www.ioirp.com
International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
ISSN: 2394-6364, Volume – 1, Issue – 6. May 2015

A Novel AQM Algorithm Named CHOKeR For Proportional Bandwidth Allocation and TCP Protection

[1]Shabeena Lylath.D.B, [2]Miss.Pallavi.R
Department of computer science and Engg.
Akshya Institute of Technology
Tumkur, India
[1]shabeena.l@rediffmail.com

***Abstract*— A recently proposed active queue management, CHOKe, is stateless, simple to implement, yet surprisingly effective in protecting TCP from UDP flows. They have presented an equilibrium model of TCP/CHOKe. They have proven that, provided the number of TCP flows is large, the UDP bandwidth share peaks at (+ 1) 1 = 0 269 when UDP input rate is slightly larger than link capacity, and drops to zero as UDP input rate tends to infinity. They clarified the spatial characteristics of the leaky buffer under CHOKe that produce this throughput behavior. Specifically, they proved that, as UDP input rate increase s, even though the total number of UDP packets in the queue increases, their spatial distribution becomes more and more concentrated near the tail of the queue, and drops rapidly to zero toward the head of the queue. In stark contrast to a non leaky FIFO buffer where UDP bandwidth shares would approach 1 as its input rate increases without bound, under CHOKe, UDP simultaneously maintains a large number of packets in the queue and receives a vanishingly small bandwidth share, the mechanism through which CHOKe protects TCP flows.**

Keywords*— Active queue management, bandwidth share, CHOKe, leaky buffer, spatial characteristics.

I. INTRODUCTION

TCP is believed to be largely responsible for preventing congestion collapse while the Internet has undergone dramatic growth in the last decade. Indeed, numerous measurements have consistently shown that more than 90% of the traffic on the current Internet is still TCP packets, which, fortunately, are congestion controlled. Without a proper incentive structure, however, this state of affair is fragile and can be disrupted by the growing number of nonrated-adaptive (e.g., UDP-based) applications that can monopolize network bandwidth to the detriment of rate-adaptive applications. This has motivated several active queue management schemes, e.g., [2]–[4], [7], [that aim at penalizing aggressive flows and ensuring fairness. The scheme, CHOKe, of [11] is particularly interesting in that it does not require any state information and yet can provide a minimum throughput to TCP flows. In this paper, we provide an analytical model of CHOKe that explains both its throughput behavior and the spatial characteristics of its leaky buffer that underlies the throughput behavior. 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. An important component of the many QoS architectures proposed is the packet scheduling algorithm used by routers in the network. The packet scheduler determines the order in which packets of various independent owes are forwarded on a shared output link. One of the simplest algorithms is First Come First Served (FCFS), in which the order of arrival of packets also determines the order in which they are forwarded over the output link. While almost trivial to implement, FCFS clearly cannot enforce QoS guarantees, as it allows rogue ows to capture an arbitrary fraction of the output bandwidth. In general, a packet scheduler should have the following properties.

II. ISSUES IN CHOKER

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 Diff- Serv. 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 burst. 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.

III. Motivational Goal and Algorithm

Our work is motivated by the need for a simple, stateless algorithm that can achieve flow isolation and/or approximate fair bandwidth allocation. As mentioned in the introduction, existing algorithms (like RED, FQ and others) are either simple to implement or able to achieve flow isolation, but not both simultaneously. We seek a solution to the above problem in the context of the Internet. Thus, we are motivated to find schemes that differentially penalize "unfriendly" or "unresponsive" Rows4, which implies bad implementations of TCP, and UDP-based Rows. Further, we seek to preserve some key features that RED possesses; such as its ability to avoid global synchronization5, its ability to keep buffer occupancies small and ensure low delays, and its lack of bias against bursty traffic. By doing this, in the absence of unfriendly or unresponsive flows our algorithm will perform similarly to RED. Next, we need a benchmark to compare the extent of fairness achieved by our solution. Maxmin fairness suggests itself as a natural candidate for two reasons: (a) It is well-defined and widely understood in the context of computer networks (see [11, page 526, or [IO]), and (b) the FQ algorithm is known to achieve it. However, for any scheme to achieve perfect maxmin fairness without flow state information seems almost impossible. Maxmin fairness is not suitable in our context since we do not identify the flow(s) with the minimum resource allocation and maximize its (their) allocation. Instead we identify and reduce the allocation of the flows which consume the most resources. In other words, we attempt to minimize the resource consuinption of the maximum flow or seek to achieve minmax fairness6. The resource freed up as a result of minimizing the maximum flow's consumption is distributed among the other flows. In the Internet context the former flows are either unfriendly TCP or UDP, and the latter flows are TCP.

IV. The Choke Algorithm

Suppose that a router maintains a single FIFO buffer for queuing the packets of all the flows that share an outgoing link. they described an algorithm, 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 observable for the algorithm is the total occupancy of the buffer. CHOKe calculates the average occupancy of the FIFO buffer using an exponential moving average, just 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 less than minth, every arriving packet is queued into the FIFO buffer. If the aggregated arrival rate is smaller than the output link capacity, the average queue size should not build up to minth very often and packets are not dropped frequently. If the average queue size is greater than maxth, every arriving packet is dropp.ed. This moves the queue occupancy back to below maxth. When the average queue size is bigger than mznth, each arriving packet is compared with randomly selected packet, called drop candidate packet, from the FIFO buffer. If they have the same flow ID, they are both dropped. Otherwise, the randomly chosen packet is kept in the buffer (in the same position as before) and the arriving packet is dropped with a probability that depends on the average queu size. The drop probability is computed exactly as in RED. In if they arrive when the average queue size exceeds muxth.

A flow chart of the algorithm is given in Figure 1. In order to bring the queue occupancy back to below muxth as fast as possible, we still compare and drop packets from the queue when the queue size is above the maxth. In general, one can choose m > 1 packets from the buffer, compare all of them with the incoming packet, and drop the packets that have the same flow ID as the incoming packet. Not surprisingly, we shall find that choosing more than one drop candidate packet improves CHOKe's performance. This is especially true when there are multiple unresponsive flows; indeed, as the number of unresponsive flows increases, it is necessary to choose more drop candidate packets. However, since we insist on a completely stateless design, wt cannot a priori know how many unresponsive flows are active at any time (and then choose a suitable value form). It turns out that we can automate the process so that the algorithm chooses the proper value of m 2 1. One way of achieving this is to introduce an intermediate threshold intth which partitions the interval between mznth and muxth into two regions. When the average buffer occupancy is between minth and intth the algorithm can set m =. 1 and when the average buffer occupancy is between intth and muxth it sets m = 2.' More generally, we can introduce multiple thresholds which partition the interval between minth and maxth into k regions R I , Rz, . . . , R k and choose different values of m depending on the region the average buffer occupancy falls in. For example, we could choose m = 2 . i (i = 1 . . . , I C) , when the average queue size lies in region Ri. Obviously, we need to let m increase monotonically with the average queue size.

CHOKe is a truly stateless algorithm. It does not require any special data structure. Compared to a pure FIFO queue, there are just a few simple extra operations that CHOKe needs to perform: draw a packet randomly from $e queue, compare flow IDS, and possibly drop both the incoming and the candidate packets. Since CHOKe is embedded in RED, it inherits the good features of RED mentioned previously. Finally, as a stateless algorithm, it's nearly as simple to implement as RED. To see this, let us consider the details of implementation. Drawing a packet at random can be implemented by generating a random address from which a packet flow ID is read out. Flow ID comparison can be done easily in hardware. It is arguably more difficult to drop a randomly chosen packet since this means removing it from a linked-list. Instead of doing this, we propose to add one extra bit to the packet header.

V. CONCLUSION

They 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] 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.

[2] S.Wen,Y. Fang, andH. Sun, "Differentiated bandwidth allocationwith TCP protection in core routers," IEEE Trans. Parallel Distrib. Syst., vol. 20, no. 1, pp. 34–47, Jan. 2009.

[3] 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.

[4] R. Mahajan and S. Floyd, "Controlling high-bandwidth flows at congestion router," in Proc. 9th Int. Conf. Network Protocols, 2001, pp. 192–201.

[5] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation," in Proc. 19th IEEE INFOCOM, pp. 942–951, 2000

[6] A. Tang, J.Wang, and S. Low, "Understanding CHOKe: Throughput and Spatial Characteristics," in Proc. 23rd Annu. Joint Conf. IEEE Comput. and Commun., pp. 114–124, 2003

[7] 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

[8] 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

[9] 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.

[10] A. Schranzhofer, J. J. Chen, and L. Thiele, "Dynamic power-aware mapping of applications onto heterogeneousMPSoC platforms," IEEE Trans. Ind. Inf., vol. 6, no. 4, pp. 692–707, Nov. 2010.

[11] Wu, Hsien-Ming, and Woei Lin. "On the Server Fairness of Congestion Control in the ISP Edge Router." In Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on, pp. 234-241, 2005

[12] Wang, Chonggang, Jiangchuan Liu, Bo Li, Kazem Sohraby, and Yiwei Thomas Hou. "LRED: a robust and responsive AQM algorithm using packet loss ratio measurement." Parallel and Distributed Systems, IEEE Transactions on 18, no. 1, pp. 29-43, 2007

[13] Tan, Liansheng, Zhongxun Zhu, Cao Yuan, and Wei Zhang. "A novel approach for bandwidth allocation among soft QoS traffic in wireless networks." Transactions on Emerging Telecommunications Technologies , 2012

[14] Bahl, Paramvir, Imrich Chlamtac, and András Faragó. "Resource assignment for integrated services in wireless ATM networks." International Journal of Communication Systems 11, no. 1, 29-41, 1998

[15] Stoica, Ion, Hui Zhang, and T. S. Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority services. Vol. 27, no. 4. ACM, 1997.

[16] Harks, Tobias. "Utility proportional fair bandwidth allocation: An optimization oriented approach." In Quality of Service in Multiservice IP Networks, pp. 61-74. Springer Berlin Heidelberg, 2005.

[17] Huang, Yanan, F. A. N. G. Xuming, and Z. H. A. O. Yue. "An energy saving scheduling scheme for OFDMA two-hop relay systems." IEICE transactions on fundamentals of electronics, communications and computer sciences 93, no. 11, pp. 2320-2327, 2010

[18] Meng, Xiaojing. "An efficient scheduling for diverse QoS requirements in WiMAX." PhD diss., University of Waterloo, 2007.

[19] Todinca, D., H. Graja, P. Perry, and J. Murphy. "Novel admission control algorithm for GPRS/EGPRS based on fuzzy logic", pp. 342-346, 2004

[20] Thing, Vrizlynn LL, P. Shum, and M. K. Rao. "Bandwidth-efficient WDM channel allocation for four-wave mixing-effect minimization." Communications, IEEE Transactions on 52, no. 12, pp. 2184-2189, 2004

[21] Mohit Kumar Saini, R.K. Chuhan , " Performance Evaluation of Resource Allocations in WiMAX Network", GJESR Review Paper Vol. 1 [ISSUE 3] APRIL, 2014

[22] Katabi, Dina, Mark Handley, and Charlie Rohrs. "Congestion control for high bandwidth-delay product networks." ACM SIGCOMM Computer Communication Review 32, no. 4 (2002): 89-102.

[23] Filali, Fethi, and Walid Dabbous. "A simple and scalable fair bandwidth sharing mechanism for multicast flows." In Network Protocols, 2002. Proceedings. 10th IEEE International Conference on, pp. 248-258. IEEE, 2002.

[24] Zhang, Jianyong, Anand Sivasubramaniam, Alma Riska, Qian Wang, and Erik Riedel. An interposed 2-level I/O scheduling framework for performance virtualization. Vol. 33, no. 1. ACM, 2005.