Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090436

Pushout with Differentiated Dropping Queue Management for High-Speed Networks

*Jui-Pin Yang**

Department of Information Technology and Communication, Shih-Chien University, Kaohsiung 84552, Taiwan, R.O.C.

Received: 6 Nov. 2014, Revised: 6 Feb. 2015, Accepted: 7 Feb. 2015
Published online: 1 Jul. 2015

Abstract: Queue management is critical for high-performance and high-speed routers. Pushout scheme (PO) performs well in terms of buffer utilization and packet loss probability, but requires identifying the longest flow queue and executing frequent pushout operations whenever a packet arrives at a full buffer. Additionally, PO cannot effectively protect lightly loaded flows against bandwidth aggression from heavily loaded flows under bursty traffic conditions. To overcome the disadvantages of the PO, this study proposes a simple but efficient queue management scheme, namely pushout with differentiated dropping (PDD). PDD uses a weight function to estimate the weights of active flows based on their traffic intensity; moreover, it maintains the flow states of two approximate maximum and sub-maximum differentiated factors. By comparing differentiated factors of arriving packets with both maintained differentiated factors, PDD can correctly deal with each arriving packet by discard, pushout or acceptance. Simulation results verify that PDD has better fair bandwidth sharing and much lower packet pushout probability than PO under a variety of traffic conditions.

Keywords: differentiated dropping, queue management, fair bandwidth sharing, packet pushout probability

1 Introduction

When a feasible queue management scheme is applied to manage the buffer, it benefits high buffer utilization and low packet loss performance. Furthermore, this scheme is useful to enhances fair bandwidth sharing and upgrade congestion control mechanisms [1,2]. Queue management thus is important for network routers, which could be divided into three types including static threshold, dynamic threshold and pushout. The static threshold schemes generally designate one or more fixed thresholds in advance which are used to control the growth of queue lengths [3,4]. Ideally, the control thresholds should be dynamically adjusted in order to fit for varying traffic. Accordingly, many dynamic threshold schemes are proposed [5,6,7,8,9]. In a word, the threshold-based types are easy to implement, but they both are unable to achieve robust performance.

In order to improve the threshold-based types, the pushout (PO) type is proposed [10,11,12,13,14,15,16,17,18]. In general, the PO type fully utilizes the buffer capacity and allows each flow to increase in queue length until they reach fair buffer allocation. When the buffer is full, a residing packet belonging to the longest flow queue should be pushed out to make room for the new arrival. On the other hand, no additional constraint is applied before the buffer is full. The PO type is superior to the threshold-based types in several performance metrics inclusive of buffer utilization, fair buffer usage and packet loss probability. However, the PO type should find out the longest flow queue and then execute a pushout operation whenever a packet arrives at a full buffer. This causes PO type too difficult to deploy in high-speed networks. In addition, this type favors the heavily loaded flows under bursty traffic conditions that could lead to unfair bandwidth sharing on the lightly loaded flows.

We propose a novel queue management scheme which possesses low complexity and excellent fair bandwidth sharing namely pushout with differentiated dropping (PDD). PDD is a variant of the PO type, but it eliminates the original disadvantages. The PDD first estimate the traffic intensity of active flows and then use a weight function to transform traffic intensity into the corresponding weight. Next, the PDD evaluated the differentiated factor (df) of each arriving packet. The df is equal to the product of current queue length and weight. By comparing the df of each arriving packet with two approximate maximum and sub-maximum df, the PDD can deal with arriving packets correctly. A representative

* Corresponding author e-mail: juipinyang@gmail.com

PO scheme is selected from the PO type as a comparison [10]. Considering the implementation overheads of the PDD and PO, we analyze packet pushout probability of both schemes.

An adequate scheduling algorithm greatly contributes to improve the fairness of the queue management schemes. In this study, we consider the PDD and PO cooperating with a well-known scheduling algorithm, namely deficit round robin (DRR) [19]. The main idea of the DRR is that arriving packets are classified first before they are dispatched to a dedicated flow queue. Next, the residing packets in the forefront of each nonempty flow queue are served in turns until their respective deficit count is smaller than the size of the following residing packet. We summarize the differences between PDD and PO;

1. PDD supports differentiated dropping by simply comparing the differentiated factors. Otherwise, the PO merely considers current queue lengths. Consequently, PDD can effectively protect the lightly loaded flows from bandwidth aggression by the heavily loaded flows. PDD thus achieves better fair bandwidth sharing than the PO.
2. PDD has lower implementation overheads than PO. PO needs to execute a pushout operation whenever a packet arrives at a full buffer. However, PDD only needs to execute a necessary pushout operation by early and precisely discarding arriving packets. PDD thus has much lower packet pushout probability than the PO.
3. PO needs to identify the longest flow queue among all active flows. However, PDD only needs to compare the differentiated factor of each arriving packet with two approximate maximum and sub-maximum differentiated factors. PDD thus has lower implementation complexity than the PO.

The remainder of this paper is organized as follows. Section 2 illustrates previous studies related to the queue management schemes and scheduling algorithms. Section 3 consists of three components. First, we describe the detailed PDD scheme, including intensity estimation, weight function and then complete PDD algorithm; second, we define two performance indexes, namely normalized bandwidth ratio and packet pushout probability, and use both for performance measurement; third, we analyze the complexity of the PDD and PO. Computer simulations are used to compare the fairness and pushout behavior of the PDD with PO under various traffic conditions in Section 4. Finally, we conclude with a summary in Section 5.

2 Related work

Many queue management schemes have been proposed to improve specific performance metrics such as throughput, fairness, packet loss probability, multiple loss priorities and so on. Irland proposed a queue management scheme that optimizes growth in queue lengths according to traffic conditions [3]. This scheme is too complicated to implement because it needs instant measurement of current traffic behavior. A simplified square-root rule was proposed. However, it leads to performance degradation. On the other hand, several static threshold schemes were proposed and analyzed based on product form solutions, including complete sharing (CS), complete partitioning (CP), sharing with maximum queue lengths (SMXQ), sharing with a minimum allocation (SMA), and sharing with a maximum queue and minimum allocation (SMQMA) [4]. Although these schemes are easy to implement, they perform well under limited traffic conditions.

To improve the disadvantages of the static threshold schemes, dynamic threshold schemes that automatically adjust control thresholds according to buffer variations were proposed [5,6,7,8,9]. Dynamic queue length threshold (DT) changes control threshold in association with residual buffer size [5]. When the queue length of a flow equals or exceeds the control threshold, the DT discards all arriving packets from that flow until its queue length is smaller than the control threshold. DT has low buffer utilization because excessive buffer size is reserved to guarantee high throughput and low packet loss probability of the lightly loaded flows. An extended version of the DT was proposed by considering multiple packet loss priorities [6]. Hierarchal queue management (HBM) can improve buffer utilization and achieve fair buffer usage at the same time [7]. In addition, HBM works as CS or CP based on the setting of a control threshold.

Considering buffer utilization and fair buffer usage, partial sharing and partial partitioning (PSPP) was proposed which has better performance than the HBM [8]. PSPP first classifies flows as active or inactive. Next, it reserves sufficient buffer size for all inactive flows. Finally, the residual buffer size is fairly allocated among each active flow. Although the PSPP performs well under many traffic conditions, it is still unable to fairly allocate buffer size. Consequently, a threshold-based selective drop (TSD) that originates from the concept of the max-min fairness was proposed [9]. Dynamic threshold schemes often have better and more robust performance than static threshold schemes in terms of throughput, fair buffer usage and packet loss probability. However, all threshold-based queue management schemes (static threshold and dynamic threshold) cannot fully utilize buffer size because they have to reserve a certain amount of buffer size so as to implement effective queue management.

PO is also named as drop on demand policy [10]. When the buffer is not full, each arriving packet is admitted to the buffer. Otherwise, one residing packet belonging to the longest flow queue should be removed. PO always performs much better than the threshold-based schemes. Unfortunately, the PO may degrade under

asymmetric traffic conditions. For instance, when the lightly loaded flows have larger traffic burstiness than the heavily loaded flows, their queue lengths suddenly increase. As a result, the lightly loaded flows possess higher packet loss probability, lower throughput and unfair bandwidth sharing. A variant of the PO was proposed to alleviate the drawback, namely pushout with threshold (POT) [11]. The main difference between PO and POT is that the latter assigns a dedicated control threshold to each flow. When the buffer is full, the control threshold is used to determine whether a residing packet should be pushed out by other flows. The POT is sophisticated because it has to assign adequate thresholds to each flow where a lot of current flows exist. To simplify POT, a maximum busy period scheme was proposed by modifying the original assumptions of the POT (poisson arrival, exponentially distributed service time) and considering correlated contiguous time slots [12].

Many network applications require differentiated packet treatment, and therefore several modified schemes based on PO were proposed to support multiple packet loss priorities [13,14,15,16,17,18]. PO-based schemes often keep promising performance, but they suffer at least three critical drawbacks. First, they are relatively complex since they have to find out the longest flow queue. Second, they need to execute frequent pushout operations whenever a packet arrives at a full buffer. This means that they have high implementation overheads. Third, they are unable to achieve fair bandwidth sharing under asymmetric traffic conditions.

To resolve the drawbacks of traditional PO-based schemes, pushout with differentiated dropping (PDD) queue management scheme is proposed. PDD benefits fair bandwidth sharing while it keeps low implementation complexity and overheads. On one hand queue management schemes are used to determine which arriving packets could be admitted to enter the buffer, and on the other hand scheduling algorithms are used to schedule the transmission of residing packets. Both mentions thus must cooperate well so as to achieve the optimal performance. Deficit round robin (DRR) is a well-known scheduling algorithm which provides excellent fair bandwidth sharing accompanying with low implementation complexity [19]. Many modified scheduling algorithms based on the DRR have been proposed to deal with various quality of service (QoS) requirements such as proportional bandwidth, delay, jitter and so on [20,21]. To summarize, we analyze the fairness and packet pushout probability of the PDD and PO under a variety of traffic conditions where DRR works as a default scheduling algorithm in the following simulations.

3 Pushout with differentiated dropping

The details of the PDD queue management scheme are described as follows; (1) intensity estimation for active flows, (2) weight function for transforming intensity into corresponding weight, and (3) a complete PDD algorithm. Next, we describe the definitions of two performance indexes and analyze the implementation complexity of the PDD and PO.

3.1 Intensity estimation

Before explaining the intensity estimation method, we define a flow that is composed of a stream of packets with the same source and destination IP addresses. Furthermore, a flow is identified as active only if it has at least one residing packet in the buffer. We use a weighted moving average to estimate the traffic intensity of the active flows, which resembles in random early detection (RED) [22]. Equation (1) is the method to estimate the intensity where $I_{i,k}$ denotes the estimated intensity of active flow i at the beginning of time interval k, and $m_{i,k-1}$ denotes the amount of arriving packets belonging to active flow i during time interval k-1. T_d denotes the duration of a time interval and C denotes output link capacity. In addition, w_a is a parameter that affects the dependency of intensity estimation in association with short-term or long-term traffic conditions where $0 \le w_a \le 1$. PDD enrolls the estimated intensity of each active flow into the ActiveList of DRR especially. To obtain accurate intensity estimation, PDD changes the maintenance rule on the ActiveList by removing the records of active flow i from the ActiveList at the beginning of time interval k only if $m_{i,k-1} = 0$. PDD thus can get rid of frequent updates to the ActiveList.

$$I_{i,k} = w_a \cdot \frac{m_{i,k-1}}{C \cdot T_d} + (1 - w_a) \cdot I_{i,k-1} \quad (1)$$

3.2 Weight function

The weight function is used to transform estimated intensity into corresponding weights related to active flows. PO only compares current queue lengths of the active flows and then selects a residing packet to be pushed out on demand. Consequently, PO cannot effectively protect the lightly loaded flows from bandwidth aggression of the heavily loaded flows that degrades fairness and increases packet pushout probability. The basic principle of the weight function is to transform small intensity to a small weight. Otherwise, large intensity is transformed to a large weight. The simplest weight function involves allocating weight linearly in proportion to intensity but it may result in insufficient flow differentiation that greatly degrades the fairness of the lightly loaded flows.

Based on the analysis, we establish a two-phase weight function, as described in Equation (2). The α is a parameter which denotes the differentiated degree where $\alpha > 1$. Additionally, $WF_{i,k}$ denotes the weight of active

flow i at the beginning of time interval k. When $0 \le I_{i,k} \le 1$, $WF_{i,k}$ is estimated using a logarithmic function in order to provide sufficient flow discrimination for the lightly loaded flows. On the other hand, $WF_{i,k}$ is linearly proportional to intensity when $I_{i,k} > 1$. $I_{i,k}$ is larger than 1 only if the incoming links have higher capacity than the outgoing link. This weight function is motivated by the idea of colored layers from rainbow fair queueing (RFQ) **[23]**. The weight function is critical because it dominates the PDD performance on fairness and packet pushout probability.

$$WF_{i,k} = \begin{cases} log_{\alpha}(1+(\alpha-1)\cdot I_{i,k}) & 0 \le I_{i,k} \le 1 \\ I_{i,k} & I_{i,k} > 1 \end{cases} \quad (2)$$

3.3 PDD algorithm

In general, the arriving packets that have larger weights and current queue lengths should be discarded with higher probability. Accordingly, a differentiated factor (df) is defined that is equal to the product of weight and current queue length. Figure 1 illustrates the complete PDD algorithm.

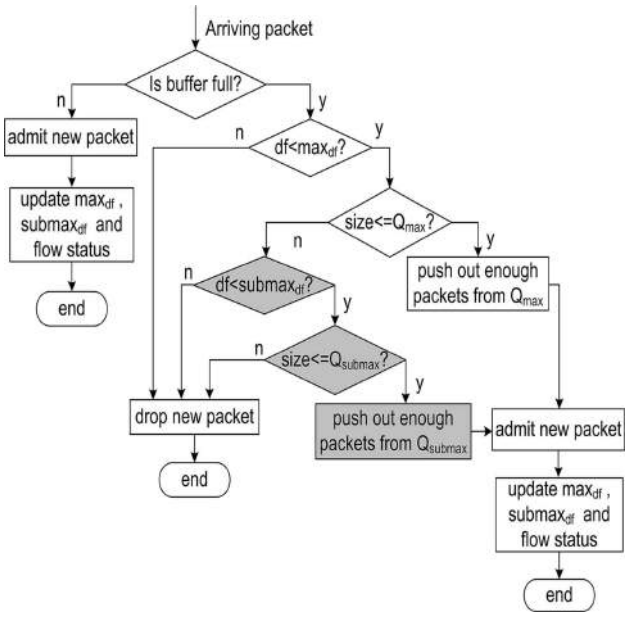

Fig. 1: PDD algorithm

When the buffer is not full, all arriving packets are accepted. Furthermore, PDD compares the df of each arriving packet with the max_{df} and $submax_{df}$ where they both represent the approximate maximum and sub-maximum df respectively. After comparisons, PDD updates the flow states of the max_{df} and $submax_{df}$. The flow state includes the weight, current queue length and flow identification (a pair of IP addresses). PDD can use the flow identification to locate the residing packets of a specific active flow which are candidates for being pushed out on demand. In this phase, the goal of the PDD is to update related flow states of both max_{df} and $submax_{df}$, and prepare for consequent pushout operations.

When the buffer is full, the comparisons of df differ from the previous phase. In the first place, PDD only compares the df of the arriving packet with max_{df}. Furthermore, it adopts a different way to estimate the current queue length. For a new arriving packet, PDD adds its packet size to current queue length in the buffer within the same flow; meanwhile, the current queue length of the max_{df} is subtracted from the size of the arrival. Without exception, PDD adopts the same rule for the $submax_{df}$. The benefit to the PDD is that it only executes necessary pushout operations. If the df of the arriving packet is larger than the max_{df}, the new arrival is directly discarded without any additional procedures. Otherwise, PDD has to compare the size of arriving packet with Q_{max}, where Q_{max} denotes current queue length of a given flow with max_{df}. If the Q_{max} equals or exceeds the packet size of the arrival, the new arrival can be accepted by pushing out a residing packet. PDD selects the residing packet to be pushed out is from tail to head. Next, PDD may need to update the flow states related to the max_{df} or $submax_{df}$. If Q_{max} is smaller than that of the new arrival, PDD will compare its df with the $submax_{df}$ further that is similar to the comparison with the max_{df}. If Q_{submax} is larger than packet size of the arrival, the arriving packet is admitted to enter the buffer where Q_{submax} denotes current queue length of a given flow with $submax_{df}$. Particularly, the flow states related to the max_{df} and $submax_{df}$ is invariable herein.

Two more differentiated factors are preferable to PDD because they make for better fairness. Unfortunately, this may also increase packet pushout probability and implementation complexity at the same time. Based on previous analysis, two differentiated factors are sufficient for the PDD to cope with various traffic conditions. In this phase, the goal of the PDD is to determine the most optimal residing packet to be pushed out. As a result, PDD can achieve excellent fairness and avoid unnecessary pushout operations.

3.4 Performance indexes

To completely analyze the PDD and PO, two performance indexes are defined, inclusive of packet pushout probability and normalized bandwidth ratio (NBR). The former is equivalent to the number of pushed out packets divided by the number of arriving packets for a flow. The latter is associated with the max-min fairness **[24]**, as shown in Equation (3) where N denotes the number of active flows and f denotes the max-min fair rate. Also, r_i denotes the mean arrival rate of flow i.

$$\sum_{i=1}^{N} min\{r_i, f\} = C \quad (3)$$

Equation (4) is the definition of normalized bandwidth ratio where NBR_i denotes the NBR of flow i and D_i denotes the mean departure rate of flow i. If the NBR of each active flow equals 1, a queue management scheme achieves the optimal fairness.

$$NBR_i = D_i / min\{r_i, f\} \quad (4)$$

3.5 Complexity analysis

Whenever there is an arriving packet, the PDD needs a maximum of B comparisons to obtain current queue length belonging to the same flow where B denotes the number of residing packets. Meanwhile, both Q_{max} and Q_{submax} are available. As a result, PDD has time complexity of O(B). PDD has to maintain traffic intensity and the weight of each active flow, so its space complexity is proportional to the number of active flows, namely O(N). On the other hand, PO has to find out the longest flow queue among the active flows, so its time complexity is around $O(N) \sim O(N^2)$, depending on select sorting algorithms. In addition, PO has to maintain queue lengths related to the active flows, so its space complexity is the same as that of PDD. Generally, N significantly exceeds B, so PDD has lower implementation complexity than PO. Based on above analysis, PDD is more suitable to be deployed in high-speed network environments.

4 Simulation Results

In Figure 2, we consider a single congested link topology. Also, all network links have the same capacity of 10 Mbps. Moreover, there are 10 flows and buffer size is set at 20 KB. Each flow generates packets based on a specific ON-OFF traffic model. To simplify computer simulations, assume that all arriving packets have the same packet size of 1 KB. The simulation time for each experiment is set to 200 seconds. In the DRR, the quantum size is set at 1 KB. The parameters for the PDD are set as follows; $w_a = 0.3$, $T_d = 16$ ms and $\alpha = 10$. Unless otherwise specified, all mentioned configurations are applied to successive simulations all the time. We compare the normalized bandwidth ratio and packet pushout probability of PDD with that of PO in subsections 4.1 and 4.2, respectively.

4.1 Fairness

This subsection analyzes the fairness of PDD with PO under various traffic conditions by comparing their NBRs. All flows are numbered from 1 to 10, and transmit several times over the max-min fair rate. The max-min fair rate is

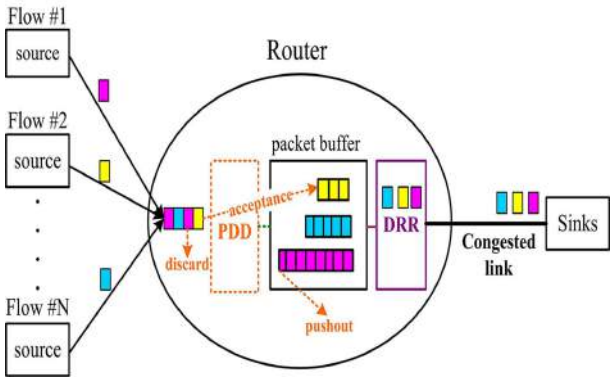

Fig. 2: A single congested link topology

1 Mbps. In other words, the average arrival rate for each flow is set to [1 2 3 4 5 6 7 8 9 10 (Mbps)], respectively. Figure 3 illustrates the NBR versus different buffer sizes. If the buffer size is set at 10 KB, the fair buffer usage for each flow equals 1 KB. This leads to extremely unfair bandwidth sharing in both schemes due to an extreme lack of buffer size. In the PO, the NBR of flow 1 is close to 0.52. However, the NBR of flow 1 is close to 0.72 in the PDD. This reason is that PDD assigns lower weights to the lightly loaded flows, so their packets are more likely to be accepted and less likely to be pushed out. Furthermore, flow 10 is the heaviest loaded flow which obtains the largest NBR near 1.15 in the PO. This may encourage network users to violate congestion control mechanisms so as to maximize their bandwidth. In the PDD, flow 8 has the largest NBR near 1.08 not flow 10. Thus PDD does not favor the heavily loaded flows like the PO. Both PDD and PO have the optimal fairness if the buffer size is set at 60 KB. The fair buffer usage for each flow equals 6 KB, which is sufficient for all flows to increase their queue lengths during bursty traffic conditions. Undoubtedly, an excellent queue management scheme is critical for achieving excellent fairness even if the DRR scheduling algorithm is applied to schedule the residing packets. Simulation results show that PDD achieves better fairness than PO.

In Figure 4, the traffic conditions are similar to Figure 3, except the average arrival rate of each flow is changed to [1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 (Mbps)], respectively. If the buffer size is set at 10 KB, flow 10 obtains the largest NBR, approaching 1.21 in the PO. Also, flow 10 in the PDD obtains the largest NBR near 1.11. Each flow has approaching average arrival rate, so PDD does not assign a sufficient weight to constrain the arriving packets of flow 10. Furthermore, flow 1 has better NBR in both schemes as compared with Figure 3. The buffer is relatively unlikely to be filled up because of the lower total arrival rate, leading to the acceptance of more arriving packets of flow 1. Both schemes improve fairness along with the increment of buffer size.

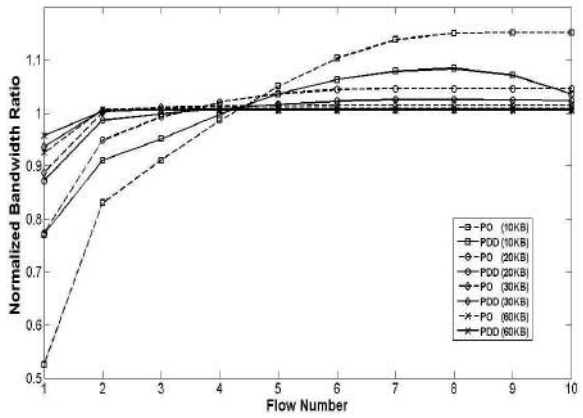

Fig. 3: Normalized bandwidth ratio versus different buffer sizes where average arrival rate varies from 1 Mbps to 10 Mbps

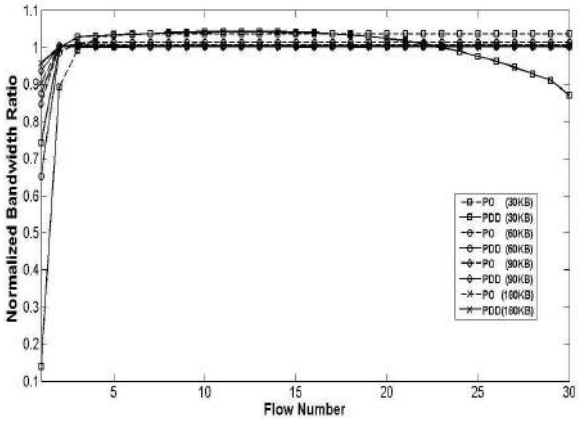

Fig. 5: Normalized bandwidth ratio versus different buffer sizes where average arrival rate of 30 flows varies from 1/3 Mbps to 10 Mbps

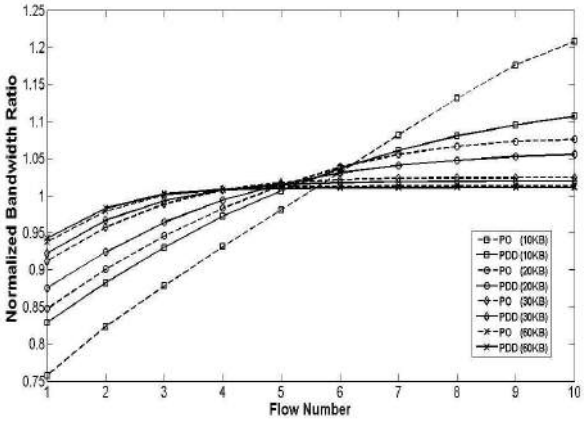

Fig. 4: Normalized bandwidth ratio versus different buffer sizes where average arrival rate varies from 1 Mbps to 1.9 Mbps

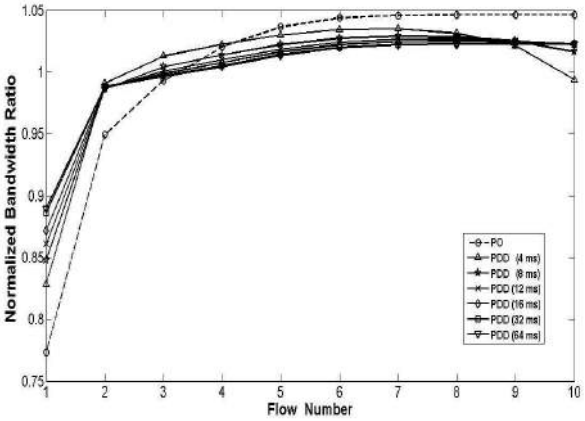

Fig. 6: Normalized bandwidth ratio versus different time intervals

In Figure 5, we consider 30 flows that have average arrival rates several times larger than the max-min fair rate, and hence the max-min fair rate is 1/3 Mbps. Accordingly, the average arrival rates of these flows are set to [1/3 2/3 1 4/3 5/3 2 ..... 10 (Mbps)], respectively. Flows 5 to 30 have approximate NBRs near 1.03 in the PO if the buffer size is set at 30 KB. As for the PDD, flows 24 to 30 all have NBRs below 1. PDD thus punishes flows with higher average arrival rate. Otherwise, the PO still allows the heavily loaded flows to seize bandwidth from the lightly loaded flows. Flow 1 has the worst NBR near 0.13 in the PO but the NBR grows near 0.74 in the PDD. PDD can effectively protect the lightly loaded flows against bandwidth aggression from the heavily loaded flows while it punishes the heavily loaded flows with less bandwidth below the max-min fair rate.

Figure 6 shows the effect of time intervals on the PDD. Additionally, the other traffic conditions are similar to Figure 3, except the buffer size is set at 20 KB. If T_d is set at 4 ms, PDD cannot correctly estimate flow intensity owing to insufficient statistical information. Accordingly, flow 1 has the lowest NBR near 0.83. When the T_d increases, PDD can estimate flow intensity more accurately. Consequently, the fairness of PDD is thus improved. If T_d is set at 32 ms, the NBR of flow 1 is near 0.88. Regardless of the value of the T_d , PDD always has better fairness than PO.

Figure 7 shows the effect on fairness under different burst lengths of flow 1. The traffic conditions are the same as Figure 6. When the burst length increases, both the PDD and PO schemes degrades their fairness. The average arrival rate of flow 1 is 1 Mbps which equals max-min fair rate. Consequently, a larger burst length will prevent arriving packets of flow 1 from being accepted because of short-term traffic burstiness. If the burst length is larger than 33, the NBRs of both schemes slowly decrease and finally approach a constant. This occurs because a certain amount of arriving packets of flow 1 are admitted to the buffer. Simulation results show that PDD is capable of resisting traffic burstiness.

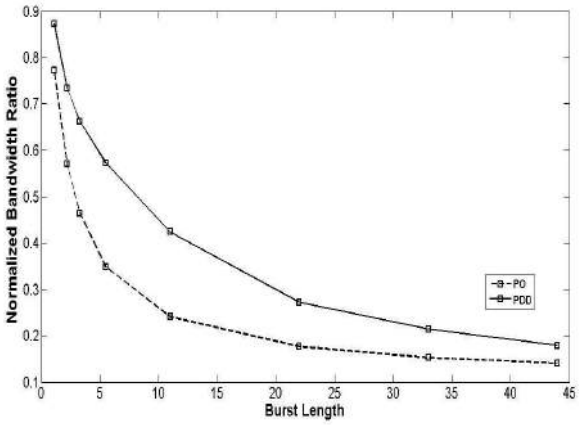

Fig. 7: Normalized bandwidth ratio versus different burst lengths

In Figure 8, the average arrival rate for each flow is set at either 1 Mbps or 10 Mbps. In addition, there are 10 flows and buffer size is set at 20 KB. In the "3 flows" case, flows 1 to 3 are all set at 1 Mbps, while the others are all set at 10 Mbps. The same explanations are used to illustrate "5 flows" and "8 flows" cases. In the "8 flows" case, two 10 Mbps flows in PDD obtain the highest NBR, approaching 1.22, higher than in the other two cases because only two flows share grabbed bandwidth from eight 1 Mbps flows. As for the PO, both flows have higher NBR near 1.26. In all cases, the PDD has better restraints on 10 Mbps flows than PO. On the other hand, PDD can protect 1 Mbps flows from the bandwidth aggression of 10 Mbps flows. Figures 3 to 8 show that PDD is capable of supporting better fairness than PO under a variety of traffic conditions while it is beneficial for congestion control.

4.2 Packet pushout probability

This subsection analyzes the packet pushout probability of PDD and PO. In Figure 9, the traffic conditions are the

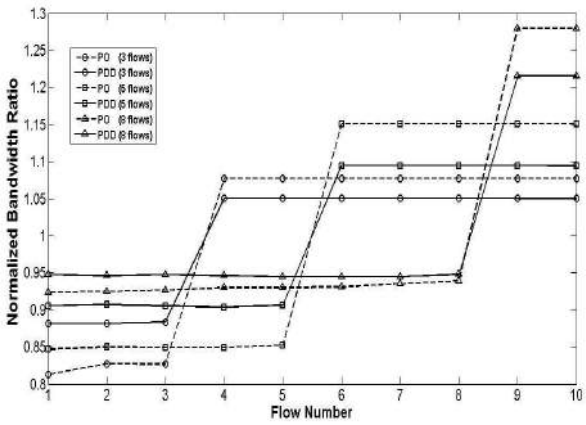

Fig. 8: Normalized bandwidth ratio versus different number of flows that consist of two kinds of average arrival rate

same as Figure 3. When a pushout operation is needed, both PDD and PO have to push out a residing packet from the buffer. In the PO, the operation involves finding out the longest flow queue. Inversely, PDD only needs simple comparisons. That is, PDD is much simpler to implement than PO even if both have the same packet pushout probability. In the PDD, the buffer size displays less improvement in packet pushout probability because it can correctly push out residing packets on demand. Hence, the packet pushout probability of PDD is relatively independent of buffer size than that of PO. Particularly, a large buffer size is helpful to improve the packet pushout probability of PO particularly for flow 1. Simulation results show that PDD has much lower packet pushout probability than PO. In a word, PDD has much lower implementation overheads than PO.

In Figure 10, the traffic conditions are the same as Figure 6. When T_d increases, PDD enhances the accuracy of flow intensity estimation and thus assigns more adequate weights to differentiate flows. All flows in PDD repeatedly have much lower packet pushout probability than that of PO especially for flow 1. If T_d is larger than 32 ms, PDD has a little improvement on packet pushout probability. In a word, the PDD has much lower packet pushout probability than PO under various time intervals.

Figure 11 considers the burst lengths of flow 1 and the traffic conditions are the same as Figure 7. In the "*overall*" case, the average packet pushout probability of all flows in PDD is near 0.1. However, it is near 0.85 in PO. PDD obviously shows much lower overall packet pushout probability than PO. In the case of "flow 1", a large burst length increases the packet pushout probability of flow 1 in both schemes. Flow 1 has more arriving packets because of traffic burstiness, and hence its packets are more likely to be discarded or pushed out. If the burst length exceeds 11, the packet pushout probability of flow

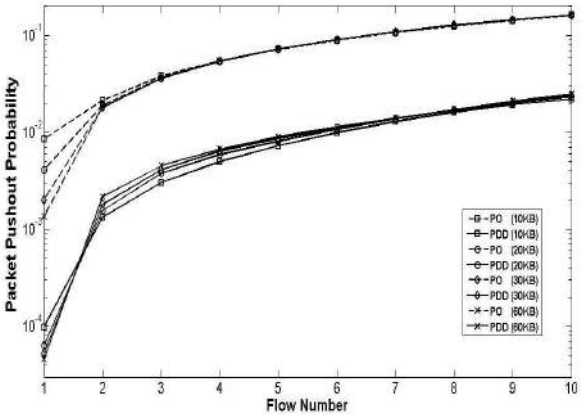

Fig. 9: Packet pushout probability versus different buffer sizes where average arrival rate varies from 1 Mbps to 10 Mbps

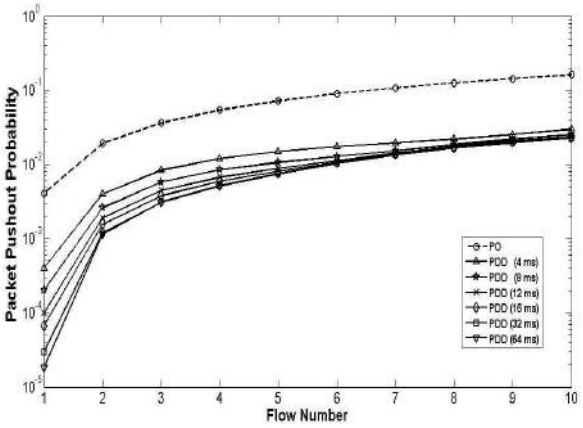

Fig. 10: Packet pushout probability versus different time intervals

1 is close to a constant, being either PDD or PO. The reason is that at least a certain amount of arriving packets of flow 1 are accepted in both schemes. Based on Figures 9 to 11, we conclude that PDD can support much lower packet pushout probability than PO under various traffic conditions. In summary, PDD can cooperate well with the DDR, contributing to excellent fairness and low packet pushout probability. Besides, the PDD is simpler to implement than PO. Undoubtedly, PDD queue management scheme is suitable for deployment in high-speed and high-performance routers.

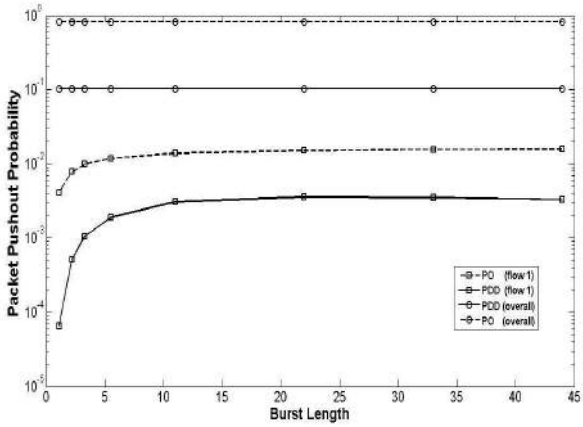

Fig. 11: Packet pushout probability versus different burst lengths

5 Conclusions

Queue management schemes are critical because they predominantly affect the performance of a router. The threshold-based schemes are easy to implement, but they perform well under limited traffic conditions. On the other hand, PO-based schemes achieve better performance, but they are too sophisticated to apply to high-speed networks. To overcome the above issues, a simple but efficient queue management scheme is proposed, namely pushout with differentiated dropping (PDD). PDD uses a weight function to transfer the traffic intensity into corresponding weights. In addition, PDD maintains flow states related to two approximate maximum and sub-maximum differentiated factors. By comparing and evaluating the differentiated factors, the PDD optimally determines packet treatment on arriving packets. Based on the simulation results, PDD achieves better fair bandwidth sharing, as well as much lower packet pushout probability and implementation complexity than PO under a variety of traffic conditions. In summary, PDD is suitable for high-speed and high-performance network environments. In the future, we would like extend the PDD by taking TCP Vegas [25] into account.

Acknowledgement

The work was supported by Shih-Chien University and National Science Council under grant number USC-101-05-05007 and NSC 100-2221-E-158-008 respectively, Taiwan, R.O.C..

References

[1] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge,

L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski and L. Zhang, Recommendations on queue management and congestion avoidance in the Internet, RFC **2309** (1998).
[2] A. Mankin, K. Ramakrishnan, Gateway congestion control survey, RFC **1254** (1991).
[3] M. Irland, Buffer management in a packet switch, IEEE Transactions on Communications, **26** (3) 328-337 (1978).
[4] F. Kamoun, L. Kleinrock, Analysis of shared finite storage in a computer node environment under general traffic conditions, IEEE Transactions on Communications, **28** (7) 992-1003 (1980).
[5] A. K. Choudhury, E. L. Hahne, Dynamic queue length thresholds for shared-memory packet switches, IEEE/ACM Transactions on Networking, **6** (2) 130-140 (1998).
[6] E. L. Hahne, A. K. Choudhury, Dynamic queue length thresholds for multiple loss priorities, IEEE/ACM Transactions on Networking, **10** (3) 368-380 (2002).
[7] R. Fan, A. Ishii, B. Mark, G. Ramamurthy, Q. Ren, An optimal buffer management scheme with dynamic thresholds, Proceedings of the GLOBECOM, 631-637 (1999).
[8] Y. S. Chu, J. P. Yang, C. S. Wu, M. C. Liang, Partial sharing and partial partitioning buffer management scheme for shared buffer packet switches, IEICE Transactions on Communications, **E85-B** (1) 79-88 (2002).
[9] J. P. Yang, Performance analysis of threshold-based selective drop mechanism for high performance packet switches, Performance Evaluation, **57** (2) 89-103 (2004).
[10] S. X. Wei, E. J. Coyle, M.-T. T. Hsiao, An optimal buffer management policy for high-performance packet switching, Proceedings of the GLOBECOM, 924-928 (1991).
[11] I. Cidon, L. Georgiadis, R. Guerin, Optimal buffer sharing, IEEE Journal on Selected Areas in Communications, **13** (7) 1229-1239 (1995).
[12] S. Sharma, Y. Viniotis, Optimal buffer management policies for shared-buffer ATM switches, IEEE/ACM Transactions on Networking, **7** (4) 575-587 (1999).
[13] H. Kroner, G. Hebuterne, P. Boyer, A. Gravey, Priority management in ATM switching nodes, IEEE Journal on Selected Areas in Communications, **9** (3) 418-427 (1991).
[14] A. K. Choudhur, E. L. Hahne, Space priority management in a shared memory ATM switch, Proceedings of the GLOBECOM, 1375-1383 (1993).
[15] C. G. Kang, H. H. Tan, Queueing analysis of explicit policy assignment push-out buffer sharing schemes for ATM networks, Proceedings of the INFOCOM, 500-509 (1994).
[16] R. Roy, S. S. Panwar, Efficient buffer sharing in shared memory ATM systems with space priority traffic, IEEE Communications Letters, **6,** (4) 162-164 (2002).
[17] V. Zaborovsky, O. Zayats, V. Mulukha, Priority queueing with finite buffer size and randomized push-out mechanism, Proceedings of the ICN, 316-320 (2010).
[18] Y. S. Lin, C. B. Shung, Quasi-pushout cell discarding, IEEE Communications Letters, **1**, (5) 146-148 (1997).
[19] M. Shreedhar and G. Varghese, Efficient fair queuing using deficit round-robin, IEEE/ACM Transactions on Networking, **4** (3) 375-385 (1996).
[20] S. Ramabhadran, J. Pasquale, The stratified round robin scheduler: design, analysis and implementation, IEEE/ACM Transactions on Networking, **14** (6), 1362-1373 (2006).
[21] S. C. Tsao, Y. D. Lin, Pre-order deficit round robin: a new scheduling algorithm for packet-switched networks, Computer Networks, **35** (2-3) 287-305 (2001).
[22] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking, **1** (4), 397-413 (1993).
[23] C. Zhiruo, W. Zheng, E. Zegura, Rainbow fair queueing: fair bandwidth sharing without each-flow state, Proceedings of the INFOCOM, 922-931 (2000).
[24] I. Stoica, S. Shenker, H. Zhang, Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks, IEEE/ACM Transactions on Networking, **11** (1) 33-46 (2003).
[25] M Shin, M Park, D Oh, B Kim, J Lee, Adaptive Logarithmic Increase Congestion Control Algorithm for Satellite Networks, KSII Transactions on Internet and Information Systems (TIIS) **8**, 2796-2813 (2014).

Jui-Pin Yang was born in Kaohsiung, Taiwan, R.O.C., in 1972. He received the Ph.D. degree in department of electrical engineering at National Chung Cheng University in 2003. From 2004 to 2008, he served as an engineer and project leader in Industrial Technology Research Institute (ITRI). He is currently an associate professor with the department of information technology and communication, Shih Chien University, Taiwan. From 2010 to 2014, he was elected as a special outstanding researcher from National Science Council, Taiwan. His research interests include computer network, information system design and development and cloud technology.