ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.3, March 2007, pp.765−774
DOI: 10.1360/jos180765

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种短流优先的公平带宽分配机制*

张鹤颖+, 蒋 杰, 窦文华

(国防科学技术大学 计算机学院,湖南 长沙 410073)

A Fair Bandwidth Allocation Mechanism with Preference to Short Flows

ZHANG He-Ying+, JIANG Jie, DOU Wen-Hua

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573678 ext 803, E-mail: hey_zhang@hotmail.com

Zhang HY, Jiang J, Dou WH. A fair bandwidth allocation mechanism with preference to short flows. *Journal of Software*, 2007,18(3):765−774. http://www.jos.org.cn/1000-9825/18/765.htm

Abstract: This paper proposes a fair bandwidth allocation mechanism FPIP (fair PIP). Dealing differently with the packets of long flows and short flows at routers, the mechanism can preferentially allocate the bandwidth of the router to short flows and allocate the remaining among the competing long flows. Furthermore, it can keep the queue length of the router at a reference value using a well-designed active queue management AQM (active queue management) algorithm. The simulation results show that this new mechanism can outperform CSFQ (core-stateless fair queueing) in terms of fairness, queue length and the response time of Web flows.
Key words: fair bandwidth allocation; queue length; response time; short flow

摘 要: 提出一种短流优先的公平带宽分配机制 FPIP (fair PIP).通过区别处理短流和长流的报文,FPIP 能够将带宽优先分配给短流,然后将剩余的带宽在长流之间公平分配.此外,FPIP 采用主动队列管理机制 AQM (active queue management)检测拥塞并控制队列长度.仿真结果表明,FPIP 在保证公平性、控制队列长度、减小 Web 流的响应时间等方面具有良好的性能.
关键词: 公平带宽分配;队列长度;响应时间;短流
中图法分类号: TP393 文献标识码: A

1 Introduction

The simple queue management and forwarding mechanism of the router will probably cause the starvation of conformant flows, even the danger of congestion collapse. Until recently, fair bandwidth allocation is mainly achieved by packet scheduling and active queue management. Most of the scheduling algorithms can achieve nearly perfect fairness[1,2]. However, they usually maintain per-flow state and perform per-flow processing, which prevents them from being widely deployed in the high-speed backbone core routers. In contrast, the queue management mechanisms with preferential dropping can achieve approximate fairness by maintaining full or partial state

* Supported by the National Natural Science Foundation of China under Grant No.60603064 (国家自然科学基金)
Received 2004-06-28; Accepted 2006-02-23

information of the flows rather than operating on per-flow queueing[3−8]. Although the proposed queue management algorithms are more scalable, they usually need to take a tradeoff between the degree of fairness and scalability.

Recently, the stateless core architecture, or SCORE for short, has been proposed to achieve approximate fairness and reasonable scalability simultaneously. The key technique used to implement the SCORE network is the dynamic packet state (DPS), which inserts the flow state information into the header of the incoming packets. In the SCORE/DPS architecture, routers are divided into edge routers and core routers. Edge routers maintain per-flow state and insert it into the header of the incoming packet[9−12]. Core routers use the simple first-in first-out (FIFO) queueing and drop the incoming packet based on the state information carried in its header when congestion occurs. Unfortunately, the existing mechanisms based on SCORE/DPS architecture have the following limitations. First, they label each packet passing through edge routers, which is not really necessary since only packets of high-bandwidth flows will be dropped probabilistically when the network becomes congested. Moreover, it is showed by recent measurement that most of the traffic is actually carried by a small number of flows, while the large remaining amount of flows is very small both in size and lifetime[4,13]. So it is reasonable to just maintain the state of these minority flows that tend to occupy more bandwidth and label their packets. Second, these mechanisms deal equally with short flows and long flows. In fact, the throughput and delay of the short-lived transmission control protocol (TCP) flows will deteriorate severely when competing with the long-lived one due to lack of sufficient packets to activate duplicate acknowledgments and the dependence on timeout to detect packet loss. Although several approaches have been proposed to deal with short flows preferentially, they cannot allocate bandwidth fairly among the competing long flows[14,15]. Third, to the best of our knowledge, none of these proposed SCORE/DPS mechanisms applies specific approach to control the queue length in routers, which corresponds to the queueing delay experienced by the backlogged packets.

In order to address these issues, we propose a new fair bandwidth sharing mechanism in this paper. In the mechanism, edge routers classify the flows into short and long based on the amount of traffic they send. The packets of long flows are labeled with their flow rate while the packets of short flows are not labeled. Core routers allocate the bandwidth of the output link to short flows as they need and allocate the remaining bandwidth fairly among the contending long flows. The new mechanism with the name of fair PIP (FPIP) uses a well-designed active queue management (AQM) algorithm, called proportional integral based series compensation and position feedback compensation (PIP), to detect congestion and control the queue length[16].

The rest of the paper is organized as follows. In Section 2, we describe FPIP in detail. In Section 3, the performance of FPIP is evaluated through extensive simulations. In Section 4, we discuss some miscellaneous issues related to the implementation of FPIP. Finally, we conclude in Section 5.

2 FPIP Framework

2.1 Overview

We present FPIP, a packet labeling and queue management mechanism that significantly simplifies the operation of edge router and core router without affecting the performance of the mechanism by taking into account the ubiquitous heavy-tailed distribution of the Internet traffic. Since the bandwidth demanded by short flows is comparatively little, proper protection of short flows will not cause persistent congestion. Instead, the bandwidth allocated to long flows should be limited for they tend to use up bandwidth more aggressively. Furthermore, in the current Internet, long flows are mainly used to transfer the bulk data, which are not very sensitive to the packet loss and delay. So, it is feasible to satisfy the bandwidth demands of short flows first and then fairly allocate the remaining bandwidth among long flows. To achieve this goal, FPIP consists of a set of edge-router and core-router

mechanisms as shown in Fig.1. The edge router classifies the flows as long and short, and then estimates the rate of long flows and labels their packets. For short flows, their traffic counters are updated. The core router uses AQM algorithm to detect congestion, estimates the fair share of the bandwidth, and drops packets with probability.

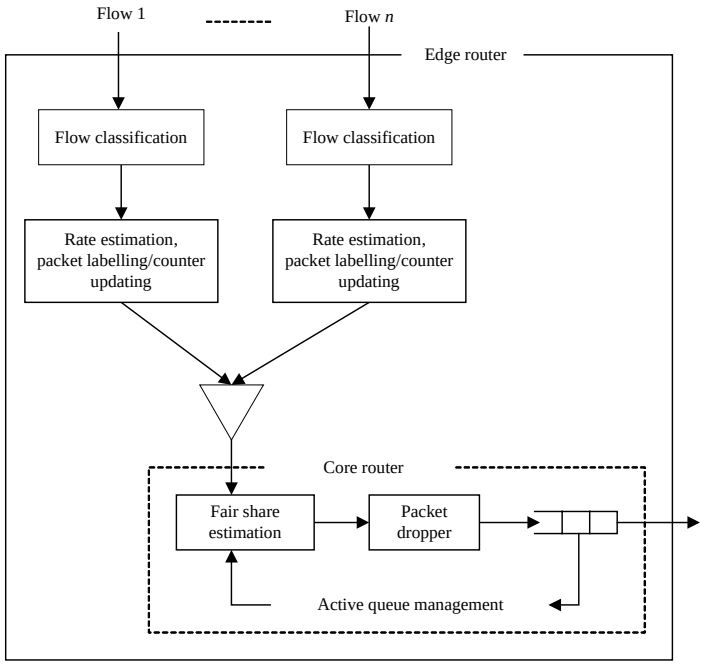


Fig.1 Overall architecture of FPIP

2.2 Router mechanisms

2.2.1 Flow classification and packet labeling

To protect short flows, we should distinguish them from long flows at first and then decrease their loss rates. Since the routers determine the packet drop probability based on its label, it is applicable to differentiate between short and long flows by their labels. In our mechanism, the edge router maintains a traffic counter for each active flow, which is used to record the number of bits sent by this flow. Once the traffic counter exceeds a certain "bit threshold", noted as *bitThresh*, the flow will be considered long. Otherwise, it is considered short. For the long flows, the edge routers estimate their arrival rates and label their packets. Instead, for the short flows, only their traffic counters increase.

To estimate the long flow arrival rate, we use the exponential averaging formula. Specifically, let Δt_i^k be the time interval between the k^{th} and the $(k-1)^{th}$ packet of flow i. The estimated rate of flow i is calculated as

$$r_i^{new} = (1 - e^{-\Delta t_i^k / K_r}) l_i^k / \Delta t_i^k + e^{-\Delta t_i^k / K_r} r_i^{old} \quad (1)$$

where l_i^k is the length of the k^{th} arrival packet of flow i and K_r is a constant.

Since the short Web flows are the main traffic in the current Internet[14], the new label approach proposed above will significantly decrease the state information maintained by edge routers and simplify the operation

2.2.2 Estimating the aggregate arrival rate of short flows

To calculate the bandwidth that can be allocated to long flows, we should estimate the aggregate arrival rate of short flows at first. For each arrival packet, the router checks its label to see which kind of flow it comes from. If the packet label equals to zero, the packet is thought of as coming from short flow. Let l be the length of the arrival

packet and Δt be the inter-arrival time of the consecutive packets that come from short flows. The router calculates the aggregate arrival rate of short flow *sRate* as follows:

$$sRate_{new} = (1 - e^{-\Delta t/K_s}) l/\Delta t + e^{-\Delta t/K_s} sRate_{old} \quad (2)$$

where K_s is a constant.

If the label of the arrival packet is greater than zero, the packet is thought of as coming from long flow. *sRate* is also updated according to (2), where *l* equals to zero. Then, we yield

$$sRate_{new} = e^{-\Delta t/K_s} sRate_{old} \quad (3)$$

Since $0 < e^{-\Delta t/K_s} \le 1$, we have $sRate_{new} \le sRate_{old}$. By doing so, *sRate* will reflect the real aggregate arrival rate of short flows even if there have been no packets from short flows for a long period of time. Now, the bandwidth that can be obtained by long flows is readily available. It is calculated as $C_l = \max\{0, C - sRate\}$.

2.2.3 Estimating the number of active long flows

In our mechanism, the routers calculate the fair share rate based on two variables: the bandwidth allocated to long flows and the number of the active long flows (*Nactivel*). The former has been determined easily. However, the latter is a lot harder to estimate. Several approaches have been proposed previously to address this issue[3,8,17]. Since these approaches are motivated by some specific goals, none of them can be copied here.

We introduce a new method to estimate the number of the active long flows, which is very straightforward. According to our method, the router is required to maintain a state table for tracking the arrival time (denoted by *prevtime*) of the packet that has lately arrived from each long flow. For each arrival packet, if its label is greater than zero, the *prevtime* of the corresponding flow in the state table is checked. If it equals to zero, the number of active long flows increases and the *prevtime* is set to the current time. Otherwise, only the *prevtime* is replaced by the current time. In order to estimate the number of the flows sharing the bandwidth during a longer period of time rather than that of the flows currently having packets in the buffer, the flow table is not updated when there is packet leaving the queue. Instead, it is updated periodically with a constant frequency, which can be viewed as a background task, for it is shifted from the high-speed data-forwarding path. When the update timer expires, items of the table are checked one by one. If the interval between the current time and the *prevtime* of a flow is greater than a certain threshold (*Tn*), which means there is no packet from that flow in the last *Tn* time units, the flow is considered terminated. Thus, *Nactivel* is reduced and the *prevtime* of the flow is reset to zero.

2.2.4 Estimating and adjusting the fair share rate

The problem of the fair bandwidth sharing occurs along with the presence of the network congestion, and the estimation of the fair share rate depends further on it. Therefore, it is of great importance to correctly detect the congestion state of the network. In this paper, we apply AQM scheme in congestion detection for the following reasons: (1) The packet drop probability calculated by AQM is a good representation of the degree of congestion; (2) AQM scheme is able to detect congestion and control queue length simultaneously. In our mechanism, we use a robust AQM scheme designed previously, which is called PIP[16]. In PIP, the packet drop probability is calculated as

$$p(k) = p(k-1) + \frac{\Delta t}{T}(q(k) - q_0) + \left(\frac{\tau}{T} - K_h\right)[q(k) - q(k-1)], k \ge 1.$$

where, Δt is the packet interarrival time, q_0 is the reference queue length, $q(k)$ is the current queue length. τ, T and K_h are parameters of PIP, which are determined by the stability conditions of the TCP/PIP system.

Here, we don't drop the packet based on $p(k)$ directly. Instead, it is regarded as a measure of congestion. When $p(k)$ is greater than a random variable, it is likely that the link is congested and the fair share rate (R_{fair}) should be calculated. Let the capacity of the output link be C. Suppose *sRate* is less than C. R_{fair} is calculated as

$$R_{fair}=\frac{C-sRate}{Nactivel},\ Nactivel>0 \tag{4}$$

Next, we will show that the estimated value of R_{fair} is approximate to the accurate value.

When the aggregate packet arrival rate is greater than the link capacity, the accurate fair share rate is the unique solution of the following equation

$$C=\sum_{i=1}^{n}\min(r_i(t),R_{fair}(t)) \tag{5}$$

where n is the number of active flows sharing the same link. Without loss of generality, suppose $r_1(t)\le r_2(t)\le r_3(t)\le ...\le r_n(t)$. There necessarily exists $k\in\{1,2,...,n\}$ and $k<n$, which makes

$$C=\sum_{i=1}^{k}r_i(t)+(n-k)R_{fair} \tag{6}$$

From Eq.(6), we have

$$R_{fair}=\frac{C-\sum_{i=1}^{k}r_i(t)}{n-k} \tag{7}$$

Let R_{slow} be the aggregate rate of slow flows and *Nactivef* be the number of active fast flows, then we have

$$\begin{cases}R_{slow}=\sum_{i=1}^{k}r_i(t)\\ Nactivef=n-k\end{cases}.$$

Eq.(7) can be written as

$$R_{fair}=\frac{C-R_{slow}}{Nactivef} \tag{8}$$

Comparing Eq.(4) with Eq.(8), we can find that they are similar to each other in the form. The difference between them lies in the implications of the variables. In Eq.(8), R_{slow} is the aggregate arrival rate of the slow flows and *Nactivef* is the number of the fast flows, while in Eq.(4) *sRate* is the aggregate arrival rate of short flows and *Nactivel* is the number of long flows. Although short flows are not necessarily slow flows, they can be treated equally for both of them tend to demand less bandwidth than long flows and fast flows. Thus, we conclude that the fair share rate calculated by Eq.(4) is a good approximation to the accurate value.

When $p(k)$ is less than a random variable, the link is considered uncongested. To avoid under-utilization of the link, when the estimated rate of the accepted traffic (*cRate*) is less than the output link capacity C, the fair share rate is adjusted as follows

$$R_{fair}^{new}=\frac{C}{cRate}R_{fair}^{old} \tag{9}$$

The accepted rate is also estimated by exponential averaging:

$$cRate_{new}=(1-e^{-\Delta t/K_c})l/\Delta t+e^{-\Delta t/K_c}cRate_{old} \tag{10}$$

The implications of the parameters in Eq.(10) are similar to those in Eq.(2). The adjusting method ensures high utilization of the link even if the estimation of *sRate* or *Nactivel* is not very accurate. In addition, when the link is not congested, the fair share rate will increase according to Eq.(9).

After R_{fair} is estimated, the incoming packet of the long flow will be dropped with a probability that is calculated as

$$p=\frac{r_i-R_{fair}}{r_i} \tag{11}$$

where r_i is the rate of flow i carried by the packet label.

3 Simulations

We use NS simulator to evaluate the performance of FPIP under a variety of conditions, and compare it with Core-Stateless Fair Queueing (CSFQ) and Random Early Detection (RED) respectively[18]. Since CSFQ is one of the most well-known mechanisms for the fair bandwidth sharing based on SCORE/DPS architecture, we use similar configurations of simulation used by CSFQ to verify the performance of FPIP.

3.1 Single congested link

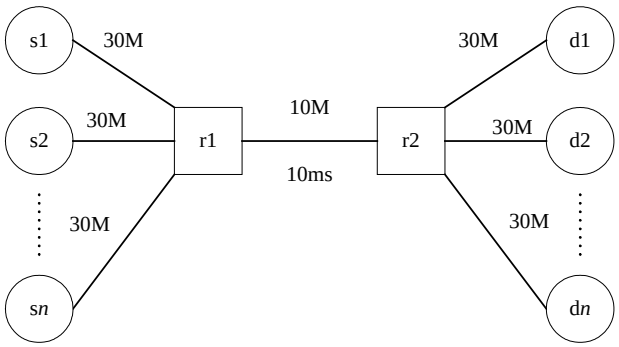


Fig.2 Simple network topology

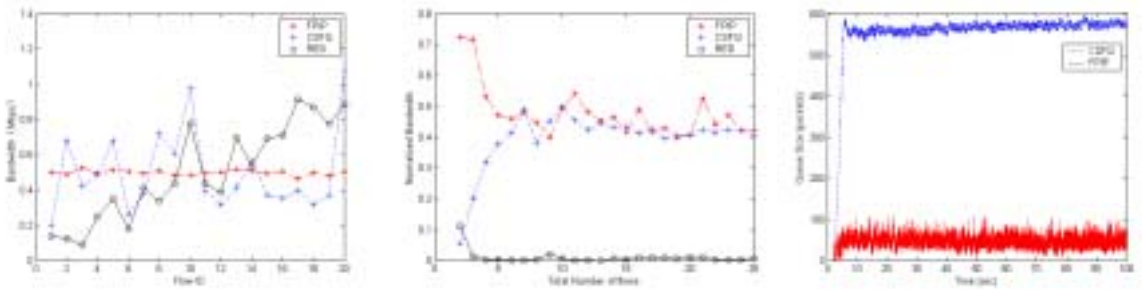
We start with the single congested link topology shown in Fig.2, where the only bottleneck link on the route of a connection is between the routers r1 and r2. The buffer sizes in both r1 and r2 are 300 Kbytes. The capacity of the bottleneck link is 10 Mbps and the propagation delay is 10 ms. The bandwidths of all the other links are 30 Mbps with the propagation delay of 5 ms. r1 is configured as edge router and r2 is core router. The packet size of TCP flow is 1 Kbytes and that of UDP flow is 500 bytes. In FPIP, the *bitThresh* is set to 10 Kbytes, T=200, τ=0.4, K_h=0.0014, q_0 is 50 packets. The averaging constants K_r, K_s and K_c are all set to 100 ms. Unless otherwise specified, the buffer threshold in CSFQ is set to 50 Kbytes. Other parameters of CSFQ are selected as the default values in NS. In RED, the minimum threshold is set to 25 Kbytes, while the maximum threshold is 75 Kbytes.

At first, we evaluate the fairness property when all the flows are UDP flow. It is well known that UDP flows do not reduce their sending rates when congestion occurs. So one of the important goals for the fair bandwidth sharing mechanisms is to restrict the rate of greedy UDP flows to the fair share rate. Here we consider 20 CBR flows sharing a single bottleneck link. Each flow sends at $10\times i/20$ Mbps, where $i\in\{1,2,...,20\}$ is the flow number. Thus flow 1 sends at 0.5 Mbps, flow 2 sends at 1 Mbps, and so on. Fig.3(a) shows the average throughput of each flow over a 10 second interval. FPIP achieves the highest degree of fairness; while RED fails to ensure fairness with each flow getting a share proportional to its incoming rate. CSFQ achieves a less precise degree of fairness, for the deviations between the throughputs of most flows and the ideal value are very large. Moreover, for CSFQ the estimated fair share rate is about two times more than the ideal value 0.5 Mbps, while for FPIP the estimated value matches the ideal value pretty well.

Next, we will study the performance of a TCP flow sharing the same bottleneck link with (N−1) unresponsive UDP flows. All the UDP flows send at twice their fair share rate. We compare the bandwidth achieved by each flow through the normalized bandwidth, which is defined as the ratio of the allocated bandwidth to the ideal bandwidth. When the total number of flows changes from 2 to 25, the normalized bandwidths of the TCP flow are illustrated in Fig.3(b). FPIP performs better than CSFQ when there are less than six flows. With the increase of the traffic flow, the TCP flow can achieve about 45% of the ideal bandwidth under CSFQ and FPIP. Again, RED performs worst.

We also evaluate the response time of the short Web flows. In the experiment, we use a mixture of 30 Web flows, 30 FTP (file transfer protocol) flows and 15 CBR (constant bit rate) flows. Sources (1,2,…,30) are Web servers and sinks (1,2,…,30) are clients. The request interval of the clients follows the exponential distribution and the average value is 5 seconds. The page size is 52 Kbytes. The sending rate of CBR flow is 2 Mbps. The response

time is shorter under FPIP (0.085s) than that under CSFQ (0.499s). For RED, we cannot even measure the response time of Web flows for they cannot get any bandwidth when competing with the high-speed unresponsive UDP flows. The big difference between the response time of the Web flows under FPIP and CSFQ is partially caused by their queue length, as illustrated in Fig.3(c). The queue length under CSFQ is much higher than that under FPIP, which leads to larger queueing delay. Even though the queue length has a great impact on the response time of Web flows, it is not the only factor. We should also protect short Web flows from high-speed unresponsive flows and ensure reasonable bandwidth to them. As in RED, even if the queue length is low (between 25 Kbytes and 75 Kbytes in our simulations), the response time is infinite due to lack of bandwidth guarantees for Web flows.



(a) The average throughput of each flow (b) The normalized bandwidth of a TCP flow that competes with (N–1) UDP flows (c) The queue dynamics under mixed traffic

Fig.3

3.2 Multiple congested links

Next, we will consider a traffic flow traversing multiple congested links, as shown in Fig.4.

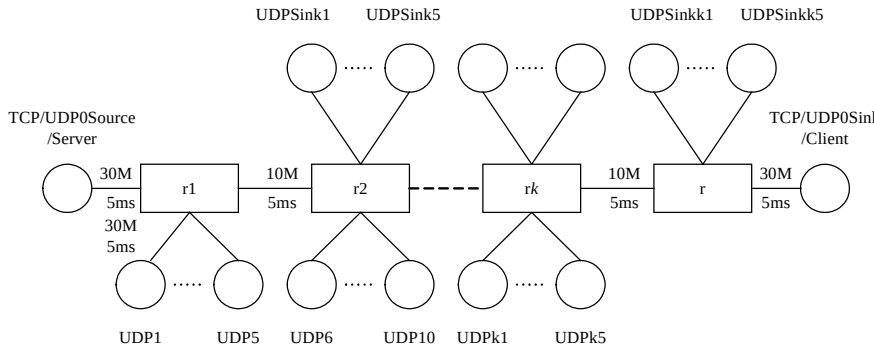


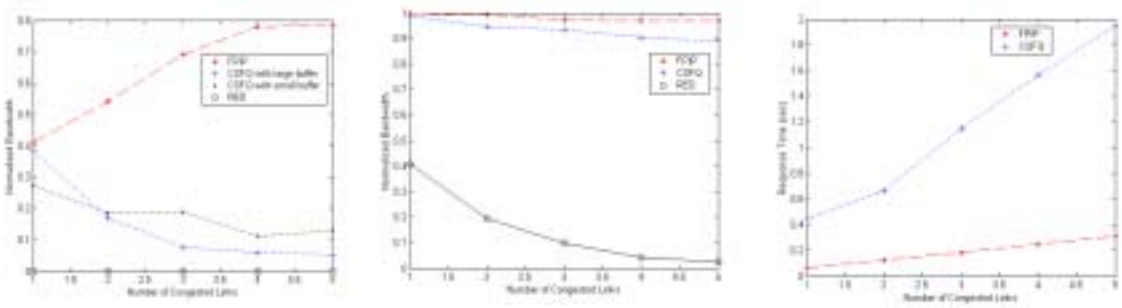
Fig.4 Topology with multiple congested links

In our simulations, the number of congested links varies from one to five. The capacities of all the access links are 30 Mbps, and those of the congested links are 10 Mbps. The propagation delay of each link is 5ms. Each router is connected with five UDP flows which terminate at the next router and send at 4 Mbps. Thus, all the links between neighboring routers are congested.

When a TCP flow traverses all the congested links, Fig.5(a) shows the bandwidths achieved by it as a function of the number of congested links. In RED, the TCP flow is submerged by the high-speed unresponsive flows. The TCP flow achieves more and more bandwidth under FPIP when the number of the congested links is less than four, while it achieves less bandwidth under CSFQ. This can be explained as follows. For a TCP flow, the round trip time increases gradually with the increase of the congested links, which leads to the decrease of its throughput and the

packet drop rate. If the packet drop rate decreases faster than the increase of the round trip time, the throughput of the TCP flow may increase just as observed in FPIP. On the other hand, if the round trip time increases faster than the decrease of the packet drop rate, the throughput of the TCP flow may decrease as observed in CSFQ. To further verify our analysis, we set the buffer size and the buffer threshold for CSFQ to 30Kbytes and 25Kbytes respectively. With the decrease of the queueing delay, the bandwidth achieved by the TCP flow will increase. The simulation result is also shown in Fig.5(a). The bandwidth achieved by the TCP flow increases just as expected.

When the TCP flow is replaced by a UDP flow (denoted as UDP0 in Fig.4) sending at its fair share rate 1.67Mbps, Fig.5(b) shows the normalized bandwidth achieved by UDP0. Similarly, RED has the worst performance. FPIP performs slightly better than CSFQ. In the last experiment, a Web flow traverses all the congested links. Fig.5(c) shows the response time of the Web flow traversing different numbers of the congested links. We cannot show the response time of the Web flow under RED, for the client of the Web flow cannot even receive a single response from the server. For the other two mechanisms, when the number of the congested links increases, the response time of the Web flow also increases. Moreover, the increase under CSFQ is faster than that under FPIP. For the space limitation, the queue dynamics are not shown. They are similar to those shown in Fig.3(c).



(a) The normalized bandwidth achieved by a TCP flow (b) The normalized bandwidth of a UDP flow (c) The response time of Web flow

Fig.5

4 Discussion

4.1 Choosing *bitThresh*

The value of *bitThresh* determines the boundary between short flow and long flow. A larger value for *bitThresh* has many advantages, such as more flows being classified as short flows, more bandwidth being allocated to short flows, less state requirement, lower computing overhead involved in the estimation of the flow arrival rate at the edge router, the calculation of the packet drop probability, and rewritten of the packet label at the core router. On the contrary, a smaller value for *bitThresh* leads to the opposite results.

Several possible approaches can be used to set *bitThresh*. First, we can set *bitThresh* to the size of three packets, which ensures that each TCP flow can send at least three packets so that they are guarded against timeout. This is a conservative setting. Second, it is appropriate to choose the value of *bitThresh* based on the average size of the short TCP flows measured in the real networks. Typically a short TCP flow has less than 20 packets to transmit[15]. In our simulations, we empirically choose *bitThresh* to be 10 Kbytes, which indicates 10 packets with the size of 1 Kbytes. This setting is more aggressive. Instead of a fixed, predetermined value for *bitThresh*, another possibility is to vary *bitThresh* dynamically according to the measure of congestion in a network. If the network is severely congested, *bitThresh* should be small, which indicates less bandwidth can be allocated to short flows. On

the contrary, if the network is not congested, *bitThresh* should be large for enough bandwidth can be allocated to short flows.

4.2 Choosing T_n, the longest idle time of long flows

T_n is the longest idle time of long flows, which indicates that a long flow is considered terminated if no packet from it is observed in the last T_n time units. T_n can be naturally chosen as the maximum RTT of long TCP flows traversing the link. Due to the high overhead involved in the measure of RTT, T_n can be set empirically. A smaller value for T_n can lead to some disadvantages, such as the fluctuation of the estimated number of long flows, the unnecessary updating of the long flow state table, and larger estimated fair share rate. On the contrary, if T_n is too large, the stopping of a long flow may not be detected immediately and the estimated number of the active long flows will be larger than the real value. According to Eq.(4), the estimated fair share rate will be smaller than the accurate value, which can then lead to the potential under-utilization of the output-link capacity. Fortunately, this issue has been addressed in our mechanism by adjusting the fair share rate when the estimated aggregate rate of the accepted traffic is less than the output-link capacity. Therefore, we argue that the performance of FPIP is not very sensitive to T_n as long as it is large enough. In our simulations, we set T_n to a conservative value that is about two magnitudes of order larger than the commonly observed RTT in the real networks, for example 5 seconds.

4.3 State requirements

In addition to the aggregate state needed by regular SCORE/DPS mechanisms[9–11], FPIP needs to maintain the state of long flows at the core routers to estimate the number of the active long flows, which is at variance with the original spirit of the stateless core proposal. Even so, it is not likely to result in severe scalability problems for the following reasons. First, only the state of packets from active long flows is maintained. The amount of memory required will be considerably small considering the ubiquitous heavy-tailed distribution of the Internet traffic in terms of the flow size[4]. Second, the core routers only need to maintain the arrival time of the lately arrived packet from each long flow. Thus, the item of the state table contains only one column of content, which greatly simplifies the construction of the table and the processing done on it.

5 Conclusions

In this paper, we present a new fair bandwidth allocation mechanism based on the Internet traffic characteristics. By labeling only the packets of long flows at edges, the mechanism greatly reduces the amount of the flow state required and the processing done on it. To provide low delay service, the core routers apply AQM algorithm to detect congestion and control queue length, which is beneficial to the adaptive flows and delay-sensitive applications. The results of the extended simulations show that FPIP performs well in many aspects.

References:

[1] Alan D, Srinivasan K, Scott S. Analysis and simulation of a fair queueing algorithm. In: Proc. of the ACM SIGCOMM'89. Austin: ACM Press, 1989. 1–12.

[2] Shreedhar M, Varghese G. Efficient fair queueing using deficit round robin. In: Proc. of the ACM SIGCOMM'95. Boston: ACM Press, 1995. 231–243.

[3] Dong L, Robert M, Dynamics of random early detection, In: Proc. of ACM SIGCOMM'97, Cannes: ACM Press. 1997, 127–137.

[4] Ratul M, Sally F. Controlling high bandwidth flows at the congested router. Technical Report, TR-01-001, AT&T Center for Internet Research at ICSI (ACIRI), 2001.

[5] Wu CF, Kang GS, Dilip DK, Debanjan S. The BLUE active queue management algorithms. IEEE/ACM Trans. on Networking, 2002,10(4):513–528.

[6] Rong P, Balajii P, Konstantinos P, CHOKe-A stateless active queue management scheme for approximating fair bandwidth allocation. In: Proc. of the IEEE INFOCOM 2000. Tel-Aviv: IEEE Press, 2000. 942−951.

[7] Rong P, Lee B, Balaji P, Scott S. Approximate fairness through differential dropping. ACM Computer Communication Review, 2003,33(2):23−39.

[8] Jung-Shian L, Ming-Shiann L. Network fair bandwidth share using hash rate estimation. Networks, 2002,40(3):125−141.

[9] Ion S, Scott S, Hui Z. Core-Stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In: Proc. of the ACM SIGCOMM'98. Vancouver: ACM Press, 1998,118−130.

[10] Cao ZR, Zheng W, Ellen Z. Rainbow fair queueing: Fair bandwidth sharing without per-flow state. In: Proc. of the IEEE INFOCOM 2000. Tel-Aviv: IEEE Press, 2000. 922−931.

[11] Antoine C, Walid D. Tag-Based fair bandwidth sharing for responsive and unresponsive flows. No.3846, INRIA Sophia Antipolis, 1999.

[12] Hoon-Tong N, Chen-Khong T. A control-theoretical approach for achieving fair bandwidth allocations in core-stateless networks. Computer Networks, 2002,40(2):727−741.

[13] Nevil B, Claffy KC. Understanding Internet traffic streams: dragonflies and tortoises. IEEE Communications Magazine, 2002, 40(10):110−117.

[14] Zhang Y, Qiu L, Srinivasan K. Speeding up short data transfers: Theory, architecture support, and simulation results. In: Proc. of the 10th Int'l Workshop on Network and Operating System Support for Digital Audio and Video. Chapel Hill, 2000.

[15] Liang G, Ibrahim M. The war between mice and elephants. BU-CS-2001-005, Boston: Boston University, 2001.

[16] Zhang HY, Liu BH, Dou WH. Design of a robust active queue management algorithm based on feedback compensation. In: Proc. of the ACM SIGCOMM 2003. Karlsruhe: ACM Press, 2003. 277−286.

[17] Teunis JO, Lakshman TV, Larry W. SRED: Stabilized RED. In: Proc. of the IEEE INFOCOM'99. New York: IEEE Press, 1999, 1346−1355.

[18] Sally F, Van J. Random early detection gateways for congestion avoidance. IEEE/ACM Trans. on Networking, 1993,4(1):397−413.

ZHANG He-Ying was born in 1976. She is a lecturer at the School of Computer, National University of Defense Technology. Her research areas are network congestion control, QoS and wireless network.

JIANG Jie was born in 1976. He is a lecturer at the School of Computer, National University of Defense Technology. His research areas are network security and sensor network.

DOU Wen-Hua was born in 1946. He is a professor at the National University of Defense Technology and a CCF senior member. His research areas are network congestion control, network management and wireless network.