CFD: an efficient active queue management algorithm with CFD

Xianliang Jiang and Guang Jin✉

The existence of the bufferbloat and high-bandwidth non-adaptive flows results in the high queuing delay, which affects the performance of interactive applications, and the inter-flow unfairness, respectively. To solve above-mentioned issues, an efficient active queue management algorithm with *C*ontrolling *F*airness and *D*elay (*CFD*) is proposed. Different from most other schemes, CFD can achieve the low queuing delay and inter-flow fairness simultaneously without the queue isolation. Extensive simulation results also prove the effectiveness and feasibility of CFD.

Introduction: The bufferbloat [1] seriously affects the performance of delay-sensitive applications in the current Internet. To mitigate the issue, the delay-controlled active queue management (AQM) algorithms such as controlled delay (CoDel) [2] and proportional integral controller enhanced (PIE) [3] were proposed. They can minimise the queuing delay and guarantee the performance of delay-sensitive applications. However, in the presence of high-bandwidth (HBF) non-adaptive flows (e.g. selfish user datagram protocol (UDP) flows), the delay-controlled AQM algorithms are hard to protect the performance of adaptive flows (e.g. delay-sensitive transmission control protocol (TCP) flows) such as the inter-flow fairness.

In [4], Floyd and Jacobson proposed the random early detection (RED) algorithm. It adopts the average queue length to calculate the packet dropping probability (PDP) and can acquire the better queue stability. However, RED cannot achieve the inter-flow fairness in the presence of HBF non-adaptive flows and ensure the low queuing delay. To achieve the inter-flow fairness, the stateless and fair AQM algorithms, e.g. geometric choose and keep for responsive flows (gCHOKe) [5], were proposed. They penalise HBF non-adaptive flows and ensure the fair share of the bottleneck bandwidth. To obtain the low queuing delay, CoDel [2] and its variants [3, 6] were developed to limit the sojourn time of packets in a queue. Although they can reduce the queuing delay, the inter-flow fairness is still not ensured. In this Letter, we devote to solve the fairness issue, protect adaptive flows, and maintain the queuing delay around a referred value under the circumstance of HBF non-adaptive flows.

Inspired by the enhanced proportional–integral (PI) controller of PIE [3] and the *sample-match* method of gCHOKe [5], we propose an AQM scheme with *C*ontrolling *F*airness and *D*elay (*CFD*) in this Letter. *The contributions include* a fairness-controlled algorithm and a method to compute the differentiated PDP (D-PDP) of different flows. To aid the computation of the D-PDP, CFD exploits the landmark least-recently-used (L-LRU) cache [7] to store partial flows states. Different from most other AQM schemes, CFD can achieve the low queuing delay and inter-flow fairness simultaneously without the queue isolation [6] when HBF non-adaptive flows arise. Compared with CoDel [2] and PIE [3], CFD can achieve the inter-flow fairness when HBF non-adaptive flows arise. Different from RED [4] and its variants [5], CFD can achieve the low queuing delay. To deal with the unfairness, some other variants of RED were presented. However, they were insufficient to protect adaptive flows effectively.

Framework of CFD: In CFD, we design a method to compute the D-PDP of different flows based on the enhanced PI controller and *sample-match* method. Therein, the former can solve the bufferbloat problem [1]. Moreover, the latter is usually used to identify HBF flows. To improve the accuracy of the HBF identification, we adopt an L-LRU cache to record the states (e.g. hit counts and flow identification) of HBFs. Using an enhanced PI controller, CFD estimates the PDP with the queuing delay (the queue backlog divides the output link speed) and a referred delay periodically. Using the flow states in the L-LRU cache, CFD computes the D-PDP and manages the arriving packets.

Fig. 1 presents the structure of CFD. It consists of three parts: delay controller (DC), fairness controller (FC), and queue manager (QM). Thereinto, the DC, which consists of the rate estimator, delay estimator, and PDP calculator, can estimate the queuing delay and then compute the PDP. The FC, which consists of the HBF detector, L-LRU cache, and hit frequency of flows, can identify HBFs and compute the hit frequency of different flows. On the basis of a D-PDP calculator, the QM can compute the PDP of different flows combining the results of FC and DC. After that QM can manage arriving packets with the D-PDP generated by the D-PDP calculator. To achieve the controlled fairness, we design an equation to compute the D-PDP of the ith flow, p_i^{d}

$$p_i^{\mathrm{d}} = p_k \cdot \left(1 - \frac{f_i}{\sum_{j=1}^{n}(f_j)}\right) \tag{1}$$

where p_k is the PDP generated by the DC in the kth time interval (e.g. 10 ms). f_i and f_j are the hit counters of ith and jth flows recorded in the L-LRU cache, respectively. The smaller p_i^{d} means that the arrived packet of ith flow has a larger probability to be discarded. If $u > p_i^{\mathrm{d}}$, the packet from the ith flow is dropped. Note, u is a random number in [0, 1]. Next, we present the details of FC, DC, and QM. Note that $f_i = 0$ and $p_i^{\mathrm{d}} = p_k$ when the state of the ith flow is not in an L-LRU cache.

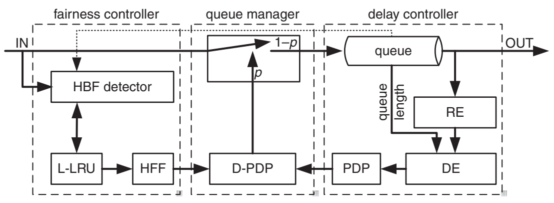

Fig. 1 *Structure and key components (FC, DC, and QM) of CFD*

Delay controller: In DC, an enhanced PI controller similar to [3] is used to compute the PDP. Such as PIE, CFD regards the queuing delay instead of the queue length as an input. As described in [3], the PI controller has the following form: $p_k = p_{k-1} + \alpha \cdot (c_{\mathrm{d}} - r_{\mathrm{d}}) + \beta \cdot (c_{\mathrm{d}} - l_{\mathrm{d}})$, where p_{k-1} is the previous value of p_k. Moreover, c_{d}, r_{d}, and l_{d} are the current queuing delay, referred queuing delay, and the previous queuing delay of c_{d}, respectively. α and β are two scaling factors. To balance the rapid response rate and the stability of the queuing delay, such as [3], CFD adopts the following method to adjust α and β:

$$[\alpha,\ \beta] = \begin{cases} [\overline{\alpha}/8,\ \overline{\beta}/8], & \text{if } p_k < 0.01 \\ [\overline{\alpha}/2,\ \overline{\beta}/2], & \text{if } p_k < 0.1 \\ [\overline{\alpha},\ \overline{\beta}], & \text{other} \end{cases} \tag{2}$$

where $\overline{\alpha}$ and $\overline{\beta}$ are static configurable parameters. To estimate the current queuing delay (c_{d}) as accurate as possible, the DC employs Little's law such as [3]. Suppose that q_{len} and d_{r} are the current queue length and the estimated departure rate, c_{d} equals $q_{\mathrm{len}}/d_{\mathrm{r}}$. With regard to d_{r}, the DC adopts the method such as [3] to estimate it.

Fairness controller: The identification of HBFs is a key to achieve the inter-flow fairness. FC uses the *sample-match* method similar to [5] to identify HBFs. Specifically, it draws n (≥ 1) packets from a queue randomly and compares them with the arriving packet. If they belong to the same flow, the hit counter of the flow state in an L-LRU increases. Note that, to improve the accuracy of the HBF identification, FC employs an L-LRU cache such as [7] to store the states of HBFs. Using the flows' states in an L-LRU, FC can distribute different weights, calculated by $f_i/\sum_{j=1}^{n}(f_j)$, to different flows and achieve the inter-flow fairness in the presence of HBFs.

Queue manager: The QM drops or marks packets randomly according to a dropping probability, p_i^{d}, that is calculated by (1). If $u > p_i^{\mathrm{d}}$, the packet of the ith flow is dropped or marked. Therein, u is a random number.

Evaluation and analysis: We conduct extensive experiments using the packet-level simulator NS-2 [8] in the presence of HBF non-adaptive flows, which is generated by long-lived UDP flows. To prove the performance of CFD reasonably, we pick RED [4], gCHOKe [5], CoDel [2], and PIE [3] as compared algorithms and adopt the Jain's fairness index (JFI) [9] and average flow completion time (AFCT) [10] metrics. Without any other statements, $\overline{\alpha} = 0.125$ and $\overline{\beta} = 1.25$ are the same as [3]. The number of TCP and UDP flows generated by NewReno and UDP are 50 and 1, respectively. The sending rate of UDP flows is 0.5 Mbit/s. Moreover, the packet sizes of TCP and UDP are 1000 and 500 B, respectively. The update interval of p_k is 30 ms and $r_{\mathrm{d}} = 20$ ms. The size of the L-LRU cache is 1000 records. In addition, the configurable parameters of compared schemes are the optimum value recommended by their respective authors. To ensure the credibility of experimental results, we adopt the dumbbell (single bottleneck link) and parking-lot (multiple bottleneck links) topology, which are widely used in compared schemes.

Fig. 2 is the dumbbell topology. The bottleneck queue size is 100 packets. All compared schemes are deployed in the bottleneck queue. Other queues use DropTail. The delay and capacity of the bottleneck and non-bottleneck links are provided in Fig. 2. Note that all flows are from C_i to *Server* ($i = 1, \ldots, n$). To measure the fairness and AFCT of compared schemes in the dumbbell topology, we build two different configuration to conduct experiments. In the first one, the sending rate of UDP flows varies in [0.1, 1.0] Mbit/s and other configured values are default. In another case, the number of TCP flows varies in [5, 50] and other configured values are default.

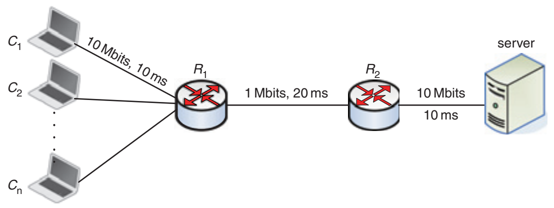

Fig. 2 *Topology with single bottleneck link (R_1–R_2) for measuring fairness and AFCT of different algorithms*

First, we measure the JFI values of different schemes. Note that the throughput for computing the JFI is an average value in [1 s, 200 s]. As shown in Fig. 3, CFD can achieve better performance than other schemes when the sending rate of UDP flows and the number of TCP flows vary. These results also indicate that CFD can protect responsive flows.

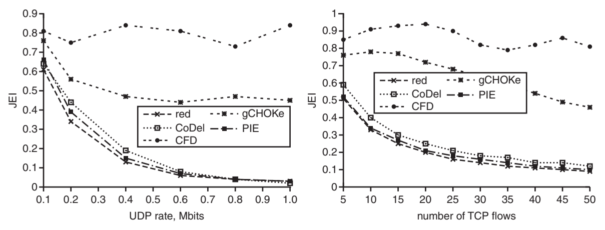

Fig. 3 *Fairness of different algorithms with different UDP rates (left) and number of TCP flows (right)*

Secondly, short-lived TCP flows are used to measure the AFCT of different algorithms. The flow size subjects to a uniform distribution with mean values in [10, 100] packets. The number of short-lived flows varies in [100, 1000]. Other configured values are default. At 0 s, short-lived TCP flows start to transmit data. Note that the simulation time is un-limited. The results shown in Fig. 4 indicate that CFD can achieve better performance than other schemes when the size and number of flows vary.

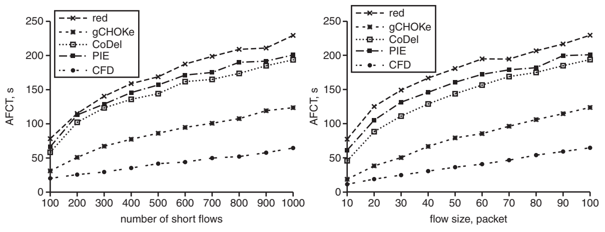

Fig. 4 *AFCT of different algorithms with different numbers of short flows (left) and flow size (right)*

Thirdly, we use a parking-lot topology (Fig. 5) to evaluate CFD. The bottleneck queue size, link delay, and capacity are 100 packets, 20 ms, and 1 Mbit/s, respectively. All compared schemes are deployed in bottleneck queues and other queues use DropTail. The delay and capacity of non-bottleneck links are shown in Fig. 5. Three UDP flows are established from S_i to S_{i+1} ($i = 1, 2, 3$). Owing to space limitation, only the AFCT results are listed in this Letter. We use the same model as previous to build short-lived TCP flows. Note that all short-lived flows are from C_i to *Server* ($i = 1, \ldots, n$). Fig. 6 is the results and we can see that CFD outperforms other schemes in the AFCT.

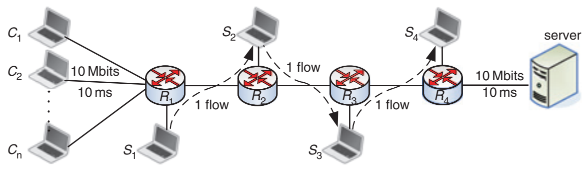

Fig. 5 *Topology with multiple bottleneck links (R_1–R_2, R_2–R_3, and R_3–R_4) for measuring AFCT of different algorithms*

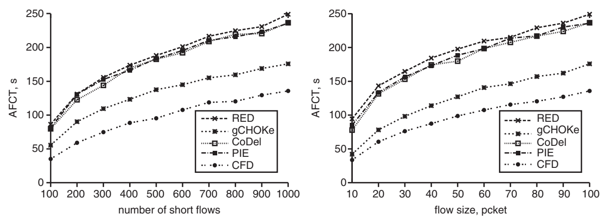

Fig. 6 *AFCT of different algorithms with different numbers of short flows (left) and flow size (right)*

In terms of space efficiency, CFD consumes some additional memory (e.g. L-LRU cache), whose size is often a constant. Owing to the simplicity and efficiency, we consider CFD can be easily deployed.

Conclusion: This Letter presents a novel AQM algorithm, CFD, based the PI controller, the *sample-match* method, and the L-LRU cache. It can achieve the inter-flow fairness, reduce the AFCT of adaptive flows, and hold the queuing delay around a referred value when HBF non-adaptive flows arise. Furthermore, the scheme is easy to deploy in current Internet routers. In future, we will adopt the control theory to analyse the stability of CFD and optimise its parameters in more complex scenarios etc.

Acknowledgment: This work was supported by the NBU Research Project.

© The Institution of Engineering and Technology 2016
Submitted: *1 July 2016* E-first: *3 November 2016*
doi: 10.1049/el.2016.2370
One or more of the Figures in this Letter are available in colour online.

Xianliang Jiang and Guang Jin (*Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, People's Republic of China*)

✉ E-mail: jinguang@nbu.edu.cn

References

1 Gettys, J.: 'Bufferbloat: dark buffers in the Internet', *Internet Comput.*, 2011, **15**, (3), pp. 95–96
2 Nichols, K., and Jacobson, V.: 'Controlling queue delay', *ACM Queue*, 2012, **10**, (5), pp. 1–15
3 Pan, R., Natarajan, P., Piglione, C., *et al.*: 'PIE: a lightweight control scheme to address the bufferbloat problem'. Proc. HPSR'13, Taipei, Taiwan, July 2013
4 Floyd, S., and Jacobson, V.: 'Random early detection gateways for congestion avoidance', *ACM Trans. Netw.*, 1993, **1**, (4), pp. 397–413
5 Eshete, A., and Jiang, Y.: 'Generalizing the CHOKe flow protection', *Comput. Netw.*, 2013, **57**, pp. 147–161
6 Joergensen, T.H., McKenney, P., Taht, D., *et al.*: 'FlowQueue-CoDel'. Available at https://www.tools.ietf.org/html/draft-hoeiland-joergensen-aqm-fq-codel-00, March 2014
7 Che, L., Qiu, B., and Wu, H.: 'Improvement of LRU cache for the detection and control of long-lived high bandwidth flows', *Comput. Commun.*, 2005, **29**, pp. 103–113
8 Chiu, D., and Jain, R.: 'The Network Simulator version 2'. Available at http://www.isi.edu/nsnam/ns/. 2012
9 Chiu, D., and Jain, R.: 'Analysis of the increase and decrease algorithms for congestion avoidance in computer networks', *Comput. Netw. ISDN Syst.*, 1989, **17**, pp. 1–14
10 Dukkipati, N., and McKeown, N.: 'Why flow-completion time is the right metric for congestion control', *ACM SIGCOMM Comput. Commun. Rev.*, 2006, **36**, (1), pp. 59–62