CHOKeR: A Novel AQM Algorithm With Proportional Bandwidth Allocation and TCP Protection

Lingyun Lu, Haifeng Du, and Ren Ping Liu, *Member, IEEE*

***Abstract*—Although differentiated services (DiffServ) networks have been well discussed in the past several years, a conventional Active Queue Management (AQM) algorithm still cannot provide low-complexity and cost-effective differentiated bandwidth allocation in DiffServ. In this paper, a novel AQM scheme called CHOKeR is designed to protect TCP flows effectively. We adopt a method from CHOKeW to draw multiple packets randomly from the output buffer. CHOKeR enhances the drawing factor by using a multistep increase and single-step decrease (MISD) mechanism. In order to explain the features of CHOKeR, an analytical model is used, followed by extensive simulations to evaluate the performance of CHOKeR. The analytical model and simulation results demonstrate that CHOKeR achieves proportional bandwidth allocation between different priority levels, fairness guarantee among equal priority flows, and protection of TCP against high-speed unresponsive flows when network congestion occurs.**

***Index Terms*—Active Queue Management (AQM), CHOKe, differentiated services (DiffServ), fairness, quality of service (QoS).**

I. Introduction

QUALITY of service (QoS)-related problems in the Internet have been studied for a number of years, but have not been solved completely. One of the challenges is to provide a reliable and cost-effective method to support multiple services at different priority levels within a core network that can support thousands of flows.

DiffServ can provide a variety of services for IP packets based on their per-hop behaviors (PHBs) [1]. It categorizes routers into edge routers and core routers. Sophisticated operations, such as per-flow classification and marking, are implemented at edge routers. Core routers only need to forward the packets according to PHB values in the packets. Packets with high priority based on their PHB values have relatively low drop probability. Because of its simplicity and scalability, DiffServ has caught the most attention nowadays in IP backbone networks. In core networks, many routers often experience congestion. It is important to guarantee QoS at all time. In differentiated services (DiffServ) architecture, Active Queue Management(AQM) algorithms are among the most effective network control mechanisms and can achieve a satisfactory tradeoff between drop probability and throughput [2], [3]. Different AQM parameters lead to different QoS performance. Moreover, the implementation of an AQM in DiffServ must provide bandwidth differentiation and TCP protection simultaneously.

Manuscript received April 07, 2013; revised June 20, 2013; accepted August 07, 2013. Date of publication August 15, 2013; date of current version December 12, 2014. This work was supported by the National Natural Science Foundation of China under Grant 61201200. Paper no. TII-13-0186.

L. Lu is with the School of Computer and Information Technology, Beijing JiaoTong University, Beijing 100044, China (e-mail: lylu@bjtu.edu.cn).

H. Du is with the School of Software Engineering, Beijing JiaoTong University, Beijing 100044, China.

R. P. Liu is with CSIRO, Sydney, NSW 1710, Australia (e-mail: ren.liu@csiro.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2013.2278618

A. Related Work on AQM

The importance of TCP protection has been discussed by Floyd and Fall [4]. The problem of TCP Protection originates from TCP flows competing with unresponsive UDP flows in order to occupy scarce bandwidth [5]. After the TCP flows reduce sending rates, the unresponsive UDP flows can seize the available bandwidth and cause starvation of TCP flows. This results in unfairness to large volume TCP flows. Conventional AQM algorithms such as Random Early Detection (RED) [6] and BLUE [7] cannot protect TCP flows. Per-flow schemes such as RED with preferential dropping (RED-PD) can punish non-TCP-friendly flows, but it requires reserved parameters for each flow, which significantly increases the memory requirement [8]. CHOKe proposed by Pan *et al.* is simple and does not require per-flow state maintenance [9]. However, CHOKe only serves as an enhancement filter for RED in which a buffered packet is drawn at random and compared with an arriving packet. If both packets come from the same flow, they are dropped as a pair (matched drops); otherwise, the arriving packet is delivered to RED. The validity of CHOKe has been explained using an analytical model by Tang *et al.* [10].

Bandwidth differentiation is an important feature in DiffServ. RED with in/out bit (RIO) [11], is one of the most popular schemes designed for bandwidth differentiation. In RIO, an "out" flow may be starved because there is no mechanism to guarantee the bandwidth share for low-priority traffic. Some scheduling schemes, such as weighted fair queueing (WFQ) may also support differentiated bandwidth allocation [12], [13]. The main disadvantage of scheduling schemes, such as WFQ, is that they require constant per-flow state maintenance, which is not scalable in core networks.

CHOKeW, proposed by Wen *et al.*, can provide bandwidth differentiation among flows at multiple priority levels [5], [14]. CHOKeW borrows the idea of matched drops from CHOKe for TCP protection. While CHOKe draws a single packet, CHOKeW draws more than one packet to compare. In CHOKeW, the adjustable number of draws is not only used

for restricting the bandwidth share of high-speed unresponsive flows but also used as signals to inform TCP of the congestion status. CHOKeW is capable of providing higher bandwidth share to flows with higher priority, maintaining good fairness among flows in the same priority, and protecting TCP against high-speed unresponsive flows when network congestion occurs.

However, there are three problems that CHOKeW cannot solved. First, the bandwidth differentiation at multiple priority levels becomes smaller after the number of flows increases. In particular, as the scale of the Internet becomes larger, all types of flows push into the Internet. In such complex environment, the drop probability of CHOKeW is close to CHOKe. The advantage of differentiation will be difficult to standout. Second, CHOKeW shows poor performance if the traffic is bursty. From [5], when the drawing factor increases quickly, the fairness diminishes. And CHOKeW cannot provide the assured bandwidth allocation for different priority flow. Third, when network congestion becomes worse, the bandwidth allocation in CHOKeW cannot cope with nonresponsive flows.

B. Our Contributions

We adopt the idea of CHOKeW that draws multiple packets from the output buffer and compares them with the arriving packet to design a novel AQM algorithm, CHOKeR, where R stands for the ratio of bandwidth allocation. We improve the algorithm by designing a multistep increase and single-step decrease (MISD) drawing factor. The performance of CHOKeR is evaluated with both analytical model and extensive simulations. Our results show that CHOKeR provides stable bandwidth allocation and maintains differentiated bandwidth ratios for different priority levels. Additionally, CHOKeR maintains fairness among flows of the same priority.

The remainder of this paper is organized as follows. Section II describes our algorithm, CHOKeR. Section III derives the equations to analyze the features and effectiveness of CHOKeR, such as proportional bandwith allocation and fairness. Section IV presents and discusses the simulation results, including the effect of supporting three and four priority levels, TCP protection, and fairness. We conclude our paper in Section V.

II. CHOKeR Algorithm

CHOKeR uses the strategy of matched drops to protect TCP flows without the necessity to maintain per-flow state. As such, CHOKeR is capable of working in core networks where a myriad of flows are served.

A. MISD Drawing Factor

CHOKeR adopts and enhances the idea of drawing factor from CHOKeW to adjust the number of draws that can be used as signals to inform TCP of the congestion status. The drawing factor was originally defined in CHOKeW [5] to determine how many packets should be drawn from the buffer. We enhance the update process of the drawing factor in CHOKeR by designing a MISD mechanism. MISD dictates that the drawing factor takes a multistep increase when the length of the queue becomes larger and a single-step decrease when the length becomes smaller. Such MISD adjustment of the drawing factor can avoid overflow of the queue when larger number TCP connections go through the router.

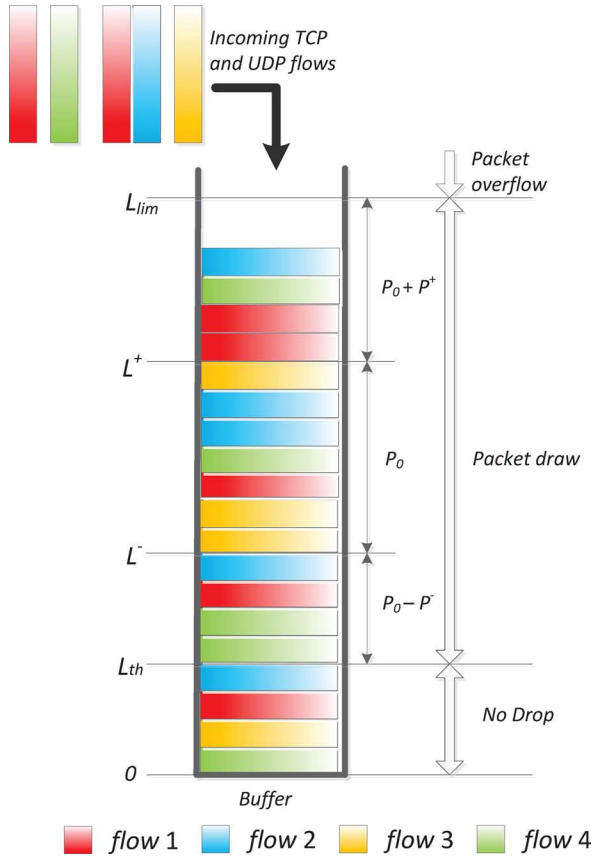


Fig. 1. Update process of the drawing factor p_0 in a buffer with mixed TCP and UDP flows.

In CHOKeR, we use p_0 to denote the drawing factor, which is the number of random draws from the queue upon a packet arrival. The precise meaning of drawing factor p_0 depends upon its value. The value of p_0 is usually beyond 1 and is divided as an integral part and a fractional part described as follows:

$$p_0 = m + f \tag{1}$$

where m is the integer part and f is the fractional part of p_0. In this case, the probability of drawing m packets from the queue for comparison is $1 - f$, and the probability of drawing $m + 1$ is f.

Algorithm Description

In practical networks, the congestion status of a router may become heavier or lighter after a period of time. In order to cooperate with TCP and to improve the system performance, CHOKeR needs to inform TCP senders to lower their sending speeds by dropping adequate number of packets when the network congestion becomes worse. CHOKeR can adaptively update p_0 based on the congestion status. Fig. 1 illustrates the update process of the drawing factor p_0 in a buffer with mixed TCP and UDP flows. Different colors represent different flows of packets.

When a packet, denoted as $pkt(k)$ of priority k, arrives, the router checks the buffer occupancy L_k and compare with a set of predefined thresholds. If L_k is less than the minimum threshold $L_{\rm th}$, there will be no packet drop by resetting the drawing factor $p_0 = 0$; When L_k is larger than $L_{\rm th}$, but less than the lower

threshold L^-, the drawing factor is decreased by a single step p^- with

$$p_0 = |p_0 - p^-|^+ \tag{2}$$

where $|x|^+ = \max(0, x)$. When L_k is between the lower and upper thresholds, L^- and L^+, p_0 is kept unchanged. When L_k is larger than the upper threshold, the router will take a more aggressive action to increase the drawing factor by multiple steps with

$$p_0 = p_0 + a \cdot p^+ \tag{3}$$

where the number of steps is calculated as

$$a = \left\lceil \frac{L_k - L^+}{L^+ - L^-} \right\rceil. \tag{4}$$

When the buffer occupancy L_k is over the buffer limit $L_{\lim}$, the arrival packet will be dropped with probability of 1. l_a denotes the number of packet arrivals, and l_b denotes the number of packets drawn from the queue. The details of the computation of the drawing factor p_0 and the packet draw actions are shown in Algorithm 1.

Algorithm 1 CHOKeR

1: {**Initialization**}
2: $p_0 \leftarrow 0$
3: {**Update drawing factor** p_0}
4: **for** each incoming packet $pkt(k)$ with priority k **do**
5: **if** $L_k < L_{th}$ **then**
6: $p_0 \leftarrow 0$
7: $L_k \leftarrow L_k + l_a$
8: **else if** $L_{th} < L_k < L^-$ **then**
9: $p_0 \leftarrow |p_0 - p^-|^+$
10: **else if** $L^- < L_k < L^+$ **then**
11: $p_0 \leftarrow p_0$
12: **else if** $L^+ < L_k < L_{lim}$ **then**
13: $a \leftarrow \lceil (L_k - L^+)/(L^+ - L^-) \rceil$
14: $p_0 \leftarrow p_0 + a * p^+$
15: **else if** $L_k > L_{lim}$ **then**
16: drop $pkt(k)$
17: **end if**
18: $m \leftarrow \lfloor p_0 \rfloor$
19: $f \leftarrow p_0 - m$
20: $v \leftarrow rand()$ {generate random number $v \in [0, 1)$}
21: **if** $v < f$ **then**
22: $m \leftarrow m + 1$
23: **end if**
24: {**Packet draw actions**}
25: **while** $m > 0$ **do**
26: $m \leftarrow m - 1$
27: **if** $\xi_a = \xi_b$ **then**
28: {Draw packets with matching flow ID}
29: $L_k \leftarrow L_k - l_a - l_b$
30: break
31: **else if** $L_k/L_{lim} > R(k)$ **and** priority of drawn packet pkt' is k **then**
32: {Draw packets with matching priority}
33: $L_k \leftarrow L_k - l_b$
34: **end if**
35: **end while**
36: **end for**

There are two match draw actions in Algorithm 1. The first match drop action is to match flow ID. The router selects packets with flow ID ξ_b matching the arrival packet's flow ID ξ_a, and drop them with the drawing factor p_0. The second match drop action is to match priority of the packets. When the number of packets with priority k in the queue exceeds a threshold $R(k)$, the excess packets are drawn from the queue and dropped. Such priority match drop action provides proportional bandwidth allocation for different priority levels. The bandwidth allocation for priority k is $R(k)$, and we have

$$\sum_{k=1}^{M} R(k) = 1. \tag{5}$$

From Algorithm 1, we can see that the drawing factor increases faster with multiple steps if the congestion becomes heavier and it decreases with a single step when congestion becomes lighter. This MISD mechanism of CHOKeR is able to response to bursty traffic effectively, thus avoid buffer overflow. The bursty traffic may be generated in core networks by a large number of TCP flows when they begin sending traffic simultaneously.

B. Complexity

In terms of complexity, CHOKeR is similar to CHOKeW. CHOKeR needs to remember only $R(k)$ for each predefined priority level k, ($k = 1, 2, \ldots, M$), instead of per flow information. Thus, the complexity of CHOKeR is proportional to the number of priority levels M, rather than the number of flows N. That is, the complexity of CHOKeR is $\mathcal{O}(M)$, while for conventional per-flow schemes, e.g., [19], the complexity is $\mathcal{O}(N)$. In DiffeServ networks, it is expected that M is much smaller than N, i.e., $M \ll N$.

III. Performance Evaluations

We now evaluate performance of CHOKeR using a leaky buffer model, which has been proposed and validated in [10], [15]. In particular, we analyze the fairness and proportional bandwidth allocation of CHOKeR.

In previous work [10], [15], a leaky buffer model has been proposed to explain the effectiveness of CHOKe. In a leaky buffer model, a packet can be dropped while it moves toward the head of the queue. In CHOKeR, a packet is dropped according to the matched drops and the ratio of the allocated bandwidth for the priority after the packet entered the queue. According to [10], [18], the goodput of a flow is determined by the source node sending rate, the probability of matched drops, and the ratio of the allocated bandwidth for the priority. We adopt the leaky buffer model for the performance analysis of CHOKeR.

A. Fairness

The goal of DiffServ is to provide differentiated service for flows with different priority levels. So we first consider the fairness under the circumstances where flows have the same priority, similar to [5]. We study two TCP flows, i and j with the priority k. Let $r_{k,i}$ be the probability that matched drops occur at one draw, which is dependent on the current length of queue

L_k, and the number of packets from flow i in the queue [16] as follows:

$$r_{k,i} = \frac{L_{k,i}}{L_k} \tag{6}$$

where $L_{k,i}$ is the number of packets from flow i with priority k in the queue. Thus, the probability that a packet from flow i is dropped upon its arrival is

$$p_{k,i} = 1 - (1 - r_{k,i})^{p_0}. \tag{7}$$

For each packet arrival from flow i, the probability that it is dropped before entering the queue is $p_{k,i}$. According to the rule of flow ID matched drops, a packet from the same flow, which is already in the buffer, should also be dropped if the arriving packet is dropped. Also, there is another dropped probability ϕ_k, which is defined as the ratio of the number of packets with priority k in the queue and the total number in the queue beyond the predefined value $R(k)$. Thus, in the steady state, we have

$$\lambda'_{k,i} = \lambda_{k,i}(1 - (2p_{k,i} + \phi_k)) \tag{8}$$

where $\lambda_{k,i}$ is the arrival rate of flow i with priority k, and $\lambda'_{k,i}$ is the rate that flow i leave the queue. When CHOKeR router is in steady state, from (7), we have

$$\frac{r_{k,i}}{r_{k,j}} = \frac{\lambda_{k,i}(1 - r_{k,i})^{p_0}}{\lambda_{k,j}(1 - r_{k,j})^{p_0}}. \tag{9}$$

Previous works [5] and [14] have shown that the approximate sending rate of TCP can be described as

$$\lambda_{k,i} = \alpha_{k,i} R(p_{k,i}) \tag{10}$$

where $\alpha_{k,i} > 0$ denotes the combination of packet size, round-trip time, and other parameters such as the TCP version and the speed of user's computers. The sending rate of TCP decreases as networking congestion deteriorate [19], [20], so we have

$$\frac{\partial R}{\partial p_{k,i}} < 0. \tag{11}$$

Based on RED [6] and BLUE [7], we have

$$\left(\frac{\lambda'_{k,i}}{\lambda'_{k,j}}\right)_{RaB} = \frac{\alpha_{k,i}}{\alpha_{k,j}}. \tag{12}$$

If $\alpha_{k,i} = \alpha_{k,j}$, according to (6), (7), (8), and (10), we have $\lambda_{k,i} = \lambda_{k,j}$ and $\lambda'_{k,i} = \lambda'_{k,j}$. If $\alpha_{k,i} > \alpha_{k,j}$, then

$$\left(\frac{\lambda'_{k,i}}{\lambda'_{k,j}}\right)_{RaB} > 1.$$

If $\alpha_{k,i} > \alpha_{k,j}$, $\lambda'_{k,i}/\lambda'_{k,j}$ is close to 1, and we can obtain better fairness. Considering CHOKeR algorithm, the ratio of average throughput from i to j is less than $\left(\lambda'_{k,i}/\lambda'_{k,j}\right)_{RaB}$. Thus, we draw a conclusion that CHOKeR is capable of providing better fairness than RED [4] and BLUE [7].

B. Proportional Bandwidth Allocation

In the CHOKeR router, let r_k be the probability that match priority k occurs at one draw, which is dependent on the current length L and the number of packet with priority k in the queue, denoted by L_k. We know that

$$r_k = \frac{L_k}{L}. \tag{13}$$

Thus, the probability of priority match drop is that

$$\begin{cases} \psi_k = 1 - \left(1 - \frac{L_k}{L}\right)^{p_0}, & \frac{L_k}{L} \geq R(k) \\ \psi_k = 0, & \frac{L_k}{L} < R(k) \end{cases}. \tag{14}$$

According to the algorithm's description, the probability of flow ID match drop is determined by $L_{k,i}/L$ and p_0. From (14), we know that ψ_k is determined by L_k/L, $R(k)$, and p_0, the drawing factor p_0 is the same for packet with any priority level. We also noticed that the interactions among L_k/L, $R(k)$, and ψ_k may cause some confusion. On the one hand, a large value of $R(k)$ results in a smaller probability of $L_k/L \geq R(k)$, which can leads to $\psi_k > 0$. On the other hand, if $L_k/L \equiv R(k)$, a large value of $R(k)$ means a larger L_k/L, which leads to a larger value of ψ_k. That is to say, the aggregate flows with larger $R(k)$ will be given more penalty on the condition that $L_k/L \geq R(k)$. This ensures that the system can maintain stable status even when congestion occurs.

Now, let us study the relation between the aggregate throughput of priority level k and its allocation $R(k)$.

Theorem 1: Stability Condition

The number of flows which go through the $M/M/1$ queue is N. The packets leave the queue with rate C and all packets are of the same size. The condition that the system is stable is as follows: if the ratio of the number packets of flow i in the queue and total number of packets in the queue can be maintained at $R(i)$, then the flow i will leave the queue with the speed $C*R(i)$.

Proof: In the time interval τ, the number of packets from flow i is $C \cdot \tau \cdot R(t)$. Thus, flow i leave the queue with the speed as follows:

$$\frac{C \cdot \tau \cdot R(t)}{\tau} = C \cdot R(t). \tag{15}$$

■

As described in Algorithm 1, the priority match drop only occurs when $L_k/L \geq R(k)$ and $L > L_{th}$. From (14), we know the value ψ_k will increase as the drawing factor F increases. Meanwhile, the increase of the drawing factor p_0 means that congestion becomes heavy. Thus, it can ensure that $L_k/L \approx R(k)$. According to the theorem 1, we get that the speed of aggregate flows with priority k leave the queue is approximate to $C(t) \cdot R(t)$ when system is in stable status, where $C(t)$ is the throughput of router.

IV. Simulation Results

To validate our proposed algorithm extensive simulations are conducted using ns simulator version 2 [21], [22]. The proposed CHOKeR algorithm is compared with other state of art AQM schemes, namely, CHOKeW [5], RED [6], RIO [11], and

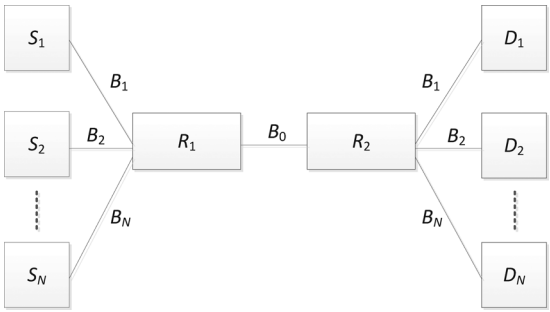


Fig. 2. Network topology for simulation.

BLUE [7], under the network topology as shown in Fig. 2, where $B_0 = 1$ Mb/s and $B_i = 10$ Mb/s, $i = 1, 2, \ldots, N$. Although such a topology is simple, it allows us to evaluate and compare the AQM algorithms. Moreover, such a topology represents a typical network access architecture for Internet Service Providers and has been used for performance evaluations in a number of works, e.g., [5] and [10].

The buffer limit is 500 packets, and the average packet size is 1000 bytes. TCP flows are driven by FTP applications, and UDP flows are driven by constant bit rate (CBR) traffic at the speed 10 Mbs/s. All TCPs are TCP SACK. Each simulation runs for 500 s[1]. We use another parameter $W(k)$ to denote weight for priority k in simulations. The ratio of allocated bandwidth for priority k is

$$R(k) = \frac{W(k)}{\sum_{j=1}^{M} W(j)}. \tag{16}$$

Simulation parameters of the evaluated AQM schemes are listed in Table I. p_w^+ and p_w^- are the increased drop probability and decreased drops probability in CHOKeW, respectively. P_r^+ and p_r^- are the increased drop probability and decreased drops probability in CHOKeR, respectively.

Fig. 3 illustrates the RCF of goodput for flows at each priority level of CHOKeR, CHOKeW, and RIO. We simulate 200 TCP flows. Each priority level is assigned an equal number of flows. In Fig. 3, we can see that the network goodput is zero when RCF of goodput for flow RIO out is 0.2. In other words, 20 of the 100 "out" flows are starved. Flow starvation is very common in RIO, but it rarely happens in CHOKeR and CHOKeW because CHOKeR and CHOKeW are using the matched drops.

Figs. 4 and 5 demonstrate the aggregate TCP goodput for each priority level versus the number of TCP flows for three priority levels and four priority levels of CHOKeW and CHOKeR, respectively. In Fig. 4, the number of TCP flows ranges from 30 to 300 at each level. When the number of TCP flows changes, the ratios of allocated bandwidth for each priority level in CHOKeR are still stable, while CHOKeW cannot support the bandwidth allocation. Moreover, as the number of priorities increases, CHOKeR is still stable, while CHOKeW becomes worse in terms of bandwidth allocation. These results verify the correctness of Theorem 1 for proportional bandwidth allocation.

In Fig. 6, it can be seen that for CHOKeW, CHOKeR and WFQ, the aggregate goodput of high-priority flows rises as the weight ratio $w(2)/w(1)$ increases, and the aggregate goodput of low-priority flows falls. For CHOKeR, the ratio of the aggregate goodput of two priority levels is approximately equal to the weight ratio. However, for WFQ, the aggregate goodput of high-priority flows rises slowly even if the ratio increases

[1]The simulation time of 500 s is decided by observing that the network performance measures, throughput and queue length, stabilize by 500-s simulation time.

TABLE I
SIMULATION SETTINGS

AQM	Parameters	Value
CHOKeW [5]	L_{th}	100 packets
	L^-	125 packets
	L^+	175 packets
	p_w^+	0.002
	p_w^-	0.001
CHOKeR	L_{th}	100 packets
	L^-	125 packets
	L^+	175 packets
	p_r^+	0.002
	p_r^-	0.001
RED [6]	*min_th*	100 packets
	max_th	200 packets
	EWMA weight	0.02
	p*max*	0.02
RIO [11]	*minth_out*	100 packets
	maxth_out	200 packets
	minth_in	150 packets
	maxth_in	250 packets
	pmax_in	0.01
	pmax_out	0.02
BLUE [7]	p_b^+	0.0025
	p_b^-	0.00025
	update interval	100 ms

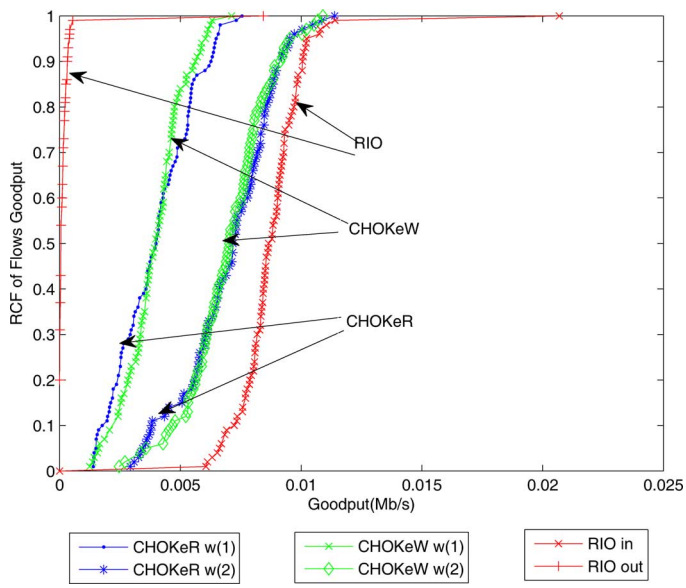


Fig. 3. RCF of CHOKeW, CHOKeR, and RIO under a scenario of two priority levels.

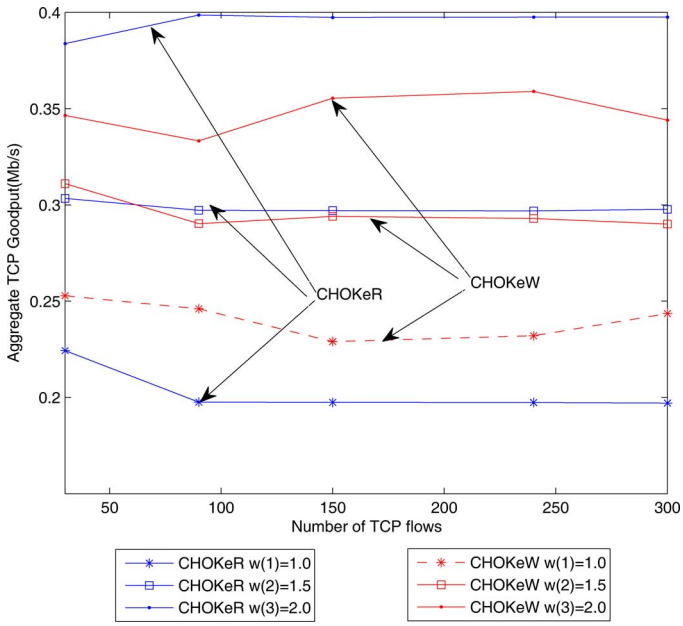


Fig. 4. Aggregate goodput versus the number of TCPflows under a scenario of three priority levels.

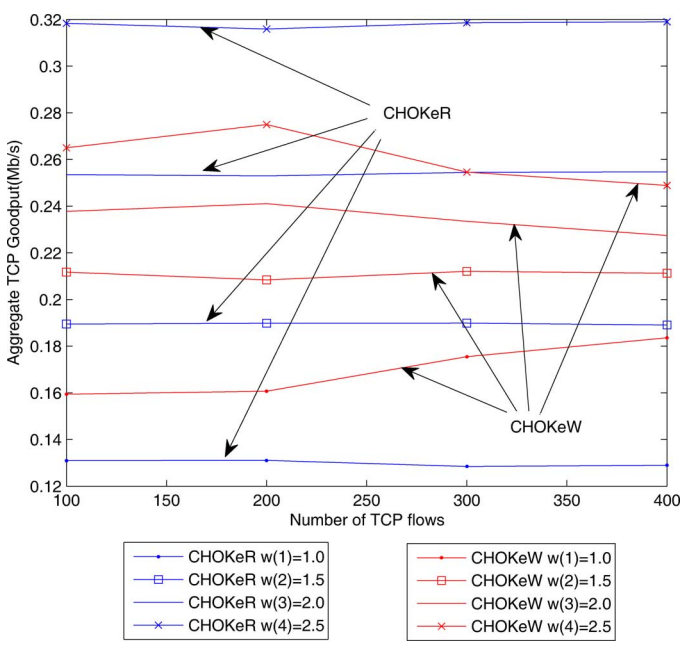


Fig. 5. Aggregate goodput versus the number of TCP flows under a scenario of four priority levels.

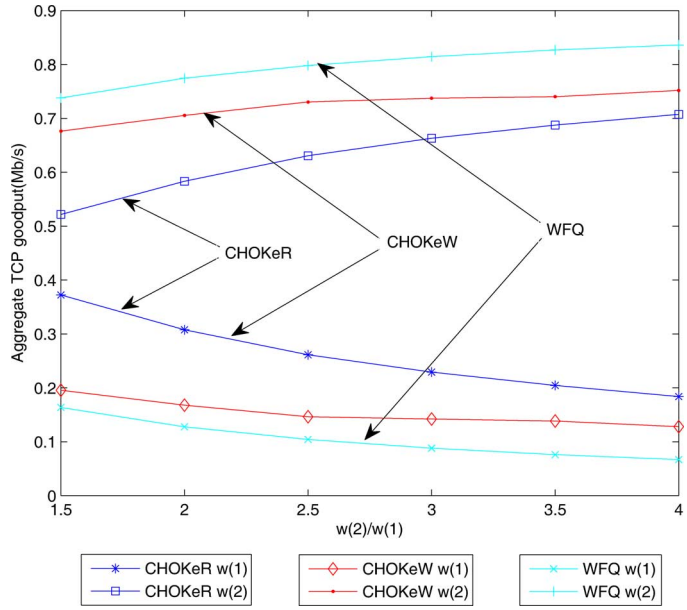


Fig. 6. Aggregate TCP goodput versus $w(2)/w(1)$ when 15 flows are assigned $w(1)$ and 75 flows $w(2)$.

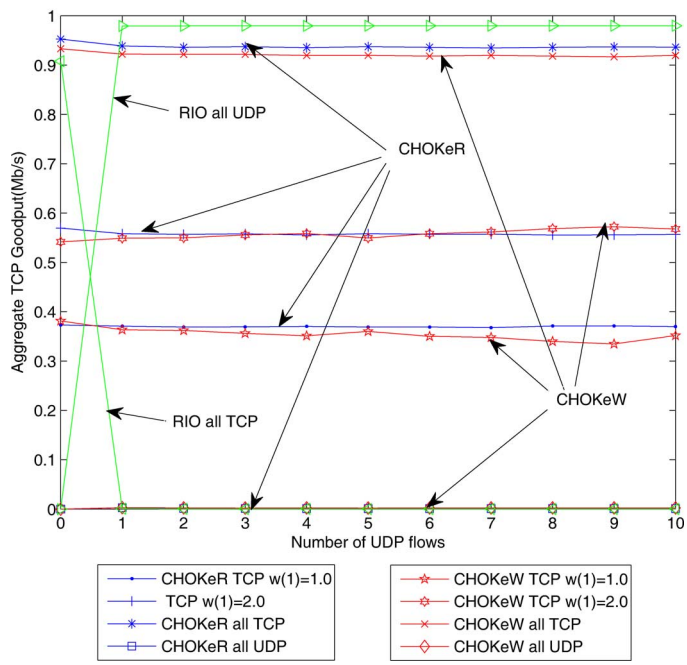


Fig. 7. Aggregate goodput versus the number of UDP flows under a scenario to investigate TCP protection.

rapidly. The performance of CHOKeW is inferior to that of CHOKeR, but is better than that of WFQ.

The goodput versus the number of UDP flows is shown in Fig. 7, where CHOKeR is compared with RIO. For CHOKeR, even if the number of UDP flows increases from 1 to 10, the TCP goodput in each priority level is quite stable, and the UDP goodput is nearly zero. By contrast, the bandwidth share for TCP flows in a RIO router is nearly zero, as high-speed UDP flows occupy almost all the bandwidth.

In order to compare the average queue length of CHOKeR and CHOKeW, all flows in the simulations are assigned the same priority. In Fig. 8, we can see that the average queue length for CHOKeW increases as the number of TCP flows increases. In contrast, the average queue length of CHOKeR can be maintained at a steady range. That is to say, when the number of TCP flows increases, the queue length of the router does not increase. The reason that CHOKeR can maintain a steady average queue is that it uses a MISD drawing factor.

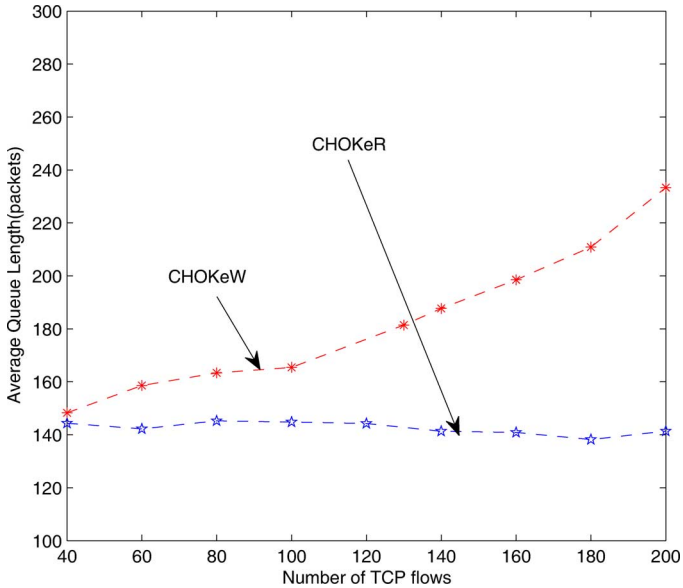


Fig. 8. Average queue length of CHOKeR and CHOKeW.

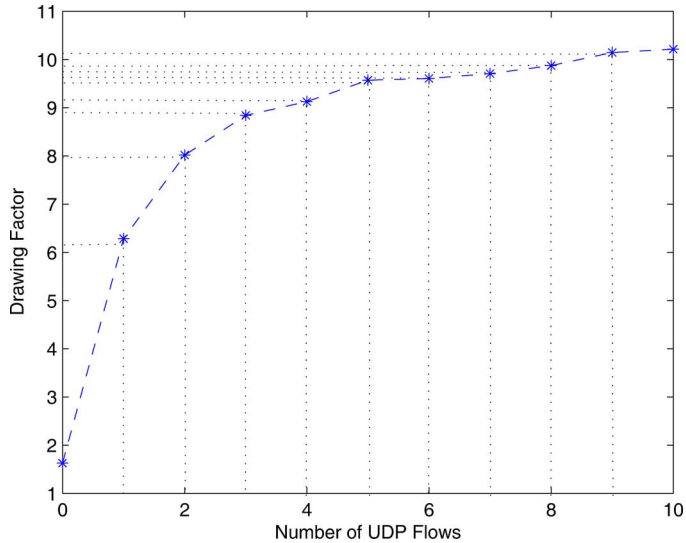


Fig. 10. Drawing factor versus the number of UDP flows under a scenario to investigate TCP protection.

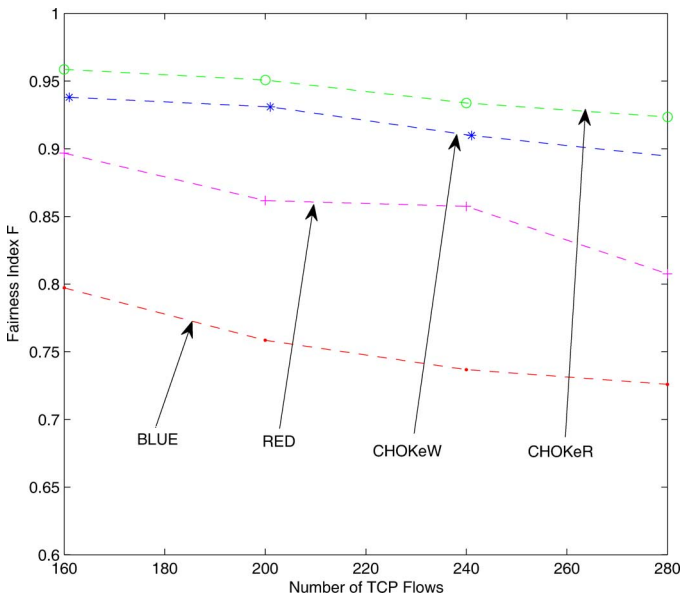


Fig. 9. Fairness index versus the number of flows for CHOKeR, RED, and BLUE.

We use the fairness index of [5] to evaluate the fairness of the schemes as follows:

$$F = \frac{\left(\sum_{i=1}^{N} g_i\right)^2}{N \sum_{i=1}^{N} g_i^2} \tag{17}$$

where g_i represents the goodput of flow i. The closer the value of fairness index is to 1, the better the fairness.

Fig. 9 shows the fairness index of CHOKeW, CHOKeR, RED, and BLUE versus the number of TCP flows ranging from 160 to 280. It can be seen that while the fairness indexes of RED and BLUE decease as the number of flows increases, CHOKeW and CHOKeR provide much better fairness. The second match drop action, i.e., the priority match drop in Algorithm 1, enables proportional bandwidth allocation in CHOKeR, which is a significant improvement over CHOKeW.

Fig. 10 illustrates the relationship between the drawing factor p_0 of CHOKeR and the number of UDP flows. As more UDP flows start, p_0 increases, but it rarely reaches the high value of starting to block TCP flows, while those high-speed UDP flows are blocked.

V. Conclusion

We proposed a cost effective AQM scheme named CHOKeR. CHOKeR uses MISD to update the drawing factor, such that large-scale and burst data can be processed fast, and congestion can be avoided. When the number of flows increase abruptly, CHOKeR's proportional bandwidth allocation with multiple priority is able to guarantee TCP protection. Both the analytical model and simulation results demonstrate that the CHOKeR achieved proportional bandwidth allocation for flows in different priorities, and fairness for flows in the same priority. With the increase of audio and video flows, CHOKeR reduces the UDP flows to protect TCP flows, and allocate a fair share of bandwidth to UDP flows.

In our future work, we will focus on the TCP protection in heterogeneous environment [23]. In its current design, CHOKeR algorithm only controls the state of the output queue. In practical network device, it is feasible to control input queue and cooperate congestion control at the output queue. Such cooperative control may further improve the fairness among users.

References

[1] S. Blake, D. black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," in *Proc. IETF RFC*, Dec. 1998, vol. 2475.

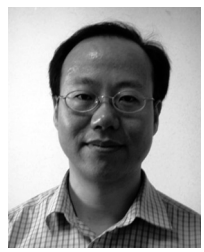
[2] N. N. Zhang, M. Y. Yang, Y. W. Jing, and S. Y. Zhang, "Congestion control for DiffServ network using second-order sliding mode control," *IEEE Trans. Ind. Electron.*, vol. 56, no. 9, pp. 3330–3336, Sep. 2009.

[3] I. D. Barrera, S. K. Setephan, and G. R. Arce, "Statistical detection of congestion in routers," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 957–968, Mar. 2010.

[4] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Network.*, vol. 7, no. 4, pp. 458–472, Aug. 1999.
[5] S. Wen, Y. Fang, and H. Sun, "Differentiated bandwidth allocation with TCP protection in core routers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 1, pp. 34–47, Jan. 2009.
[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Network.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
[7] W. Feng, K. Shin, D. Kandlur, and D. Saha, "The BLUE acitve queue management algorithm," *IEEE/ACM Trans. Network.*, vol. 10, no. 4, pp. 513–528, Aug. 2002.
[8] R. Mahajan and S. Floyd, "Controlling high-bandwidth flows at congestion router," in *Proc. 9th Int. Conf. Network Protocols*, 2001, pp. 192–201.
[9] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation," in *Proc. 19th IEEE INFOCOM*, 2000, pp. 942–951.
[10] A. Tang, J. Wang, and S. Low, "Understanding CHOKe," in *Proc. 23rd Annu. Joint Conf. IEEE Comput. and Commun.*, 2003, pp. 114–124.
[11] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. Network.*, vol. 6, no. 4, pp. 362–373, Aug. 1998.
[12] K. McLaughlin, S. Sezer, H. Blume, X. Yang, F. Kupzog, and T. Noll, "A scalable packet sorting circuit for high-speed WFQ packet scheduling," *IEEE Trans. Very Large-Scale Integr. (VLSI) Syst.*, vol. 16, no. 7, pp. 781–791, 2008.
[13] Z. Guo and H. Zeng, "Simulations and analysis of weighted fair queuing algorithm in Opnet," in *Proc. IEEE Comput. Modeling Simulat.*, 2009, pp. 114–118.
[14] S. Wen, Y. Fang, and H. Sun, "CHOKeW: Bandwidth differentiation and TCP protection in core network," in *Proc. MILCOM*, 2005, pp. 1456–1462.
[15] K. J. Lee and Y. Lim, "Performance analysis of the congestion control scheme in signaling system no.7," in *Proc. 8th INFOCOM*, 1989, pp. 691–700.
[16] N. Teresa, F. Jose, and D. Jose, "Buffer management strategies to reduce HoL blocking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 6, pp. 739–753, Jun. 2010.
[17] F. Renato, G. Filippo, M. Bartolomeo, and R. Maurizio, "A fair and high throughput reader-to-reader anticollision protocol in dense RFID networks," *IEEE Trans. Ind. Inf.*, vol. 8, no. 3, pp. 697–706, Aug. 2012.
[18] K. Natori, R. Oboe, and K. Ohnishi, "Stability analysis and practical design procedure of time delayed control systems with communication disturbance observer," *IEEE Trans. Ind. Inf.*, vol. 4, no. 3, pp. 185–197, Aug. 2008.
[19] S. Ramabhadran and J. Pasquale, "Stratified round robin: A low complexity packet scheduler with bandwidth fairness and bounded delay," *IEEE/ACM Trans. Network.*, vol. 14, no. 6, pp. 1362–1373, Dec. 2006.
[20] L. L. H. Andrew, S. H. Low, and B. P. Wydrowski, "Understanding XCP: Equilibrium and fairness," *IEEE/ACM Trans. Network.*, vol. 17, no. 6, pp. 1697–1710, Dec. 2009.
[21] Website: ns-2 (Network Simulator Version 2). 2008 [Online]. Available: http://www.isi.edu/nsnam/ns/
[22] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Network.*, vol. 8, no. 1, pp. 133–145, Feb. 2000.
[23] A. Schranzhofer, J. J. Chen, and L. Thiele, "Dynamic power-aware mapping of applications onto heterogeneous MPSoC platforms," *IEEE Trans. Ind. Inf.*, vol. 6, no. 4, pp. 692–707, Nov. 2010.

Linyun Lu received the M.E. degrees from Shenyang Institute of technology, Shenyang, China, and the Ph.D. degree from Beijing Jiaotong University, Beijing, China.

She is an Associate Professor with Beijing Jiaotong University, Beijing, China. Her research interests include cross-layer network performance, heterogeneous networks, and multi-user signal processing. She has been involved with and led projects for the National Natural Science Foundation of China and other commercial projects ranging from OpenFlow and congestion control.



Haifeng Du received the M.E. degrees from Shenyang Institute of technology, Shenyang, China, and the Ph.D. degree from Beijing Jiaotong University, Beijing, China.

He is a Senior Engineer with Pica8 as well as a Postdoc Fellow with Beijing Jiaotong University, Beijing, China. His research interests include TCP/IP protocol and QoS design. He has been involved with and led a number of projects from the National Natural Science Foundation of China. As a technology manager, he led the projects of Open Switches in Pica8.

Ren Ping Liu (M'09) received the B.E. and M.E. degrees from Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree from the University of Newcastle, Newcastle, Australia.

He is a Principal Scientist of networking technology with CSIRO, Sydney, Australia. His research interests include Markov chain modelling, QoS scheduling, and security analysis of communication networks. He has also been heavily involved with and led commercial projects ranging from QoS design, TCP/IP inter-networking, security, wireless networking, to next generation network architectures. As a CSIRO consultant, he delivered networking solutions to government and industrial customers, including Optus, AARNet, Nortel, Queensland Health, CityRail, Rio Tinto, and DBCDE.