SPF: to improve the performance of packet-mode scheduling

Wenjie Li*, Bin Liu

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People’s Republic of China

Received 27 May 2004; revised 4 February 2005; accepted 11 February 2005

Available online 7 March 2005

Abstract

Recent researches present that packet-mode scheduling may provide overall performance advantages over cell-mode scheduling in input queueing switches. But one significant drawback of existing packet-mode schedulers is that they tend to result in longer waiting time of short packets, whose delay is critical to the performance of IP networks. To mitigate the problem, this paper proposes a simple but effective packet-mode scheduling algorithm called Short Packets First (SPF), studies the corresponding buffering architecture, and evaluates the performance on throughput and packet delay. SPF achieves 100% throughput under an accurate Internet traffic model. Theoretical analysis proves that SPF can reduce the average packet waiting time of short packets as well as overall packets, under the condition of uniform Poisson arrival process and low to medium offered load. Furthermore, extended simulation studies show that SPF is superior to general packet-mode and cell-mode scheduling, especially under heavy offered load.

q 2005 Elsevier B.V. All rights reserved.

Keywords: Input queueing switches; Scheduling algorithm; Cell-mode; Packet-mode; Short packets first

1. Introduction

Tremendous growth of IP traffic coupled with the availability of advanced optical technologies, such as dense wavelength division multiplexing (DWDM), stimulates the fast development of Internet. However, restricted by electrical technologies, high performance core routers face more and more challenges from the rapid increase of line rate (e.g. OC-768 SONET). In traditional low-speed routers, output queueing architecture is very popular owing to its 100% throughput and excellent QoS provided to users [1]. But this architecture requires a very high speedup of switch fabric, which is deemed impractical in terabit core routers. Then in recent years, input queueing (IQ) architecture coupled with the virtual output queueing (VOQ) technique is becoming an attractive alternative due to its high scalability. Many state-of-the-art core routers have employed this architecture, such as Cisco 12000 series core routers [2], the Tiny-Tera [3] and the BBN [4].

There are many scheduling algorithms in IQ switches. One class of IQ scheduling algorithms is based on the Birkhoff–von Neumann decomposition for a doubly stochastic matrix [5,6]. It is capable of providing guaranteed-rate services when giving the arrival rates of all flows, but this scheduling algorithm may select empty VOQs and thus causes large cell delay, and its off-line computational complexity is up to $O(N^{4.5})$ for an $N \times N$ switch. The cell delay may be reduced using the enhanced Birkhoff–von Neumann decomposition [7], where those matches of empty VOQs are substituted by other new ones of non-empty VOQs. Furthermore, the maximum credit first (MCF) scheduling algorithm [8] can achieve a tighter delay bound with a lower complexity of $O(N^3)$. But the problem is that all of these algorithms require the arrival rates of all flows. There are lots of simultaneously running flows in core routers and these flows change dynamically. It is rather difficult and costly to measure each flow’s rate and then complete the complex scheduling algorithms fast enough to reconfigure the switch fabric timely. Due to the computational complexity, these algorithms based on Birkhoff–von Neumann decomposition are still not practical with the current silicon technologies.
Another class of IQ scheduling algorithms just requires the state of VOQs at each input port. The VOQ scheme entirely eliminates head-of-line (HOL) blocking in IQ architecture, and even achieves 100% throughput with appropriate scheduling algorithms [9]. To now, many simple and practical scheduling algorithms have been proposed to achieve high performance, such as iSLIP [10], iLPF [11] and DRRM [12]. In order to build high-speed core routers, we mainly focus on those practical IQ scheduling algorithms in this paper. All of these mentioned scheduling algorithms operate upon the time slot, which is defined as the duration of a cell, or a fixed-size unit. This type of scheduling is generally called cell-mode scheduling. In IP routers, they require segmentation and reassembly (SAR) module to segment packets into cells for scheduling and switching in the ingress direction, and then reassemble cells after being switched for recovering original packets in the egress direction. SAR becomes tough in high-speed core routers, and wastes usable bandwidth of the switch fabric due to the necessary large overhead of each cell.

To reduce the SAR module in cell-mode scheduling, an approach is to schedule variable-size packets directly. In this situation the scheduler needs to monitor the arrival and departure time of every packet, which is in continuous time domain, and thus the scheduling is essential to be finished in every clock cycle rather than a time slot. However, this task is impossible, or at least extremely difficult to be fulfilled in high-speed terabit core routers.

Through modification of general cell-mode scheduling, the idea of transferring cells of the same packet like a train without being interleaved with cells of other packets is first systematically studied in [13]. This type of scheduling is called packet-mode scheduling. Compared to cell-mode scheduling, packet-mode scheduling does not bring about performance penalty from the viewpoint of user QoS. First of all, it has been proven that under any admissible Bernoulli i.i.d. arrival traffic, packet-mode maximum-weight-matching algorithms are stable [13], and this result has been extended to general re-generative admissible traffic model in [14]. Moreover, Zhang et al. [15] shows that fair queueing can be also implemented in packet-mode scheduling to achieve better QoS control for various flows. Finally, since cells from the same packet remain continuous during transmission, packet-mode scheduling can simplify the switch architecture and reduce the reassembly delay at output ports.

However, by analyzing Internet traffic characteristics we find a critical drawback in general packet-mode scheduling: short packets suffer from large waiting time because of the continuous transferring of cells of the same long packets. From previous research results on the traffic characteristics of Internet in [16] and the latest study in [17], we know that packets not more than 64 bytes in size occupy a large fraction of Internet traffic, and most of them are TCP ACK packets and TCP control packets, such as SYN, FIN and RST packets. Blocking these short packets may cause more TCP data packets to be retransmitted due to timeout of TCP ACK packets, and therefore wastes the network bandwidth by resending the same packet through networks twice or more. Consequently, reducing the delay of short packets will improve the performance of TCP flows. At the same time, most of VoIP traffic in User Datagram Protocol (UDP) flows is also of short packets, such as at ITU-T G.729 codecs, the packet size of VoIP is 50 bytes (20 byte IP header + 12 byte UDP header + 8 byte RTP header + 10 byte payload) for real-time transport protocol (RTP) [18] and much less bytes for compressed RTP (cRTP) [19]. Lowering the delay of short packets will also upgrade the QoS of VoIP and attach more users to Internet.

To improve the QoS of short packets in general packet-mode scheduling, we propose the Short Packets First (SPF) scheduling algorithm and study its buffering architecture and performance. The key idea of this algorithm is to buffer short packets in separate VOQs and to give short packets first priority. SPF is simple, but its performance tradeoff between short and long packets is non-trivial. If long packets experience too much delay, the advantages of SPF will get counteracted. Fortunately, the performance degradation of long packets is very slight in SPF (see Section 5 for detail). Through theoretical analysis and performance simulations, this paper shows that SPF achieves better average packet waiting time performance (for both short packets and overall packets) than general packet-mode scheduling. Moreover, the throughput of SPF is almost 100% and the iteration number of SPF is just two, rather than four in cell-mode scheduling for an example of a 16×16 switch.

The rest of the paper is organized as follows. Section 2 discusses the criterion of classifying short and long packets, and describes logical architectures of both cell-mode and general packet-mode scheduling, as well as the modified architecture for SPF. Section 3 illustrates the iterative scheduling process of SPF and then gives a scheduling example. Section 4 presents theoretical analysis on the performance of average packet waiting time. Section 5 performs many simulations and then analyzes the results of SPF, cell-mode and packet-mode scheduling, respectively. Finally, the concluding remarks are given in Section 6.

2. Logical architecture

In ATM networks, data packets are segmented and reassembled at each terminal, so fixed-size ATM cells are transferred through ATM networks. But in Internet, variable-size IP packets are forwarded and switched directly through IP networks, so SAR processing is usually the responsibility of each IP router. In this section, we will describe the criterion of classifying short and long packets in detail, and then present cell-mode scheduling, packet-mode scheduling and SPF to deal with variable-size IP packets.
2.1. The criterion of classifying short and long packets

Although the traffic in Internet may change from time to time, its basic characteristics will not vary much because of the wide use of TCP. We analyze a real link trace called Auckland-II from the National Laboratory for Applied Network Research (NLANR) [20], and obtain that the fraction of TCP packets, UDP packets and other protocol packets is about 86.5, 12.8 and 0.7%, respectively. This obviously demonstrates that TCP is the dominant protocol in Internet. Then from Auckland-II we can get the cumulative distribution of packet sizes in Internet as shown in Fig. 1. From this figure we further understand that the fraction of packets not more than 64 bytes is about 60% of overall packets. Among those short packets, TCP data packets are only about 3.9%, and most of others are TCP ACK packets and TCP control packets.

Based on the Internet traffic characteristics, we choose 64 bytes as the cell size (in IPv6, the cell size should be a little larger). The cell size is used widely, which is a compromise between the utilization and convenience of switching and scheduling: smaller cell size offers higher utilization and finer granularity of the switch fabric, while larger cell size relaxes the timing requirements imposed on switching and scheduling algorithms.

From the packet size distribution in Fig. 1 and the chosen cell size of 64 bytes, we make the criterion of classifying short and long packets.

Criterion. A packet is classified as a short packet if its size is not more than 64 bytes. Otherwise, a packet is classified as a long packet.

Short packets only need one cell, and this makes it rather simple to buffer and schedule short packets separately. In one hand, of these packets not more than 64 bytes, TCP data packets are only about 3.9%, so most of these classified short packets are TCP ACK packets and TCP control packets that we try to optimize. On the other hand, if we classify more packets with larger size as short packets, such as packets not more than 128 bytes, more common TCP data packets will be included in short packets, instead of TCP control packets. What is more, the criterion of classifying short and long packets just needs to check the IP length, so it can be easily implemented in high-speed switches.

2.2. Cell-mode scheduling

Fig. 2 shows the typical packet switch architecture based on an internal crossbar switch fabric. This switch includes N input ports and N output ports, where parameter N is the port number.

Arrival IP packets are segmented into fixed-size cells by the segmentation module, and then buffered into different VOQs according to their destination ports by the buffer manager module. The function of the input scheduler module is to send connection requests to the switch.
scheduler module and then transmit the head cell in a
granted VOQ through the crossbar switch fabric to the
destination output port. The switch scheduler executes a
cell-mode scheduling algorithm, controls connections
between input and output ports and then reconfigures the
crossbar switch fabric.

In cell-mode scheduling, the scheduling algorithm
operates upon independent cells, so the correlation between
cells coming from the same packets is not taken into
account. Cells from different packets may be interleaved
through the crossbar switch fabric, so reassembly is
necessary. At each output port, cells are buffered according
to their source input ports. This kind of queuing structure
for reassembly is called virtual input queueing (VIQ). Only
after a packet is reassembled completely can the output
scheduler transmit it to the external link. When speedup is
one, at most one packet can be reassembled in one time slot.
But when the speedup is greater than one, more than one
packet will be reassembled in one time slot, and thus the
design of output reassembly and output scheduler becomes a
little complicated.

2.3. Packet-mode scheduling

The logical architecture of packet-mode scheduling is
very similar to that of cell-mode scheduling. The difference
is that scheduling algorithms in packet-mode scheduling
consider cells of the same packet as a whole unit and grant
an input port continuously until all these cells of a packet are
transferred completely. This means that granted input ports
can send cells of the same packet in their acknowledged
queues in the following time slots continuously until the end
of the served packets. Since cells of different packets are not
interleaved, VIQs and the output scheduler are no longer
required, which greatly reduces the buffer of reassembly and
the complexity of output control. Obviously, we can achieve
higher switch speed when requiring less buffer and simpler
control logic under the same hardware condition.

Moreover, a cell-mode scheduling algorithm can be
easily modified to be suitable for packet-mode scheduling.
The only complexity increased in the implementation is to
add a register to keep track of the current connected input
port and a variable to indicate whether the current packet
transmission is over. However, one significant problem
introduced by general packet-mode scheduling is that short
packets may be blocked. To effectively solve this problem,
we modify the queueing architecture a little bit and propose
the SPF scheduling algorithm and the corresponding
buffering architecture for SPF.

2.4. SPF scheduling architecture

Fig. 3 plots the logical architecture of SPF. There are two
types of VOQs at each input port: $N$ queues for short packets
and $N$ queues for long packets. When a packet arrives at an
input port, it will be segmented and then buffered in a
corresponding VOQ according to its packet type (short or
long) and the destination port number. The input scheduler
module sends connections for both short and long packets.
The switch scheduler module executes the SPF algorithm,
which will be detailed in Section 3.

Same as general packet-mode scheduling, SPF guaran-
tees the continuous transferring of cells of the same packet,
so reassembly is also not required in SPF. Cells from the
switch fabric are buffered in the output FIFO to shield clock
difference between the switch fabric and lower layer
processing chipsets. When the head cell of a packet arrives
at the output FIFO, it will be sent to the output link

![Fig. 3. Architecture of switches with SPF.](image-url)
immediately. This scheme makes the size of output FIFO much small and the output control logic quite simple. Even if the speedup is greater than one, the output control logic in SPF is still not complex because just one FIFO should be managed, rather than N VIQs in cell-mode scheduling.

3. SPF scheduling algorithm

SPF is an iterative packet-mode scheduling algorithm. The arrival and departure traffic are variable-size packets, while cells are scheduled internally as illustrated in the previous section. The scheduling process of SPF in each time slot includes three steps: connection request, output grant and input accept.

Step 1: Connection request. In each time slot, an unmatched input port can send at most two connection requests to each output, one for short packets and the other for long packets, if it has queued at least one short/long packet for the output port. Each connection request includes the packet type and the destination port number.

Step 2: Output grant. Each output port (1 ≤ j ≤ N) maintains two pointers: OP_S(j) for short packets and OP_L(j) for long packets. One-bit register Reg_Con(j) stores the connection status of output j.

Reg_Con(j) = 1 means that output port j is currently occupied by an input port, and the matched input port number is stored in the register Reg_Port(j) (1 ≤ Reg_Port(j) ≤ N).

Reg_Con(j) = 0 means that output j is idle in the current time slot.

The granting process at an output j is illustrated in pseudo algorithm language that is shown in Fig. 4. Its essence is to acknowledge connection requests for short packets first if an output port is idle or current connection will end in this time slot, and then acknowledge requests for long packets if this output port does not receive any requests for short packets. Reg_Con(j) is set if and only if an input port accepts output j’s acknowledgment. If the acknowledgment is for a short packet, pointer OP_S(j) is updated to the one beyond the granted input port (modulo N). Otherwise, OP_L(j) is updated if and only if an input port accepts output j’s acknowledgment for a long packet.

Step 3: Input accept. Each input i (1 ≤ i ≤ N) maintains two pointers: IP_S(i) for short packets and IP_L(i) for long packets, which denote the highest priority for output ports in a fixed round-robin schedule. If input port i receives any grants for short packets, it will accept the one, saying k, that appears next in the round-robin schedule from IP_S(i) and then update IP_S(i) to the one next to output k (modulo N). Otherwise, if input port i receives grants for long packets, it will accept one and update pointer IP_L(i) using the similar scheme for short packets.

Multiple iterations of SPF can be performed to improve the overall system performance. Similar to iSLIP, in SPF all pointers are updated only during the first iteration and this strategy avoids starvation of some input ports [10]. After the scheduling is finished, crossbar switch fabric is reconfigured at the beginning of next time slot and kept unchanged until the end of next time slot.

Fig. 5 gives an example of SPF where N = 4. R_S(i, j) and R_L(i, j) represent the requests for short packets and long packets, respectively. R_S(i, j) = 1 or R_L(i, j) = 1 means that there is a request for short or long packets from input port i to output port j. At the beginning of the scheduling in this example, all outputs are idle and all pointers equal one. In step 1, output port 1 only receives a request for short packets, output port 2 and 3 only receive requests for long packets, while output 4 receives requests for both kinds of packets. In step 2, input port 1 receives an acknowledgment for short packets from output port 1, input port 2 receives that for long packets from output port 2, and input port 3 receives acknowledgments for long and short packets from output port 3 and output port 4. Then in step 3, input port 1, 2 and 3 accept the acknowledgments from output port 1, 2 and 4, respectively. Finally, after the 3-step scheduling, all the pointers of matched input and output ports are updated.

4. Performance evaluation

In this section, we present the theoretical analysis of SPF, cell-mode and packet-mode scheduling.

Packets arriving at an input port follow a discrete-time stochastic process A^n, where i (1 ≤ i ≤ N) is an input port number, and n (n ≥ 0) is a time slot index. For convenience, two notations are defined first.

(1) \( \lambda_i^j \): the average arrival rate of packets destined to output j from input port i per time slot (1 ≤ i, j ≤ N).
(2) \( L_j \): the average size of packets destined to output \( j \) from input port \( i \) \((1 \leq i, j \leq N)\).

For a packet switch model, the arrival process is said admissible if no input or output port is overloaded, i.e.

\[
\sum_{i=1}^{N} \lambda_i L_i < 1, \forall i, \quad \text{and} \quad \sum_{j=1}^{N} \lambda_j L_j < 1, \forall j
\]  

(1)

4.1. Performance indices

Under admissible traffic, we mainly study the average delay performance. Generally speaking, the delay of a packet in a router mainly consists of the following parts: physical layer processing delay, routing table lookup delay, segmentation and reassembly delay, delay in waiting queues and delay in the switch fabric. Here we will ignore constant delays and only focus on the last two random delays, which are closely related to the switch fabric.

For an internal cell-based switch fabric in an IP router, there are two types of delay: cell delay and packet delay. Cell delay is the time spent by cells in VOQs and the switch fabric. Packet delay is the time interval starting from the instant when the last cell of a packet arrives at an input buffer and ending when the last cell of this packet is transferred through the switch fabric. From the viewpoint of IP routers, packet delay is a better user QoS parameter than cell delay because each packet itself represents more information for end users. Without the special notation, delay in this paper refers to packet delay minus packet service time, whatever in cell-mode scheduling, packet-mode scheduling, or SPF.

4.2. Packet waiting time estimation

By queueing theory, we can give an intuitive and quantitative estimation of packet waiting time.

In input queueing architecture, there are two types of conflicts in the switch fabric: one at each output port and the other one at each input port. When an output receives multiple requests from different input ports, only one request can be granted. At the same time, when an input port receives multiple grants from different output ports, only one can be accepted. To estimate the average packet delay with a queueing model, we neglect the conflict occurring at each input port, which means an input port can accept multiple grants and can send more than one packet in one time slot. As in [13], the following analysis and results are reasonably accurate for low to medium load.

For the simplicity and generality of arrival traffic, we make the following assumptions:

(1) The arrival traffic of an input port follows the i.i.d. Poisson process. The arrival process among different input ports is also independent.

(2) Destinations of all incoming packets are uniformly distributed over all output ports.

(3) Continuous queueing model is used to estimate the discrete-time arrival process.

Moreover, we focus on an output port \( j \) \((1 \leq j \leq N)\), and notations by queueing theory are defined here.

(1) **Packet arrival rate**: \( \lambda_{\text{short}} \), \( \lambda_{\text{long}} \) and \( \lambda \) represent the packet arrival rate of short, long and overall packets to output \( j \). Obviously,

\[
\lambda = \lambda_{\text{short}} + \lambda_{\text{long}}.
\]  

(2) **Packet service rate**: \( u_{\text{short}} \), \( u_{\text{long}} \) and \( u \) denote the packet service rate of short, long and overall packets,
respectively. They have the following relation:
\[
\frac{1}{u} = \frac{\lambda_{\text{short}}}{\lambda} \frac{1}{u_{\text{short}}} + \frac{\lambda_{\text{long}}}{\lambda} \frac{1}{u_{\text{long}}}. \tag{3}
\]

(3) **Offered load**: offered load is defined as the proportion of packet arrival rate and packet service rate. We use $\rho_{\text{short}}$, $\rho_{\text{long}}$ and $\rho$ to represent the offered load of short, long and overall packets. From (3) we can get
\[
\rho = \rho_{\text{short}} + \rho_{\text{long}}. \tag{4}
\]

(4) **$E(S)$**: the average packet service time of overall packets, which is directly proportional to the average packet size.
\[
E(S) = \frac{\text{Average packet size in byte}}{\text{Transferred bytes per time slot}}. \tag{5}
\]

(5) **$C_v$**: the coefficient of variation of the packet service time.

(6) **$E(W_{\text{cell}}), E(W_{\text{packet}})$**: the average packet waiting time of overall packets in cell-mode and packet-mode scheduling, respectively.

(7) **$E(W_{\text{SPF,S}}, E(W_{\text{SPF,L}}), E(W_{\text{SPF}})$**: average packet waiting time of short, long and overall packets in SPF.

Under admissible arrival traffic, a cell-mode scheduling algorithm can be paralleled to the processor-sharing service model [21]. This is because a cell-mode scheduling algorithm, such as iSLIP [10], tends to distribute the output bandwidth over all input ports equally with the cell interleaving. Therefore, we can obtain
\[
E(W_{\text{cell}}) = \frac{\rho E(S)}{1 - \rho}. \tag{6}
\]

Since each packet is served separately with no interleaving of cells of other packets, a packet-mode scheduling algorithm is corresponding to M/G/1 FCFS queueing model [22].
\[
E(W_{\text{packet}}) = \frac{1 + C_v^2}{2} E(W_{\text{cell}}). \tag{7}
\]

Marsan et al. [13] compares the performance of cell-mode and packet-mode scheduling and gets the following results:

When $C_v < 1$, $E(W_{\text{cell}}) > E(W_{\text{packet}})$;
When $C_v = 1$, $E(W_{\text{cell}}) = E(W_{\text{packet}})$;
When $C_v > 1$, $E(W_{\text{cell}}) < E(W_{\text{packet}})$.

In SPF, there are two priorities: high priority for short packets and low priority for long packets. With the non-preemptive priority model [22], we can get
\[
E(W_{\text{SPF,S}}) = \frac{\rho E(S)}{1 - \rho_{\text{short}}} \frac{1 + C_v^2}{2}, \tag{8}
\]
\[
E(W_{\text{SPF,L}}) = \frac{\rho E(S)}{(1 - \rho_{\text{short}})(1 - \rho)} \frac{1 + C_v^2}{2}. \tag{9}
\]

So the average packet waiting time of overall packets is
\[
E(W_{\text{SPF}}) = \frac{\lambda_{\text{short}}}{\lambda} E(W_{\text{SPF,S}}) + \frac{\lambda_{\text{long}}}{\lambda} E(W_{\text{SPF,L}})
= \frac{1 - \lambda_{\text{short}}/u_{\text{short}}}{1 - \lambda_{\text{short}}/u_{\text{short}}} E(W_{\text{packet}}). \tag{10}
\]

Since $u_{\text{short}} > u$ is always true when the arrival rate of long packets is greater than zero, from (10) we can get $E(W_{\text{SPF,L}}) < E(W_{\text{SPF,S}})$. So when $C_v < 1$, $E(W_{\text{SPF,S}}) < E(W_{\text{SPF,L}})$.

In general packet-mode scheduling, since short packets and long packets are not discriminated, the average packet waiting time of short packets is the same as the average packet waiting time of overall packets. From (7) and (8), we can get
\[
E(W_{\text{SPF,S}}) = \frac{1 - \rho}{1 - \rho_{\text{short}}} E(W_{\text{packet}}). \tag{11}
\]

From (11), we know that the average packet waiting time of short packets in SPF is less than that in general packet-mode scheduling.

Above theoretical analysis verifies that SPF reduces the average packet waiting time of both short packets and overall packets, which meets the goal of the proposed SPF.

5. **Performance simulations**

Previous section gives the theoretical analysis of system performance under low to medium offered load. This section presents simulation results and extends our conclusions under heavy offered load.

5.1. **Simulation model**

The switch size is $16 \times 16$, i.e. $N=16$. One million time slots are simulated and only results between 200,000 and 800,000 are recorded in the steady state.

The scheduling algorithm used in cell-mode is 2i-SLIP [10], in packet-mode is 2i-SLIP packet modification [13], and SPF with two iterations as described in Section 3.

Poisson arrival process is used widely in theoretical analysis, but it does not take care of the overlapping ingress time of packet transferring. In other words, the interval time between any two packets is definitely no less than the time to transfer a minimum IP packet. Therefore, in our
simulations we use a more accurate two-state ON–OFF model to characterize 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 of each input port.

ON state: packets are received in this state. Destinations of arrival packets are uniformly distributed over all output ports. ON state ends when a packet is completely transferred.

At the same time, we omit the cell padding process of packets whose sizes are not integral times of a cell unit, and use the TRIMODEL model to characterize the sizes of arrival packets.

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, for a cell is 64 bytes as mentioned in Section 2. This traffic model and these parameters are the same as results shown in [13], which are measured in actual backbone networks. Similar parameters are also reported in [17] and [23]. Therefore TRIMODEL(1, 9, 24, 0.559, 0.200) is a rather accurate model to describe the real Internet traffic.

5.2. Performance on throughput

It has been proven in [14] that IQ switches using packet-mode scheduling can achieve 100% throughput when arrival traffic is admissible and re-generative. This result directs the research on packet-mode scheduling. Fig. 6 shows the throughput of SPF, cell-mode and packet-mode scheduling under TRIMODEL(1, 9, 24, 0.559, 0.200). In this figure, three curves overlap almost everywhere, and the output link utilization in all three scheduling modes is the same as the offered load. In other words, all of them can provide approximate 100% throughput. In SPF, the adoption of separate queues for short packets and the strategy of scheduling short packets first do not degrade the throughput of general packet-mode scheduling. This result establishes the foundation for the following performance simulations.

5.3. Performance on average packet waiting time

Under TRIMODEL(1, 9, 24, 0.559, 0.200), we first calculate the queueing parameter E(S) and C_v. By basic probability theory [21], we can get

\[ E(S) = aP_a + bP_b + c(1 - P_a - P_b) = 8.143, \]  

and

\[ C_v = \frac{a^2P_a + b^2P_b + c^2(1 - P_a - P_b)}{E(S)^2} - 1 = 1.346. \]  

With the increase of offered load from 0.2 to 1.0, simulation results on average packet waiting time of overall packets are shown in Fig. 7. When \( \rho < 0.8 \), the average packet waiting time of cell-mode scheduling is less than general packet-mode scheduling. This result accords with what we get in Section 4 in theory: when \( C_v \) is greater than 1.0, cell-mode scheduling performs better under low to medium traffic. However, when \( \rho > 0.8 \), the average packet waiting time of cell-mode scheduling becomes the worst. This is because with the increase of offered load, conflicts, which should be arbitrated by the scheduling algorithm, occur more and more frequently. In cell-mode scheduling, all cells of the same packet need to be scheduled separately. While in packet-mode scheduling, only one scheduling is needed by every packet. If a packet is granted, all the following cells of this packet can be transferred continuously, so the scheduling task in cell-mode scheduling is much heavier than that in packet-mode scheduling.
Light scheduling task may make the design of algorithms in packet-mode scheduling easier.

In Fig. 7 we can also get that when \( r < 0.7 \), SPF is better than general packet-mode scheduling, just as the results analyzed in theory. When \( r > 0.7 \), the average packet waiting time of SPF becomes the best among three curves. With the increase of offered load, the advantage of SPF becomes more and more obvious, e.g. when the offered load is 0.95, the average packet waiting time of SPF, packet-mode and cell-mode scheduling is 227, 334, and 714 cells.

Furthermore, Fig. 8 plots the average packet waiting time of short packets. From this figure, we can observe that under low to medium load, the performance of cell-mode scheduling is the best, and the performance curve of SPF is between cell-mode and packet-mode scheduling. However, when the offered load goes up beyond 0.8, SPF becomes the best and the waiting time both in cell-mode and packet-mode scheduling increases more and more sharply. Even when the offered load is 1.0, the average packet waiting time of short packets in SPF is still less than 160 cells.

In SPF, the first priority of short packets may increase the average packet waiting time of long packets. Among these three curves, cell-mode scheduling is the worst. This is because interleaving of cells increases the average packet waiting time of long packets greatly. Compared to general packet-mode scheduling, in SPF the average packet waiting time of long packets is just a little larger and does not suffer a great degradation from the first priority of short packets, e.g., the average packet waiting time of long packets is 161 and 139 cells in SPF and packet-mode scheduling when the offered load is 0.9. This is mainly because the service time of short packets is rather small, which is just one time slot.

In brief, compared to general packet-mode scheduling, SPF is always better on the average packet waiting time of short packets and overall packets under any offered load, at the cost of a slight unavoidable performance degradation of long packets. Compared to cell-mode scheduling, SPF performs better under heavy offered load, especially for short packets, while slightly worse under low to medium load. But in cell-mode scheduling, the necessary reassembly processing at output ports will introduce extra delay to every packet. In SPF, however, the reassembly module is removed and there is no reassembly delay any more. Therefore, when taking into account the reassembly delay at each output port, the benefit for short packets achieved under low to medium load in cell-mode scheduling will be partially counteracted, and this makes SPF preferable even under low to medium offered load. Moreover, SPF achieves better average packet waiting time of long packets than cell-mode scheduling.

5.4. Impact of multiple iterations

In cell-mode scheduling, multiple iterations of scheduling are required to achieve maximal matching. SPF can achieve better performance with less iterations. Fig. 10 shows the impact of iteration number on the average packet waiting time of overall packets under TRIMODEL(1, 9, 24, 0.559, 0.200). When the iterations increase from two to four, the average packet waiting time of cell-mode scheduling is reduced greatly under heavy offered load, but the performance of SPF and general packet-mode scheduling is not obviously improved. This result again indicates that the scheduling task in cell-mode scheduling is much heavier than SPF and general packet-mode scheduling.

Fig. 10 also illustrates that the performance of 2\textsuperscript{i}-SPF (2 iteration SPF) is almost identical to the performance of 4\textsuperscript{i}-SPF and better than both 4\textsuperscript{i}-Cell and 4\textsuperscript{i}-Packet. The benefit of 2\textsuperscript{i}-SPF is that it can be implemented more easily under the same hardware condition because it reduces the iteration number of the scheduling algorithm.
5.5. Simulations under heavy load of short packets

TRIMODEL(1, 9, 24, 0.559, 0.200) is rather an accurate model to describe the current Internet traffic characteristics and from it we have obtained many worthwhile results. When the load of short packets is large enough, the waiting time of long packets will increase in SPF. We extend to do an extreme model to study the scenario where the offered load of short packets is high. We increase the percentage of short packets from 0.559 to 0.900, and then we use the TRIMODEL(1, 9, 24, 0.900, 0.050) to do more simulations. Four iterations of SPF, packet-mode scheduling and cell-mode scheduling are executed, respectively.

Fig. 11 shows the average packet waiting time of long packets. When the percentage of short packets is added to 0.900, the difference of long packets’ average packet waiting time between SPF and general packet-mode scheduling becomes a little larger. e.g., when the offered load is 0.9, the average packet waiting time of long packets in general packet-mode scheduling, SPF and cell-mode scheduling is 80, 136 and 174 cells, respectively. We, however, can still get that the average packet waiting time of long packets in SPF is less than that in cell-mode scheduling. Although there is no service discrimination between short and long packets in cell-mode scheduling, it causes large waiting time of long packets. Therefore, even preferring short packets in SPF, the average packet waiting time of long packets in SPF will be still more satisfying than cell-mode scheduling. Generally speaking, the offered load of short packets in Internet cannot be too large because the dominant TCP protocol utilizes as many large packets as possible to avoid silly window syndrome [24].

The performance of TCP flows is determined by the delay of large data packets as well as short packets (such as TCP ACKs). Fig. 12 shows the average packet waiting time of short packets under TRIMODEL(1, 9, 24, 0.900, 0.050). In this figure we can also obviously see the advantages of SPF under heavy offered load. The average packet waiting time of short packets in SPF is less than 90 cells under the offered load of 1.0, but it increases sharply from the offered load of 0.85 to 1.0 in both cell-mode and packet-mode scheduling. Herein, we can also get that the average packet waiting time of short packets in SPF is always less that in general packet-mode scheduling under TRIMODEL(1, 9, 24, 0.900, 0.050).

5.6. Discussion on SPF

In SPF the first priority and separate queues for short packets may cause out-of-sequence of a TCP flow with
a mix of short and long data packets. Out-of-sequence does affect the performance of IP networks, but it is not a pathological behavior.

First of all, out-of-sequence is allowed [25] and does exist in Internet for local parallelism in routers and links [26]. Second, out-of-sequence is not always destructive on the performance of TCP. In SPF, most of short packets are TCP ACK packets and TCP control packets [17]. For TCP traffic, earlier arrival of these control packets will cause a TCP sender to open its congestion window and send TCP flows more quickly. The consequent result is that the delay of TCP flows is reduced, and this is exactly required in high bandwidth networks. Finally, in terabit core routers, traffic is heavily aggregated and the fraction of TCP data packets belonging to short packets is very small, so we expect that there is a very low probability that two packets of the same TCP flow arriving consecutively are close enough to cause out-of-sequence of short and long data packets in SPF.

6. Conclusions

Although many input queueing scheduling algorithms are proposed to guarantee reserved bandwidth and delay performance of each flow in IP networks, this paper first demonstrates that short packets in a TCP flow should be guaranteed higher priority and lower delay. We further study a simple but effective scheduling algorithm, called SPF, to optimize general packet-mode scheduling. By combining the advantages of cell-mode scheduling for short packets and packet-mode scheduling for long packets, SPF can provide advantages over both cell-mode and packet-mode scheduling.

Both theoretical analysis under low to medium offered load and simulation results show that SPF can reduce the average packet waiting time of short packets, especially under moderate and heavy offered load. At the same time, the average packet waiting time of long packets is only increased slightly. As a good result of the performance tradeoff between short and long packets, the average packet waiting time of overall packets is also lowered with SPF. For networks filled with TCP flows, the result is particularly valuable. Simultaneously, SPF favors QoS of VoIP (of short packets), a kind of hot real time traffic increasing rapidly in recent years.

Acknowledgements

We are likewise indebted to Dr Martin Collier and Dr Heng Liao for their instructive comments and suggestions on this work. We also wish to thank the anonymous reviews of this paper for their remarkable comments to enhance the quality of the manuscript.

References

| Wenjie Li was born in Henan Province, China, on November 13 1978. He received his B.E. Degree from Department of Computer Science and Technology, Tsinghua University, China in 2000. He is currently working toward the PhD Degree in Institute of Computer Network Technology, Department of Computer Science and Technology, Tsinghua University, China. His research interests include high-speed broadband networks, core router architecture, practical scheduling algorithms and QoS. |
| Bin Liu was born in Shandong Province, China, in July 1964. He received his PhD Degree from Department of Computer Science and Engineering, Northwestern Polytechnic University, China in 1993. He is currently a full professor in Department of Computer Science and Technology, Tsinghua University, China. His research interests include methodology and theory of modern switching technology, high-speed network protocol and performance evaluation, QoS control and the behavior analysis of multimedia traffic. |