JCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-2, Issue-11 E-ISSN: 2347-2693

Survey on CHOKe AQM Family

Vijith C [1*] and M. Azath [2]

[1*2]*Department of Computer Science and Engineering, Met's School of engineering, Kerala, India*

cjvijith@gmail.com, mailmeazath@gmail.com

www.ijcaonline.org

Received: 24 Oct 2014 Revised: 04 Nov 2014 Accepted: 20 Nov 2014 Published: 30 Nov 2014

Abstract— One of the most important applications in internet is load balancing. For load balancing in IP networks, there are different approaches including Active Queue Management (AQM) which has a proportionate development in research. CHOKe is an AQM method ensures Quality of Service in congested traffic by differentiating responsive flows and unresponsive flows. The survey attempts to study the CHOKe with its descendants and investigates the algorithms based on various congestion metrics and short lived as well as long lived TCP traffic and UDP flows.

Keywords— AQM, CHOKe, IP Networks, Load balancing.

I. Introduction

Networks provide communication between computing devices. An Internet Protocol network (IP network) is a network of computers which communicate using internet protocol (IP) [1]. The Internet is the largest and best known IP network. Load balancing is the main networking solution responsible for distributing incoming traffic and dividing the amount of work between two or more computers so that work can be done faster [2]. The main idea is to map the part of the traffic from the heavily loaded paths to some lightly loaded paths to avoid congestion in the shortest path route and to increase the network utilization and network throughput. One of the main reasons for congestion in traffic which causes load unbalance is short or long lived TCP/UDP flows which may be bursty in nature. Bursty flows are widely used in internet and they generate issue of global synchronization in packet receiving nodes. Congestion leads to high packet loss resulting in large network delays and non-process data transmissions. Queue management in routers plays an important role in congestion avoidance and congestion control.

Various scheduling and queue management algorithms are implemented to avoid congestion. Fair queuing (FQ), which can be construed as a packet approximation of generalized processor sharing (GPS), is a scheduling algorithm used by network schedulers, to allow flows in network to fairly share the link [3]. Weighted fair queuing (WFQ) is a modified FQ allowing different scheduling priorities for data flows [4]. In general, FQ cannot be used for handling different flows bandwidth requirements and WQ loses granularity and they cannot be adjusted. Explicit Congestion Notification (ECN) is introduced in TCP/IP which allows notification of congestion in network without dropping packets [5]. The main problem faced in ECN is that setting a non-compliant TCP connection to indicate it was ECN-capable which leads to loss of ECN messages in the network.

There is another queue management which works adaptively, based on the traffic called Active Queue Management [6]. An AQM system is used to control the length of a queue so that it does not run full, adding its maximum delay under load. Such management also enables TCP/UDP flows to do its job of sharing links properly. Random early detection (RED), also known as random early discard or random early drop is an active queuing discipline for a network scheduler suited for congestion avoidance [7]. RED has several variants like WRED, ARED, RRED, FRED, SRED, DRED, BLUE etc. These methods are good enough for queue management but they possessed drawbacks in modern networks. The main drawbacks are

1) They fail to recognize TCP and UDP flows
2) They failed to handle large busty traffic
3) They suffer from lockout and global synchronization when the parameters are not tuned properly
4) They cannot handle short lived internet packets
5) They never penalize unresponsive flows
6) They do not provide fairness and quality of service in network

To overcome these limitations new active queue management mechanism is implemented called CHOKe. CHOKe stands for Choose and Keep for Responsive Flows, Choose and Kill for Unresponsive Flows. Responsive flows are the information flow that are identical and unique to each other and ready to get serviced in a communication link. Unresponsive flows are other types of flows, mainly the duplicate and similar information flows that will intersect each other. The network is bursty and the majority of flows will be unresponsive and short lived. The flows will be mixture of TCP and UDP flows. So the loads are balanced effectively and the fairness is guaranteed by handling unresponsive flows using CHOKe and its descendants which lead to ensure QoS and enhanced differentiated services.

II. CHOKe

CHOKe, Choose and Keep for Responsive flows, Choose and Kill for Unresponsive flows, an Active Queue Management method, is stateless, controls misbehaving flows with a minimum overhead [9]. It is simple to implement, based on queue length and differentially penalizes unresponsive flows using the information of each flow. It shields responsive flows from unresponsive or aggressive flows and provides Quality of Service (QoS) to all users. CHOKe does not require any special data structure for its operation. CHOKe aims to provide maximum fairness flows that pass through a congested router [10]. It inherits the good feature of RED where the idea behind the CHOKe is that the contents of FIFO buffer form a sufficient statistic about the incoming traffic. CHOKe calculates the average occupancy of the buffer using an exponential moving average called EWMA (Exponential Waited Moving Average). On the arrival of a packet at congested router, CHOKe randomly draws a packet from buffer and compares arriving and drawing packets. If they both have same flow information then the both is dropped, else the randomly chosen packet is placed back and the arriving packet is dropped or admitted with a probability which is computed exactly as in RED. The reason is that the FIFO buffer more likely to have packets from a misbehaving flow, thus they have greater chances to be chosen for comparison. Further, CHOKe assumes packets belonging to misbehaving flows implicitly have same statistical characteristics and QoS requirements. So it can easily differentiate responsive and unresponsive flows and manages buffers without per-flow state information. As a result packets of misbehaving flows are dropped more than packets of well-behaved flows. Algorithm and flowchart for CHOKe is given fig.1 and fig.2 respectively. CHOKe marks two thresholds on the buffer, a minimum threshold min^{th} and max^{th}. If the average queue size is less than min^{th}, every arriving packet is queued into the FIFO buffer [8]. If the average queue size is lesser than min^{th} then the packets are not dropped and arriving packet is allowed to enter in buffer. If the average queue size exceeds max^{th}, then every arriving packet is immediately dropped. This moves the queue occupancy back to below max^{th}.

```
For every packet arrival {
    Calculate $Q_{avg}$
    if $Q_{avg} < min^{th}$ {
        Admit new packet
    }
    else {
        Draw a packet randomly from the queue
        if Both packets have same flow {
            Drop both packets
        }
        else {
            if $Q_{avg} < max^{th}$ {
                Admit packet with probability p
            }
            else {
                Drop the new packet
            }
        }
    }
}
```

Fig.1 Algorithm for CHOKe

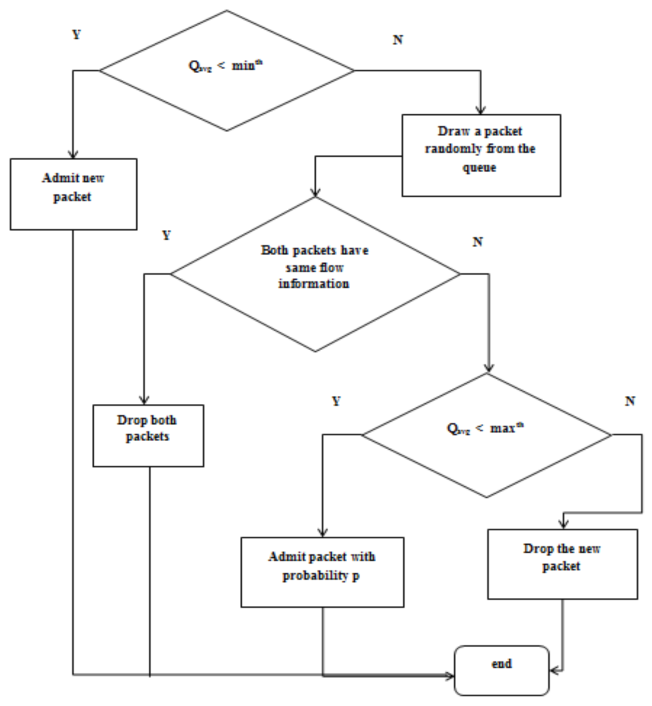
Y
N
$Q_{avg} < min^{th}$
Draw a packet randomly from the queue
Admit new packet
Y
Both packets have same flow information
N
Drop both packets
Y
$Q_{avg} < max^{th}$
N
Admit packet with probability p
Drop the new packet
end

Fig.2 Flowchart for CHOKe

When the average queue size is between min^{th} and max^{th}, each arriving packet is compared with a randomly chosen packet (drop candidate packet), from the FIFO buffer. If they have the same flow ID, they are both dropped. Otherwise, the randomly chosen packet is left intact in the buffer and the arriving packet is dropped with a probability p that depends on the average queue size. The drop probability is computed as in RED. In particular, this means that packets are dropped with probability 1 if they arrive when the average queue size exceeds max^{th}. In order to bring the queue occupancy back to below max^{th} when the buffer occupancy exceeds max^{th}, CHOKe repeatedly compare and drop packets from the queue. CHOKe works effectively in TCP flows. It could provide better QoS in a bursty TCP traffic. The differentiation and dropping policy of CHOKe makes global synchronization in bursty flows. Busty flows are characterized based on their flow information and they may be allowed to occupy buffer if the buffer has enough space to accommodate all connected bursty packets otherwise they are dropped even without holding a single packet from the bursty flow. So CHOKe gives same performance level for individual and aggregate TCP flows. When the UDP flows are present CHOKe acts similar to TCP flows. In normal UDP rate under bandwidth capacity CHOKe performs similar to RED. When UDP rate supersedes the link bandwidth, CHOKe could penalize the UDP flow to make average queue size around minth, and enable TCP flows to get better bandwidth share. There are a growing number of UDP based applications running in the Internet, such as packet voice and packet video. Increasing number of UDP flows increase the percentage of UDP packets in the heterogeneous network, which will lead to high dropping probability for TCP flows since the buffers are almost occupied by the orphan UDP flows. The differentiation in UDP flows is not that much effective compared to TCP traffic. This leads CHOKe to face similar dropping problems as seen in RED. Also CHOKe doesn't show any characterization for short lived and long lived traffic. There are several variations on basic CHOKe scheme where basic one is the Original CHOKe, in which the drop candidate packet is chosen randomly from the queue. [9][10] The descendants of CHOKe are listed below and they are summarized in table 1.

A. Front CHOKe

The drop candidate is always the packet at the head of the queue.

B. Back CHOKe

The drop candidate is always the packet at the tail of the queue.

C. Multi-drop CHOKe (M-CHOKe)

m 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 scheme 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. GCHOKe

gCHOKe (geometric CHOKe), is another method which provides an advanced flow protection which is realized by introducing an extra flow matching trial upon each

Method	Pros	Cons
CHOKe	Differentiation between flows into responsive and unresponsive	Single trail and vulnerable by bursty flows
M-CHOKe	Multiple trails for matching	Global synchronization and lockout problems
XCHOKe/RECHOKe/CSa-XCHOKe	Uses historical data for flow protection	Lacks stateless property
SAC	Provides simplicity and lower processing cost	Does not provide support for multiple priority levels
A-CHOKe	Protects well-adaptive flows from non-adaptive flows	Fluctuation in heavy loads
P-CHOKe	It provides better packet delivery ratio and low queue delay.	Fair bandwidth allocation
GCHOKe	Extra shield of protection by using bonus trial on successful matching	Does not work well in long lived and short lived TCP traffic
FAVQCHOKe	Uses virtual queue to guaranty more security for internal queue	Imperfection in differential service since load factor based
CHOKeW	Supports Differentiated Services	Cannot provide the assured bandwidth allocation for different priority flow
CHOKeR.	Uses MISD scheme	No sustainability in heterogeneous real time network

Table.1 Pros and Cons of CHOKe family

successful matching of packets [15]. The difference between CHOKe and gCHOKe is the number of trials that they use to differentiate. CHOKe punishes 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 bonus trial. The succession of bonus trials provides an extra shield of protection to rate adaptive flows from unresponsive ones [7]. By tuning the defined maximum number of trials, a desired protection level may be achieved. This makes traffic control more tractable, which is lacking in the original plain CHOKe where flow protection is flat.

K. FAVQCHOKe

Flow based Adaptive Virtual Queue CHOKe (FAVQCHOKe) uses both queue dimensions and load based factors for tuning CHOKe parameters [16]. It is a congestion avoidance scheme which utilizes the implementation of virtual queues in packet receiving nodes. The actual buffers are updated based on the virtual queue status. It helps to protect internal buffers before they get vulnerable by malicious unresponsive flows.

L. CHOKeW

CHOKeW uses "matched drops" created by CHOKe to control the bandwidth allocation, but excludes the RED module for bandwidth differentiation and TCP protection which is important for implementing Quality of Services (QoS) [17]. In DiffServ networks where flows have different priority, CHOKe fails to support all levels .CHOKeW does not need per flow state and it supports multiple bandwidth priority levels by giving high priority flows with high throughput. CHOKeW avoids flow starvation. 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 shares to flows with higher priority, maintaining good fairness among flows with the same priority and protecting TCP against high speed unresponsive flows when network congestion occurred.

J. CHOKeR

CHOKeR is advancement to CHOKeW algorithm which overrides the problems of bandwidth differentiation in multiple priority levels and poor performance on bursty traffic in large congested network which are experienced by CHOKeW [18]. CHOKeR does not maintain per flow state information and it uses MISD (Multi step Increase and Single step Decrease) model for congestion avoidance.

CHOKe family haven't yet proved better reliability and stability in modern social IP networks because they fail to recognize and isolate link prediction problems raised during the on-need formation of dynamic links [20].

III. CONCLUSION

This survey presents CHOKe from its formal beginnings to gradual improvements. The stateless nature and queue length with flow information based mechanism of CHOKe family enhance adaptive queue management in the IP network. It could be concluded that CHOKe family ensure fairness in the network through differentiation of flows into responsive and unresponsive flows. Although CHOKe and its descendants are good solutions for congestion control in the network, they fail to provide a stable mechanism for load balancing when the heterogeneous network is dealing with long and short lived TCP traffic and large UDP flows.

References

[1] en.wikipedia.org/wiki/IP_address.

[2] searchnetworking.techtarget.com/definition/load-balancing.

[3] en.wikipedia.org/wiki/Fair_queuing.

[4] en.wikipedia.org/wiki/Weighted_fair_queueing.

[5] en.wikipedia.org/wiki/Explicit_Congestion_Notification.

[6] gettys.wordpress.com/active-queue-management-aqm-faq.

[7] B. Kiruthiga and Dr. E. George Dharma Prakash Raj, "Survey on AQM Congestion Control Algorithms", IJCSMC, Vol. 2, Issue. 2, pp.38–44, Feb 2014.

[8] G.F.Ali Ahammed, Reshma Banu, "Analyzing the Performance of Active Queue Management Algorithms", IJCNC, Vol. 2, pp. 19, Mar 2010.

[9] Rong Pan, Balaji Prabhakar, Konstantinos Psounis, "CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation", INFOCOM 2000, vol.2, pp. 942-951, Mar 2000.

[10] Ao Tang, Jiantao Wang and Steven H. Low, "Understanding CHOKe: Throughput and Spatial Characteristics", IEEE/ACM Trans. Networking, vol. 12, No. 4, pp. 694-707, Aug 2004.

[11] Jiang Ming, WU Chumming, Zhang Min and Bian Hao, "CSa-XCHOKe: A Congestion Adaptive CHOKe Algorithm", Chinese Journal of Electronics, Vol.19, No.4, Oct 2010.

[12] Ying Jiang, and Jing Liu, "Self adjustable CHOKe: an active queue management algorithm for congestion control and fair bandwidth allocation", IEEE computers and comm., Vol.2, No.4, pp. 1018-1024, Jul 2013.

[13] K.Chitra and Dr. G.Padamavathi, "Adaptive CHOKe: An algorithm to increase the fairness in Internet Routers", IJANA, vol. 01, Issue. 06, pp. 382-386, Apr 2010.

[14] G. Sasikala and E. George Dharma Prakash Raj, "P-CHOKe: A Piggybacking-CHOKe AQM Congestion Control Method", IJCSMC, Vol. 2, Issue. 8, pp.136–144, Aug 2013.

[15] Addisu Eshete and Yuming Jiang, "Protection from Unresponsive Flows with Geometric CHOKe", Centre for Quantifiable Quality of Service in Communication Systems, Feb 2012.

[16] K.Chitra and Dr.G.Padmavathi, "FAVQCHOKE: To Allocate Fair Buffer To A Dynamically Varying Traffic In An Ip Network", IJDPS, Vol. 2, Issue. 1, pp.73–82, Jan 2011.

[17] Shushan Wen, Yuguang Fang and Hairong Sun, "CHOKeW: Bandwidth Differentiation and TCP Protection in Core Networks", IEEE Trans. Parallel and Distributed Sys. , Vol. 20, NO. 1, pp. 34-47, Jan 2009.

[18] Lingyun Lu, Haifeng Du and Ren Ping Liu, "CHOKeR: A Novel AQM Algorithm with Proportional Bandwidth Allocation and TCP Protection", IEEE Trans. Industrail Informatics, Vol. 10, No. 1, pp.637–644, Feb 2014.

[19] Addisu Eshete and Yuming Jiang, "Generalizing the CHOKe Flow Protection", Preprint submitted to Computer Networks, pp.1–28, Feb 2012.

[20] Shalki Chahar, "Social Networking Analysis", International Journal of Computer Sciences and Engineering, Vol. 02, No. 5, pp.159–163, May 2014.

AUTHORS PROFILE

Vijith C has completed B Tech in CSE from Sahrdaya College of Engineering and Technology, Thrissur, Kerala, in 2012. Presently he is pursuing his M Tech in CSE from Met's School of engineering, Thrissur, Kerala. His research interests include Networking, Computer Vision and Algorithms.

Dr. M. Azath is Head of Department of Computer Science and engineering, Met's School Of Engineering, Mala. He has received Ph.D. in Computer Science and Engineering from Anna University in 2011. He is a member in Editorial board of various international and national journals and also a member of the Computer society of India, Salem. His research interests include Networking, Wireless networks, Mobile Computing and Network Security.