ISSN:0975-9646

CHOKe: A Stateless Queue for Securing Flows

Saniya Elza Dominic[1], Gayathri .R Krishna[2]

Department of CSE, Mangalam College of Engineering
Kottayam, Kerala, India- 686631

***Abstract*— A CHOKe is a simple to implement and stateless active queue management (AQM) schème. The top notch property that makes CHOKe attractive is that it can protect responsive TCP flows from unresponsive UDP flow. Studies have proved that both bandwidth share and buffer share of UDP traffic in a link can be bound together using a CHOKe. These studies were done and focus only for a steady state where the queue reaches equilibrium in the presence of many TCP flows and an non responsive UDP flow of fixed arrival rate. Studies failed to prove whether the protection property of CHOKe is valid especially when UDP traffic rate changes over time. For instance when the UDP rates falls to zero, the unresponsive flow may assume close to full utilization in sub-round-trip-time (sub-RTT) scales, thereby making the TCP flows to starve. This paper investigates about the CHOKe queue properties in a transient regime and tries to explain the aforementioned discrepancy. A transient regime is defined as the time period of change between two steady states of the queue, initiated as the rate of the unresponsive flow changes. The results derived from the explicit expressions that characterize flow throughputs in transient regimes helps understand a CHOKe and provide ample explanation on its intriguing behavior in the transient regime.**

***Index Terms*— CHOKe, flow protection, queue management (QM), Random Early Detection (RED), TCP.**

INTRODUCTION

A. Overview of the Flow Protection

Widely classifying, there are two distinct ways to enforce flow fairness and protection in the Internet. One way is to follow the end-to-end architectural design principle of the Internet [1] and the other being a more classical way which is via congestion control algorithms, which are typically implemented in the transport protocols (e.g., TCP) of end-hosts [2]. All users are required to abide by the scheme and respond to the network congestion properly to ensure fairness in the system globally. However, for two reasons, these requirements cannot be satisfied. First reason being no performance incentive to end-users. This is mainly because the users, who lack the congestion control algorithms, intentionally may end to use with a major share of band- width.

The second reason is that in order to meet real-time requirements, many applications do not implement congestion control to protect responsive (e.g: TCP) flows from unresponsive (e.g: UDP) ones. It is thereby necessary to introduce some mechanisms in the network since relying solely on the end-to-end schemes can be unfair or risky. The second approach is provided through router mechanisms. Such a router mechanism can be either: Per-flow fair queueing (PFFQ) scheme, e.g: Weighted Fair Queueing (WFQ) [3]: PFFQ schemes share link bandwidth among flows in a fair manner by isolating the flows into separate logical or physical first-in–first-out (FIFO) queues and maintain flow-level state information [4][5]. Flow isolation protects well-behaved flows and enables performance guarantees to such flows by building firewalls around heavy users. For high-speed implementation the maintenance of per-flow state and the dynamic management of complex queue structure are believed to be problematic·

1. 1) Queue management (QM) scheme, e.g: Random Early Detection (RED):

QM schemes focus on the buffer allocation. With a single queue shared by all flows [6], QM scheme designs are generally simpler. Among other QM schemes, RED is the most widely known scheme because it maintains an exponentially moving average queue size which indicates the level of congestion in the router. Depending on the queue size, a congested RED router drops incoming packets with a certain probability. Both high-rate and low-rate flows can be punished in equal measures, since the dropping probabilities are applied globally to all flows. The same ambient drop rate of RED can be more detrimental and highly unfair to some flows based on the nature of Internet flows (e.g: flow sizes, underlying transport protocol) and resultant differences in their responsiveness to congestion. This unfairness can be overcome by more complex variants of RED, e.g: Flow RED (FRED) [11] and RED with Proportional Differentiation (RED-PD) [12], by applying differential per-flow drop rates. But these schemes generally need to maintain partial flow state to be able to discriminate drop rates among flows [9]. It is to be noted that with no flow isolation, fairness afforded by a QM scheme is only approximate.

B. CHOKe

Unlike FRED or RED-PD, the CHOKe is a highly novel QM scheme that does not require flow state to be maintained in the router. A CHOKe [13] is designed to protect responsive (rate-adaptive) flows from unresponsive ones. A CHOKe can be implemented by a few modification of the RED algorithm. To be concise “when a packet comes at a congested router, CHOKe draws a packet at randomly from the FIFO (first-in-first-out) buffer and compares it with the arriving packet from the flow. If they both belong to the same flow, then they are both dropped, else the randomly chosen packet is left intact and the arriving packet is admitted into the buffer with a probability (exactly as computed in RED) that depends on the level of congestion [13].

An interesting feature of CHOKe is that it provides analytically proven protection of responsive flows from an unresponsive flow at the congested router. The following steady-state properties of CHOKe have been derived in the presence of many flows:[14] [16]

- Limits: An unresponsive UDP flow cannot exceed certain limits in buffer share and link bandwidth share.

The maximum UDP bandwidth share is of link capacity, and the maximum buffer share is 50% [15] [16]

- Asymptotic property: As the UDP rate increases without bound, its buffer share can asymptotically reach 50%, and queueing delay can be reduced by half, but its link utilization drops to zero.
- Spatial distribution: The spatial packet distribution in the queue can be highly non uniform [14]. The probability of finding a packet belonging to a high-rate flow in the queue diminishes as we move toward the head of the queue. The flow distribution in queue is skewed with most packets of high-rate flows found closer to queue tail, where as packets of low rates are found closer to queue head.

C. CHOKe Algorithm

The algorithm here used is CHOKe, that differentially penalizes unresponsive and unfriendly flows. The state, taken to be the number of active flows and the flow ID of each of the packets, is assumed to be unknown to the algorithm. The only feature for the algorithm is the total occupancy of the buffer. CHOKe calculates the average occupancy of the FIFO buffer using an exponential moving average, similar as RED does. It also marks two thresholds on the buffer, a minimum threshold minth and a maximum threshold maxth**.**

If the average queue size is smaller than minth**,** every arriving packet is queued into the FIFO buffer. If the aggregated arrival rate is lesser than the output link capacity, the average queue size should not constructed up to minth very often and packets are not dropped frequently. If the average queue size is larger than maxth**,** every arriving packet is dropped. This moves the queue occupancy back to below maxth**.** When the average queue size is bigger than minth**,** each arriving packet is compared with a randomly chosen packet, called drop candidate packet, from the FIFO buffer. If they have the similar flow ID, they are both dropped. Otherwise, the randomly chosen packet is kept in the buffer and the arriving packet is dropped with a probability that depends on the average queue size. The drop probability is computed same as in RED**.**

A flow chart of the algorithm is given in Figure **1.**

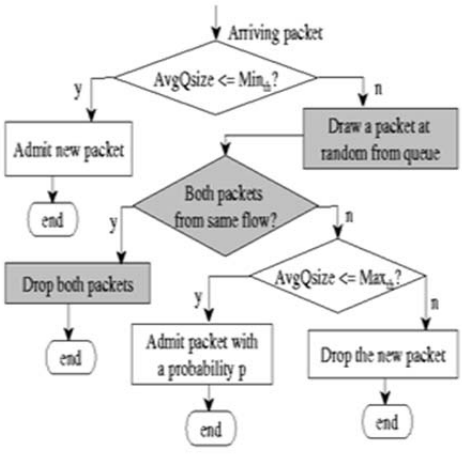

Fig 1: Choke algorithm

The descendants of CHOKe are listed below:

A. Front CHOKe

The drop candidate is chosen from the front or head of queue.

B. Back CHOKe

The drop candidate is chosen from end of queue.

C. Multi-drop CHOKe (M-CHOKe)

n packets are chosen from the buffer to compare with the incoming packet, and drop the packets that have the same flow ID as the incoming packet.

D. XCHOKe

Algorithm uses data structure to store state information. Maintains a table to hold flow's hit counter ***n***. In XCHOKe, n depends solely on CHOKe hits [19].

E. RECHOKe

It is similar to XCHOKe. The flow's hit counter ***n*** depends on table hit when the packet's flow ID is found in the table, CHOKe hit when the arriving packet's ID matches that of the randomly chosen packet and RED hit whe n the packet is chosen for dropping / marking with the RED drop probability [19].

F. CSa-XCHOKe

It improves XCHOKe by calculating packet dropping probability based on congestion level and link load. Congestion level is determined from link load and average queue length [11].

G. Self-Adjustable CHOKe (SAC)

The mechanism treats TCP and UDP flows differently, and can adaptively adjust its parameters according to the current traffic status [12].

H. A-CHOKe

CHOKe algorithm is a good solution for lockout and global synchronization problems but it sometime results in worsening TCP performance and does not work well in case of only few packets from unresponsive flows in the queue. Adaptive CHOKe (A-CHOKe) provides a stable operating point for the queue size and fair bandwidth allocation irrespective of the dynamic traffic and congestion characteristics of the flows [13]. It also obtains high utilization, low queuing delay and packet loss by tuning parameters adaptively. The dynamic value of parameter adapts itself to the varying nature of the congestion and traffic. A-CHOKe is a more sophisticated way to do M-CHOKe such that the algorithm automatically chooses the proper number of packets chosen from buffer where the buffer is divided into a number of regions [7].

I. P-CHOKe

P-CHOKe (Piggybacking CHOKe) is an algorithm based on Adaptive CHOKe. It aims to protect well-behaved flows from misbehaving flow and adaptive flows from non-adaptive flows [14]. P-CHOKe provides a stable operating point for the queue size and fair bandwidth sharing regardless of dynamic traffic and congestion characteristics of flows. P-CHOKE obtains high Packet delivery Ratio and throughput, low queuing delay and process time than the existing Adaptive CHOKe. The algorithm has a gateway module which draws, compares, admits or drops the packets randomly and sends collected acknowledgements from packet receiving nodes to sending nodes.

J. G-CHOKe

gCHOKe (geometric CHOKe), is another method which provides an advanced flow protection which is realized by introducing an extra flow matching trial upon each successful matching of packets. The difference between CHOKe and gCHOKe is the number of trials that they use to differentiate. CHOKe penalize the unresponsive flow from possibly dominating the use of the buffer and the link using a single trial of flow matching per packet arrival. However, gCHOKe additionally rewards each successful matching with a additional trial. The succession of additional trials gives an extra shield of protection to rate adaptive flows from unresponsive ones . By reducing the defined maximum number of trials, a desired protection level may be gained. This makes traffic control more tractable, which is lacking in the original plain CHOKe where flow protection is flat.

EVALUATION

In this section, we validate the results using simulations performed in ns-2.34. The network setup shown in Fig. 2 with the following settings is used: C = 20 Mb/s or 2500 packets/s, link latency 1 ms,4 buffer size 1000 packets, N = 1to10 TCP flows each of type
UDP is 1 to 5. RED buffer thresholds (in packets) $\min_{th}$ = 20 and $\max_{th}$ = 1000. Packet sizes are 1000 B. Flows start randomly on the interval [0. 2] s.

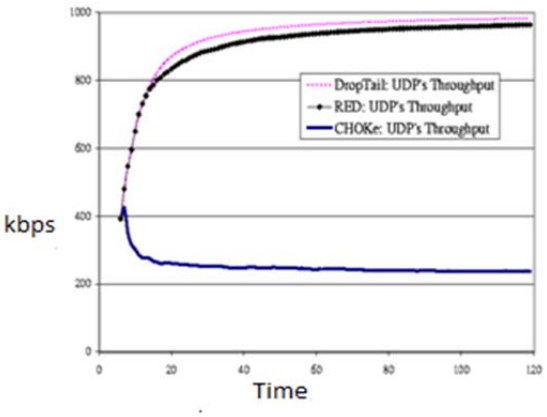

Fig: 2 UDP throughput comparisons

VI. CONCLUSION AND FUTURE WORK

This paper looks in a packet dropping scheme, CHOKe. This helps to attain fairness at a minimal implementation overhead. While existing works mainly concentrated on the Choke which limit the behavior of queue, when queue reaches equilibrium. The reaches equilibrium in the presence of long-lived TCP flows and UDP flows. The lack of showing properties where exogenous rates of unresponsive flows may dynamically change. This paper concern with Choke queue behavior when time changes. Which we model by transition from one steady queue to another. And found that the performance limits in steady state rarely hold for such transient regimes. When the rate change the queue exhibits instant fluctuation of UDP bandwidth sharing in reverse direction. This affect the smooth operation of internet. Here we consider only one UDP flows. When multiple UDP flows arrive there are two possible ways to analysis. One is to treat all UDP flows as single UDP flow. Then the same analysis flows hold. But the result is not suited to all .If one is interested in the individual buffer share and link bandwidth share each UDP flow, the current result are not applicable, future work is needed.

ACKNOWLEDGMENT

The first author would like to thanks all those people, who guided and supported. Without their valuable guidance and support, this task was not possible and also likes to thank colleagues for their discussions and suggestions.

REFERENCES

[1] J.H.Saltzer,D.P.Reed,andD.D.Clark,"End-to-end arguments in system design," *Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277-288, 1984.
[2] M. Allman, V. Paxson, and E. Blanton,"TCP congestion control," *RFC 5681*, Sep. 2009.
[3] A.Demers,S.Keshav,andS.Shenker,"Analysis and simulation of a fair queuing algorithm," *in Proc. ACM/SIGCOMM*, 1989, pp. 1-12.
[4] S. Floyd and V.Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw*., vol. 1, no. 4, pp. 397-413, Aug. 1993.
[5] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," *Tech. Rep*., 2001.
[6] S. Keshav, "An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network. Reading", *MA, USA: Addison-Wesley-Longman*, 1997.
[7] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, "Evaluating the number of active flows in a scheduler realizing fair statistical band¬width sharing," *in Proc. ACM/SIGMETRICS*, 2005, pp. 217-228.
[8] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury, "Buffer man¬agement schemes for supporting TCP in gigabit routers with per-flow queueing," *IEEE J. Sel. Areas Commun*., vol. 17, no. 6, pp. 1159-1169, Jun. 1999.
[9] A. Eshete and Y. Jiang, "Approximate fairness through limited flow list," *in Proc. Int. Teletraffic Cong*., 2011, pp. 198-205.
[10] C. Hu, Y. Tang, X. Chen, and B. Liu, "Per-flow queueing by dynamic queue sharing," *in Proc. IEEE INFOCOM*, 2007, pp. 1613-1621.
[11] D. Lin and R. Morris, "Dynamics of random early detection," *Comput. Commun*. Rev., vol. 27, no. 4, pp. 127-137, 1997.
[12] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," *in Proc. IEEE ICNP*, 2001, pp. 192-201.
[13] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe—A stateless active queue management scheme for approximating fair bandwidth alloca¬tion," *in Proc. IEEE INFOCOM*, 2000, pp. 942-951.
[14]A.Tang,J.Wang,andS.H. Low,"UnderstandingCHOKe: Throughput and spatial characteristics," *IEEE/ACM Trans. Netw*., vol. 12, no. 4, pp. 694-707, Aug. 2004.
[15] J. Wang, A. Tang, and S. H. Low, "Maximum and asymptotic UDP throughput under CHOKe," *in Proc. ACM SIGMETRICS*, 2003, pp. 82-90.
[16] R. Pan,C.Nair, B. Yang, and B. Prabhakar, "Packet dropping schemes, some examples and analysis," *in Proc. Allerton Conf Commun. , Contro, Comput*., 2001, pp. 563-572.
[17] H. Jiang and C. Dovrolis, "Why is the Internet traffic bursty in short time scales?' " *in Proc. ACM SIGMETRICS*, 2005, pp. 241-252.
[18] F. Baccelli and D. Hong, "AIMD, fairness and fractal scaling of TCP traffic," *in Proc. IEEE INFOCOM*, 2002, pp. 229-238.
[19] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *in Proc. ACM SIGMETRICS*, 1996,pp.160-169.
[20] A. Eshete and Y. Jiang, "Generalizing the choke flow protection," *Comput. Netw.*, vol. 57, no. 1, pp. 147-161, 2013.
[21] A. Tang, L. L. H. Andrew, K. Jacobsson, K. H. Johansson, H. Hjal-marsson, and S. H. Low, "Queue dynamics with window flow control," *IEEE/ACM Trans. Netw*., vol. 18, no. 5, pp. 1422-1435, Oct. 2010.