Comparison of CHOKe and gCHOKe Active Queues Management Algorithms with the Use of Fluid Flow Approximation

Adam Domański[1], Joanna Domańska[2], and Tadeusz Czachórski[2]

[1] Institute of Informatics, Silesian Technical University
Akademicka 16, 44-100 Gliwice, Poland
adamd@polsl.pl

[2] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences
Baltycka 5, 44-100 Gliwice, Poland
{joanna,tadek}@iitis.gliwice.pl

Abstract. In the article we examine a model of TCP connection with Active Queue Management in an intermediate IP router. We model a system where CHOKe or gCHOKe are the AQM policy. We use the fluid flow approximation technique to model the interactions between the set of TCP/UDP flows and two variants of the CHOKe algoithms. The obtained results confirm the superiority of these algorithms over a standard RED algorithm.

Keywords: active queue management, TCP flow control, RED.

1 Introduction

Congestion control mechanisms in TCP/IP networks are one of the most important topics in the field of today Internet and their modeling remains a vital problem. The development of new AQM (Active Queue Management) routers allow to improve the performance of Internet applications.

In recent years a number of analytical models of AQM in IP routers was presented, e.g. [1–5] in open-loop scenario, because of the difficulty in analyzing AQM mathematically inside the whole closed loop of TCP congestion control. This paper extends a nonlinear dynamic model of TCP proposed earlier [6, 7] to analyze the AQM systems with RED. Here, we use a similar model to investigate the performance of CHOKe or gCHOKe mechanisms.

The models based on fluid flow approximation, e.g. [8], are able to capture the dynamics of TCP flows [9] and allow to analyze networks with a large number of flows. The article describes the use of this method to compare routers having different active queue management principles (classical RED, CHOKe and gCHOKe) and transmitting TCP/UDP flows. The model allows to study not only the steady-state behavior of the network, but also the transient one when a set of TCP flows start or finish transmission. We focus on transient average router queue length for different AQM strategies.

The rest of this article is organized as follows. The Section 2 introduces two variants of CHOKe algorithm. The Section 3 describes the fluid flow model of AQM router supporting TCP/UDP flows. The Section 4 presents the obtained results. The conclusions are presented in Sect. 5.

2 The CHOKe Algorithm and Its Variant – gCHOKe

The CHOKe (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows), [10] is a stateless AQM algorithm slightly similar to RED (Random Early Drop), proposed not only to control TCP packets but also to prevent uncontrollable UDP connections to monopolize the links [11]. It uses incoming packages to punish streams with the highest demand for bandwidth.

Similarly to the RED mechanism, there are two threshold values: Min_{th} and Max_{th}. At the arrival of a new package, the new walking average queue length is calculated. If the average queue length is less than Min_{th}, the packet is placed in the buffer. When the average queue length is greater than Min_{th}, CHOKe pulls randomly one packet from the FIFO buffer a ("CHOKe victim") and verifies whether it belongs to the same stream as an incoming packet.

If the both packets belong to the same stream, they are removed (this situation is called "CHOKe hit"). Otherwise, the randomly selected packet is returned to the buffer and the arrived packet is placed into the queue with probability P. This probability is calculated in the same manner as in the case of RED algorithm. The event is called "CHOKe miss".

The geometric CHOKe algorithm (gCHOKe) [12] is a modification of the CHOKe having an additional, configurable parameter $maxcomp \in [1, \ldots, \infty)$. This parameter determines the maximum number of successful comparisons. As previously, the algorithm compares the incoming packet with a random packet drawn from the queue. The comparison is successful when both packages are from the same stream. The comparison ends when: the comparison is unsuccesfull or the number of comparisons exceeds $maxcomp$. In this case all matching packets (plus the incoming ones) are removed from the queue. If the first match is not successful, the random packet ("CHOKE victim") comes back to the queue. The arriving packet is placed into the queue with probability of P (see Fig. 1). Hence, CHOKe is a special case of gCHOKe algorithm ($maxcomp = 1$).

3 The Fluid-Flow Model of TCP/UDP Streams

This section presents a fluid flow model of a TCP connection having a bottleneck router with AQM policy. The router transmits also UDP packets, [8]. This model ignores the TCP timeout mechanisms and allows to obtain the average value of key network variables. This model is based on the following nonlinear differential equations:

$$\frac{dW(t)}{dt} = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))} p(t-R(t)) \tag{1}$$

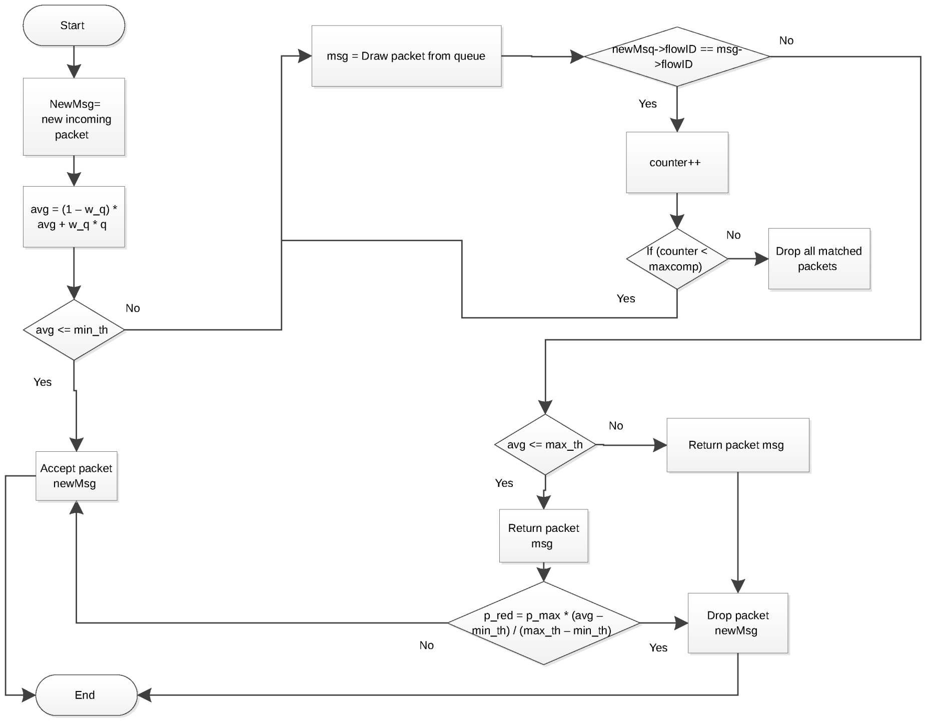

Fig. 1. Diagram of the gCHOKe algorithm

$$\frac{dq(t)}{dt} = \frac{W(t)}{R(t)} N(t) - C \qquad (2)$$

where:

W – expected TCP congestion window size (packets),
q – expected queue length (packets),
R – round-trip time $q/C + T_{\mathrm{p}}$ [s],
C – link capacity [packets/s],
T_{p} – propagation delay [s],
N – number of TCP sessions at the router,
p – packet drop probability.

The maximum values of q and W depend on the buffer capacity and maximum window size. The dropping probability p depends on the AQM queue algorithm. The first term on the right side of the Equation (1) represents the rate of increase of congestion window due to incoming acknowledgements, the second represents the rage with which the congestion window decreases due to packet losses. The Equation (2) gives the speed of router queue changes due to incoming and leaving flows of packets.

The traffic composed of TCP and UDP streams has been considered in [13]. For this model a single router supports N TCP sessions. Each TCP stream is a TCP-Reno connection and each UDP sender is a CBR (Constant Bit Rate) source. The total rate of UDP sessions is denoted by λ, it is assumed that the rate λ is associated at equal intensity λ/N with each TCP connection. Fluid-flow equations of TCP and UDP mixed traffic become:

$$\frac{dW(t)}{dt} = \frac{1}{R'(t)} - \frac{W(t)W(t-R'(t))}{2R'(t-R'(t))}p(t-R'(t)) \tag{3}$$

$$\frac{dq(t)}{dt} = \frac{W(t)}{R'(t)}N(t) - (C-\lambda) \tag{4}$$

where R' = round-trip time $= q/(C-\lambda) + T_\mathrm{p}$ [s].

In RED AQM mechanism, at arrival of each packet, the average queue size x is calculated as an exponentially weighted moving average using the following formula: $x_i = (1-\alpha)x_{i-1} + \alpha q_\mathrm{inst}$ where q_inst is the current queue length. Then the RED drop function is applied: there are two thresholds Min_th and Max_th; if $x < Min_\mathrm{th}$ the packet is admitted, for $Min_\mathrm{th} < x < Max_\mathrm{th}$ the packet is dropped with probability p_RED growing linearly from 0 to p_max

$$p_\mathrm{RED} = p_\mathrm{max}\frac{x - Min_\mathrm{th}}{Max_\mathrm{th} - Min_\mathrm{th}} \tag{5}$$

and if $x > Max_\mathrm{th}$ the packet is dropped.

The CHOKe algoritm pulls from the FIFO buffer "CHOKe victim" and if the package is from the same stream as an incoming packet, we drop it. So prabability p_CHOKe depends on the number of packets of a stream i relative to the total buffer occupancy, $p_\mathrm{CHOKe} = q_i/q$. Probability p_CHOKE depends also on the buffer occupancy and the number of streams. For simplicity, the model assumes that the number of packets belonging to a single stream in the queue is the same for all streams, $q_i = q/N$. It follows that the probability of packet loss is inversely proportional to the number of the streams, $p_\mathrm{CHOKe} = 1/N$.

Geometric CHOKE repeats the selection of "CHOKe victim" until the packet has been drown from another stream or the number of draws exceeds *maxcomp*. For each incoming packet we can drop $1, 2, 3, \ldots, maxcomp$ packets. If we assume that the probability of selecting a package of the same stream in one selection is $\frac{1}{N}$ then

$$p_\mathrm{gCHOKe} = \frac{1}{N} + \frac{1}{N^2} + \cdots + \frac{1}{N^n} = \sum_{k=1}^{n}\frac{1}{N^k} \tag{6}$$

where $n = maxcomp$. Using the sum of a geometric sequence:

$$S_n = a + aq + \cdots + aq^{n-1} = a\frac{1-q^n}{1-q} \tag{7}$$

we can write:

$$p_\mathrm{gCHOKe} = S_n - a + q^k \ . \tag{8}$$

In our case $a = 1$ and $q = \frac{1}{N}$, hence

$$p_\mathrm{gCHOKe} = \frac{1-\frac{1}{n^k}}{1-\frac{1}{n}} - 1 + \frac{1}{n^k} = \frac{n^k + n^{k-1} - n^{-1} - 1}{n^k - N^{k-1}} \ . \tag{9}$$

4 Numerical Results

Computations were made with the use of PyLab (Python numeric computation environment) [14] which is a combination of Python, NumPy, SciPy, Matplotlib, and IPython. The graphs shown below present transient system behavior, the time axis is drawn in seconds.

We assume the following parameters of the AQM buffer: $Min_{\mathrm{th}} = 10$, $Max_{\mathrm{th}} = 15$, buffer size (measured in packets) = 20, weight parameter $\alpha = 0.007$, and the parameters of TCP connection:

- transmission capacity of AQM router: $C = 0.075$,
- propagation delay for i-th flow: $T_{\mathrm{p}_i} = 2$,
- initial congestion window size for i-th flow (measured in packets): $W_i = 1$.

The obtained mean queue lengths for TCP connections and various AQM policies are presented in Table 1.

Table 1. The obtained mean AQM queue lengths Q

Algorithm	Nb of streams	Nb of packets
CHOKE	1	7.98664081264
CHOKE	2	8.63146812018
CHOKE	5	10.0998514529
CHOKE	10	11.0167546717
CHOKE	11	11.7731309893
RED	1	8.57089136683
RED	2	9.05376778822
RED	5	10.3805817389
RED	10	11.1549893996
RED	11	11.7731309893
gCHOKe ($maxcomp = 2$)	1	7.88714079539
gCHOKe ($maxcomp = 5$)	1	7.81053339265
gCHOKe ($maxcomp = 10$)	1	7.78926315534
gCHOKe ($maxcomp = 2$)	2	8.61223411769
gCHOKe ($maxcomp = 5$)	2	8.60737680727
gCHOKe ($maxcomp = 10$)	2	8.61023081937
gCHOKe ($maxcomp = 2$)	10	11.0059435834
gCHOKe ($maxcomp = 5$)	10	11.0108325703
gCHOKe ($maxcomp = 10$)	10	11.0108321896

Figures 2, 3, 4, present the queue behavior in the case of two flows and respectively RED, CHOKe and gCHOKe queues. The size of congestion window increases until the buffer reaches the Min_{th} value. Algorithm drows "CHOKe victim" and the probability of removing the package is equal to $\frac{1}{2}$ (probability of removing the packet by RED mechanism is much smaller). Packets are dropped and the size of congestion window decreases causing a slow decrease of the queue length – this pattern is repeated periodically.

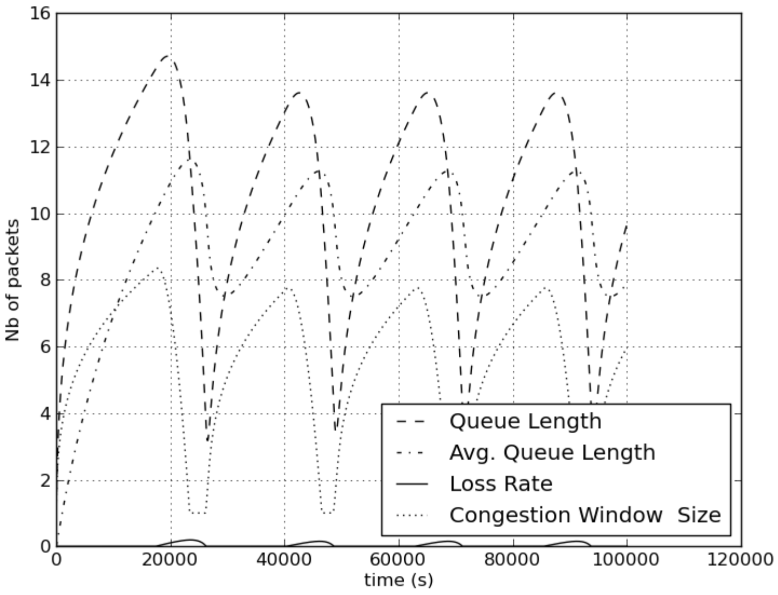

Fig. 2. RED queue, 2 TCP/UDP flows

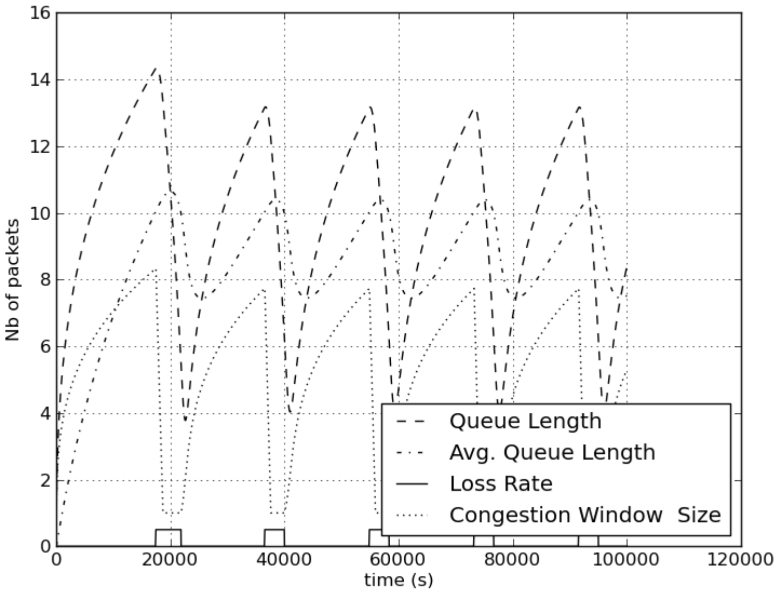

Fig. 3. CHOKe queue, 2 TCP/UDP flows

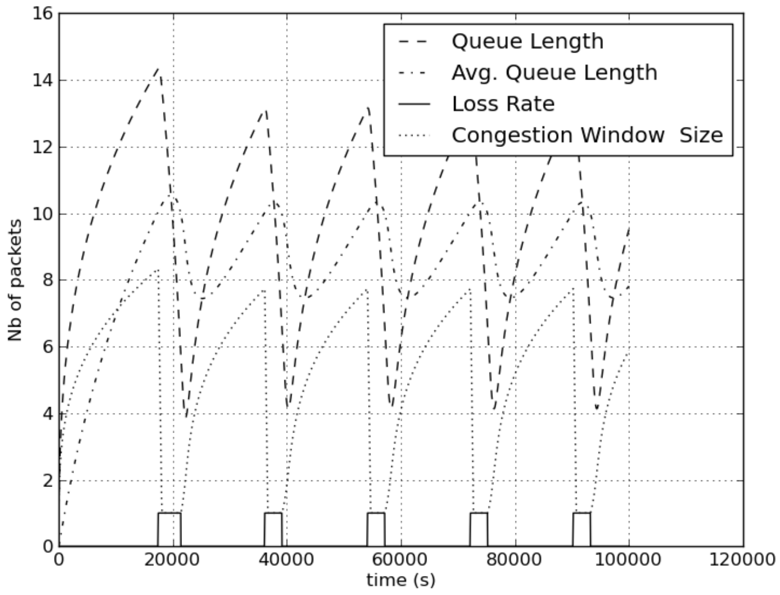

Fig. 4. gCHOKe queue ($maxcomp = 10$), 2 TCP/UDP flows

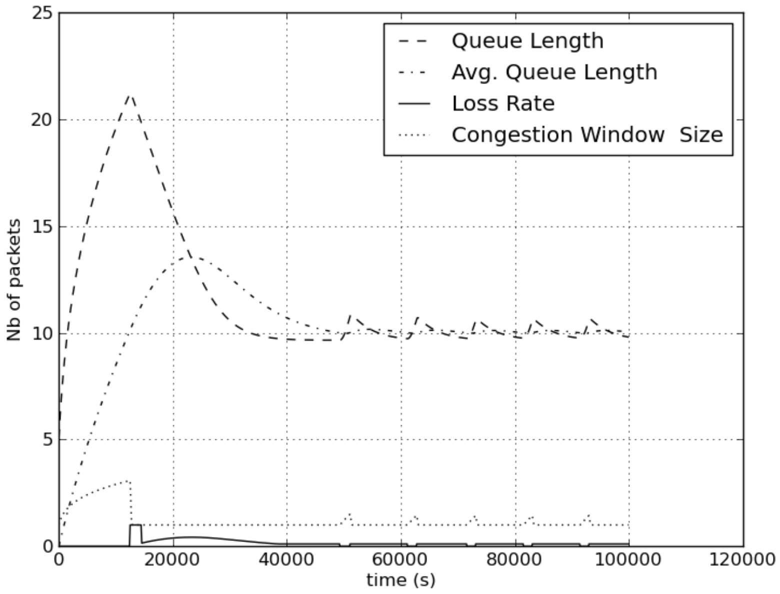

Fig. 5. gCHOKe queue ($maxcomp = 10$), 10 TCP/UDP flows

Comparing the behavior of the CHOKe algorithm with the RED algorithm one can see that the CHOKe algorithm works better in the case of aggressive (stealing most of the bandwidth) streams. When the number of streams grows, the importance of the choke algorithm decreases. The probability of selecting a good victim decreases and the packets are removed by the RED mechanism. Comparing the results shown in Table 1 one can see that the differences between the obtained average queue length for CHOKe and for RED algorithms decreases when the number of streams increases.

In our tests of gCHOKe algorithm we assumed that the maximum number of draws can not exceed the Min_{th} parameter. Figures 3 and 4 show that compared to CHOKe algorithm, gCHOKe brings a slight improvement. The average buffer occupancy is also slightly reduced (Table 1). When the number of streams increases, the influence of algorithm becomes invisible (Fig. 5).

5 Conclusions

This article confirms the advantage of CHOKe algorithm over standard RED for aggressive streams. The use of the choke algorithm is insignificant in the case of a large number of streams with the similar intensity. It confirms also the advantage of the algorithm in presence of mixed TCP/UDP traffic. Unfortunately, weaknesses of the model (UDP data closely associated with the TCP streams) could not allow us to show the advantages of the gCHOKe algorithm in shaping the intensified traffic of UDP datagrams. Our future work will concern this issue.

Acknowledgements. This research was partially financed by Polish Ministry of Science and Higher Education project no. N N516479640.

References

1. Liu, C., Jain, R.: Improving explicit congestion notification with the mark-front strategy. Computer Networks 35(2-3) (2000)
2. Domańska, J., Domański, A., Czachórski, T.: The Drop-From-Front Strategy in AQM. In: Koucheryavy, Y., Harju, J., Sayenko, A. (eds.) NEW2AN 2007. LNCS, vol. 4712, pp. 61–72. Springer, Heidelberg (2007)
3. Augustyn, D.R., Domański, A., Domańska, J.: Active Queue Management with non linear packet dropping function. In: 6th International Conference on Performance Modelling and Evaluation of Heterogeneous Networks HET-NETs (2010)
4. Augustyn, D.R., Domański, A., Domańska, J.: A Choice of Optimal Packet Dropping Function for Active Queue Management. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2010. CCIS, vol. 79, pp. 199–206. Springer, Heidelberg (2010)
5. Domańska, J., Domański, A., Czachórski, T.: Implementation of modified AQM mechanisms in IP routers. Journal of Communications Software and Systems 4(1) (March 2008)
6. Hollot, C.V., Misra, V., Towsley, D., Gong, W.-B.: On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In: IEEE INFOCOM (2002)

7. Rahme, S., Labit, Y., Gouaisbaut, F.: An unknown input sliding observer for anomaly detection in TCP/IP networks. In: Ultra Modern Telecommunications & Workshops (2009)
8. Misra, V., Gong, W.-B., Towsley, D.: Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In: ACM SIGCOMM (2000)
9. Yung, T.K., Martin, J., Takai, M., Bagrodia, R.: Integration of fluid-based analytical model with Packet-Level Simulation for Analysis of Computer Networks. In: SPIE (2001)
10. Pan, R., Prabhakar, B., Psounis, K.: CHOKe, A stateless AQM scheme for approximating fair bandwidth allocation. IEEE INFOCOM, 942–952 (2000)
11. Hollot, C.V., Misra, V., Towsley, D.: A control theoretic analysis of RED. IEEE/INFOCOM (2001)
12. Eshete, A., Jiang, Y.: Generalizing the CHOKe flow protection. Computer Network Journal (2012)
13. Wang, L., Li, Z., Chen, Y.-P., Xue, K.: Fluid-based stability analysis of mixed TCP and UDP traffic under RED. In: 10th IEEE International Conference on Engineering of Complex Computer Systems (2005)
14. `www.scipy.org`