Equilibrium-RED: Adjusting RED's Dropping Probability for Maximum Fairness and Performance

S. Dimitriou, and V. Tsaoussidis
Dept. of Electrical and Computer Engineering
Democritus University of Thrace, Greece

***Abstract*— RED is an Active Queue Management technique that has been around for many years. Contrary to the Droptail algorithm, RED manages to increase fairness dramatically, although occasionally against network performance. RED's efficiency is due to a dropping mechanism that, based on a function of average queue length, drops probabilistically packets from the queue. Unlike RED, Droptail may favor performance over fairness. In this paper we analyze RED's dropping algorithm and based on the results we attempt to create a RED variant for maximum performance and fairness: Equilibrium-RED. We present results that demonstrate EQU-RED's ability to balance the trade-off between Droptail and classic RED behaviors.**

I. INTRODUCTION

RED [7] is an AQM technique used by routers to avoid buffer overflows, which cause bursts of drops. It uses proactive packet dropping: RED drops probabilistically packets from the queue, before it reaches its maximum capacity, in order to notify the senders that a congestion event is imminent. As more packets are dropped, more flows detect packet losses and retreat. However, retreating has a cost: lost packets have to be retransmitted, resulting in additional energy consumption; a critical matter for battery-supplied devices. Therefore, RED has two main characteristics: a lot of retransmissions and high fairness. On the other hand, Droptail has no proactive dropping. It accepts packets as long as there is buffer space available and rejects all the incoming packets after the router has reached its maximum capacity. Unlike RED, in Droptail, only a small percentage of flows is notified of the occurrence of a congestion event and thus only few flows detect lost data and retreat. Thus, retransmissions will be minimized leading, however, to unfair distribution of resources.

Although the above remarks are not by law, there are many cases where Fairness and Badput (packet retransmissions) are related to the number of packets proactively dropped. In this work we investigate this correlation; we show that the level of this correlation depends on the portion of forced (drops caused by buffer limitations) and unforced (proactive drops caused by the AQM mechanism) drops and that in many cases fairness and network overhead are inversely related. We then propose Equilibrium-RED, a tunable RED variant which aims to offer maximum fairness and minimum retransmissions by adjusting the dropping probability in order to achieve equal numbers of unforced and forced drops. Contrary to RED, the dropping probability is constant, almost for the entire length of the queue and equals to p_{EQU}. For a given time period, EQU-RED measures the number of forced and unforced drops made by the router. If unforced drops are more than forced drops then it decreases the p_{EQU}, otherwise it increases it.

We demonstrate through simulations that EQU-RED can achieve good performance with only a small loss in fairness. Moreover, we show that most AQM methods, although they may work better under specific conditions (networks with real-time flows or small-lived flows), they tend to follow this Badput-Fairness trade-off. While some manage to maintain high fairness with many retransmissions (RED, REM), others cause less retransmissions and weak fairness. Only EQU-RED, however, can be tuned in order to lean towards the first or the second approach.

In section 2, we present the modifications that have been proposed towards the improvement of the original RED, as well as the various RED-derived protocols. In section 3, we review RED's algorithm and emphasize on the analysis of the dropping mechanism. Section 4 is focused on the EQU-RED proposal and its experimental evaluation. In section 5, we define the simulation scenarios and in section 6, we analyze the relevance between dropping probability and network metrics. In section 7 we measure the effect of different dropping schemes. Section 8 concludes and sets the framework for future work.

II. RELATED WORK

In [7] S. Floyd and V. Jacobson proposed the RED algorithm. The original RED scheme proposed a dropping mechanism based on average queue length. RED sets a maximum threshold for the average queue length, beyond which it discards all incoming packets. The gentle RED modification extends this threshold to double the previous value, making full exploitation of the buffer space. However, some applications that generate a small amount of critical data, like Telnet, do not exit from slow start and may delay from packet drops. In [13] the authors propose for the first time ECN (Explicit Congestion Notification). In ECN packets

are not dropped, but are marked by the router instead, and their marking will have the same effect on senders as packet loss.

One more issue of RED gateways is the lack of adaptability on different traffic levels. Adaptive RED [3], [5] avoids link underutilization by maintaining the average queue length among the two thresholds by adjusting p_{max}. Weighted RED (WRED) [14] is designed to serve Differentiated Services based on IP precedence. Packets with a higher IP precedence are less likely to be dropped, thus high priority traffic will be delivered with higher probability than low priority. Flow RED (FRED) [9] uses per-active-flow accounting to impose on each flow a loss rate that depends on the flow's buffer use. Unfortunately, extended memory and processor power is required for a big number of flows. On the other hand RED-PD (Preferential Dropping) [11] maintains a state only for the high-bandwidth flows. Loss Ratio based RED (LRED) [16] measures the latest packet loss ratio, and uses it as a complement to queue length in order to dynamically adjust packet drop probability and decrease response time. Lastly, Exponential-RED (E-RED) [10] sets the packet marking probability to be an exponential function of the length of a virtual queue whose capacity is slightly smaller than the link capacity.

In section 7, during our simulations we will use three well known techniques, REM, BLUE and the PI controllers. REM's [1] main characteristic is that it matches user rates to network capacity by sending the right signal to the senders. BLUE [4] manages dropping, based on packet loss and link idle events; if the queue drops packets due to buffer overflows, BLUE increases the dropping probability, whereas if the queue becomes empty or idle, BLUE decreases the dropping. Last, the Proportional-Integral (PI) controller [8] is a robust controller that outperforms the RED controller.

III. Analyzing RED Dropping Policies

Instead of using queue's actual length, RED uses a weighted moving average of the queue's length. We have

$$avg = p \cdot q_{sample} + (1 - p) \cdot avg$$

where the p variable takes small values, usually around 0.002. RED also uses two thresholds, minimum and maximum threshold, which are determined by the buffer size. We have three distinct cases:

1. If the average length is less than the minimum threshold then no packet is dropped.

2. If the average length is greater than the maximum threshold or if the queue is full, then the last packet of the queue is definitively dropped (forced drop).

3. If the average length is between these two thresholds then the last packet is dropped with a given probability (unforced drop), which increases as the average length approaches maximum threshold. The dropping probability function for the original RED algorithm is shown in Fig. 1.

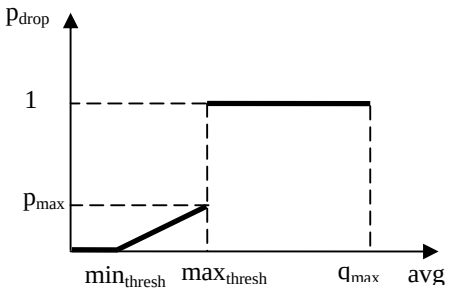

Fig. 1. RED's Dropping Probability Function

Thus, when the flow traffic exploits the available buffer space, RED causes some unforced and some forced drops. The number of unforced drops is typically much higher than the number of forced drops; however, this is determined by the number of flows and the thresholds. Unlike RED, the original Droptail causes forced drops only when the queue exceeds buffer's capacity. These two types of drops have some specific characteristics.

Forced Dropping: Forced dropping occurs when the queue's length exceeds a threshold, either user-defined (max_{thresh} in RED) or physical (buffer's capacity). Forced drops occur in bursts and most times they involve a small number of participating flows. The arising problem is twofold; some flows will not be aware of the occurrence of a congestion event and they will continue to increase their sending rates, whereas the rest will be synchronized. However, as contention increases, more flows will be notified of the congestion event. In many cases the system could reach better performance since the packets will be rejected only when it is "absolutely necessary". Unfortunately, in cases of TCPs that create stable queue lengths, as TCP Vegas or TCP Real [17], a threshold set too low would result in an average length practically equal to the physical capacity of the queue, and hence buffer underutilization will be inevitable.

Unforced Dropping: Unforced dropping is a proactive measure to avoid buffer overflow. The dropped packets typically belong to various flows, thus notifying more senders about high contention. This enhanced diversity of notifying receivers may result in wide transmission retreat which causes inevitably underutilization. Moreover, some highly-sophisticated TCP versions might conceive these drops as corrupted data caused by wireless links and might maintain the same sending rate. For the protocol senders the unbalanced forced/unforced dropping policy may be experienced as a paradox: More transmission effort may not result in higher throughput gains [12]. An interesting characteristic of unforced drops is their location independence. The reaction of the sender on a packet drop, is based on the way it perceives the loss (multiple DACKs or timeout), not on the buffer state the time the drop occurred. This leads to the conclusion that the router could be more aggressive even for low level of queue utilization.

From Fig. 1 we can deduce that the parameters that determine the dropping probability of RED are min_{thresh}, max_{thresh} and p_{max}; in order to adjust the number of dropped packets each time, we should alter one or more of these parameters. Minimum and maximum threshold are inappropriate for this purpose, since they may lead to queue underutilization for protocols that generate stable queue lengths. Thus, p_{max} is the only parameter left that seems to have a direct connection with packet drops. However, experiments show that increasing p_{max} may lead to less unforced dropping than with a lower value. These results indicate that maybe the current dropping probability function is inadequate to balance the two types of drops.

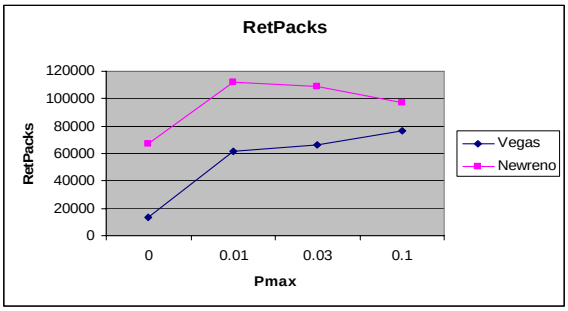

Fig. 2. Retransmitted packets vs p_{max} using gentle RED

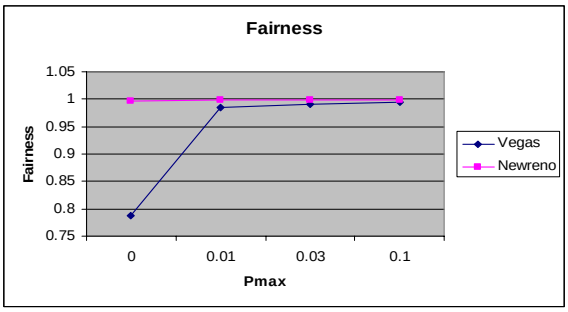

Fig. 3. Fairness vs p_{max} using gentle RED

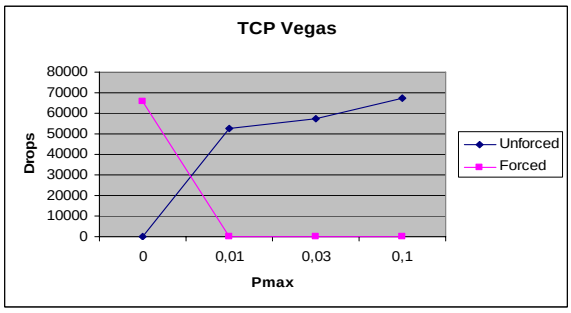

Fig. 4. Types of drops vs p_{max} using gentle RED and TCP Vegas flows

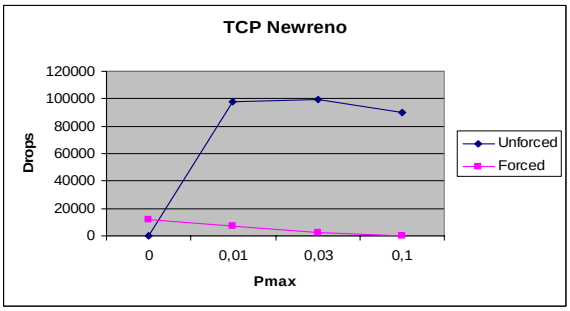

Fig. 5. Types of drops vs p_{max} using gentle RED and TCP Newreno flows

In order to justify our claims we made some experiments with the current gentle RED scheme. We used a dumbbell topology with 100 flows and we ran some ns-2 [15] simulations with different values of p_{max}. We can observe, that even with gentle RED, p_{max} increase does not necessarily correspond to more unforced drops. Moreover, even for the cases where there is an increase in retransmitted packets, this is increase is not linear; significant increase in p_{max} doesn't necessarily result in a significant increase of RetPacks, thus adjusting p_{max} to modify the number of drops, might lead to oscillation and slow convergence.

IV. EQUILIBRIUM RED

To achieve the desired balance we propose a modified dropping function. In order to decrease complexity we use a constant function where the dropping probability is the same for almost the entire of the queue length (apart from 0-10% because of low link utilization in this area). The proposed function can be seen in Fig. 6. The rest of RED's functionality still applies.

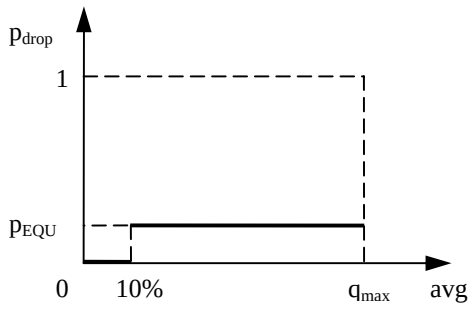

Fig. 6. The EQU-RED dropping probability function

Minimum and maximum threshold should be extended in order to occupy the available buffer space. The functionality of EQU-RED gateways includes measurement and adjustment. First they measure the number of forced and unforced dropped packets for a predetermined period of time (e.g. one second). If the number of forced drops is greater than the number of unforced drops, p_{EQU} will be multiplicatively increased, otherwise decreased. We adjust multiplicatively p_{EQU} to achieve faster convergence. After some seconds the router will adjust to an equilibrium state where the number of forced and unforced drops will be equal.

Using only one point of control (p_{EQU}) and choosing the equilibrium point equally between forced and unforced drops, we achieve three goals:

1. We avoid buffer underutilization. In many cases RED leads to buffer underutilization since the dropping probability is so high that the queue never reaches its full capacity. Our proposed scheme imposes forced dropping leading to a more efficient buffer use.

2. We ensure adaptability on various network conditions. Every second the router will recalculate the dropping probability to capture the current network conditions.

3. We achieve full control of the dropping function. Although the gentle RED proposal gives better results over original RED, the connection between p_{EQU} and drops is not explicit. Using our function we establish a corresponding relation between p_{EQU} and the amount of packet drops.

EQU-RED's algorithm is simple. The differences from the original RED algorithm are the modification of the dropping probability function and the addition of two more counters: total_forced (the total number of forced drops in a second) and total_unforced (the total number of unforced drops in a second).

```
if(forced_dropping_occurs()) total_forced++;
if(unforced_dropping_occurs()) total_unforced++;

every_second() {
   if(total_forced>total_unforced) p_equ=p_equ*1.1;
   else p_equ=p_equ/1.1;
   total_forced=0;
   total_unforced=0;
}
```

The time interval between measurements and the factor we use to adjust p_{EQU} should be dynamically modified to reflect network conditions; however, in this first approach we will consider them constant throughout the transmission, 1sec and 1.1 respectively.

By and large, our algorithm can be further adjusted in order to give weight either to fairness or performance, by altering the Forced to Unforced proportion appropriately. The choice depends on the network topology and the requirements of the specific applications e.g. battery-powered devices might tolerate unfair treatment for less retransmissions.

Having defined the EQU-RED scheme we demonstrate through simulations that the proposed function can adjust the unforced-to-forced ratio better; let alone that EQU-RED combines effectively the desired properties of both Droptail and RED.

V. Network Topology and Scenarios

We conducted simulations with ns-2 on a cross-traffic topology (Fig. 7). The topology was designed in the way that all flows have the same propagation delay in order to allow for direct, accurate measurements of fairness. In both cases the number of flows varies from 2 to 200 and the simulation time is 100sec. **R1_1** to **R1_N/2** and **R2_1** to **R2_N/2** are the receiving nodes for the **S1_1** to **S1_N/2** and **S2_1** to **S2_N/2** sending nodes, respectively.

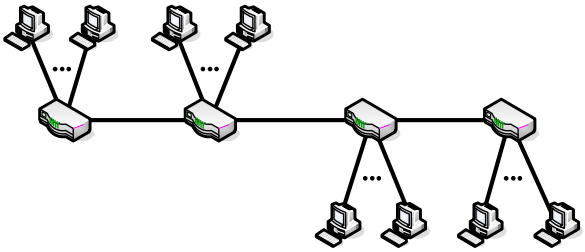

Fig. 7. Cross-traffic topology

We should note that although all flows have the same propagation delay, the system is very unfair, due to the fact that the flows S2-R2 establish connection faster than the flows S1-R1.

In the first part of the experiments we vary the unforced-to-forced ratio with fixed number of flows using EQU-RED. In the second part we vary the number of flows and compare EQU-RED with Droptail and RED. Both of these sets of simulations were done with TCP Vegas and Newreno. In the third part we adjust the unforced-to-forced ratio with different flows and in the final fourth part we compare different AQM mechanisms.

VI. Connection of Dropping Probability and Network Metrics

Using the function described in Fig 6, we examine EQU-RED's behavior when we alter gradually the unforced-to-forced ratio from 5-1 to 1-5. We present (1/Badput)-to-Fairness diagrams for each protocol.

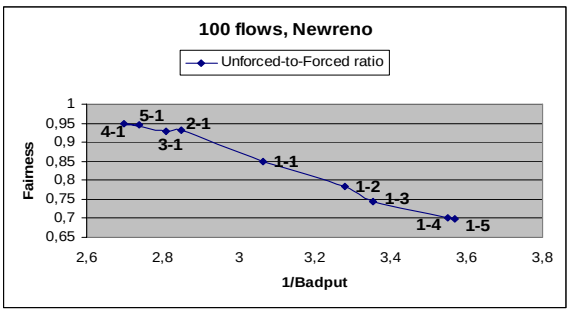

Fig. 8. 100 Newreno flows

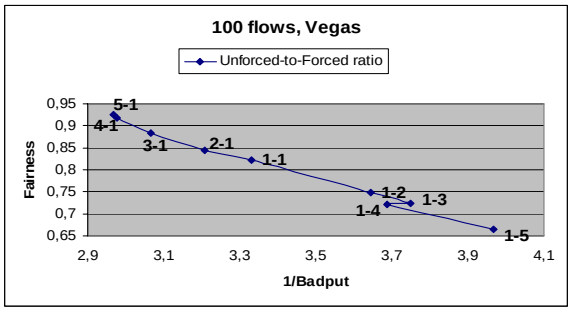

Fig. 9. 100 Vegas flows

Generally, a remarkable result is that the decrease of the unforced-to-forced ratio is combined with a decrease of Fairness and increase of 1/Badput (Badput is decreased). An unforced-to-forced ratio 1-1 gives an equilibrium point, where we have a fair trade-off between Fairness and Badput. However, this equilibrium point is not stable as it shifts left for TCP Vegas flows. As we move to extreme high and low values of unforced-to-forced ratio the results of the simulations seem close to each other, and this is due to the fact that either the algorithm was not fast enough to reach the expected equilibrium point, or that after some point, such small divergences from the equilibrium point may make no difference to the network behavior.

We should point out that Fig. 8,9 above capture an average case. In some cases there is no trade-off, since the increase of fairness due to RED dropping mechanism is accompanied with decrease of Badput. In these cases setting this ratio equal to 1 will result in a worse solution than RED, but in a better solution than Droptail.

VII. MEASURING THE EFFECT OF UNFORCED DROPPING

During the second part of the experiments we compare side-by-side Droptail, RED and EQU-RED. We increase the number of flows from 2 to 200 and we measure Retransmitted Packets and Fairness. We omit Goodput, since the differences among the various AQM mechanisms are minor. The results we get are indicative of the compromise EQU-RED does to achieve good metrics. EQU-RED tends to follow the high-fairness behavior of RED, whereas it has a significant gain over network overhead. This is more apparent with TCP Vegas, where we observe extensive unfairness. EQU-RED manages to achieve 20% less retransmitted packets than RED, with only 2% decrease of fairness.

We should note that for specific number of flows, EQU-RED achieves better results than Droptail in means of RetPacks, showing the dynamic nature of the protocol. Yet, EQU-RED has a stable behavior, contrary to RED, thanks to the adjusting probability mechanism that allows it to follow network dynamics.

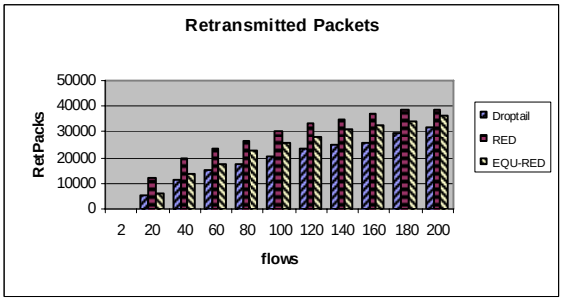

Fig. 10. Retransmitted packets – Newreno

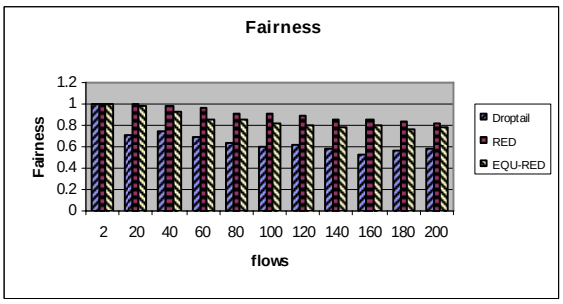

Fig. 11. Fairness – Newreno

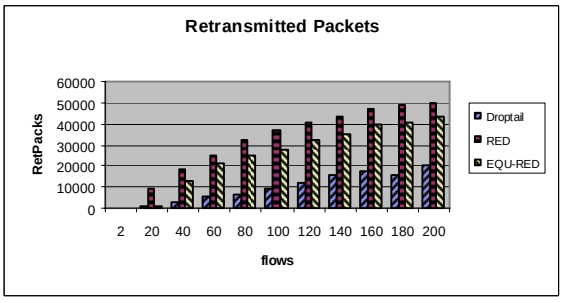

Fig. 12. Retransmitted packets – Vegas

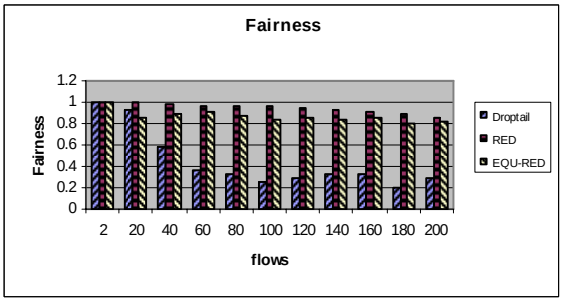

Fig. 13. Cross-traffic topology – Vegas

Modifying the target unforced-to-forced ratio alters EQU-RED behavior the way we demonstrated in Section 6 Depending on the application or the type of network, we may want to favor either fairness or performance, e.g. in cases of networks of battery-supplied devices performance is more important than fairness. We compare three variations of EQU-RED with 1-2, 1-1, and 2-1 unforced-to-forced ratio and we show that by adjusting the ratio we can balance the Badput-Fairness to the desired level. We only present results for Newreno (Fig. 14,15). Same things apply for TCP Vegas.

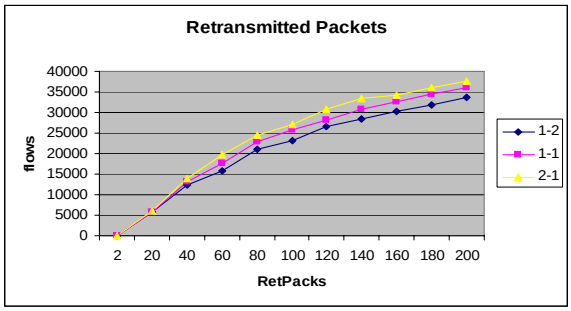

Fig. 14. Cross-traffic topology – Newreno

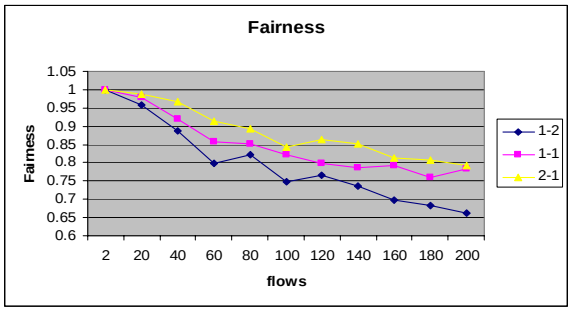

Fig. 15. Cross-traffic topology – Newreno

In order to prove our initial hypothesis, that is that most AQM mechanisms favor either fairness or performance, we made additional simulations with 200 flows and BLUE, PI and REM gateways. We illustrate the results in Fig. 16,17. We see that each technique tends to follow a specific Badput-Fairness trade-off pattern. Droptail, BLUE and PI provide less retransmissions with small fairness, while RED, REM and EQU-RED high fairness with more retransmissions.

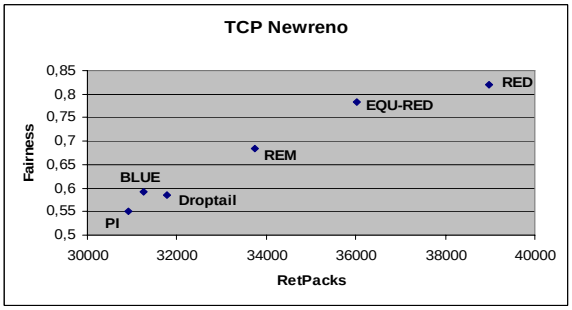

Fig. 16. Cross-traffic topology – Vegas

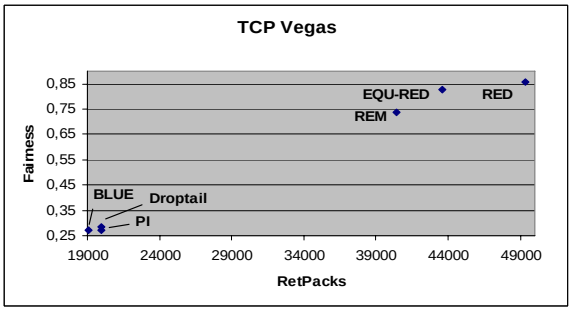

Fig. 17. Cross-traffic topology – Vegas

VIII. Conclusion and Future Work

We have examined the distinctive impact of unforced and forced drops on various performance metrics. The next step was to propose a protocol that exploited forced and unforced dropping in order to achieve the trade-off among these metrics. However, at present, we omitted to evaluate protocols that have a different reaction to transient drops, such as TCP Westwood. Multiple backbone topologies are also a matter of concern as they favor unfairness and represent more real-life applications. Further extension should also concern the granularity issue and include high-speed networks.

References

[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, “REM: Active Queue Management”, *IEEE Network Magazine*, May 2001

[2] Brakmo et al, “TCP Vegas: New Techniques for Congestion Detection and Avoidance”, *SIGCOMM*, August 1994

[3] W.C. Feng, D. Kandlur, D. Saha, and K. Shin, “A Self-Configuring RED Gateway”, *Infocom*, March 1999

[4] W. Feng, D. Kandlur, D. Saha, and K. Shin, “BLUE: A New Class of Active Queue Management Algorithms”, *U. Michigan CSE-TR-387-99*, April 1999

[5] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive RED: An Algorithm for Increasing the Robustness of RED’s Active Queue Management”, August 2001

[6] S. Floyd, T. Henderson, and A. Gurtov, “RFC3782 – The NewReno modification to TCP’s fast recovery algorithm”, April 2004

[7] S. Floyd, and V. Jacobson, “Random Early Detection gateways for Congestion Avoidance”, *IEEE/ACM Transactions on Networking*, August 1993

[8] C. Hollot, V. Misra, D. Towsley, and W. Gong, “On Designing Improved Controllers for AQM Routers Supporting TCP Flows,” *Infocom*, April 2001

[9] D. Lin, and R. Morris R, “Dynamics of Random Early Detection”, *SIGCOMM*, September 1997

[10] S. Liu, T. Basar, and R. Srikant, “Exponential-RED: A Stabilizing AQM Scheme for Low- and High-Speed TCP Protocols”, *IEEE/ACM Transactions on Networking*, October 2005

[11] R. Mahajan, and S. Floyd, “Controlling High Bandwidth Flows at the Congested Router”, *ICNP*, November 2001

[12] L.Mamatas and V.Tsaoussidis, “Protocol Behavior: More Effort, More Gains?”, *PIMRC*, September 2004

[13] K. Ramakrishnan, and S. Floyd, “RFC2481 – A Proposal to Add Explicit Congestion Notification (ECN) to IP”, January 1999

[14] Technical Specification from Cisco, *Distributed Weighted Random Early Detection*, URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf

[15] The Network Simulator, *http://nsnam.isi.edu/nsnam*

[16] C. Wang, B. Li, Y. Hou, K. Sohraby, and Y. Lin, “LRED: A Robust Active Queue Management Scheme Based on Packet Loss Ratio”, *Infocom*, March 2004

[17] V. Tsaoussidis, and C. Zhang, “TCP-Real: Receiver-oriented Congestion Control”, *The Journal of Computer Networks*, November 2002