ISSN No. 0976-5697

Volume 2, No. 1, Jan-Feb 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Scheduling in Frame Based Packet Mode for Queued Switches

Mrs. S. A. Ade
Deptt. of Computer Science & Engeering
Sipna's College of Engg. & Tech.
Amaravati,M.S. , India
shobha_chavan@rediffmail.com

Prof. P. A. Tijare
Deptt. of Computer Science & Engeering
Sipna's College of Engg. & Tech
Amaravati,M.S. , India
pritishtijare@rediffmail.com

Abstract: Proportional fair bandwidth allocation in packet switches is a fundamental issue for quality of service (QoS) support in IP networks. Input-queued switches performing packet-mode scheduling deliver all the segments of a packet contiguously from the input port to the output port, thus greatly simplify the output reassembly and yield performance advantage over switches with cell-mode scheduling under certain conditions. An important issue of packet-mode scheduling is how to achieve fair bandwidth allocation among flows with different packet sizes. Most packet scheduling algorithms for input-queued switches operate on fixed-sized packets known as cells. In reality, communication traffic in many systems such as Internet runs on variable-sized packets. Motivated by potential savings of segmentation and reassembly, there has been increasing interest in scheduling variable-sized packets in a non preemptive manner known as packet-mode scheduling. The dissertation work will study frame-based packet-mode scheduling for better scalability. The relation between the frame size and packet sizes will be derived and the speedup will be analyzed.

Keywords: Cell-mode scheduling, frame-based scheduling, input-queued switch, packet-mode scheduling

I. INTRODUCTION

Input-queued switches are able to overcome head-of-line (HOL) blocking and achieve high throughput by using virtual output queueing (VOQ) [1] and efficient scheduling algorithms. In this case, the buffers and control logic work at the line speed [2], which enables such architecture to be scaled up to backbone routers with very high speed, as realized in Tiny Tera [3], Cisco GSR [4], Lucent GRF [5], etc. Although Internet Protocol (IP) packets have variable lengths, switches are usually operated in fixed-size cells internally. Figure 1 shows a typical architecture, where IP packets are segmented into cells in the input ports, switched in the crossbar, and reassembled in the output ports. Cell delivery in the crossbar is controlled by a scheduler that calculates a bipartite match from the input ports to the output ones according to the backlog in the VOQs.

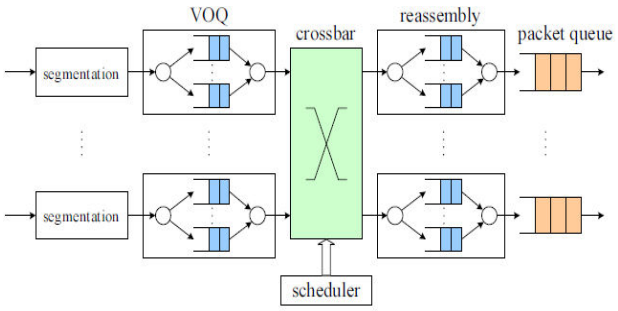

Fig. 1 Architecture of input-queued packet switch.

If all the input-output matches are recalculated in each slot, the algorithm is called cell-mode scheduling. In this case, cells from different input ports may interleaved in an output port, thus reassembly modules are needed. On the other hand, if an input-output match is kept until all the cells of a packet are delivered, the algorithm is called packet-mode scheduling. In this case, cells belonging to the same packet arrive at the output port contiguously, thus the reassembly is greatly simplified with savings in both complexity and memory. It has been shown that packet-mode scheduling achieves 100% throughput and yields performance advantage over cell-mode scheduling in terms of packet delay under certain conditions [1][6].

To support quality of service (QoS) in IP networks, schedulers are required to perform proportional fair bandwidth allocation among competing flows, which means: first, a flow with arrival rate less than its reservation is fully served; second, excess bandwidth (not used up by light load flows) is allocated among heavy load flows proportionally to their reserved shares [7]. Fairness mechanism is also important in best-effort networks where misbehaving flows should be prevented from grabbing too much bandwidth so as to protect normal ones. Although fair scheduling has been intensively investigated in output-queued switches, it is non-trivial for input-queued architecture to achieve good fairness and high throughput simultaneously since a bipartite match conforming the bandwidth regulation may not guarantee throughput performance, and vice versa [8].

A. IQ packet switches

logical architecture for an IQ packet switch is shown in Fig. 2. At each input an Input Segmentation Module (ISM) segments the incoming packet into cells. PLS is the external packet line speed. Since the ISM operates in store-and-forward mode, it must be equipped with enough memory to store a maximum size packet, and the segmentation process starts only after the complete reception of the packet. The cells resulting from the segmentation are transferred to the cell-switch input at a speed (called ILS) equal to the line speed PLS incremented to account for segmentation overheads. The capacity of input queues at the cell-switch is limited to Q_{max}, hence losses can occur. We assume that the entire packet is discarded if the input queue of the cell-switch does not have enough free space to store all the cells

deriving from the segmentation of the packet *when the first of these cells hits the queue.* This is of course a pessimistic assumption, but has the advantage of ease of implementation, and of avoiding the transmission of incomplete packet fractions through the switch.

The cell-based switching fabric transfers cells from input to output queues, according to a scheduling algorithm, as previously discussed. These cells are delivered to the Output Reassembly Module (ORM) at speed ILS. Here packets (i.e., IP datagrams) are reassembled. In general, cells belonging to different packets can be interleaved at the same output, hence more than one reassembly machine can be active in the same ORM. However, at most one cell reaches each ORM in a slot time, hence at most one packet is completed at each ORM in a slot. Once a packet is complete, it is logically added to an output packet queue, called *packet FIFO* in the figure, from which packets are sequentially transmitted onto the output line. Note that the *packet FIFO* functionality is typically implemented by imposing a sequential transfer from the suitable ORM to the output line of all the cells belonging to the same reassembled packet. No internal speedup is required to support ISM and ORM, but for compensating internal overheads. It is possible to further simplify the structure of the switch, and to improve its performance, by enforcing additional constraints on the scheduling algorithm. Indeed, the cells belonging to the same packet are contiguous in the input queue of the internal cell-switch. It is possible to modify some well-known scheduling algorithms in such a way that, once the transfer through the switching fabric of the first cell of a packet has started towards the corresponding output port, no cells belonging to other packets can be transferred to that output. In such a way, cells belonging to the same packet are kept contiguous also in the output queue, and the ORM modules are not necessary any longer (or at most one per output is used). We call this class of scheduling algorithms *packet-mode scheduling*. Note that enforcing packet contiguity is not possible in OQ switches, where cell interleaving in output queues cannot be avoided.

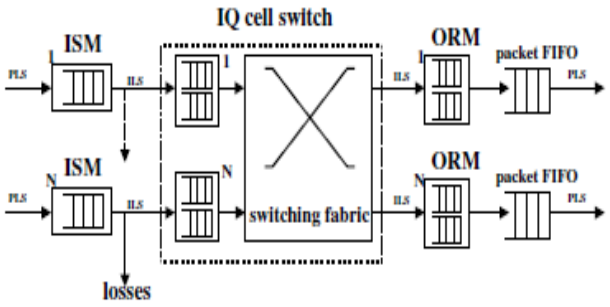

Figure. 2. Logical architecture for an IQ packet switch

B. OQ packet switches

Fig. 3 shows the logical architecture of an OQ packet switch. Packets arrive at input ports where they are segmented by ISM modules, similarly to what happens in IQ routers. The cells obtained with the segmentation are sent to the cell-switch at speed ILS, and are immediately transferred to the output queues of the cell-switch, thanks to a speed-up equal to N in the switching fabric. Losses may occur at output queues, whose capacity is limited to N X Q_{max} for each queue, since we are assuming the same buffering capacity for OQ and IQ switches. From the output queues of the cell-switch, cells are delivered at speed ILS to an ORM module for reassembly.

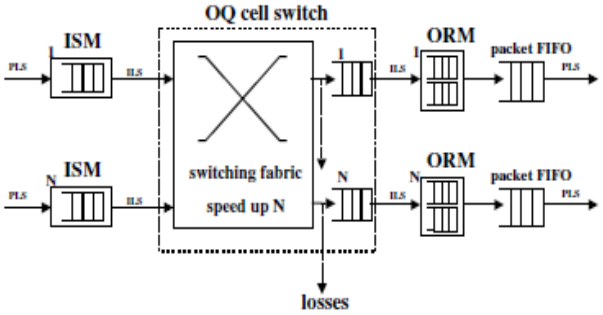

Figure. 3. Logical architecture for an OQ packet switch

The discarded cell may be in the output queue, or already in the ORM. We assume to be unable to identify and discard the other cells in the output queue: those cells will be discarded by the ORM module, which has knowledge of the existence of the packet. This means that cells belonging to packets that suffered partial losses unnecessarily use system resources. The IQ architecture has the advantage that all the cells belonging to the same packet are contiguous in input queues, where losses occur. In the OQ case, losses occur in queues where cells of different packets are interleaved.

II. RELATED WORK

Since switches are operated in cells internally, the time axis can be divided into fixed-length cell slots, where each slot allows at most one cell to be delivered from/to an input/output port. Note that an input may have cells destined to multiple outputs and an output may be competed by multiple inputs, a scheduler is needed to find a bipartite match that establishes one-to-one match pairs from inputs to outputs. A cell-mode scheduler releases *all the match pairs* and performs recalculation in *each slot*. This type of scheduling has been intensively researched in literature such as *i*SLIP [9], oldest cell first (OCF), longest queue first (LQF) [10], parallel iterative matching (PIM) [11], and dual round-robin matching (DRRM) [1]. Since cells from different packets may be interleaved in output ports, cell-mode scheduling introduces additional complexity of packet reassembly in output ports when it is applied to packet switches. Packet-mode schedulers differ from cell-mode ones in that a match pair is *kept unchanged* until all the cells belonging to the same packet are delivered. In each slot, only the idle unmatched ports and those that just delivered a complete packet in the previous slot are taken for match recalculation [1]. This mechanism guarantees that cells belonging to a single packet arrive at the destination output port contiguously, thus greatly facilitates output assembly.

A. Cell-Mode and Packet-Mode Scheduling

Since switches are operated in cells internally, the time axis can be divided into fixed-length cell slots, where each slot allows at most one cell to be delivered from/to an input/output port. Note that an input may have cells destined to multiple outputs and an output may be competed by multiple inputs, a scheduler is needed to find a bipartite match that establishes one-to-one match pairs from inputs to outputs.

A cell-mode scheduler releases *all the match pairs* and performs recalculation in *each slot*. This type of scheduling has been intensively researched in literature, such as *i*SLIP, oldest cell first (OCF), longest queue first (LQF), parallel iterative matching (PIM), and dual round-robin matching (DRRM). Since cells from different packets may be interleaved in output ports, cell-mode scheduling introduces additional complexity of packet reassembly in output ports when it is applied to packet switches.

Packet-mode schedulers differ from cell-mode ones in that a match pair is *kept unchanged* until all the cells belonging to the same packet are delivered. In each slot, only the idle unmatched ports and those that just delivered a complete packet in the previous slot are taken for match recalculation. This mechanism guarantees that cells belonging to a single packet arrive at the destination output port contiguously, thus greatly facilitates output assembly. It has been shown that this type of scheduling achieves 100% throughput and yields low average delay. The properties of low complexity and good performance make packet-mode scheduling a strong candidate for high speed IP routers.

B. Fair Scheduling in Input-Queued Switches

Existing research on fair scheduling mainly focuses on output-queued switches, where only the output ports experience contention. Several algorithms have been proposed, such as generalized processor sharing (GPS), deficit round-robin (DRR) and elastic round-robin (ERR). Input- queued switches differ from output-queued ones in that contention takes place at both input and output ports, thus makes it difficult to achieve high throughput and good fairness simultaneously. Most existing algorithms are designed for cell-mode schedulers. Weighted PIM (WPIM) employs weight-based masks in the matching process and guarantees the reserved bandwidth of each flow. However, the excess bandwidths are allocated equally (rather than proportionally) among the competing flows. Based on GPS, iterative fair scheduling (*i*FS) derives a virtual time stamp for each arriving cell and calculates bipartite match according to the time stamps of the HOL packets in the VOQs. It is well know that GPS-based algorithms suffer from the complexity of time stamp calculation and sorting, which makes *i*FS unsuitable for high speed switches.

III. ANALYSIS OF PROBLEM

Datagram networks have long suffered from performance degradation in the presence of congestion [GerBO]. The rapid growth, in both use and size, of computer networks has sparked a renewed interest in methods of congestion control [DEC87abcd,Jac88a, Man89, Nag871. These methods have two points of implementation. The first is at the source, where flow control algorithms vary the rate at which the source sends packets, Of course, flow control algorithms are designed primarily to ensure the presence of free buffers at the destination host, but we are more concerned with their role in limiting the overall network traffic. Queuing algorithms can be thought of as allocating three nearly independent quantities: bandwidth (which packets get transmitted), promptness (when do those packets get transmitted), and buffer space (which packets are discarded by the gateway).Currently, the most common queuing algorithm is first-come-first-serve (FCFS).

This paper investigates the cost of packet-mode scheduling for the combined input output queued (CIOQ) switch architecture. We devise frame-based schedulers that allow packet-mode CIOQ switch with small speedup to mimic an ideal output-queued switch, with bounded relative queuing delay. The schedulers are pipelined and are based on matrix decomposition. Our schedulers demonstrate a trade-off between the switch speedup and the relative queuing delay incurred while mimicking an output-queued switch. When the switch is allowed to incur high relative queuing delay, a speedup arbitrarily close to 2 suffices to mimic an ideal output-queued switch. This implies that packet mode scheduling does not require higher speedup than a cell-based scheduler.

IV. PROPOSED WORK

A. Packet-Mode Scheduling

This algorithm considers bandwidth allocation for coarse grain flows and leaves fine grain control to line cards before packets are segmented and put into VOQs. The concepts of flow and proportional fairness in this algorithm are defined in the following:

Definition 1 **:** In an N X N switch, flow $F_{i;j}$ is defined to be the sequence of packets from input i to output j, where $i; j = 0, 1, ..., N-1$.

Definition 2 **:** Denote the arrival rate of $F_{i;j}$ with $a_{i;j}$, let $b_{i;j}$ and $b^*_{i;j}$ stand for the reserved and allocated bandwidth of $F_{i;j}$, respectively, the proportional fairness factor of the scheduling algorithm is defined as

$$f = \max_{i,j,m,n} \left(\left| \frac{b^*_{i,j}}{b_{i,j}} - \frac{b^*_{m,n}}{b_{m,n}} \right| e_{i,j} e_{m,n} \right)$$

where

$$e_{i,j} = \begin{cases} 0 & \text{if } a_{i,j} \le b^*_{i,j} \\ 1 & \text{otherwise} \end{cases}, \quad i,j = 0,\ldots,N-1$$

Since the aggregated reservation of a link never exceeds its capacity, $f = 0$ indicates that the light load flows ($a_{i;j} <= b^*_{i;j}$) are fully served and the excess bandwidth (if non-zero) is allocated among the heavy load ones ($a_{i;j} > b^*_{i;j}$) proportionally to their reservations.

Algorithm is based on three steps: *request* → *grant* → *accept*, where fairness mechanism is introduced in the grant and the accept steps to control the bipartite matching. The grant arbiter G_j in output j and the accept arbiter A_i in input i have the same structure and maintain pointers to indicate port index from 0 to $N – 1$.

request **:** Each unmatched input i sends a request to every output j if the queue of $F_{i;j}$ is non-empty;

grant **:** If output j is unmatched, G_j is activated to select an input from the requests as the grant. If the grant is accepted in the next step, pointer g_j is increased one location beyond the selected input.

accept **:** Upon the grants to input i, A_i is used to select an output to generate a match pair, and a_i is increased one location beyond the accepted output.

Once a match pair is established, it will be kept unchanged until a complete packet is delivered. In iterative algorithm, the three steps can be repeated multiple times to improve throughput.

B. Frame Based Packet Mode Scheduling

Consider an N X N input-queued non blocking switch. All inputs and outputs are assumed to run at the same speed. The time is divided into slots and each slot allows a basic data unit (cell) to be transferred from an input port to an output port. The line speed of each port is thus one cell per time slot. A packet may consist of *l* cells that need *l* consecutive slots to complete the transmission. If we use a speedup s, then up to s cells can be transferred from an input port to an output port in one slot. During each slot, the scheduler sets up a switching configuration that connects each input port to exactly one output port and vice versa.

A frame consists of f time slots. Slots in each frame are numbered from 0 to f - 1. Slot k covers time interval (t_k; t_{k+1}). A sequence of consecutive slots k, k+1, ……..,m-1, m, is denoted by slots [k, m]. We associate a triple (i, j, *l*) to each packet if it consists of *l* cells to be transferred from input port i to output port j. Note that multiple packets can have the same triple (i, j, *l*). Let S be a set of packets to be scheduled, introduce the following notations.

Let L be the set of packet sizes that occur in S. For each $i, j (1 \le i, j \le N)$ and $l (l \in L)$ define

$$S_{i,j,l} = \{\text{any packet in } S \text{ that has the same triple} (i, j, l)\};$$

$$S_{i,j} = \bigcup_{l \in L} S_{i,j,l};$$

$$p_{i,j} = \sum_{l \in L} (l \times |S_{i,j,l}|) \text{ (total number of cells in } S_{i,j});$$

$$p_i = \sum_{j=1}^{N} p_{i,j} \text{ (total number of cells at input } i); \text{ and}$$

$$q_j = \sum_{i=1}^{N} p_{i,j} \text{ (total number of cells destined to output } j)$$

Definition 1. Let S be a set of packets. S is called schedulable under the frame-based packet mode if a schedule exists such that every packet can be transmitted within a frame non preemptively without contention.

Definition 2. The schedulability problem under the frame-based packet mode is to determine whether a given packet set S is schedulable under the frame-based packet mode.

Definition 3. Given a packet set S, the minimum finishing time is the minimum number of time slots needed to transmit all packets in S non preemptively without contention.

Finding the minimum finishing time for a packet set S under packet mode is an NP-complete problem when N >= 3. Apparently, finding the minimum finishing time is equivalent to finding a minimum frame size such that set S is schedulable within the frame. Therefore, the schedulability problem under the frame-based packet mode will also be NP-complete if the frame size is variable. Hence, we assume that a fixed frame size be used in the study of frame-based packet-mode scheduling problem. Given a packet set S, the following condition is called admissible condition that is necessary for S to be schedulable in a frame of length f

$$p_i \le f \quad \text{and} \quad q_j \le f \qquad \forall i, j \in \{1, \ldots, N\}$$

This is because the total number of cells arriving at any input port i (or destined to any output port j) cannot be larger than the number f of time slots available in the frame. As a special case, where all packets are cells, the schedulability problem under the packet mode becomes the cell-mode scheduling problem.

C. Speedup for frame –based P acket –mode scheduling

As can be seen from Fig. 1, quite different from cell-mode scheduling, the packet-mode scheduling may not produce a feasible schedule even if the traffic satisfies the admissible condition . Moreover, even a feasible schedule exists, because of the NP-completeness of the general scheduling problem, we cannot find it efficiently. Therefore, using a speed up is a practical and efficient approach. In this section , we show that if the packet set satisfies the admission condition , then all packets can be transmitted within a frame if a speedup of 2 is used .Let S be a packet set that satisfies the admissible condition . We first show a scheduling algorithm without using speedup that can schedule all packets in S into2f time slots. This implies that a speedup of 2 guarantees the schedulability.

In the algorithm, we associate N output ports with arrayR[j](1, . . .,N), where R[j](1≤j≤N), holds an integer between 0 and f - 1 showing the next available slot for output port j. Initially, R[j](1≤j≤N). After we schedule a packet (i,j,l) at slot k, then R[j]=k+l. In addition, we use f sets, P0, P1, . . . , Pf-1, to record which input ports will become available at slot 0, 1, . . . , f - 1, respectively. Obviously, these sets are disjoint. Initially, P0 = {1, 2, . . .,N} and all other sets are empty. The following is a pseudo code of the algorithm.

Algorithm:-

Packet-Mode Reservation-Based Scheduling (S, R, P)
/* S is a given set of packets */
Step 1 /*Initialization */
Set R[i]=0,(1≤j≤N)
Set P0 ={1,2----N}
Set Pk = ø , (1≤k≤ f-1)
Step 2 for k = 0 to f - 1
{do {for each i Є Pk}
do { for each j(1≤j≤N)
do{check an unscheduled packet (i; j; l)
in input port i
if R[j] ≤k
then { schedule a packet (i; j; l) in
slots [k,l+k-1]
R[j] = k+l
Pk= Pk-{i}
Pk+1= Pk+1 U{i}
}
}
}
}
Pk+1 = Pk+1 U P_k /+An input available at slot
k will be also available at
slot k þ+1. */
}
Step 3 End.

The above algorithm guarantees that at each time slot, the transmitting (input, output) pairs form a maximal matching. This is because for any unscheduled packet with triple (i,j,l) at time slot k, the ports i and j cannot be both idle. Otherwise, this packet would have been scheduled at or before slot k because input port i belongs to set Pk and R[j] ≤ k. Since there are at most (f-l) cells contained in other packets in either input i or output j, from the admissible condition, this packet will be delayed by at most2(f-l) slots. The schedule length of above algorithm is hence at most 2f-

1. Since 2f-1/j<2, a speedup of 2 is sufficient to guarantee to transmit all packets within a frame of size f.

V. CONCLUSION

In this paper, we have studied the packet-mode scheduling for an input-queued switch in the frame-based context. It is shown that different from cell-mode scheduling, the admissible condition is only necessary but not sufficient
to guarantee the schedulability for packet-mode scheduling. However, this condition becomes sufficient if a speedup of 2 is used. We also investigated how the nonpreemption in
packet -mode scheduling affects the complexity of the problem. Unlike cell-mode scheduling, the schedulability can be determined efficiently only for limited cases in packet-mode scheduling.

VI. REFERENCES

[1] Jianyu Lou and Xiaojun Shen, Senior Member, IEEE, “Frame-Based Packet-Mode Scheduling for Input-Queued Switches”, IEEE Trans On Computers, Vol. 58, No. 7, Jul 2009.

[2] V. Tabatabaee and L. Tassiulas, “MNCM a new class of e±cient scheduling algorithms for input-buffered switches with no speedup," Proc. IEEE INFOCOM'03, pp.1406-1413, April 2003.

[3] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, “Tiny Tera: A packet switch core," IEEE Micro, vol.17, no.1, pp.27-40, Feb. 1997.

[4] Cisco, \Cisco 12000 Gigabit switch router." [Online] http://www.cisco.com.

[5] Lucent, \GRF-MultiGigabit routers, product overview." [Online] http://www.lucent.com.

[6] M. Rosenblum, M. Goemans, and V. Tarokh, “Universal bounds on bu®er size for packetizing °uid policies in input queued, crossbar switches," Proc. IEEE INFOCOM'04, March 2004.

[7] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm," J. of Internetworking Research and Experience, vol.1, no.1, pp.3-26, June 1990.

[8] N. Ni and L.N. Bhuyan, “Fair scheduling and bu®er management in internet routers," Proc. IEEE INFOCOM'02, pp.1141-1150, June 2002.

[9] H.J. Chao, “Saturn: a Terabit packet switch using dual round-robin," IEEE Commun. Mag., vol.38, no.13, pp.78-84, Dec. 2000.

[10] Y. Li, S. Panwar, and H.J. Chao, “On the performance of a dual round-robin switch," Proc. IEEE INFOCOM'01, pp.1688-1697, April 2001.

[11] G. Nong and M. Hamdi, “Burst-based scheduling algorithms for non-blocking atm switches with multiple input queues," IEEE Commun. Lett., vol.4, no.6, pp.202-204, June 2000.

[12] S.S. Kanhere, H. Sethu, and A.B. Parekh, “Fair and e±cient packet scheduling using elastic round-robin," IEEE Trans. Parallel Distrib. Syst., vol.13, no.3, pp.324-336, March 2002.

[13] A. Kam and K. Siu, “Linear-Complexity Algorithms for QoSSupport in Input-Queued Switches with No Speedup,” IEEE J.Selected Areas in Comm., vol. 17, no. 6, pp. 1040-1056, Jun. 1999.

[14] T. Gonzalez and S. Shani, “Open Shop Scheduling to Mininize Finish Time,” J. ACM, vol. 23, pp. 665-679, 1976.