Preemptive Packet-Mode Scheduling to Improve TCP Performance

Wenjie Li[1], Bin Liu[1], Lei Shi[1], Yang Xu[1], and Dapeng Wu[2]

[1] Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, P. R. China
lwjie00@mails.tsinghua.edu.cn, liub@mail.tsinghua.edu.cn
shijim@mails.tsinghua.edu.cn, xy01@mails.tsinghua.edu.cn
[2] Department of Electrical & Computer Engineering, University of Florida,
Gainesville, Florida 32611-6130, USA
wu@ece.ufl.edu

Abstract. Recent Internet traffic measurements show that 60% of the total packets are short packets, which include TCP acknowledgment and control segments. These short packets make a great impact on the performance of TCP. Unfortunately, short packets suffer from large delay due to serving long data packets in switches running in the packet mode, i.e. a packet is switched in its entirety. To optimize TCP performance, we apply a cross-layer approach to the design of switching architectures and scheduling algorithms. Specifically, we propose a preemptive packet-mode scheduling architecture and an algorithm called preemptive short packets first (P-SPF). Analysis and simulation results demonstrate that compared to existing packet-mode schedulers, P-SPF significantly reduces the waiting time for short packets while achieving a high overall throughput when the traffic load is heavy. Moreover, with a relatively low speedup, P-SPF performs better than existing packet-mode schedulers under any traffic load.

1 Introduction

Input queueing packet switches are widely employed in state-of-the-art core routers, such as Cisco 12000 series core routers [1], the Tiny-Tera [2] and the BBN router [3]. In these switches, a buffering scheme called virtual output queueing (VOQ) is typically deployed. In VOQ, an input port maintains one separate buffering queue for each output port. VOQ entirely eliminates the head-of-line (HOL) blocking in input queueing switches, and even achieves 100% throughput with appropriate scheduling algorithms, such as *i*SLIP [4], *i*LPF [5] and DRRM [6]. All of these algorithms operate on a fixed time slot, which is defined as the duration of a cell (a fixed-size segment). This kind of scheduling is generally called cell-mode scheduling.

This work was supported by NSFC (No. 60173009 and No. 60373007), China 863 High-tech Plan (No. 2002AA103011-1 and No. 2003AA115110), China/Ireland Science and Technology Collaboration Research Fund (CI-2003-02) and the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20040003048).

In cell-mode scheduling, IP packets are segmented into cells at an input port, then these cells are switched independently from input ports to their destination output ports, and finally IP packets are reassembled at each output port. Since cells from different input ports may be interleaved with cells of other packets when switching, virtual input queueing (VIQ) is deployed to reassemble IP packets. VIQ requires lots of logical buffers when the switch has many ports and multiple priority queues. In recent years, researchers have proposed another architecture called packet-mode scheduling. In packet-mode scheduling, scheduling algorithms consider all the cells of a packet as one scheduling unit and grants an input port continuously until the last cell of a packet. Meanwhile, a granted input port sends cells of the same packet in the acknowledged VOQ. Packet-mode scheduling simplifies the switching architecture, removes the reassembly buffer, and reduces the delay for reassembling packets at an output port. Furthermore, packet-mode scheduling does not bring about performance penalty from the perspective of user's QoS [7][8].

Although packet-mode scheduling may be more attractive due to the fact that variable-size packets are processed in routers, we find that packet-mode scheduling results in large delay for short packets because of the continuous transferring of other long data packets. This is not a simple fairness problem among packets of variable sizes, because short and long packets carry different types of traffic. We analyze a real Internet trace called Auckland-II from National Laboratory for Applied Network Research (NLANR) [9], and obtain that the fraction of Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and other protocol packets are about 86.5%, 12.8% and 0.7%, respectively. This obviously demonstrates that TCP is the dominant transport protocol in Internet, and it is invaluable to study how to optimize TCP protocol. In Auckland-II the fraction of packets not more than 64 bytes is about 60% of overall packets. Of those short packets, TCP data packets are only about 3.9% and others are TCP ACKs and TCP control segments, such as SYN, FIN and RST. TCP ACKs can be piggybacked in TCP data packets, but in practice piggybacking seldom occurs because most applications do not send data in both the forward and reverse directions simultaneously [10][11]. Blocking short packets will cause lots of TCP retransmissions due to the timeout of TCP ACKs, and thus wastes the network bandwidth by resending the same data packet twice or more. Moreover, the round-trip time (RTT) of TCP flows has a significant variability [12]. TCP performance can be improved by greatly reducing the congestion delay for short packets, while not or slightly increasing the delay for long data packets.

Differentiated Services (DS) model [13] deals with different priority flows, rather than packets within the same TCP flow. Some DS-based works [14][15] have addressed the problem of blocking TCP ACK segments, and the preferential scheduling of variable-size TCP flows has been also proposed [16]. Different from these works, we do the research from the perspective of switching architectures and scheduling algorithms in core routers, and our main objective is to reduce the delay for short packets. One approach is to retain the architecture of general packet-mode scheduling, but grant short packets first when both short and long packets at the heads of VOQs compete for the same output port. However, through simulations we obtain that this approach cannot significantly improve the performance of short packets. This is because short packets are mainly blocked by long packets that are ahead of them in the same VOQ, rather than by cells at the heads of other VOQs. Then we turn to consider another approach called Short Packets First (SPF) [17]. SPF buffers short and long packets into separate VOQs, and then always schedules short packets first without preempting the transferring of long packets. SPF can achieve 100% throughput and substantially reduce the average packet waiting time for short packets as well as overall packets.

In this paper, to lower the buffering complexity of SPF and improve the performance of SPF further, we propose a new scheduling algorithm called preemptive short packets first (P-SPF). In P-SPF, at an input port all short packets (regardless of destination output ports) are buffered in one separate FIFO queue, but they can preempt the transferring of long packets. P-SPF eliminates the VOQs for short packets in [17] and reduces its buffering complexity. Furthermore, P-SPF achieves even lower delay for short packets than SPF, and this greatly benefits TCP flows because TCP ACK segments and control segments are treated as short packets (see Section 2 for details). P-SPF achieves an overall throughput of 94% with respect to a real traffic model. With a moderate speedup, P-SPF performs better than other packet-mode scheduling algorithms under any traffic load.

The rest of this paper is organized as follows. Section 2 describes the preemptive packet-mode scheduling architecture and studies the criteria of classifying short and long packets to avoid the out-of-sequence problem for TCP data packets. Section 3 illustrates the iterative scheduling process of P-SPF. Section 4 analyzes the performance of P-SPF using standard queueing theory. Section 5 presents simulation results under a model based on the real measurement in Internet. Finally, the concluding remarks are given in Section 6.

2 Logical Architecture for P-SPF

In this section we describe the preemptive packet-mode scheduling architecture, and study the criteria of classifying short and long packets to avoid the out-of-sequence problem within a TCP data flow.

2.1 Preemptive Packet-Mode Scheduling Architecture

Fig. 1 shows the logical architecture of preemptive packet-mode scheduling, where N is the port number. When an IP packet arrives at an input port, it will be segmented into one or more cells by the segmentation module. The buffer manager module is responsible for buffering cells into *IFIFO_S* for short packets or *VOQ_k* for long packets, where k is the packet's destination port number. The input scheduler module monitors the status of *IFIFO_S* for short packets and all the VOQs for long packets, sends connection requests for both short and long packets to the switch scheduler, and then transfers the head cell in *IFIFO_S* or *VOQ_k* when it receives an acknowledgment for short packets or one for long packets. The switch scheduler module executes the preemptive packet-mode scheduling algorithm, and then reconfigures the crossbar switch fabric in the corresponding time slot.

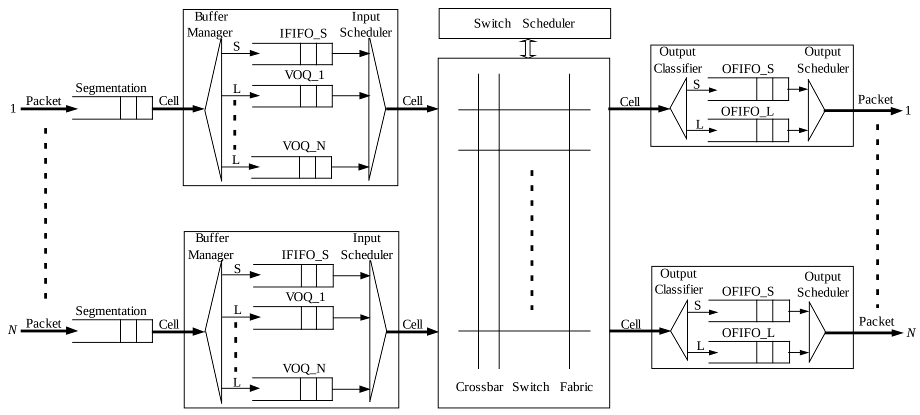

Fig. 1. Preemptive packet-mode scheduling architecture

At an output port, the output classifier module dispenses cells of short and long packets into *OFIFO_S* and *OFIFO_L*, respectively. Once a short packet or all the cells of a long packet have been transferred to an output port, the output scheduler module will send short and long packets to the external link. The output scheduler does not preempt the transferring of any packets to the link, but sends short packets first when the link is available.

2.2 The Criteria of Classifying Short and Long Packets

In many studies, the cell size of 64 bytes is adopted as a compromise between the utilization of switching bandwidth and the convenience of switching and scheduling. In this paper we also choose 64 bytes as the switching cell size[1].

For TCP data flows with mixed short and long packets, preempting the transferring of long packets may occasionally cause out-of-sequence of data packets within a TCP flow. Although this phenomenon is allowed [18] and does exist in Internet owing to local parallelism in routers and links [19], we still develop the criteria of classifying short and long packets, which can guarantee the sequence of TCP data packets.

Criteria**:** a packet is classified as a short packet if and only if the two conditions are both satisfied: 1) the packet size is not more than 64 bytes; and 2) it is a UDP packet or a TCP non-data packet (i.e. a TCP packet but not containing any data). Otherwise, a packet is classified as a long packet.

The criteria can be easily implemented in a network processor when performing the layer-4 flow classification. The IP header length, IP total length and protocol can be extracted from the IP header. If the protocol field indicates that the encapsulated payload is a TCP packet, the TCP header length field will be obtained from the TCP header. If IP total length equals IP header length plus TCP header length, we say that this packet is a TCP non-data packet.

[1] In switches for IPv6 routers, the cell size will be a little larger.

3 P-SPF Scheduling Algorithm

P-SPF is an iterative packet-mode scheduling algorithm. The scheduling of P-SPF in each time slot is composed of two parallel processes: scheduling for short packets and scheduling for long packets. Let *N* denote the port number. An input port *i* ($1 \le i \le N$) maintains one pointer $IP_L(i)$ for long packets. An output port *j* ($1 \le j \le N$) has two pointers: $OP_S(j)$ for short packets and $OP_L(j)$ for long packets. We will present the scheduling for short packets as well as for long packets as follows.

The scheduling for short packets, whose sizes are only one cell, is relatively simple.

Step 1. *Input Request.*

Each input port, where *IFIFO_S* is not empty, sends a connection request to its head cell's destination output port.

Step 2. *Output Grant.*

Each output port *j* grants a request for short packets using the round-robin schedule with the highest priority pointer $OP_S(j)$, and then updates $OP_S(j)$ to the one next to the granted input port (modulo *N*).

In the scheduling for long packets, input ports and output ports have two states.

1) *Free* state: no cells or the last cell of a packet is transferring;

2) *Busy* state: the port is occupied by the transferring of a packet except its last cell.

The scheduling for long packets involves three iterative steps.

Step 1. *Input Request.*

Each *free* input port sends connection requests for long packets at the heads of VOQs to their destination output ports.

Step 2. *Output Grant.*

If an output port has received requests for short packets while serving a long packet, it will break the current transferring of long packets and grant short packets. If not, a *busy* output port continues to grant its matched input port. If a *free* output port *j* has received multiple requests for long packets, it will grant the one which appears next in a fixed round-robin schedule starting from the highest priority input port with the pointer $OP_L(j)$. If and only if output port *j*'s acknowledgment is accepted by an input port in step 3, the pointer $OP_L(j)$ is updated to the one beyond the granted input port (modulo *N*).

Step 3. *Input Accept.*

If input port *i* receives multiple grants for long packets from output ports, it will accept the one, saying *k*, which appears next in the round-robin schedule from $IP_L(i)$, and then update $IP_L(i)$ to the one next to output port *k* (modulo *N*).

Newly matched *free* input and output ports will update their states to *busy* when the size of granted packet is larger than one cell, and still stay in *free* when the packet has just one cell. *Busy* input and output ports are set to *free* state when they have sent out and received the last cell of a packet, respectively.

Because of the simultaneous scheduling for short and long packets, an input port may accept at most two acknowledgments: one for short packets and the other for long packets. In this scenario, the input port will send the short packet in the current time slot, and then cells of the acknowledged long packet immediately if receiving no acknowledgment for short packets in the following time slots.

4 Performance Analysis

Under admissible traffic, we mainly study the performance closely related to the switch fabric, which includes the delay in waiting queues and that of traversing the switch fabric.

Packet delay: the time interval between the departure time at an output port and the arrival time at an input port for the same packet.

Packet service time: the time a packet occupies the switch fabric.

Packet waiting time: the duration when a packet stays at an input port.

We focus on an output port, take this output port as a queueing server, and define the following symbols.

1) λ_S , λ_l and λ : the packet arrival rate of short, long and overall packets.

2) ρ_s , ρ_l and ρ : the offered load of short, long and overall packets.

3) $E(S_S)$, $E(S_l)$ and $E(S)$: the average packet service time for short, long and overall packets.

4) C_V : the coefficient of variation of the packet service time.

5) $E(W_S)$, $E(W_l)$ and $E(W_P)$: the average packet waiting time for short, long and overall packets in P-SPF. In general packet-mode scheduling, the average packet waiting time for overall packets is denoted by $E(W_G)$.

6) G : preemptive gain, which is defined as the ratio of the average packet waiting time for overall packets in general packet-mode scheduling and that in P-SPF.

4.1 Packet Waiting Time Estimation of P-SPF

By queueing theory, we can give an intuitive and quantitative estimation on the average packet waiting time. In input queueing switches, there are two types of conflicts: one at each output port, and the other at each input port. When an output port receives multiple requests from different input ports, only one request can be granted. Similarly, when an input port receives multiple grants from different output ports, only one grant can be accepted. To estimate the average packet delay with a queueing model, we neglect the conflicts occurring at each input port. This means that in the following theoretical analysis we assume that an input port can accept multiple grants and can send more than one cell in one time slot. As in [7], the analysis results are reasonably accurate for low to medium load.

The service model of short packets is identical to the input queueing model in [20]. From its simulations we deduce that when the offered load is low it is accurate to use an output queueing model to characterize the average delay in the input queueing model. Let $\overline{W}$ denote the average waiting time of cells in the output queueing model, and from [20] we have

$$\overline{W} = \frac{N-1}{N} \times \frac{p}{2(1-p)} \ , \tag{1}$$

where p is the probability that a cell arrives at a particular input port.

In fact, the offered load of short packets in Internet is generally less than 0.1 (See the simulation model in Section 5 for details), so using (1) we get

$$E(W_S) = \frac{N-1}{N} \times \frac{\lambda_S}{2(1-\rho_S)} \; . \tag{2}$$

$$\text{When } N \to \infty,\; E(W_S) = \frac{\lambda_S}{2(1-\rho_S)} \; . \tag{3}$$

In P-SPF, short packets can preempt the transferring of long packets, and we use the preemptive priority queueing model [21] to estimate the approximate delay for long packets. We have

$$E(W_l) = \frac{1}{1-\rho_s} \times \left[E(S_l) + \frac{\lambda_S E(S_S^2) + \lambda_l E(S_l^2)}{2(1-\rho)} \right] - E(S_l) \; , \tag{4}$$

$$E(W_P) = \frac{\lambda_S}{\lambda} E(W_S) + \frac{\lambda_l}{\lambda} E(W_l) = \frac{1}{\lambda(1-\rho_s)} \left[\frac{1}{2}\lambda_S^2 + \rho_s \rho_l + \lambda_l \frac{\lambda E(S^2)}{2(1-\rho)} \right] . \tag{5}$$

In general packet-mode scheduling, packets are served without being interleaved with cells of other packets. Consequently, general packet-mode scheduling algorithm corresponds to the M/G/1 FCFS queueing model [7][22], so we can get

$$E(W_G) = \frac{(1+C_V^2)\rho E(S)}{2(1-\rho)} \; . \tag{6}$$

Combining (5) with (6), we get the preemptive gain

$$G = \frac{E(W_G)}{E(W_P)} = \frac{(1+C_V^2)(1-\rho_s)\rho^2}{(1-\rho)\lambda_S^2 + 2(1-\rho)\rho_s\rho_l + \lambda_l \lambda E(S^2)} \; . \tag{7}$$

4.2 The Maximum Overall Throughput of P-SPF

The preemption of short packets will break the matched connections between input and output ports for long packets. Fig. 2 shows such an example. In the current time slot, two matches for long packets are established. Then a short packet destined to O_A arrives at I_A, the new matching for this short packet will break the two previous matches and make output port O_B temporarily hanged up in the next time slot. As a result, the bandwidth of O_B is wasted and the overall throughput is degraded.

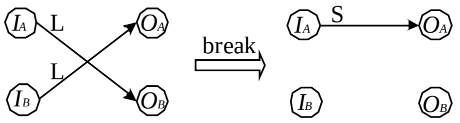

Fig. 2. An example of a broken match

We focus on the output port O_B and define two probability events.

1) $A_s(O_B)$: a short packet with the destination port O_B arrives at an input port.

2) $A_s(I_A)$: a short packet arrives at input port I_A.

When a long packet is transferring to O_B, it will be blocked if both $A_s(I_A)$ and *not* $A_s(O_B)$ occur. Therefore, the probability that O_B is blocked can be represented as

$$P_r\left(A_s\left(I_A\right)\cap\overline{A_s\left(O_B\right)}\right)=P_r\left(\overline{A_s\left(O_B\right)}\;\middle|A_s\left(I_A\right)\right)\times P_r\left(A_s\left(I_A\right)\right)=\left(1-\frac{\lambda_S}{N}\right)^{N-1}\times\left(1-\frac{1}{N}\right)\times\lambda_S\ . \qquad \textbf{(8)}$$

When $N\rightarrow\infty$, we obtain

$$P_r\left(O_B\ \text{ is blocked}\right)=\lambda_S\ell^{-\lambda_S}\ . \qquad \textbf{(9)}$$

Let T_{max} denote the maximum achievable throughput of P-SPF, and then T_{max} can be figured out as follows:

$$T_{max}=1-P_r\left(O_B\ \text{ is blocked}\right)=\begin{cases}1-\lambda_S\left(1-\dfrac{1}{N}\right)\left(1-\dfrac{\lambda_S}{N}\right)^{N-1} & N<\infty\\ 1-\lambda_s\ell^{-\lambda_s} & N=\infty\end{cases}\ . \qquad \textbf{(10)}$$

Assuming λ_S = 0.0686, we can obtain T_{max} = 0.936 when $N=\infty$, and T_{max} = 0.940 when N = 16. The result is consistent with the simulation solution in Section 5.

5 Simulations

We build a simulated switch model and run one million time slots. The switch size is 16 × 16, i.e. N = 16. *ON-OFF* model is used to simulate the packet arrival process.

OFF state: no packets arrive in this state. *OFF* state is modeled by the geometric distribution, and the probability that *OFF* state ends is fixed to a parameter, which determines the offered load at an input port.

ON state: packets are generated in this state. In *ON* state, destinations of arrival packets are uniformly distributed over all output ports. *ON* state ends when the packet is transferred completely.

We omit the process of padding packets whose sizes are not integral times of the cell size, and then use the TRIMODEL to describe the distribution of packet sizes. TRIMODEL(a, b, c, P_a , P_b): Packet sizes are chosen equal to either a cells with probability P_a , or b cells with P_b , or c cells with $1 - P_a - P_b$. In the simulations, we set the parameter a = 1, b = 9, c = 24, P_a = 0.559 and P_b = 0.200, i.e. the packet sizes are 64, 576 and 1536 bytes, respectively. These 64-byte packets are short packets and others are long data packets. The model is consistent with the real Internet traffic reported in [7][10][11], so TRIMODEL(1, 9, 24, 0.559, 0.200) is a relatively accurate model to describe the real packet size distribution in Internet.

Under general packet-mode scheduling, we simulate 4-*i*SLIP [4], maximum size matching (MSM), maximum weight matching with the weight of cell age (MWM-CA) and maximum weight matching with the weight of queue length (MWM-QL) [23].

The reason for choosing 4-*i*SLIP is its high performance and practicality, and the reason for choosing MSM, MWM-QL and MWM-CA is that they are the most typical algorithms used in the theoretical analysis. We also modify MSM, MWM-QL and MWM-CA to function in preemptive packet-mode scheduling, and call the modified algorithms P-MSM, P-MWM-QL and P-MWM-CA, respectively. These algorithms work similarly to P-SPF, except that they use MSM, MWM-QL and MWM-CA to schedule short packets first and then schedule long packets among the unmatched input/output ports.

5.1 Performance on the Maximum Throughput

Table 1 shows the maximum throughput of these selected algorithms under general packet-mode and preemptive packet-mode scheduling. The results in Table 1 show that all these algorithms under general packet-mode scheduling can achieve approximate 100% throughput, and under preemptive packet-mode scheduling, the maximum throughput is 94%. When an input port is under full utilization, we can get the arrival rate of short packets:

$$\lambda_S = \frac{aP_a}{aP_a + bP_b + cP_c} = 0.0686 \ . \qquad (11)$$

Therefore, the simulated throughput equals what we have obtained from (10).

5.2 Delay Performance of Preemptive Packet-Mode Scheduling

The maximum benefit of P-SPF is for short packets, and Fig. 3 shows the average packet waiting time for short packets. In general packet-mode scheduling, the waiting time for short packets is very large. This is mainly because short packets have to wait behind those long packets that require a large service time. In preemptive packet-mode scheduling, it is observed that the average packet waiting time for short packets is approximately zero, no matter under P-SPF, P-MSM, P-MWM-CA or P-MWM-QL. This means that the blocking probability of short packets is low, i.e. short packets will get served immediately after they arrive at an input port. Throughout the simulations we get the maximum length of *IFIFO_S* is less than 7 cells. By preempting the transferring of long packets, the waiting time for short packets falls drastically, especially when the offered load is high. In other words, the preemption provides a fast switching path for short packets.

Table 1. Throughput of simulated scheduling algorithms

Algorithms	Throughput	Algorithms	Throughput
4-*i*SLIP	0.996	P-SPF	0.940
MSM	0.997	P-MSM	0.940
MWM-QL	0.997	P-MWM-QL	0.940
MWM-CA	0.998	P-MWM-CA	0.940

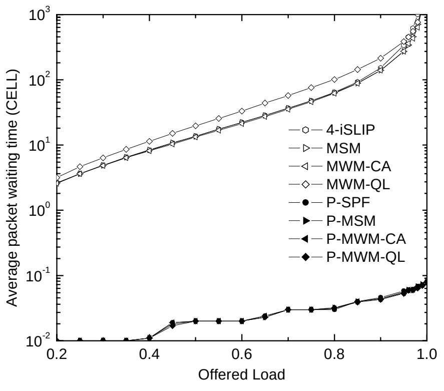

Fig. 3. Average packet waiting time for short packets

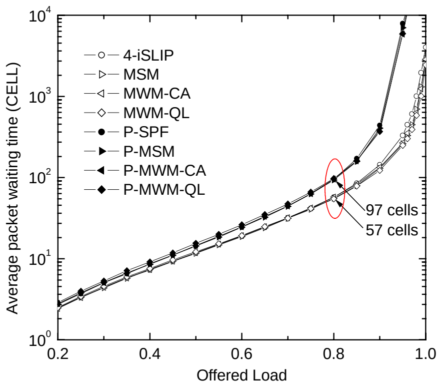

Fig. 4. Average packet waiting time for long packets

In preemptive packet-mode scheduling, the first priority and preemption of short packets may increase the average packet waiting time for long packets. Fig. 4 shows the average packet waiting time for long packets. When the offered load is less than 0.8, the performance degradation of long packets is much small. E.g., when the offered load is 0.8, the average packet waiting time for long packets is about 40 cells larger than that in general packet-mode scheduling. The absolute time is less than that transferring a maximum-size Ethernet packet (24 cells) twice at the line rate. As a conclusion, the increased delay for long packets can be almost ignored when the line rate is high, such as 10 Gb/s or 40 Gb/s. When the offered load is greater than 0.8, the difference of average packet waiting time for long packets between general and preemptive packet-mode scheduling becomes a little larger.

Fig. 5 shows the simulation results on the average packet delay for overall packets with the offered load from 0.2 to 1.0. The four curves of preemptive packet-mode scheduling algorithms overlap almost everywhere. This shows that P-SPF can achieve the performance of maximum weight matching under the simulated Internet traffic. When the offered load is less than 0.85, the performance of preemptive packet-mode scheduling is better than that in general packet-mode scheduling. When the offered load is larger than 0.85, the performance of preemptive packet-mode scheduling begins to degrade with the increase of the offered load. This is the limitation of the overall throughput of 94%, which can be improved further with the approach discussed in next subsection.

5.3 To Improve the Delay Performance of P-SPF

We deploy rather a small speedup to improve the throughput and reduce the delay for overall packets in P-SPF further. In Fig. 5 we get that when the offered load is lower than 0.85, P-SPF will always perform better than general packet-mode scheduling algorithms. Therefore, the speedup of 1.176 (1/0.85) can always guarantee the advantages of P-SPF.

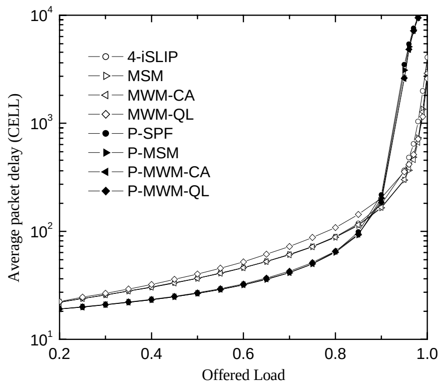

Fig. 5. Average packet delay for overall packets

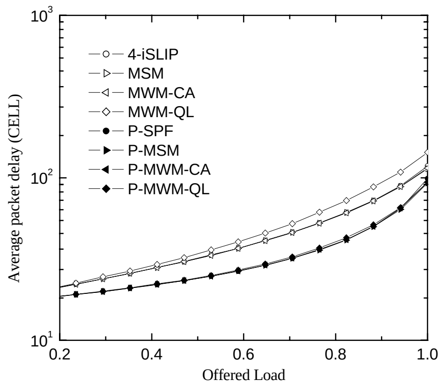

Fig. 6. Average packet delay for overall packets with the speedup of 1.176

Fig. 6 shows the average packet delay for overall packets, where all the simulated scheduling algorithms are with the speedup of 1.176. By comparing Fig. 6 with Fig. 5, we can see that the overall packet delay in P-SPF is greatly reduced under heavy offered load. E.g., when the offered load is 1.0, the average packet delay for overall packets is less than 100 cells in P-SPF, and this value is much large in Fig. 5 without the speedup. In real router designs, such a low speedup is easy to implement in hardware. E.g., to deal with 10 Gb/s links, with the mature 3.125 Gb/s high-speed serial link (HSSL) technology, four serial links can achieve the speedup of 1.25.

6 Conclusions

In this paper, we consider the packet-mode scheduling in input queueing switches and propose a scheduling algorithm called P-SPF. Compared to the general packet-mode schedulers, P-SPF can provide lower delay for short packets, resulting in an improved performance of upper layer protocols, such as TCP.

This is achieved by the following mechanisms and features. First, we took a cross-layer approach in the design of switching architectures and scheduling algorithms. Second, we proposed a preemptive packet-mode scheduling architecture. Compared to general packet-mode scheduling architecture, the added cost of P-SPF is one FIFO queue for short packets at each input port and that at each output port; the FIFO size is very small and hence the increased cost is negligible. Third, P-SPF is practical and its complexity is almost the same as *i*SLIP. Fourth, with the low speedup of 1.176, P-SPF always performs better than existing packet-mode scheduling schemes. Last but not the least, P-SPF can significantly reduce the average packet waiting time for short packets, which greatly benefits TCP flows.

Furthermore, most real time voice over IP (VoIP) traffic in UDP flows is of short packets, so lowering the delay for these short packets will also upgrade the QoS of VoIP traffic and provide a better way to support VoIP in backbone networks. This exciting topic will be researched in future works.

References

1. McKeown N.: Fast Switched Backplane for a Gigabit Switched Router. Business Commun. Review, vol. 27, no. 12, (1997) 1-30
2. McKeown N., Izzard M., Mekkittikul A., Ellersick W., Horowitz M.: Tiny Tera: a Packet Switch Core. IEEE Micro, vol. 17, no. 1, (1997) 26-33
3. Partridge C., et al.: A 50-Gb/s IP Router. IEEE/ACM Trans. Networking, vol. 6, no. 3, (1998) 237-248
4. McKeown N.: The *i*SLIP Scheduling Algorithm for Input-Queued Switches. IEEE/ACM Trans. Networking, vol. 7, no. 2, (1999) 188-201
5. Mekkittikul A., McKeown N.: A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches. IEEE INFOCOM 1998, (1998) 792-799
6. Chao H.J.: Saturn: a Terabit Packet Switch Using Dual Round-Robin. IEEE Commun. Magazine, vol. 38, no. 12, (2000) 78-84
7. Marsan M.A., Bianco A., Giaccone P., Leonardi E., Neri F.: Packet-Mode Scheduling in Input-Queued Cell-Based Switches. IEEE/ACM Trans. Networking, vol. 10, no. 5, (2002) 666-678
8. Ganjali Y., Keshavarzian A., Shah D.: Input Queued Switches: Cell Switching vs. Packet Switching. IEEE INFOCOM 2003, (2003) 1651-1658
9. The Auckland-II trace, National Laboratory for Applied Network Research (NLANR), http://pma.nlanr.net/Special/
10. Thompson K., Miller G.J., Wilder R.: Wide-Area Internet Traffic Patterns and Characteristics. IEEE Network, vol. 11, no. 6, (1997) 10-23
11. Fraleigh C., et al.: Packet-level Traffic Measurements from the Sprint IP Backbone. IEEE Network, vol. 17, no. 6, (2003) 6-16
12. Aikat J., Kaur J., Smith F.D., Jeffay K.: Variability in TCP Round-trip Times. ACM SIGCOMM Conference on Internet Measurement Workshop 2003, (2003) 279-284
13. Blake S., Black D., Carlson M., Davies E., Wang Z., Weiss W.: An Architecture for Differentiated Services. IETF RFC 2475, (1998)
14. Papagiannaki K., Thiran P., Crowcroft J., Diot C.: Preferential Treatment of Acknowledgment Packets in a Differentiated Services Network. IWQoS 2001, Springer-Verlag Lecture Notes in Computer Science, vol. 2092, (2001) 187-201
15. Wang H.N., Shin K.G.: Transport-aware IP Routers: a Built-in Protection Mechanism to Counter DDoS Attacks. IEEE Trans. Parallel and Distributed Systems, vol. 14, no. 9, (2003) 873-884
16. Rai I.A., Biersack E.W., Urvoy-Keller G.: Size-Based Scheduling to Improve the Performance of Short TCP Flows. IEEE Network, vol. 19, no. 1, (2005) 12-17
17. Li W.J., Liu B.: SPF: to Improve the Performance of Packet-Mode Scheduling. Elsevier Computer Commun., in press, (2005)
18. Baker F.: Requirements for IP Version 4 Routers. IETF RFC 1812, (1995)
19. Bennett J.C.R., Partridge C., Shectman N.: Packet Reordering is not Pathological Network Behavior. IEEE/ACM Trans. Networking, vol. 7, no. 6, (1999) 789-798
20. Karol M., Hluchyj M., Morgan S.: Input Versus Output Queueing on a Space-Division Packet Switch. IEEE Trans. Commun., vol. 35, no. 12, (1987) 1347-1356
21. Allen A.O.: Probability, Statistics, and Queueing Theory with Computer Science Applications. New York Academic Press, New York (1978)
22. Wolff R.W.: Stochastic Modeling and the Theory of Queues. Prentice-Hall Inc., Englewood Cliffs, NJ USA (1989)
23. Mckeown N., Mekkittikul A., Anantharam V., Walrand J.: Achieving 100% Throughput in an Input-Queued Switch. IEEE Trans. Commun., vol. 47, no. 8, (1999) 1260-1267