Active Queue Management in Blind Access Networks

Ronit Nossenson and Hagit Maryuma
Faculty of Computer Science
Jerusalem College of Technology
Jerusalem, Israel
nossenso@jct.ac.il, hagit.maryuma@gmail.com

***Abstract*— Per-flow AQM schemes are designed to provide fair resource management between flows and limit the negative effects of non-adaptive UDP-based flows. To overcome the flow bandwidth limitation, P2P and file sharing applications usually open multiple simultaneous TCP based flows. Although each TCP-based flow is adaptive and responds to network congestion, the aggregated demand of such application consumes network resources and dominates the queue space. In this article we suggest a new per user AQM policy designed to remedy this problem. Since user traffic in access networks is usually carried in user tunnels such as PPP or GTP, an access network device cannot identify the underlying applications and TCP/UDP flows. The new per-user AQM policy presented here operates in the dark and only uses tunnel aggregated statistics. Simulation results show that it significantly reduces the bias effects of heavy, demanding applications.**

Keywords-Active Queue Management; Resource Management; Access Networks.

I. INTRODUCTION

Peer-to-Peer (P2P), file sharing and streaming applications attract millions of users every day [8]. This has led to a deterioration in the quality of service perceived by Internet users as well as frequent network collapse [10]. Active Queue Management (AQM) is a well known network device-based form of congestion control where the network device notifies end-systems of incipient congestion. This notification consists either of dropping a packet from the queue or "marking" a packet [15]. All AQM are designed to detect an impending queue buildup and to notify the sources before the queue overflows. AQM designs fall into three categories: per packet AQM policies (e.g., RED [7], BLUE [6], REM [3]), per class of service AQM policies (e.g., Cisco's WRED [5], [9]) and per flow AQM policies (e.g., FRED [11], [2]). A well known drawback of per packet AQM and per class of service AQM policies is lock-out and bias effects from the few flows that dominate the queue space [11]. The latter per flow AQM category is designed to avoid this problem. However, a common method used by P2P applications to overcome per flow bandwidth limitations is to open multiple simultaneous flows between the peer and the content sources. Hence, a new category of AQM is now needed to provide fair resource management between users – a *per user* AQM policy.

The user traffic in access networks is usually carried in user tunnels such as the Point-to-Point Protocol (PPP) [16], the GPRS Tunneling Protocol (GTP) [17] or Radio Link Control (RLC) encryption [17]. In tunneled traffic, the access network device cannot identify the underlying applications or the underlying TCP/UDP connections and cannot analyze any application header, TCP/UDP header or IP header. A network device located inside the access network operates in complete darkness: it can only identify the tunnel (from the packet header) and can only use tunnel aggregated statistics.

In this study, we propose and evaluate a new *per user* AQM policy designed for *blind* network optimization. The new scheme, dubbed the User Random Early Drop (URED), enforces fair resource allocation among users (tunnels). It complements the admission control mechanisms of the core network such as the Policy and Charging Rules Function (PCRF) server. The proposed algorithm is a modification of the Flow Random Early Drop (FRED) algorithm [11] together with the drop function suggested in [1], adjusted for the requirements of blind access networks and per tunnel consideration.

The algorithm's performance is compared to Drop Tail, RED and FRED. The simulation shows that the new algorithm handles P2P traffic better than the other algorithms and the negative effect of heavy demanding applications is significantly curtailed. Hence, the bandwidth over a bottlenecked link is fairly divided between the P2P and the regular users. URED reduces the average throughput of the P2P user from more than 6.8 times that of a regular user with the DT algorithm to less than 1.5, which is considerably better than the RED (4.51) and FRED (3.76) algorithms.

This paper is organized as follows. In the next section, related works are outlined. We describe our new URED algorithm in section III. In section IV, we present the simulation setup and results. Finally, further research directions are discussed in section V.

II. RELATED WORK

One of the major drawbacks of per packet AQM policies and per class of service AQM policies is the lock-out and bias effects generated by non-adaptive flows that dominate the queue space [11].

To reduce the cost of maintaining flow state information, Stoica et al. [14] proposed a scheduling algorithm called Core Stateless Fair Queueing (CSFQ). A similar method, called Rainbow Fair Queueing, was proposed in [4]. In these methods, routers are divided into two categories: edge routers and core routers. An edge router maintains per flow state information and estimates each flow's arrival rate. These estimates are inserted into the packet headers and

 ISBN: 978-1-61208-205-9

passed on to the core routers. A core router simply maintains a stateless FIFO queue and during periods of congestion, drops a packet randomly based on the rate estimates. These schemes reduce the core router's design complexity. The edge router's design nevertheless remains complicated. Furthermore, because of the rate information in the header, core routers have to extract packet information differently from traditional routers. This solution is not satisfactory for access networks since this device categorization is not feasible.

The CHOKe (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows) [13] algorithm aims to approximate max-min fairness for the flows that pass through a congested router. The basic idea behind the CHOKe is that the contents of the FIFO buffer form a "sufficient statistic" about the incoming traffic and can be used in a simple fashion to penalize misbehaving flows. When a packet arrives at a congested router, CHOKe draws a packet at random from the FIFO buffer and compares it with the arriving packet. If they both belong to the same flow, they are both dropped, or the randomly chosen packet is left intact and the arriving packet is admitted into the buffer with a probability that depends on the level of congestion. Similar, a new promising method called FavourQueue aims to improve delay transfer of short lived TCP flows over a best-effort network [2]. When a packet arrives, a check is done on the whole queue to find another packet from the same flow. If no other packet is found, it becomes a favored packet and a drop protection is provided when the queue is full via a push-out scheme that drops a standard packet from the queue in order to insert a favored packet into it. These solutions are not applicable to our problem since they are basically theoretical – commercial traffic managers can drop packets only upon packet arrival and cannot drop packets from the queue.

The FRED [11] is a modified version of RED [7]. FRED uses per-active-flow accounting to impose a loss rate on each flow that depends on the flow's shared buffer use. It provides better protection for adaptive (fragile and robust) flows. In addition, FRED is able to isolate and manage non-adaptive greedy traffic. A FRED gateway maintains state only for flows for which it has packets buffered, not for all flows that traverse the gateway.

These algorithms are not completely suitable for this specific problem, although FRED comes the closest. A modification of the FRED algorithm to operate in a per user tunnel mode instead of per flow mode initially appeared promising. However, we identified several problems in the FRED algorithm that need to be addressed in blind access networks:

- In FRED, there is a requirement for a minimum guarantee of two-four packets space per active flow. It is not clear how to support this requirement in a blind access network with an unknown number of active flows. Furthermore, providing a minimum guaranteed two-four packet space per *active user* (that is, "translating" this requirement into a "per user" format) is unrealistic in a switch located high in the network hierarchy, since such a device has to handle thousands of simultaneously active user tunnels.
- The condition for identifying the non-adaptive flows should be change to a proper condition for identifying non-adaptive users. A user with multiple TCP connections can be considered non-adaptive although each of its TCP flow is adaptive.
- The actions on traffic of non-adaptive users should be less drastic if the device is not congested.
- The drop probability should increase more "smoothly" as suggested in [1]. Dropping with probability 1 results in low utilization and should be implemented only when the device is highly congested.

Below, we suggest a new algorithm inspired by FRED with modifications to handle these problems.

III. THE URED ALGORITHM FOR PER USER AQM

In this section, we present our URED algorithm for per user AQM. The URED algorithm holds a state for every active user that has packets in the queue. The state includes the following local variables (i) the user tunnel ID, as it appears in every packet header t_i; (ii) the number of packets from this tunnel in the queue q_i; and (iii) the average number of packets from this tunnel avg_i. In addition the algorithm uses the following global variables: (i) the number of active users, *Nactive*; (ii) the number of packets in the queue, *q*; and (iii) the average number of packets in the queue, *avg*.

Similar to many other AQM algorithms, queue buildup detection is based on the relationship between the average queue and two global static parameters $Gmin_{th}$ and $Gmax_{th}$. However, the URED algorithm is unique in that it considers two layers in its drop probability calculation: a *universal* layer and an *individual* layer. In each layer, a drop probability is calculated according to the Hazard function suggested in [1], denoted by P_u (universal) and P_i (individual). Obviously, as the average queue increases and a specific user tunnel is increasingly responsible for this queue buildup, the drop probability of packets from this tunnel increases. To evaluate the responsibility of a specific user tunnel in the queue buildup, the URED algorithm uses two dynamic parameters $Lmin_{th}$ and $Lmax_{th}$. These dynamic parameters are calculated on the fly and estimate the virtual individual queue size with respect to the number of active users. That is, they mark the recommended limits of space that an individual tunnel can consume at a specific moment, given the number of active tunnels at this moment. Specifically, $Lmin_{th}$ and $Lmax_{th}$ are calculated as follows: $Lmin_{th}$ = min{BufferSize/Nactive, BufferSize/3} and $Lmax_{th}$ = min{2·BufferSize/Nactive, BufferSize/2}. A user is considered non-adaptive if the global average queue size is large and its share (represented by its individual average queue size) is large compared to the current per user available space. The description of the URED algorithm is given below.

 ISBN: 978-1-61208-205-9

```
For each arriving packet P of tunnel t_i:
   Calc the average universal queue avg;
   Calc the average individual queue avg_i;
   If (avg < Gmin_th) no-drop;
   Else If (Gmin_th < avg < Gmax_th){
      If (avg_i < Lmin_th) no-drop;
      Else If (Lmin_th < avg_i < Lmax_th)
         Drop with probability P_u·P_i;
      Else If (avg_i > Lmax_th)
         Drop with probability P_u;
   }
   Else If (avg > Gmax_th){
      If (avg_i < Lmin_th)
         Drop with probability P_u·P_i;
      Else If (Lmin_th < avg_i < Lmax_th)
         Drop with probability P_u;
      Else If (avg_i > Lmax_th)
         Drop;
   }
```

Regarding the four problematic issues discussed in section II:

- The URED algorithm does not guarantee space for active users.
- The condition for identifying a non-adaptive user is proper for any user with multiple flows either TCP or UDP-based.
- The actions on a non-adaptive user are adjusted to the level of the queue buildup and consider both the global state of the universal queue and the local state of the individual queue.
- The URED algorithm uses the smooth hazard drop functions of [1].

IV. PERFORMANCE EVALUATION

The performance of our new per user AQM algorithm was evaluated by simulation in the ns2 network simulator [12]. The simulation network topology is presented in Fig. 1. It consists of one source of Constant Bit Rate (CBR) with a rate of 5 mbps, nine sources of FTP over TCP (*regular users*) and one source of P2P with five simultaneous TCP connections (*P2P user*). Two sets of experiments are presented below, with bottlenecked link of 20 mbps and 10 mbps. The simulation parameters are listed in Table I.

Regarding the user throughput, Fig. 2 plots the statistics for the average throughput per time unit for the Drop Tail (DT), RED, FRED and URED algorithms under the 20 mbps bottlenecked link. The figure shows that applying the DT algorithm results in the highest un-balanced traffic between the users. The average throughput of the P2P user ($User_{11}$) was more than 6.8 times higher than the average throughput of a regular user ($User_2$-$User_{10}$). Our new URED algorithm leads to much better results. Its average throughput for the P2P user was less than 1.5 times the average throughput of a regular user. Fig. 3 plots the statistics for the average packet delay. The figure shows that the DT algorithm results in the highest delays, and all other algorithms have similar results.

Table II depicts the average throughput gap between the P2P user and a regular user for every algorithm under bottlenecked links of 20 mbps and 10 mbps. As expected, with a 10 mpbs bottlenecked link, the traffic become more un-balanced than with a 20 mpbs bottlenecked link. An interesting observation is that RED outperforms FRED under a heavy bottlenecked link.

Regarding the user packet loss, Table III plots the statistics for the average packet loss per time unit for the Drop Tail (DT), RED, FRED and URED algorithms under a 20 mbps bottlenecked link for regular and P2P users. RED drops the fewest packets compared to the other algorithms. FRED drops more packets than the other algorithms.

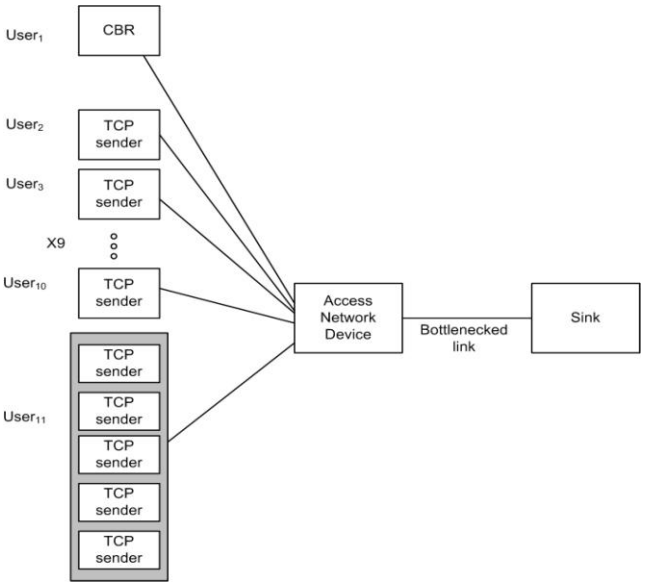

Figure 1: Simulation setup

TABLE I. THE SIMULATION PARAMETERS

Parameter	***Value***
Bottlenecked link BW	10, 20 mbps
Number of Regular Users	9
Number of CBR Users	1
Number of P2P Users	1
CBR Rate	5 mbps
FTP Burst time	2000 ms
FTP Idle Time	200 ms
FTP File Shape	1.7
Packet size	1600 B

TABLE II. THE AVERAGE THROUGHPUT GAP

Algorithm	**P2P user throughput/Regular user throughput**	
	20 mbps bottlenecked link	***10 mbps bottlenecked link***
DT	6.81	7.81
RED	4.51	4.48
FRED	3.76	4.56
URED	1.46	1.66

 ISBN: 978-1-61208-205-9

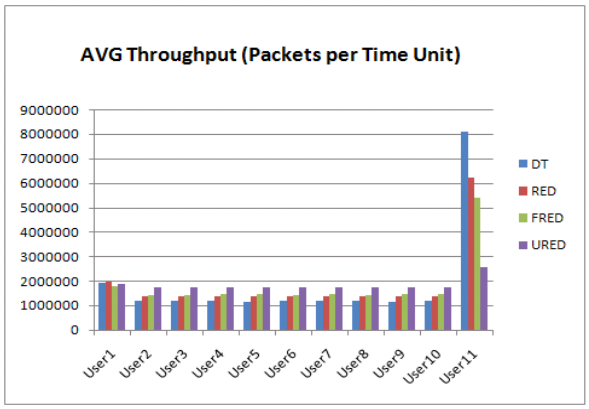

Figure 2: Average user throughput (20 mbps bottlenecked link)

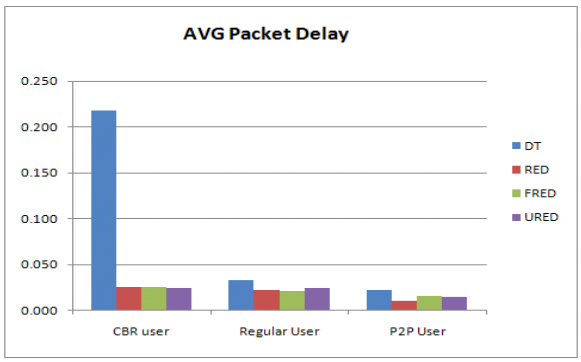

Figure 3: Average packet delay (20 mbps bottlenecked link)

TABLE III. AVERAGE PACKET LOSS

Algorithm	***Regular User***	***P2P User***
DT	72064	158847
RED	62270	275521
FRED	150064	489503
URED	106303	288112

V. CONCLUSION AND FUTURE WORK

This paper described on-going research on a new URED algorithm for AQM that is designed to handle congestion in a blind access network device. The proposed method uses available tunnel statistics and does not use information from TCP and IP headers. This method ensures fair resource allocation among network users on a congested link and can help operators limit the resource consumption of P2P applications.

Future work includes additional simulations to improve the evaluation of the algorithm's potential and identify its limitations. In addition, we are working on a self configuration method for the algorithm parameters.

ACKNOWLEDGMENT

The authors thank the anonymous referees for helpful comments on an earlier version of this paper.

REFERENCES

[1] B. Abbasov and S. Korukoğlu, "An Active Queue Management Algorithm for Reducing Packet Loss Rate", Mathematical and Computational Applications, Vol. 14, No. 1, pp. 65-72, 2009.

[2] P. Anelli, E. Lochin, and R. Diana, "FavourQueue: a Stateless Active Queue Management to Speed Up Short TCP Flows (and others too!)", ArXiv e-prints, 1103.2303, March, 2011.

[3] S. Athuraliya, S. H. Low, V. H. Li, and Y. Qinghe, "REM: active queue management", IEEE Network, Vol. 15(3)., pp. 48-53, May 2001.

[4] Z. Cao, Z. Wang, and E. Zegura, "Rainbow fair queueing: Fair bandwidth sharing without per-flow state", In Proceedings of INFOCOM'00, pp. 922–931, Tel-Aviv, Israel, March 2000.

[5] Cisco's web site http://www.cisco.com/en/US/docs/ios/12_2t/12_2t8/feature/guide/ftwrdecn.html, retrieved: May, 2012.

[6] W. C. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The BLUE Active Queue Management Algorithms" IEEE ACM Transactions on Networking, Vol. 10(4)., pp. 513-528, August 2002.

[7] S. Floyd and V. Jacobson,"Random Early Detection Gateways for congestion Avoidance" ,IEEE/ACM Transaction on Networking, Vol. 3., pp. 397-413, August 1993.

[8] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross "A Measurement Study of a Large-Scale P2P IPTV System", IEEE Transactions on Multimedia, Dec. 2007

[9] B. J. Hwang, I.S. Hwang, and P.M.Chang, "QoS-Aware Active Queue Management for Multimedia Services over the Internet", EURASIP Journal onWireless Communications and Networking, Article ID 589863, 2011.

[10] E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini, and D. Rossi, "Building a cooperative P2P-TV application over a Wise Network: the approach of the European FP-7 STREP NAPA-WINE",IEEE Communication Magazine,Vol. 64, No. 4, pp. 20-22, April 2008.

[11] D. Lin and R. Morris, "Dynamics of random early detection", Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 127-137, September 14-18, 1997, Cannes, France.

[12] NS: The Network Simulator (NS 2), http://isi.edu/nsnam/ns/ retrieved: May, 2012.

[13] R. Pan, B. Prabhakar, and K. Psounis., "Choke, a stateless active queue management scheme for approximating fair bandwidth allocation", In Proceedings of IEEE INFOCOM'00, pp. 942-951, Tel Aviv, Israel, March 2000.

[14] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks", ACM SIGCOMM Computer Communication Review, Vol. 28 (4), pp.118-130, Oct. 1998.

[15] G. Thiruchelvi and J. Raja, "A Survey On Active Queue Management Mechanisms", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, pp. 130-146, Dec. 2008.

[16] RFC 1661, http://tools.ietf.org/html/rfc1661, retrieved: May, 2012.

[17] 3GPP the mobile broadband standard web site, http://www.3gpp.org/ftp/Specs/, retrieved: May, 2012.

 ISBN: 978-1-61208-205-9