The Interaction of Forward Error Correction and Active Queue Management

Tigist Alemu, Yvan Calas, Alain Jean-Marie

▶ **To cite this version:**

HAL Id: lirmm-00089973
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00089973

Submitted on 25 Aug 2006

The Interaction of Forward Error Correction and Active Queue Management

Tigist Alemu, Yvan Calas, and Alain Jean-Marie

LIRMM
UMR 5506 CNRS and University of Montpellier II
161, Rue Ada,
34392 Montpellier Cedex 5, France
Tel: (33) 4 67 41 86 02   Fax: (33) 4 67 41 85 00
{tigist,calas,ajm}@lirmm.fr

Abstract. This paper studies the interaction of a forward error correction (FEC) code with queue management schemes like Drop Tail (DT) and RED. Since RED spreads randomly packet drops, it reduces consecutive losses. This property makes RED compatible *a priori* with the use of FEC at the packet level. We show, through simulations, that FEC combined with RED may indeed be more efficient than FEC combined with DT. This however depends on several parameters like the burstiness of the background traffic, the FEC block size and the amount of redundancy in a FEC block. We conclude generally that using FEC is more efficient with RED than with DT when the loss rate is small, a relatively important amount of redundancy and at most a moderate FEC block size is used. We complement these observations with a simple model, which is able to capture the tradeoff between the locality and the frequency of losses.

Keywords: RED, FEC, queue management.

1 Introduction

The Internet traffic suffers from heavy losses due to network congestion caused by the limited capacity of queue in the routers. There exist two end-to-end error control techniques to repair these losses: ARQ (Automatic Repeat reQuest) which consist in retransmitting dropped packets upon the destination's request, and FEC (Forward Error Correction) which consists in sending redundant packets to the destination, allowing it to repair losses without requiring packet retransmission. Because of retransmissions, ARQ is not appropriate for real-time applications. This is why FEC is increasingly used in such

applications, typically on top of the UDP transport protocol. FEC is also used in bulk data transfer applications such as Digital Fountain [1]. Several drawbacks are attached to the use of FEC. First, FEC cannot recover all lost packets. In addition, the transmission of redundant packets increases the overall network load. Finally, the effectiveness of FEC is known to depend on the way packet drops are distributed in the data stream. FEC is more efficient when packets losses are independent, and much less when they occur in groups [2].

In conjunction to end-to-end error control techniques, there exist queue management schemes operating inside routers that control network congestion. The "queue management" scheme traditionally used in the current Internet is Drop Tail (DT), which consists in discarding arriving packets when the buffer of the router overflows. *Active* queue management schemes, in particular the Random Early Detection (RED) scheme [3, 4], have been recommended recently by the IETF as an alternative, aimed at eliminating deficiencies of Drop Tail. The RED scheme basically discards packets earlier so that incipient stages of congestion can be detected.

The aim of this paper is to study the interaction of FEC with RED (RED/FEC) and to compare the obtained results with those obtained from the combination of FEC with Drop Tail (DT/FEC). This study has never been conducted to our knowledge. Indeed, FEC has always been studied in presence of the Drop Tail queue management. We believe that as compared to Drop Tail, RED may give performance improvement for the UDP sources implementing FEC since it spreads randomly packet drops between flows, reducing consecutive losses for a given flow, thereby making losses "more independent".

In Section 2, we briefly present the principles of the FEC coding scheme and of queue management algorithms. In Section 3, we describe the topology of the system and the performance metrics considered in this paper. The performance measures obtained by simulation are detailed and analyzed in Section 4. Section 5 presents a simple model with which we explain the tradeoff responsible for the fact that sometimes RED/FEC is more efficient, sometimes DT/FEC. Section 6 presents conclusions and perspectives.

2 FEC and Queue Management

We first recall some properties of codes used for Forward Error Correction. Given k data packets bearing the relevant information, the encoder (based for instance on a Reed-Solomon Erasure code [5]) generates h redundant packets useful for the recovery of the lost data packets. The concatenation of the k data packets and the h redundancy packets is called a *FEC block* of size $n = k + h$. If the total number of lost (data and redundant) packets is at most h, the decoder at the destination can retrieve successfully all lost packets. As a result all the relevant information is saved. Otherwise, if the total loss exceeds h packets, it is impossible to recover the lost packets.

The queue management schemes studied here are Drop Tail, the principle of which is straightforward, and RED. With RED, the router maintains an estimate of the average queue length, using an exponential moving average. Based on this value, it accepts or rejects incoming packets with a certain probability. The rejection probability function is a parameter of the mechanism. We have used in the following experiments the default values for RED parameters [3,6].

3 Experimental Setup

3.1 Network topology

The network setup is depicted in Figure 1. The traffic generated by nodes S_0 to S_N is multiplexed on a 10 *Mbps* bottleneck link between nodes R_1 and R_2 with a propagation delay of $30ms$. The bottleneck link is provided with a Drop Tail or a RED queue of limited capacity of 35 packets. The other links located between nodes $S_0, \ldots, S_N$ and node R_1 have a capacity of $100Mbps$. These links have different propagation delays uniformly distributed from $20ms$ to $100ms$.

We have studied a traffic mix of TCP and UDP flows, with UDP representing a minority of the traffic. Our purpose in doing so is that we wish to study the behavior of the FEC technique under bursty conditions. A background traffic generated by TCP sources is appropriate in that respect since TCP traffic is usually quite bursty. Alternately, a simulation with variable-rate UDP sources could have been used. In the future of networks, it may be that flows of real-time
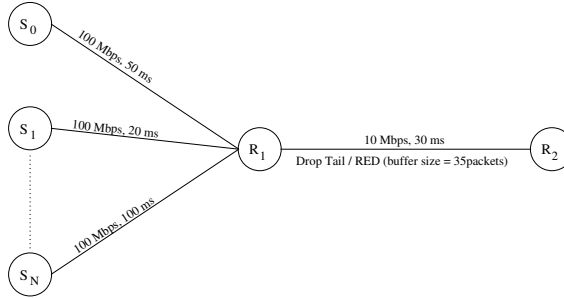
Fig. 1. Topology of the system.

applications (such as our UDP sources) will be separated from elastic TCP traffic, in which case the network conditions encountered by the real-time traffic could be quite different. But in the current Internet, where service differentiation is not yet widespread, the experimental setup we have chosen represents an average situation. The study of these conditions can actually help deciding if deploying flow differentiation at the node level can be dispensed with.

A Poisson process is used for the generation of the foreground UDP traffic at node S_0. Node S_1 to S_N generate background long-lived FTP traffics using TCP/Sack1 agents. The offered load of the UDP traffic in the absence of redundancy is set to $\rho_1 = 500 kb/s$. Without redundancy the load generated by the UDP traffic represents 5% of the bandwidth of the bottleneck link. For the generation of redundant packets, we increase the throughput ρ_1 of the UDP/FEC flow by a factor of $1 + h/k$, in order to take into account the addition of redundancy while keeping constant the rate of information generated by the source. The resulting new load ρ_{FEC} is equal to $\rho_{FEC} = \rho_1(1 + h/k)$. Under `ns-2` [7], the standard way to generate a Poisson process (approximately) is to use the exponential on/off source. Bursts of packets of size one are obtained with a very large bit rate at the source. The space between packets is controlled with the variable `idle_time_`. In order to take into account the increase of the load due to redundancy, this value is computed as follows:

$$\texttt{idle\_time\_} = \frac{\text{UDP packet size}}{500 \times (1 + h/k)} \ ,$$

where the packet size for the UDP traffic is set to 573 bytes following the recommendation of [8]. TCP packet sizes are set to 1200 bytes and the maximum TCP window size is set to 20 packets, that is, 24kBytes. Since the delay $\times$ bottleneck bandwidth product is 300 kb or 37.5kBytes, this means that a single source cannot saturate the link. Actually, given the maximum RTT of 260ms, the maximum offered throughput of one source is about 100kBytes/s. The super-position of about 12 sources should saturate the 10Mbps link. In addition, assuming a full window, the typical sending pattern of a source should be 20 packets simultaneously each RTT, thus realizing a bursty arrival pattern at the bottleneck, as intended.

Every simulation is run for 100 simulated seconds and statistics are collected every $10ms$ from the queue located between node R_1 and R_2. The results presented below are averages over 50 independent simulations.

3.2 Performance metrics

The metrics used for a specific flow (that is for the FEC flow) are:

– The *packet loss rate before correction* ($PLRBC$) that is the ratio of the average number of lost packets in a FEC block before correction to the size of the FEC block.
– The *packet loss rate after correction* (PLR) that is the ratio of the average number of lost packets in a FEC block after correction to the size of the FEC block.
– The *loss run length* [9] that is the number of packets of a particular flow that are lost consecutively. This is a random variable which gives an insight into the packet loss process. Studying this metric allows to investigate which queue management is more efficient when used with FEC.

4 Performance Measures

4.1 Influence of the number of TCP flows

Figure 2 shows the evolution of the packet loss rate before correction (PLRBC) and after correction (PLR) for the UDP source as a function of the number of TCP flows and the number of redundancy h.

(a) $k = 16, h = 1$.

(b) $k = 16, h = 4$.
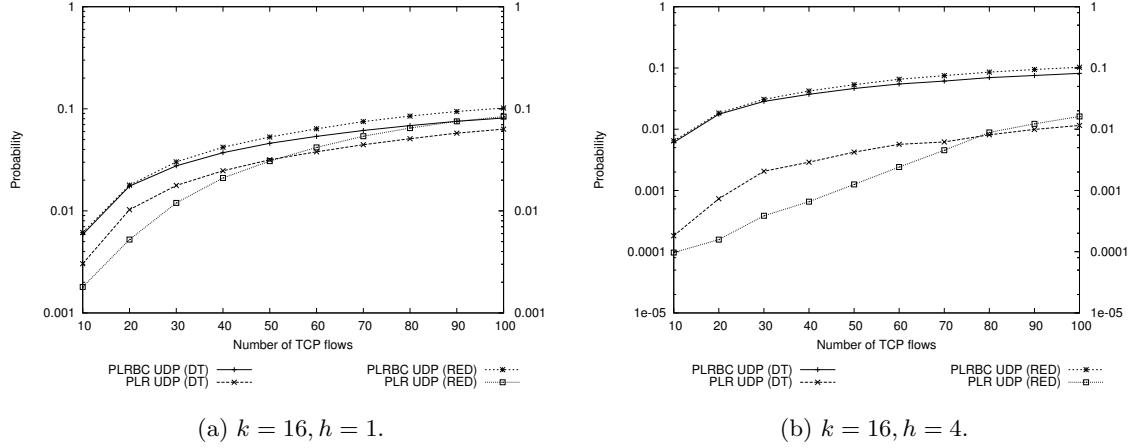
Fig. 2. Packet loss rate before (PLRBC) and after (PLR) correction by FEC.

As observed in [10] and as shown also in Figure 2, when the amount of redundancy is increased, the PLR of DT/FEC decreases for this network configuration, *i.e.* for a configuration where the number of sources implementing FEC is small. This observation is maintained for the PLR of RED/FEC in this case.

As expected and as shown by [11], we observe that the PLRBC for RED is greater than the PLRBC for Drop Tail since RED starts dropping packets earlier without reaching the buffer capacity of the queue. Nevertheless, after the correction of lost packets, RED/FEC out-performs DT/FEC and gives a lower PLR for a small number of TCP flows. For a larger number of flows, the situation is reversed: RED yields a worst performance. Indeed, for one packet of redundancy ($h = 1$) and $k = 16$ data packets, *i.e.* for an addition of 6% of load, Drop Tail can divide the PLRBC by about 1.9 for 10 TCP flows. RED does better by dividing the PLRBC by about 3.4. In the case of a 25% load increase, for $h = 4$ and $k = 16$, Drop Tail can divide the PLRBC by about 33.4 for 10 TCP flows. In this case, the correction rate of RED is more significant since it is able to divide the PLRBC by 66.5 for 10 TCP flows and by 79.6 for 30 TCP flows.

These results show that the number of TCP flows under which RED experiences an improvement on the PLR as compared to Drop Tail depends on the amount of redundancy. Hence, this threshold

number increases as the amount of redundancy increases. For instance, this number is: 45 flows for $h = 1$ and 80 flows for $h = 4$. But the increase becomes slower as h grows. We have therefore shown that even if RED increases the UDP packet loss rate (which is in accordance with previous studies [11,12]), it becomes possible to reduce its PLR by using FEC and to obtain a PLR less than the PLR for Drop Tail under certain conditions (precisely for a certain number of TCP flows and a certain amount of redundancy).
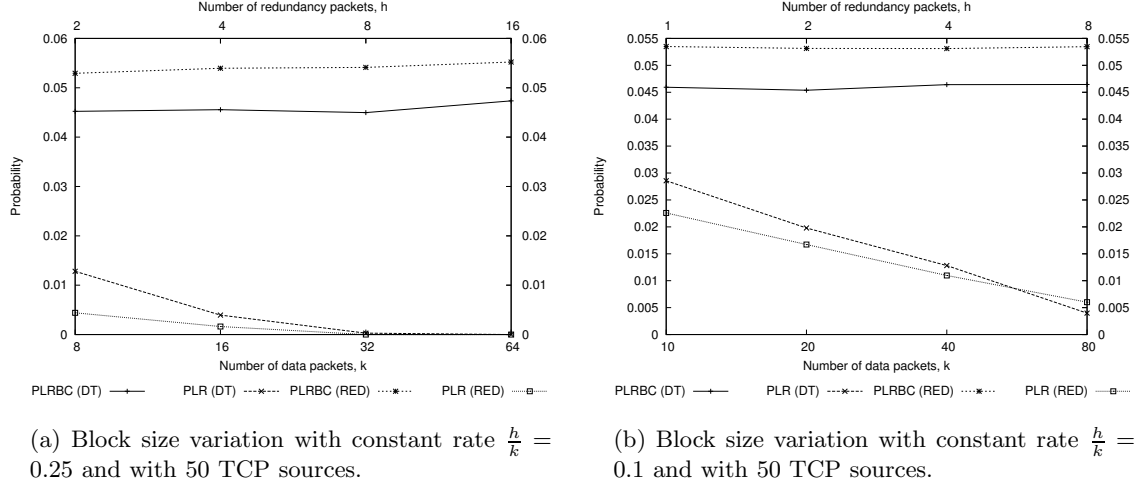
4.2 Loss run length

We have also studied the distribution of the loss run length, for both queue management mechanisms and in function of the cross traffic. The results fully reported in [13] showed that for $k = 16$ and $h = 1$, under the RED scheme, the probability to have a loss run length of size 1 packet is much larger than under Drop Tail (about 90% against 60%, respectively). This holds whatever the number of flows: the distribution does not appear to depend much on the cross traffic. This last observation shows that the situation is not as simple as initially thought. Our starting assumption was: if RED shows a smaller loss run length, then FEC will perform better with RED than with Drop Tail concerning the capacity of repairing lost packets. This turns out not to be valid for a large cross traffic, although the loss run length of RED is small throughout the range of experiments. We develop in Section 5 a model which shows that there is actually an efficiency tradeoff between the *size of bursts* of lost packets, and their *frequency*.

4.3 Influence of redundancy and FEC block size

In this experiment, we made the size of the FEC block vary, while maintaining the rate h/k constant, in order to obtain the same load increase for the UDP flow and therefore maintain the same overall network load. This way we can directly observe the influence of the FEC block size without the interference of the network load. We have also studied in [13] the case of a variable UDP load.

The load of the system being constant in this case, the PLRBC is fixed for both RED and Drop Tail as illustrated by Figure 3.

(a) Block size variation with constant rate $\frac{h}{k} = 0.25$ and with 50 TCP sources.

(b) Block size variation with constant rate $\frac{h}{k} = 0.1$ and with 50 TCP sources.

Fig. 3. Influence of the block size.

However, the PLR decreases when k (and therefore h) increases for both queue management schemes. This means that for this configuration of the system, it is interesting to increase the FEC block size. We also notice in Figure 3 that RED/FEC appears to be more advantageous than DT/FEC concerning the PLR under certain conditions depending on the number of TCP flows, the FEC block size and the amount of redundancy. This is the case in Figure 3(a), even though the PLRBC of RED/FEC is larger than the one obtained for DT/FEC. Indeed, RED queue management is able to give a slight performance improvement.

To summarize, all these results suggest that in case of a constant UDP offered load, the larger the block size, the better the correction rate is for both RED and Drop Tail. It is more interesting to use FEC with RED instead of Drop Tail in case of small number of flows, moderate FEC block size, and a relatively important amount of redundancy. Indeed, our model developed in Section 5 confirmed that RED should be more efficient if the redundancy ratio h/k is sufficiently large. Additional experiments conducted in [13] have shown that when the cross traffic is large, RED/FEC loses its advantage over DT/FEC whatever the FEC block size and the amount of redundancy.

5   A Model for FEC and its Application

We develop in this section a simplified model which is able to explain most of the phenomena observed above. This model concentrates on the sequence of packets and among them, the lost packets.

The sequence numbers $m_1, m_2, \ldots$ of the lost packets can be viewed as the instants of an "arrival" process. Call a *block* of losses a set of packets lost consecutively. If the size of blocks of losses is small compared with the time between two of these blocks, then an approximation of the situation is a *batch* arrival process, where several packets are lost simultaneously.

The most tractable of such processes is the one where grouped losses occur according to a Poisson process of rate λ. Assume that the size of the m-th group is a random variable $A_m > 0$. The sequence $\{A_m\}_m$ is assumed to be i.i.d., and we shall use the notation A for the generic random variable. The resulting process is a compound Poisson process, and the distribution of N_T, the total number of lost packets in the interval $[0, T]$, can be computed as:

$$P(N_T \leq h) \;=\; \sum_{m=0}^{\infty} \frac{(\lambda T)^m}{m!} e^{-\lambda T} P(A_1 + \ldots + A_m \leq h) \;. \quad (1)$$

The expected number of losses per unit time, which is to be interpreted as a loss rate of packets, is $p = \lambda E(A)$.

Consider now two situations where the distribution of the batch size differs, but where the average loss rate p is the same. Quantities referring to situation i will be superscripted with "(i)". Coming back to the focus of this paper, we consider that the first situation is RED/FEC and that the second is DT/FEC. We choose a simple distribution for the batch size: $A = 1$ with probability β and $A = 2$ with probability $1 - \beta$. The frequency of blocks of losses of size 1 is therefore β. According to the measurements of Section 4.2, the situations of RED and DT correspond approximately to values $\beta^{(1)} = 9/10$, and $\beta^{(2)} = 6/10$, respectively.

Assume a certain fixed T and some integer h, interpreted as the length of some FEC block, and the quantity of redundancy it contains, respectively. We are interested in the difference between the probabilities of repairing this FEC block in both situations, when

the packet loss rate is p. It can be expressed as:

$$\Delta_h(x) \; = \; P(N_T^{(1)} \leq h) \; - \; P(N_T^{(2)} \leq h) \;, \tag{2}$$

where both probabilities are obtained using (1) with $\lambda = p/m^{(i)}$ (denoting $m^{(i)} = E(A^{(i)})$), since we have assumed the same loss rate p in both situations. The difference is a function of $x = pT$, the average number of lost packets in the interval $[0, T]$.

Plotting the functions $\Delta_h(x)$, we find that for each h there exists a threshold value x_h such that $\Delta_h(x) \geq 0$ when and only when $x \leq x_h$. This difference is therefore positive if $x = pT$ is small enough (RED/FEC has a better performance), negative if x is large (DT/FEC has a better performance).

The explanation for the shape of the functions Δ_h is found analyzing Equation (1). Indeed, when $x = pT$ is small, the difference is dominated by polynomial terms in x, with coefficients $P(A_1^{(i)} + \ldots + A_m^{(i)} \leq h)$. Because $A^{(1)}$ is stochastically larger than $A^{(2)}$, this implies that $P(N_T^{(1)} \leq h)$ is larger. Here, the fact that blocks are smaller is important. On the other hand, if x is large, the terms in (2) are dominated by $e^{-x/m^{(i)}}$. Since $m^{(1)}$ is smaller, $P(N_T^{(1)} \leq h)$ decreases faster and gets smaller than $P(N_T^{(2)} \leq h)$. Here, the fact that blocks of losses are less frequent is more important.

Computing numerically these threshold values further reveals that when h is large enough, $x_h \simeq h + C$ where C is some positive constant value. Using this empirical finding, and since the size of the block T is $k + h$, we have: RED/FEC is better if

$$p(k + h) \; \leq \; h + C \iff k \; \leq \; \frac{1-p}{p}\, h + \frac{C}{p}$$
$$\iff \frac{h}{k} \; \geq \; \frac{p}{1-p} - \frac{C}{1-p}\frac{1}{k} \;.$$

This model allows therefore to predict that: a) for a given quantity of redundancy h, RED/FEC is more efficient only for a block size k small enough; b) RED/FEC is more efficient only if the redundancy ratio h/k is large enough; c) the larger the loss rate p, the smaller the block size k of RED/FEC can be. All these qualitative predictions are confirmed by simulation experiments reported in Section 4.3 or

in [13]. For instance, we have in Figure 3 situations where RED/FEC as the advantage for block sizes small enough. In other cases reported in [13], h/k is too small to compensate the loss rate.

To conclude, we observe that this preliminary model is too crude to produce accurate quantitative predictions, at least for the network simulated in Section 4. In addition, in these experiments, the loss rate (or PLRBC) p is not the same for both situations. We believe however that this first simple model reproduces adequately the principal features of the problem. Moreover, it can be checked that the principal conclusions hold when the loss rate of RED is increased (the range of parameters where RED/FEC performs better is then reduced). We have also found that an analysis based on the classical Gilbert model also exhibits the performance inversion which we have discussed here.

6 Conclusions and Perspectives

In this paper, we have studied the effect of a forward error correction (FEC) code on queue management schemes like Drop Tail and RED. It should be noted that to the best of our knowledge, no study has been conducted so far concerning FEC combined with a RED-like active queue management scheme.

It has been shown in literature that RED losses are spread as compared to Drop Tail. For this reason, one can assume that FEC would be more efficient combined with RED than with Drop Tail. But our results have also shown that RED/FEC does not always perform better than DT/FEC. This turns out to depend on certain parameters, in particular on the number of TCP flows that constitute the background traffic, the FEC block size and the amount of redundancy in a FEC block.

Our results suggest that RED/FEC performs better than DT/FEC when the loss rate is not too large, when there is a relatively important amount of redundancy in the FEC block, and a moderate FEC block size is used.

From the point of view of the queue manager, our results suggest the possibility to swap between RED and Drop Tail depending on the scenario parameters. For instance, based on the estimated number of TCP flows or the observed PLRBC, and knowing that FEC

is implemented in the UDP flows, the queue manager can take a decision. If the PLRBC is high, it is more advantageous to use Drop Tail. Otherwise, when the PLRBC is low, then RED scheme can be used by following the guidelines described just above.

Our next step will be to refine the model of Section 5 in order to investigate further the phenomenon we have observed, in which the correction capacity of FEC is worst when coupled with RED, despite the fact that losses are almost always isolated. First, it is necessary to prove the properties on the threshold values x_h which we have only empirically described. Next, we will investigate whether these properties hold with more general batch size distributions, and can be exploited to obtain decision rules concerning the use of FEC.

References

1. Byers, J., Luby, M., Mitzenmacher, M., Rege, A.: A digital fountain approach to reliable distribution of bulk data. In: Proc. ACM SIGCOMM, Vancouver, Canada (1998) 56–67
2. Cidon, I., Khamisy, A., Sidi, M.: Analysis of packet loss processes in high-speed networks. IEEE Transactions on Information Theory **39** (1993) 98–108
3. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking **1** (1993) 397–413
4. Braden, B., et al.: Recommendations on queue management and congestion avoidance in the Internet (RFC 2309). IETF. (1998)
5. McAuley, A.: Reliable broadband communications using a burst erasure correcting code. In: Proc. ACM SIGCOMM'90, Philadelphia, PA, USA (1990) 297–306
6. Floyd, S.: Discussions on setting RED parameters (1997) `http://www.aciri.org/floyd/red.html`.
7. NS: Simulator homepage (2004) `http://www.isi.edu/nsnam/ns`.
8. Brandauer, C., Iannaccone, G., Diot, C., Ziegler, T., Fdida, S., May, M.: Comparison of tail drop and active queue management performance for bulk-data and web-like Internet traffic. In: Proc. ISCC'01, Hammamet, Tunisia (2001)
9. Sanneck, H., Carle, G.: A framework model for packet loss metrics based on runlengths. In: Proc. Multimedia Computing and Networking Conference (MMCM), San Jose, CA, USA (2000) 177–187
10. Biersack, E.: A simulation study of forward error correction in ATM networks. Computer Communication Review **22** (1992) 36–47
11. Bonald, T., May, M.: Drop behavior of RED for bursty and smooth traffic. In: Proc. IEEE/IFIP IWQoS'99. (1999)
12. May, M., Bonald, T., Bolot, J.: Analytic evaluation of RED performance. In: Proc. INFOCOM'00. (2000)
13. Alemu, T.: Performance evaluation of quality of service mechanisms in the Internet. PhD thesis, University of Montpellier II (2004)