each *i* (*i*=0,1,…,*N*-1). Since all buses are separated, the output buffers of each output port can accept up to *N* cells in parallel. These parallel cell paths make it possible to transfer all arriving cells to output buffers without queuing delay at the input ports. Since all arriving cells are destined to the corresponding output ports without loss through parallel cell paths, internal speed-up is not needed and there is no input contention problem because no cells wait at the input ports. The input speed of each memory is not higher than the input link speed. The scheduling algorithm used in each output port can individually operate regardless of other ports.

One drawback of this switch is that the number of required buffers is *NM*, and thus, for large switches the number of buffers can be quite large. Fig. 2 shows a switch architecture which improves this scalability problem. The operations are similar to the switch of Fig. 1, but the difference is that there is a distributor between filters and output buffers for each output port. The distributor concentrates at most *N* input cells and distributes them over *L* output buffers evenly. It is possible to reduce the total number of buffers by decreasing *L*. However, there is a relation between the number of buffers per output port *L* and memory write speed *R* as follows:

$$L \text{ x } R = N \text{ x } \{\text{the input link speed}\}$$

Since the number of memory blocks per port is inversely proportional to the memory speed, if the number of memory blocks per port decreases, then higher speed memory is needed. Thus, there is a trade-off between the number of memory blocks per port and the memory speed.

If the distributor concentrates arriving cells and distributes them over L buffers in a round-robin fashion and the scheduler selects HOL cells in L buffers in a round-robin scheme, the difference between the number of cells in one buffer and the number of cells in another buffer does not exceed 2. If we use L buffers, it is not different from using one shared-memory of the size of L buffers from the aspect of buffer size. Therefore, all the buffers seem to be virtually shared and can be utilized efficiently. Since dividing one output buffer into several output memory blocks can lower the memory speed, the proposed quasi-shared output buffer type switch performs better than for conventional output buffered switches.

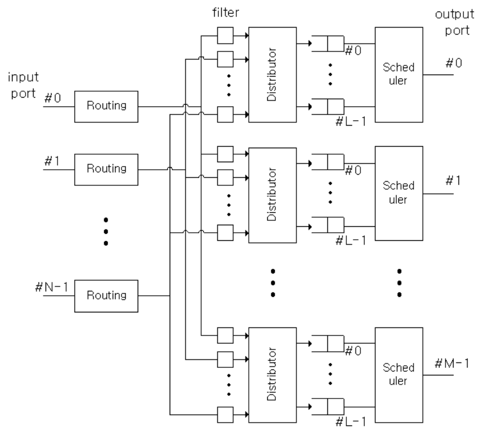

Fig. 2. Quasi-Shared Output Buffer Type Switch Architecture

B. Quasi-Shared Output Buffer Type Switch Considering QoS

If the distributor concentrates input cells and distributes them over *L* output buffers in a round-robin manner and the scheduler serves *L* buffers in a round-robin manner, the operation of a quasi-shared output buffer type switch in Fig. 2 is logically identical to the operation of output buffered switches which have a single output buffer operating as a first in first out (FIFO) queue per output port.

In real networks there are various types of traffic and users may require several different QoS. In order to provide various QoS for users with their contracts, the scheduler at the output port requires to support more efficient scheduling algorithms than the conventional FIFO scheme.

However, in the switch architecture of Fig. 2 cells belonging to the same connection are delivered to different output buffers, which makes it difficult to manage cells according to their connections or their classes. Thus, it is difficult for the switch shown in Fig. 2 to support the scheduling algorithm different from the conventional FIFO scheme.

Fig. 3 is the improved switch architecture considering QoS support. The first difference between the quasi-shared output buffer type switch of Fig. 2 and the improved switch of Fig. 3 is that the distributor of Fig. 2 concentrates input cells and distributes them over several output buffers in a round-robin fashion. However, in Fig. 3 the scheduler does not select the cell to obtain service in a round-robin scheme resulting in the difference of buffer occupancies over 2. Therefore, the distributor of Fig. 3 maintains cell occupancy of every buffer and distributes input cells to the buffer of low occupancy to utilize every buffer efficiently.

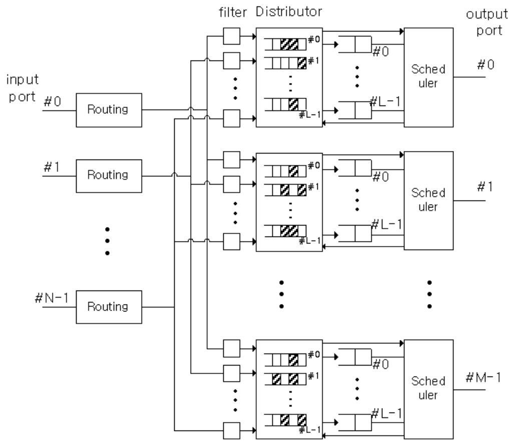

Fig. 3. Quasi-Shared Output Buffer Type Switch considering QoS

There is no difference between the procedures of Figs. 2 and 3 from the cell arrival to the filter operation. The procedure from the cell arrival at the distributor to the service of the cell at the scheduler is described in detail:

i) As shown in Fig. 3, the distributor manages all buffer occupancies and idle memory addresses of each buffer

ii) If the distributor receives a cell or multiple cells, the distributor selects the buffer which has the smallest number of cells. If the number of cells is more than one, it selects as many buffers of lowest occupancies as the number of received cells.

iii) The distributor selects idle memory address/addresses from the selected buffer/buffers.

iv) The distributor stores the cells at the selected idle memory addresses and sends the information about the cell headers and stored memory addresses to the scheduler.

v) The scheduler receives the information about the stored cells, determines service class or virtual connection information from the cell headers, and manages cell addresses as a per-class queue or a per-VC queue.

vi) The scheduler selects one queue to be served using a scheduling algorithm that can support QoS such as weighted fair queuing (WFQ) and self clocked fair queuing (SCFQ).

vii) The scheduler obtains the memory address where the head-of-line cell of the selected queue is stored from the per-VC/per-class queue and selects a cell to be delivered to the output port.

viii) The scheduler notifies the distributor of the memory address of the cell, and then the distributor updates the buffer occupancy and idle memory address list.

ix) If new cells arrive at the distributor next cell time, the sequence of operations is repeated from procedure ii).

The distributor manages the idle addresses and occupancies of all RAM buffers and the scheduler manages per-VC or per-class queue, and thus, any scheduling algorithms can be supported at this switch architecture. The scheduling algorithm of one output port solves output contentions of its own port regardless of the other output ports. However, for the input buffered switches the output contention problem is correlated with the input contention problem. Hence, the scheduling algorithm can be easily implemented at the switch architecture shown in Fig. 3 compared with the input buffered switches.

3. Performance Evaluation

A. Simulation Environment

The performance of the proposed switch is evaluated through simulation for an 8 x 8 switch. One traffic source is connected to each input port and the destined output ports of generated cells are randomly selected among 8 output ports.

Input traffic models for simulation include random and bursty traffic. For random traffic cell arrivals at each input port are generated according to a Bernoulli process with parameter *p*, where *p* is the offered load per each input port. Bursty traffic is modeled by an on-off arrival process where the on and off interval lengths are exponentially distributed with different parameter. The source alternately produces a burst of cells followed by an idle period of no cell. During on period cells are generated at the link rate and the destined output ports of the cells belonging to the same on period are all identical. The average burst length is set at 16 cells.

B. Simulation Results

The proposed quasi-shared output buffer type switch is evaluated compared with the input buffered switch which uses iSLIP and WWFA arbitration algorithms. Fig. 4 compares the average cell latency at the quasi-shared output buffer type switch and the input buffered switches as a function of offered load for random traffic. The latency of the input buffered switch using iSLIP with a single iteration is longest among them, and the latency of the quasi-shared output buffer type switch is the shortest. However, as the number of iterations increases, the performance of the iSLIP algorithm improves to a level similar to the performance of the input switch using WWFA.

Fig. 5 compares the average cell latency of the quasi-shared output buffer type switch and the input buffered switches in a bursty traffic load environment. We can observe a similar trend to Fig. 4. The input buffered switch using iSLIP with a single iteration yields the worst performance. The quasi-shared output buffer type switch yields the best performance. The performance of the 3-SLIP algorithm is almost identical to that of the WWFA algorithm. The quasi-shared output buffer type switch yields the best performance in term of the average cell latency because the operation of the quasi-shared output buffer type switch is logically identical to the output buffered switch which has a single FIFO queue per output port as described in Section 2. Since input buffered switches have a cell contention problem at input and output ports, the performance of input buffered switches can not exceed the performance of output buffered switches without input contentions.

4. Conclusions

In this paper we proposed a switch architecture with no input contentions by setting independent path from each input port to every output port. This switch can send the cells arriving at all input ports to their destined output buffers through parallel paths promptly without queuing at the input ports, but does not require internal speed-up.

The proposed switch is logically identical to the output buffered switch with a single FIFO queue per output port. The proposed switch can reduce the memory speed by increasing the number of memory blocks per output port. Hence, the proposed quasi-shared output buffer type switch can be applied to high speed ATM switching systems. The quasi-shared output buffer type switch can be extended to a structure that can support per-VC or per-class scheduling.
The simulation result shows that the quasi-shared output buffer type switch yields better performance in terms of the average cell latency compared with the input buffered switches which use iSLIP and WWFA arbitration algorithms.

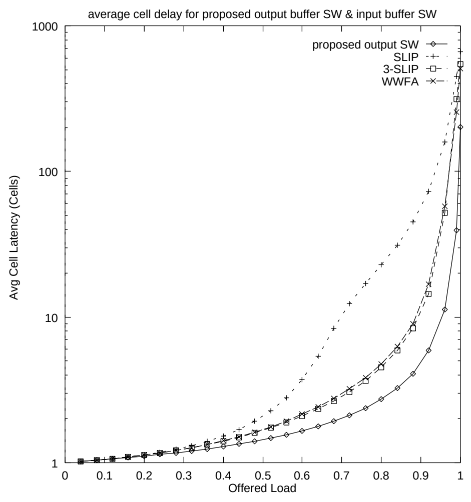

Fig. 4. Comparison of quasi-shared output buffer type switch and input buffered switch using iSLIP or WWFA under random traffic load

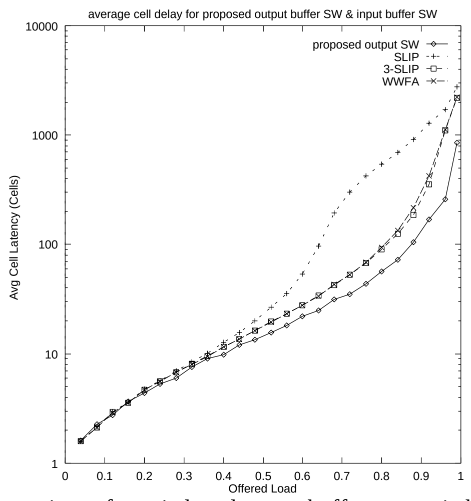

Fig. 5. Comparison of quasi-shared output buffer type switch and input buffered switch using iSLIP or WWFA under bursty traffic load

References

[1] N. McKeown, M. Izzard,, A. Mekkittikul, B. Ellersick, and M. Horowitz, “The tiny tera: A small high-bandwidth packet switch core,” IEEE Micro, vol. 17, pp. 26-33, Jan.-Feb. 1997.

[2] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, “High speed switch scheduling for local area networks,” ACM Trans. Comput. Syst., vol. 11, no. 4, pp. 319-352, Nov. 1993.

[3] N. McKeown, V. Anantharam, and J. Walrand, “Achieving 100% throughput in an input-queued switch,” in Proc. IEEE INFOCOM’96, San Francisco, CA, pp. 296-302.

[4] A. Mekkittikul and N. McKeown, “A practical scheduling algorithm for achieving 100% throughput in input-queued switches,” in Proc. IEEE INFOCOM’98, San Francisco, CA, vol. 2, pp. 792-799.

[5] N. McKeown, “The iSLIP Scheduling Algorithm for Input-Queued Switches,” IEEE/ACM Trans. On Networking, vol. 7, no. 2, pp. 188-201, Apr. 1999.

[6] Hsin-Chou Chi and Yuval Tamir, “Decomposed Arbiters for Large Crossbars with Multi-Queue Input Buffers,” IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp. 233-238, 1991.