A Packet-Based Scheduling Algorithm for High-Speed Switches

Hakyong Kim, *Student Memeber, IEEE*, Jangwoo Son, *Non-Member*, and Kiseon Kim *Senior Member, IEEE*

Abstract—**The bulk of the traffic passing through data network is variable-length packets like IP packets and, therefore, efficiently handling variable-length packets is an essential and important issue in designing a high-speed and high-performance switch. Most high-speed switches or routers using virtual output queueing as their contention resolution scheme, switch the variable-length packet in the form of small and fixed-size packets (or mini-packets) as the ATM cell, enjoying benefits in the hardware implementation and the operation speed. That is, the scheduling algorithm used in those switches is based on the cell-by-cell scheduling such as PIM and SLIP. These cell-based scheduling algorithms, however, can not fully guarantee the performance metrics of an application since performance measures relevant to the application are specified by the performance metrics of its data units (packets) rather than individual mini-packets.**

In this paper, we present an efficient algorithm called Packet-Based SLIP (PBSLIP) for scheduling variable-length packets on high-speed packet switches. PBSLIP is a packet-based variation of SLIP proposed by N. McKeown and, therefore, most operations of PBSLIP are as simple as those of SLIP. A conspicuous dissimilarity is that PBSLIP does switch a bunch of mini-packets belonged to a given packet consecutively while SLIP switches mini-packets irrespective of their correlation to adjoining mini-packets. Extensive simulation results demonstrate that PBSLIP could be a potential way to significantly improve the performance of conventional cell-based scheduling algorithms and reduce the control time.

Keywords— **SLIP, PBSLIP, variable-length packet switching.**

I. Introduction

UNIVERSAL usage of the Internet as a general means for telecommunications, works, education, and entertainments, is detonating the amount of Internet traffic to be handled (e.g., transmission and switching) by network equipments. As usage patterns of such data networks as the Internet evolve to include more and more bandwidth-intensive networking applications such as java application, video conferencing, etc., there emerges an acute need for very high-bandwidth transport network facilities. To the end, much work has concentrated on the development of broadband transmission and switching facilities.

Hakyong Kim is with Dept. Information and Communications, Kwang-Ju Institute of Science and Technology (KJIST), 1 Oryong-dong, Buk-gu, Gwangju, 500-712, Korea. E-mail: hykim@ieee.org, Web URL: http://charly.kjist.ac.kr/∼hykim/.

Jangwoo Son is with the Medialincs Co., Ltd., Keopyung B-Town 1F, 203 Nonhyun-dong, Gangnam-gu, Seoul 135-010, S. Korea. He is also the founder of Netmanias (e-mail: webmaster@netmanias.com).

Kiseon Kim is with the Department of Information and Communications, Kwang-Ju Institute of Science and Technology (K-JIST), 1 Oryong- dong, Buk-gu, Gwangju, 500-712 S. Korea, (e-mail: kskim@kjist.ac.kr).

In terms of broadband transmission, wavelength-division multiplexing (WDM) technology is used today to fully exploit the potential bandwidth of optical fiber and optical time-division multiplexing (OTDM) technology is expected to harness the power of WDM transmission in near future [1]. Regarding broadband switching, however, switching is still executed in the electrical domain for the immaturity of optical functionalities such as control and processing.

Therefore, it is a crucial issue to develop high-speed and high-performance switches/routers. In recent, many commercial vendors have developed such switches or routers and are putting them into the market. Many of them uses virtual output-queueing (VOQ) as their contention resolution scheme since VOQ queueing is considered to provide 100% throughput without any internal speedup. The Gigabit switch router in [3] and high-speed ATM switch in [4] are the example using VOQ queueing.

In these switches/routers using VOQ queueing, each input maintains N separate queues each of which stores cells destined for each output port. The selection of cells to be served in each time slot from input ports to output ports is accomplished using a scheduling algorithm which is a key factor in achieving high-performance using these switches or routers. Several algorithms such as parallel iterative matching (PIM) [2], iSLIP [5,6,7], two-dimensional round-robin (2DRR) [8], and reservation with preemption and acknowledgment (RPA) [9] have been proposed in the literature.

All of these scheduling algorithms are based on cell-by-cell scheduling since they are contrived for the ATM technology. (For the reason, they are referred to as the cell-based scheduling algorithm.) Traffic streams in the real world, however, are often characterized by variable-length packets rather than ATM cell-like fixed-sized packets. Therefore, for switches or routers to deal with such variable-length packets as IP packets, we have to choose one of two alternative approaches: The first is to switch variable-length packets as they are and the other is to switch variable-length packets in the form of a small fixed-size packet as the ATM cell.

At a glance, the first approach seems better than the second in terms that the former does not need to divide a variable-length packet into a bunch of small fixed-length packets and reassemble them later. However, handing variable-length packets is a knotty problem in implementing in a real switching system. In this paper, therefore, we study the ways to switch variable-length packets in a switch/router and present a scheduling algorithm effi-

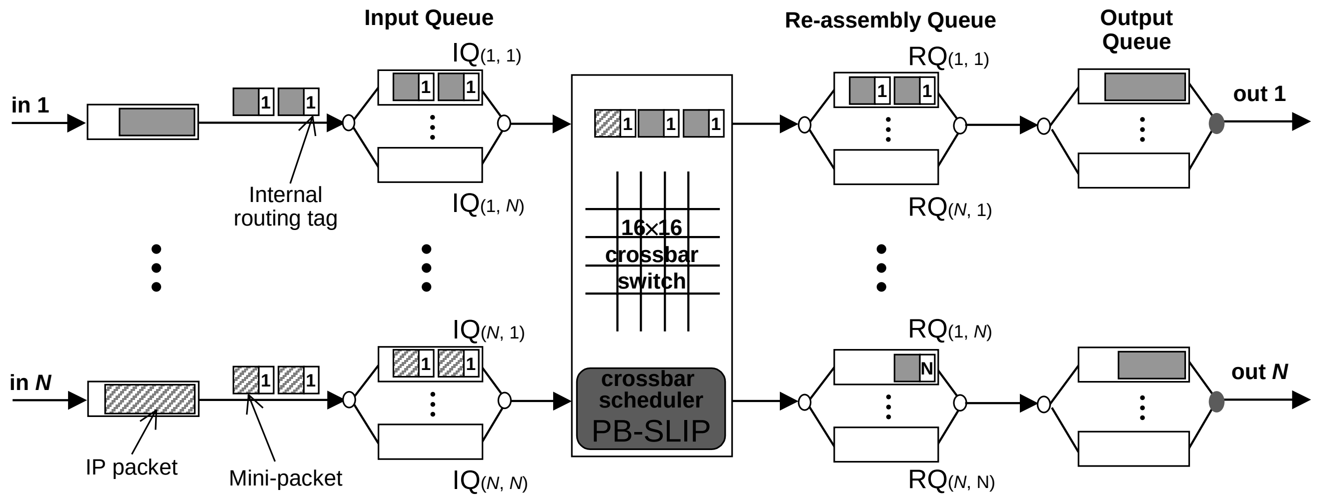

Fig. 1. A virtual output-queued switch architecture for variable-length packet switching.

ciently handling those packets in the form of consecutive fixed-length packets.

The paper is organized as follows: In Section 2, we investigate two ways of dealing with variable-length packets. They are cell-based scheduling and packet-based scheduling, respectively. In Section 3, we propose an efficient scheduling algorithm for variable-length packets based on packet-based scheduling, named the Packet-Based SLIP (PBSLIP) algorithm. In Section 4, performance metrics of PBSLIP are evaluated via extensive simulations. Finally, this paper is concluded in Section 5 with some remarks.

II. Packet-Based Scheduling

The bulk of the traffic passing through data networks is IP packets of variable length and, therefore, handling variable-length packets is an essential and important issue in designing a high-speed and high-performance switch or router. As the ways of switching those packets in a switching node, we may think of two possibilities: The first is to switch variable-length packets as they are as in the early days' router. The second is to switch variable-length packets in the form of a small fixed-size packet as in the ATM switch. This approach is called *cell-based switching.* The switches or routers using the second approach can support high-speed switching since the fast hardware switching is available as in the switching systems.

In this paper, we focus our contentions on the second approach of switching a variable-length packet in the form of a sequence of small and fixed-size packets. i.e., cell-based switching. In order to avoid a terminological confusion between variable-length packet and small fixed-length packet, let *packet* designate the variable-length packet arriving at the switching system and let *mini-packet* or *cell* designate the small fixed-length packet.

Under the cell-based switching, then, a packet arriving at the switching system must be segmented into a sequence of cells or mini-packets as shown in Fig. 1. After packet segmentation, a tag is attached to each cell and used for internal routing. Based on the tag information, each cell is transmitted to their destination output port and the internal tag is detached from the cell. If all the cells belonged to the same packet has been transmitted to the re-assembly queue (RQ), then they are reassembled into the original packet and the packet departs from the output port.

The above statement of the operation of the cell-based switching tells us that consecutive arriving cells or mini-packets are strongly correlated by having the same destination. Therefore, cell-based switching needs for the re-assembly queue at output ports to restore the original packet. It needs N separate queues at each output as well, making the switching system complex. Furthermore, it could cause significant increase in the packet delay since the last mini-packet's arrival at RQ determines the whole delay of a packet. Our intuitive idea behind this observation is that the cells of a given packet should be scheduled as a whole and transmitted continuously inside a switching system rather than scheduled on a cell-by-cell basis. We refer the scheduling to as the *packet-based scheduling* in that the cells belonging to a packet are switched successively.

The main advantage of using the packet-based scheduling is three-fold. First, by using packet-based scheduling, we don't need to perform scheduling at each single cell transmission time. This is due to the fact that once the first cell of a packet is scheduled to be transmitted across the switching fabric, all the remaining cells of the packet will be transmitted in the following time slots without interruption and without any scheduling decision. It should simplify the complexity of the scheduling process and reduce the control time needed, which will be proved in later Section. Second, packet-based scheduling could provide an enhanced performance metrics since performance measures relevant to an application are specified by the performance metrics of its data units (packets) rather than individual cells. Finally, it alleviates the hardware complexity of the switching system by removing the need for re-assembly queues at output ports.

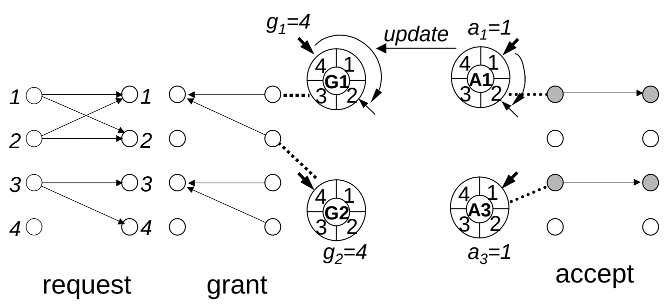

Fig. 2. Operation of the SLIP algorithm.

In [10] and [11], Ge Nong et al. proposed a similar scheduling approach to the packet-based scheduling, named burst-based scheduling. The burst-based scheduling is based on the PIM algorithm in its operation. In the studies, the authors considered only that burst-based approach can reduces the number of iteration of PIM algorithms needed for a certain QoS requirement. However, it is well known that SLIP provides better characteristics than PIM in terms of scheduling performance, control complexity, and fairness. In the following section, we proposed a packet-based scheduling algorithm based on the SLIP algorithm proposed by McKeown in [5,6,7].

III. Packet-Based SLIP (PBSLIP) Algorithm

In this paper, we propose an efficient packet-based scheduling algorithm, named the Packet-Based SLIP (PB-SLIP) algorithm. PBSLIP is a variation of SLIP proposed by McKeown in [5,6,7] and, therefore, most operations of PBSLIP are equal to those of SLIP. SLIP is a kind of three-phase scheduling algorithm and is composed of three phases of Request, Grant, and Accept. The three phases are iterated several times and cell selection operations in each phase are performed parallel on every output or every input in round-robin manner. Here, we do not elaborate on the SLIP algorithm since SLIP is the most well-known scheduling algorithm for the virtual output-queued (VOQ) switches. For more details, please refer to [5,6,7]. A conspicuous dissimilarity of PBSLIP from SLIP is that PB-SLIP does switch consecutively a bunch of mini-packets or cells belonging to a specific packet while SLIP switches cells irrespective of their correlation to adjoining cells. (That is, SLIP performs a cell-based scheduling.) If all packets arriving at the switching system have the same packet length as in ATM, then PBSLIP is indiscernible from SLIP.

For PBSLIP to conduct packet-based scheduling, we have to make some modifications on the original SLIP algorithm and the modifications are as follows:

- Only when the first mini-packet or cell, destined for output j from input i, is accepted in Accept phase, the value of pointers are fixed as $g_j = i$ and $a_i = j$ as long as the last mini-packet is switched.
- Only when the last mini-packet or cell, destined for output j from input i, is accepted for transmission, the pointers g_j and a_i are updated as $g_j = (i + 1)$ $mod\ N$ and $a_i = (j + 1)$ $mod\ N$.

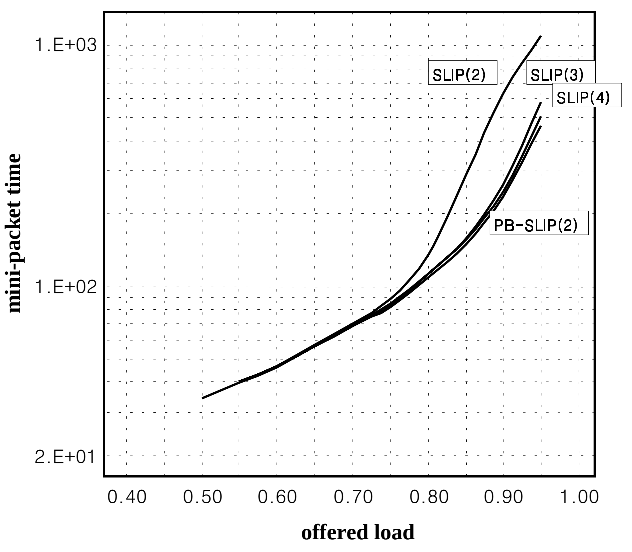

Fig. 3. Mean packet delay in input ports, D_{in}. (The number in parentheses implies the iteration times of scheduling algorithm used.)

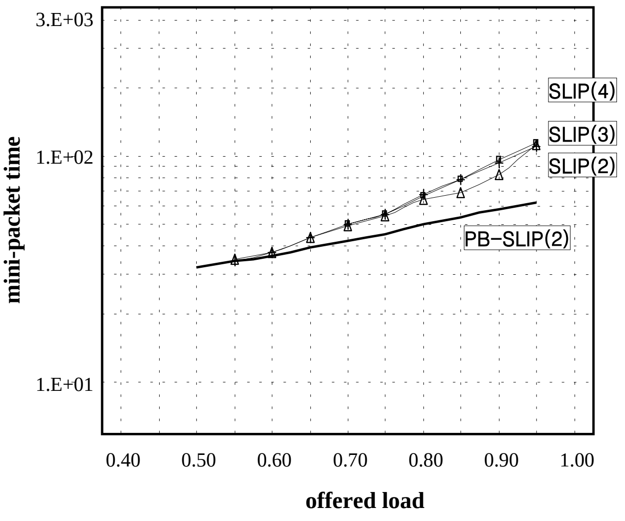

Fig. 4. Mean packet delay in output ports, D_{out}.

Then, the mini-packets belonging to the same packet can be transmitted consecutively. It means that any other mini-packets can not interrupt the successive operation of a packet-wise scheduling and, as the result, can not be inserted between mini-packets belonging to the same packet. Consequently, the switching system performs switching in packet level while the internal switching fabric performs switching in cell level. It means that the switching systems using PBSLIP requires only one re-assembly queue on each output port. By doing so, PBSLIP can reduce the packet-level delay since only one packet waits in each output port.

IV. Simulation Results

To investigate how much the original SLIP algorithm can take advantage from the packet-based scheduling concept, extensive simulations were designed and carried out. In

TABLE I
Comparison of the number of input-output matchings between SLIP and PBSLIP. PBSLIP finds most of matchings in 2 iterations.

offered	SLIP (iteration times)					PBSLIP (iteration times)				
load	1	2	3	4	$\sum$	1	2	3	4	$\sum$
0.70	9.587	1.542	0.037	0.000	11.166	10.945	0.223	0.001	0.000	11.171
0.75	9.812	2.051	0.095	0.000	11.958	11.663	0.296	0.002	0.000	11.961
0.80	9.975	2.558	0.229	0.003	12.765	12.376	0.387	0.004	0.001	12.768
0.85	10.120	2.951	0.444	0.014	13.529	13.030	0.488	0.010	0.003	13.648
0.90	10.310	3.203	0.747	0.058	14.318	13.711	0.591	0.017	0.007	14.326
0.95	11.120	2.980	0.848	0.105	15.053	14.330	0.703	0.027	0.016	15.076

these simulations, we assumed that the switch size is 16×16 and buffer size is 1000 mini-packets per queue at each input port. We also assumed the average packet length is equal to 16 cells and each cell is of 53 bytes size. Packet arrival at each input can be modeled by a 2-state Markov Modulated Bernoulli Process (2-MMBP). Packet arrival at an input is independent of packet arrival at other input ports and the arrived packets are assumed to destine for output ports with uniform distribution.

In the performance study, we focus our attention on the mean packet delay and on the number of iterations required to achieve a good performance. Since the switch manages queues on both input and output ports as described in previous sections, we observed packets on both input and output ports. Let D_{in} and D_{out} designate the mean packet delay on input ports and output ports, respectively. Then, the total mean packet delay, D_{total}, can be defined as the sum of two, viz., $D_{total} = D_{in} + D_{out}$. Each of mean packet delay is plotted in terms of mini-packet time from Fig. 3 to Fig. 5 which is the time required to process a mini-packet. The diverse offered loads were obtained by changing the transition probability used in the 2-MMBP model. In the figures, we compared the delay characteristics of PBSLIP with those of SLIP with different iteration numbers. The number in parentheses implies the iteration times of a given scheduling algorithm. As shown in the figures, PBSLIP outperforms SLIP in every case. Note that 2 iterations of PBSLIP yields much better delay characteristics than 4 iterations of the SLIP algorithms do.

Table 1 compares the number of input-output matchings between SLIP and PBSLIP. In both cases, 4 iterations of each algorithm found its maximum matching. For example, for the offered load of 0.95, both algorithms found about 15.05 matchings in 4 iterations that are equal to the multiplication of the offered load (0.95) and the switch size (16). However, the important thing is not that both algorithms find a maximal input-output matching but that PBSLIP finds its maximal matching in 2 iterations while SLIP finds in 4 iterations. Table 1 shows that PBSLIP finds most of its matchings during the first two iterations. That is, by using PBSLIP in scheduling variable-length packets, we can reduce the iteration number of the scheduling algorithm, resulting in a simpler hardware complexity and reduced control time.

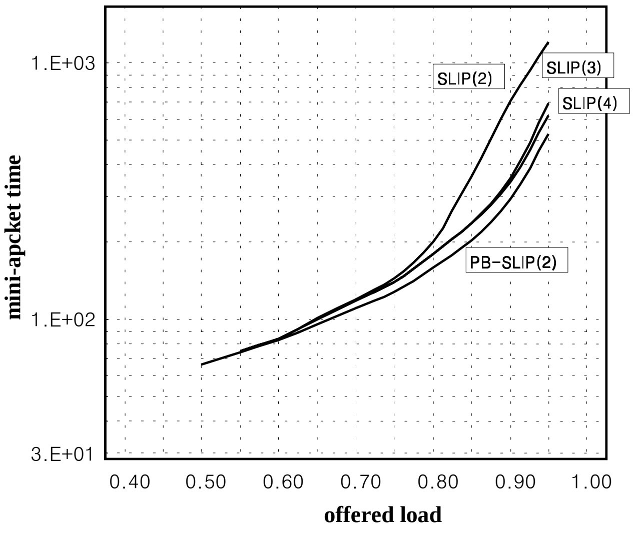

Fig. 5. Total mean packet delay, D_{total}.

V. Conclusions and Remarks

When we switch variable-length packets in the high-speed switches/routers, we first divide the variable-length packet into small fixed-length packets, called mini-packets, and then perform hardware switching of the mini-packets. Mini-packets can be switched either by the cell-based or cell-level scheduling or packet-based or packet-level scheduling. Most conventional switches/routers use the cell-based scheduling such as PIM and SLIP since the scheduling algorithms are developed for ATM technologies.

In this paper, we proposed an efficient scheduling algorithm, named Packet-Based SLIP (PBSLIP). PBSLIP is a packet-based scheduling algorithm and, therefore, switches variable-length packets like IP packets by switching small and fixed-length mini-packets belonging to the same packet consecutively. Simulation results demonstrate that PB-SLIP outperforms the SLIP algorithm in dealing with variable-length packets. 1 or 2 iterations of PBSLIP were sufficient to achieve good performance. Simulation results obtained by 2 iterations of PBSLIP outperformed those by 4 iterations of SLIP. Furthermore, packet-based scheduling simplifies the output module of a switching system by removing all re-assembly queues and control circuits relevant to re-assembly queues. Beside these, scheduling decisions made on packet-basis makes it possible to use cheap and moderate-speed microprocessors in implementing the high-speed switching system.

As further studies, we have to consider the fairness aspect of PBSLIP since PBSLIP switches mini-packets from the same queue successively. Finding an optimal mini-packet size might be another research area, even though 53 bytes is one of good candidate for the optimal size.

References

[1] B. Mukherjee, "WDM optical communication networks: progress and challenges," *IEEE JSAC*, vol.18, no.10, October 2000, pp.1810-1824.

[2] T.E. Anderson, S.S. Owicki, J.B. Saxe, and C.P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Tr. Computer Systems*, vol.11, no.4, Nov. 1993, pp.319-352.

[3] http://www.cisco.com/univercd/cc/td/doc/pcat/12000.pdf Cisco 12000 Series - Gigabit Switch Routers

[4] J.W. Lockwood, "Design and implementation of a multicast, input-buffered ATM switch for the *i*POINT testbed," Ph.D. dissertation, Univ. of Illinois at Urbaba-Champaign, 1995.

[5] N. McKeown, P. Varaiya and J. Walrand, "Scheduling cells in an input-queued switch," *IEE Electronics Letters*, vol.29, no.25, Dec. 9th 1993, pp. 2174-2175.

[6] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. thesis, University of California at Berkeley, May 1995.

[7] N. McKeown, "The *i*SLIP scheduling algorithm for input-queued switches," *IEEE/ACM Tr. Networking*, vol.7, no.2, April 1999, pp.188-201.

[8] R.O. LaMaire and D.N. Serpanos, "Two-dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Tr. Networking*, vol. 2, no. 5, Oct. 1994, pp. 471-482.

[9] M.G.A. Marsan, A. Bianco, and E. Leonardi, "RPA: A simple efficient and flexible policy for input buffered ATM switches," *IEEE Communications Letters*, vol.1, May 1997, pp.83-86.

[10] Ge Nong, M. Hamdi, and K.B. Letaief, "Efficient scheduling of variable- length IP packets on high-speed switches," *Proc. IEEE GLOBECOM '99*, vol. 2, 1999, pp. 1407-1411

[11] Ge Nong and M. Hamdi, "Burst-based scheduling algorithms for non-blocking ATM switches with multiple input-queues," *IEEE Communications Letters*, vol. 4, no. 6, June 2000, pp. 202-204.