The interaction of Forward Error Correction and Active Queue Management

Tigist Alemu, Yvan Calas, Alain Jean-Marie

LIRMM – 161 Rue Ada, 34392 Montpellier Cedex 5, France
tigist@lirmm.fr calas@lirmm.fr ajm@lirmm.fr

Résumé : The aim of this paper is to study the interaction of a forward error correction (FEC) code combined with the 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 number of TCP flows that constitute 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, and a relatively important amount of redundancy and at most a moderate FEC block size is used. On the other hand, the use of DT is more efficient for larger loss rates.

Mots-clés : Forward Error Correction, Active Queue Management, Performance Evaluation

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. The first technique called ARQ (Automatic Repeat reQuest) consists in retransmitting dropped packets upon the destination's request. The second technique, called FEC (Forward Error Correction) 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 audio and/or video applications requiring strong time constraints. This is why FEC is increasingly used in such applications, typically on top of the UDP transport protocol. 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 [Cidon *et al.*, 1993].

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 have been proposed recently as an alternative [Braden & al., 1998], aimed at eliminating deficiencies of Drop Tail. Those schemes basically discard packets earlier so that incipient stages of congestion can be detected. For instance, the Internet Engineering Task Force recommended the use of RED [Floyd & Jacobson, 1993, Braden & al., 1998], an active queue management scheme which is able to achieve high throughput and low average delay (for TCP traffic) by spreading randomly packet drops between flows.

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 reducing consecutive losses for a given flow, thereby making losses "more independent". It may therefore be interesting to add a "moderate" amount of redundancy in presence of RED queue management in order to decrease the packet loss rate for the UDP source as compared to Drop Tail. Of course, this must be done without penalizing the TCP flows.

Indeed, in case of an increase of redundancy, *i.e.* an increase of the network load, the TCP sources respond by decreasing their throughput whereas the UDP flow does not.

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 conclusions and perspectives.

2 Forward Error Correction 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) generates h redundant packets useful for the recovery of the lost data packets. The concatenation of the k data packets and the h parity packets is called a *FEC block* of size $n = k + h$. If these k data packets are transmitted without loss to the destination, it is not useful to recover the possibly lost redundant packets as they do not contain relevant data. It is only in case of loss of data packets that packet recovery is required. 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 in this paper 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 (see [Floyd & Jacobson, 1993, Floyd, 1997] for details).

3 Experimental setup

We describe in this section the experimental setup we have used for this performance study. We have made extensive simulations using the `ns-2` simulator [ISI, 2004]. We begin with the network elements used in the simulation, and proceed with the definition of the performance metrics we have collected.

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 $100ms$.

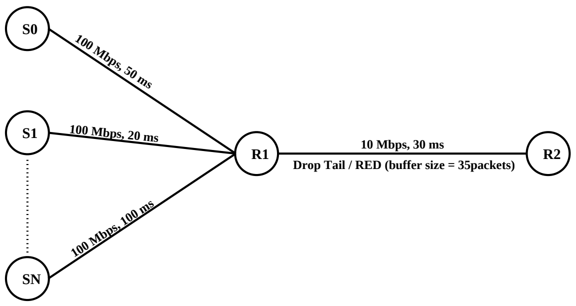


FIG. 1 – Topology of the system.

We have studied a traffic mix of TCP and UDP flows, with UDP representing a minority 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 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 distribution 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 $\rho_1 = 500kb/s$. 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 :

$$\rho_{FEC} \; = \; \rho_1 \left(1 + \frac{h}{k}\right) \; . \tag{1}$$

Under `ns` this increase of the load is obtained by decreasing the average idle time between packets. This parameter is computed as follows :

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

where the packet size for the UDP traffic is set to 573 bytes following the recommendation of [Brandauer *et al.*, 2001]. 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 superposition 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 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 5 independent simulations.

3.2 Performance metrics

We considered the aggregate traffic performance as well as the individual flow performance. The metrics used for the aggregate traffic are :

- The *average queueing delay* inferred by the average of the instantaneous queue size. It is interesting to minimize queueing delay for applications with strong real-time constraints.
- The *delay jitter* corresponding to the variance of the instantaneous queue size. Minimizing delay jitter is useful for applications such as audio and video.
- The *aggregate throughput*, that is the amount of bytes of information correctly transmitted to the destination per unit time, regardless of the flow identity.

Observe that since the bottleneck link schedules packets in the FIFO order, the aggregate average queueing delay is also the average queuing delay of UDP, and of TCP packets. Likewise, the delay jitter for UDP flows, the most interesting in practice, is equal to the aggregate average jitter. In addition to aggregate metrics, consider the following metrics for a specific FEC flow :

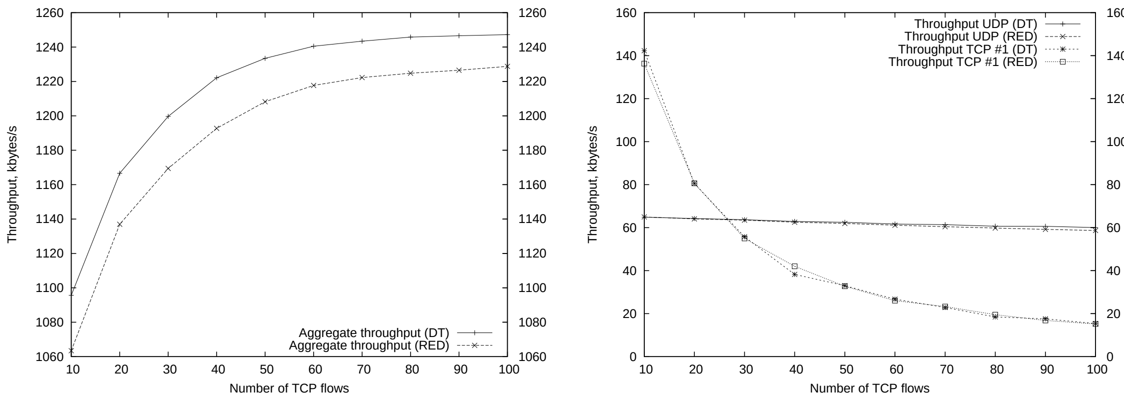
- The *packet loss rate before correction* ($PLRBC$) that is the ratio of the average number of lost packets in a FEC block before correction to the size of the FEC block.
- The *packet loss rate after correction* (PLR) that is the ratio of the average number of lost packets in a FEC block after correction to the size of the FEC block.
- The *loss run length* [Sanneck & Carle, 2000] 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

This section presents the results of our experiments. We first look at the influence of the cross traffic, represented by the number of TCP flows, on the efficiency of RED/FEC and DT/FEC (Section 4.1). We check in Section 4.1.3 that the theoretical performance advantages provided by RED are preserved in our experiments. Measurements on the Loss Run Length are presented in Section 4.1.4. Next, we study in Section 4.2 the influence of the amount of redundancy on the performance.

4.1 Influence of the number of TCP flows

4.1.1 Throughput



(a) Aggregate throughput. (b) Per-flow throughput.

FIG. 2 – Throughput as a function of the cross traffic for $k = 16$ and $h = 1$.

As shown by Figure 2(a), Drop Tail offers a higher throughput than RED (about $20kbytes/s$) whatever the number of TCP flows. The increase of the offered load due to the addition of a slight increase of redundancy ($h \leq 4$) for the UDP flow does not affect significantly the overall throughput.

Figure 2(b) also shows that whatever the type of buffer management used, the UDP flow remains almost insensitive to the increase of the number of TCP flows since we observe a very light decrease of the throughput. On the other hand, the TCP flows are constrained to reduce their throughput (from $120kbytes/s$ for 10 TCP flows to $20kbytes/s$, each for 100 TCP flows).

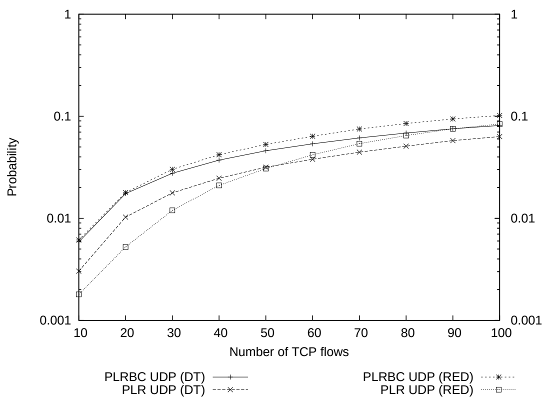
4.1.2 Packet loss rate for the UDP source

Figure 3 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.

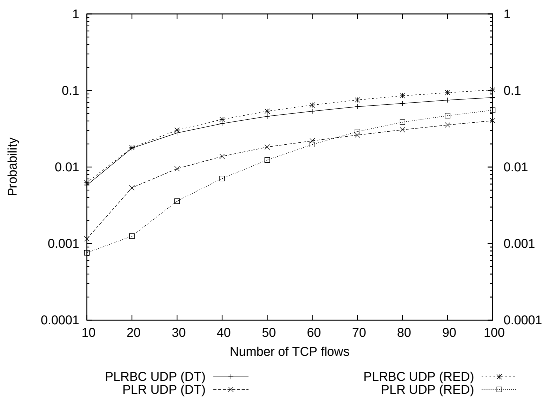
As observed in [Biersack, 1992, Calas & Jean-Marie, 2003] and as shown also in Figure 3, when the amount of redundancy is increased, the PLR decreases for this network configuration, *i.e.* for a configuration where the number of sources implementing FEC is smaller than the number of sources not implementing FEC. This observation is maintained for RED for this network configuration.

As expected and as shown by [Bonald & May, 1999], 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.

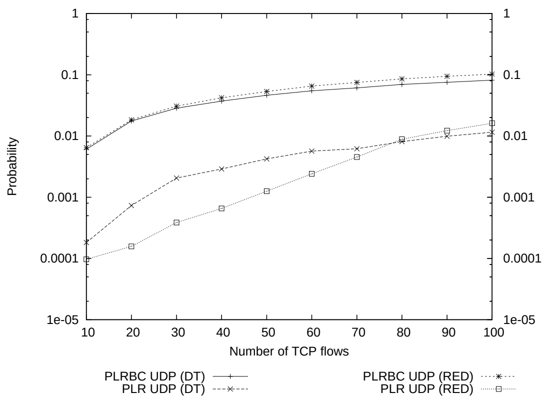
Indeed, for one packet of redundancy ($h = 1$) and $k = 16$ data packets, *i.e.* for an addition of 6% of load, Drop Tail can divide the PLRBC by about 2.2 for 10 TCP flows. RED does better by dividing the loss rate by about 4.3. In the case of a 25% load increase, for $h = 4$ and $k = 16$, Drop Tail can divide the PLRBC by about 33.4 for the same number of TCP flows. In this case, the correction rate of RED is more significant. RED is able to divide the PLRBC by 66.5 for 10 TCP flows and by 79.6 for 30 TCP flows.



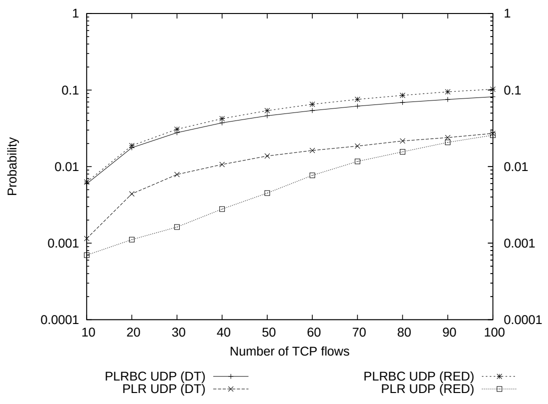
(a) $k = 16, h = 1$.



(b) $k = 16, h = 2$.



(c) $k = 16, h = 4$.



(d) $k = 8, h = 2$.

FIG. 3 – Packet loss rate before (PLRBC) and after (PLR) correction by FEC.

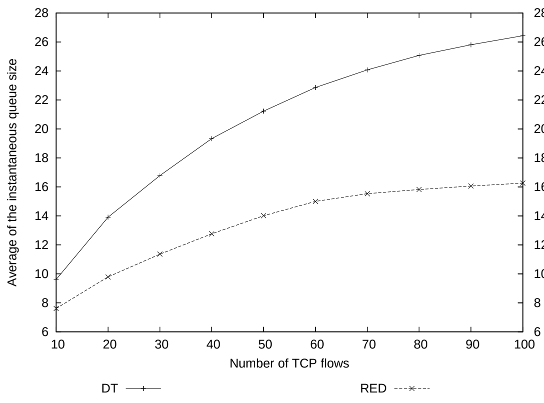
Moreover, we observe that the curve of the PLR of RED stands under the curve of Drop tail for a number of flows less than a certain threshold : 45 flows for $h = 1$ (Figure 3(a)), 65 flows for $h = 2$ (Figure 3(b)), and 80 flows for $h = 4$ (Figure 3(c)). In Figure 3(d), obtained with $k = 8$ and $h = 2$, the threshold is 100 TCP flows. Above these thresholds, FEC exhibits better performances with Drop Tail than with RED.

These results show that the number of flows under which RED experiences an improvement of the PLR as compared to Drop Tail depends on the amount of redundancy. This number of flows increases as the number of redundancy packets increases. But the increase becomes slower as h grows.

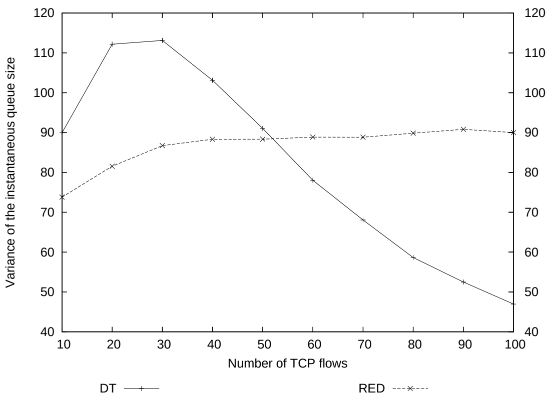
We have therefore shown that even if RED increases the UDP packet loss rate (which is in accordance with previous studies [Bonald & May, 1999, May *et al.*, 2000]), it becomes possible to reduce the PLR by using FEC and to obtain a PLR for RED less than the PLR for Drop Tail under certain conditions (precisely for a certain number of flows and a certain amount of redundancy).

4.1.3 Queuing delay and delay jitter

The advantage of RED over Drop Tail concerning the queuing delay is maintained with the addition of redundancy. The addition of redundancy does not influence this metric. Figure 4(a) shows that whatever the number of flows, RED improves the queuing delay as compared to Drop Tail. On the other hand, the improvement of RED concerning the delay jitter represented by Figure 4(b) depends on the number of flows. Indeed, for small number of flows the queue size variance of RED is lower than the one of Drop Tail. However, for large number of flows as the queue is always almost full we observe for Drop Tail a lower variance. Note that for few number of flows, FEC in conjunction with RED manages to reduce the PLR while exhibiting a lower queuing delay and delay jitter.



(a) Average queue size.



(b) Queue size variance.

FIG. 4 – Average and variance of the queue size $k = 16$ and $h = 1$.

4.1.4 Loss Run Length

We have displayed in Figure 5 the distribution of the Loss Run Length, for both queue management mechanisms and in function of the cross traffic. The different curves show the probabilities of losing consecutively 1, 2 or 3 packets. For this experiment, $k = 16$ and $h = 1$.

This figure confirms that the distributions are different under RED and DT. Under RED, the probability to have a loss run length equal to 1 packet is much larger than for Drop Tail (about 90% against about 60%, respectively). This holds whatever the number of flows, since 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 larger probability to have a loss run length of size one, 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. The conclusion must be that the fact that losses are isolated is not the only factor to take into account. The global loss rate (PLRBC), which is larger for RED, has also an impact. Preliminary analyzes, to be reported elsewhere, suggest that since bursts of lost packets are smaller for RED, they must be more frequent, which sometimes makes the FEC blocks more vulnerable under RED than under TD.

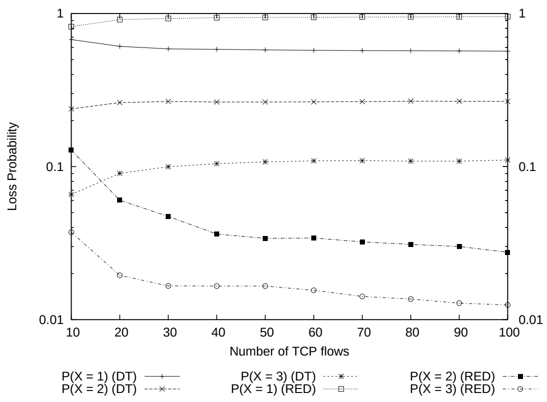


FIG. 5 – Distribution of the Loss Run Length for $k = 16$ and $h = 1$.

4.2 Influence of redundancy and FEC block size

In this section, we explore the effect of the FEC block size on the performance of RED/FEC or DT/FEC. To begin with, we consider a scenario where the number of data and redundancy packets are varied proportionally so as to maintain the UDP offered load constant. In the second scenario, the number of data packets is changed while keeping constant the number of redundancy, and the UDP offered load varies.

As it turns out, it is interesting to use large FEC block sizes in order to decrease the PLR. However, in practice the block size is limited due to the following constraints : with a larger block, it takes a larger delay for the reception of all packets belonging to the same block by the destination, and it takes more time for the coding (the generation of all redundancy packets) and the decoding of the FEC block (*i.e.* for the correction of all packets).

4.2.1 Fixed UDP load

In this section, we varied the size of the FEC block while maintaining the rate $\frac{h}{k}$ 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.

For instance, for Figures 6(a) and 6(b), we add 2 packets of redundancy for 8 data packets and proportionally 16 packets of redundancy for 64 data packets in order to obtain a load increase of 25% with respect to the situation without redundancy. For Figures 6(c) and 6(d), we add 1 packets of redundancy for 10 data packets and proportionally 8 packets of redundancy for 80 data packets in order to obtain a load increase of 10%.

The load of the system being constant, the PLRBC is fixed for both RED and Drop Tail as illustrated by Figure 6. 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.

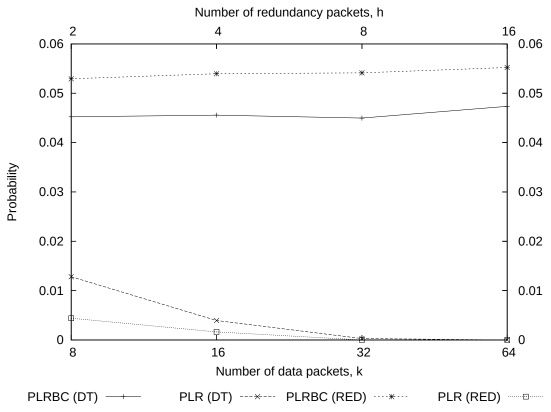
We also notice in Figure 6 that RED/FEC appears to be more advantageous than DT/FEC concerning the PLR under certain conditions depending on the number of TCP flows, the FEC block size and the amount of redundancy.

This is the case in Figure 6(a), even though the PLRBC of RED/FEC is higher than the one obtained for DT/FEC. Indeed, RED queue management is able to give a slight performance improvement : for instance, for $k = 8$ and $h = 2$, and 50 TCP flows, RED reduces the PLR by almost a factor of 2.9 as compared to Drop Tail. Moreover, the pair of values ($k = 8$, $h = 2$) shows a higher difference between the PLR of RED and the PLR of Drop Tail than the pair ($k = 16$, $h = 4$). Indeed, if we compare the intersection point between the curves of the PLR of RED and Drop Tail depicted by Figures 3(c) and 3(d), we note that the configuration with the pair of values ($k = 8$, $h = 2$) gives a better performance than the configuration with the pair ($k = 16$, $h = 4$) : this intersection point is obtained for 100 TCP flows for $k = 8$, $h = 2$ instead of 80 flows for $k = 16$, $h = 4$. Hence, in this case, RED appears to be more advantageous than Drop Tail if small sizes of FEC blocks are used. Nevertheless, if the block size is increased then this performance improvement is diminished and both queue management schemes show the same performance by repairing all lost packets.

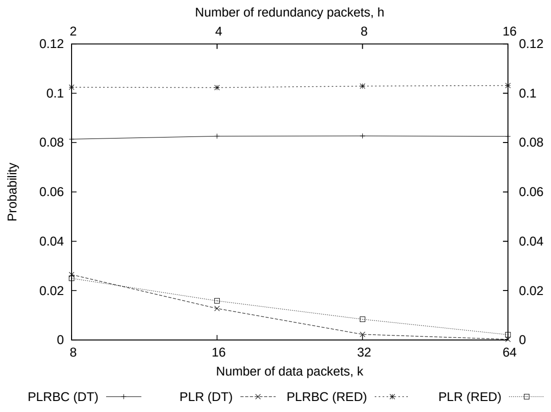
On the other hand, if we increase the number of TCP flows from 50 to 100 (Figure 6(b)), Drop Tail is more advantageous than RED. However the PLR difference of both queue management schemes is negligible (about 0.3% for $k = 16$ and $h = 4$) as compared to the difference of their PLRBC (2%). This shows that RED has managed to approach Drop Tail performances by repairing an important amount of lost packets. Indeed RED shows a higher correction rate since the absolute difference of the PLRBC and PLR is about 8.70% for RED whereas it is equal to 6.98% for Drop Tail.

Unlike the case presented in Figure 6(b), the difference between both PLR is more significant when we reduce the UDP load increase from 25% to 10% for the same number of TCP flows (100 flows). This is illustrated by Figure 6(d). For example, for $k = 20$ and $h = 2$, the absolute difference between the PLR for RED and Drop Tail is about 2%. Note that for this case, Drop Tail is far more advantageous than RED. This result can be explained by a higher rejection rate of RED combined with the presence of low redundancy in FEC blocks. In addition, when the FEC blocks are larger, their vulnerability is increased, which leads to worst performances.

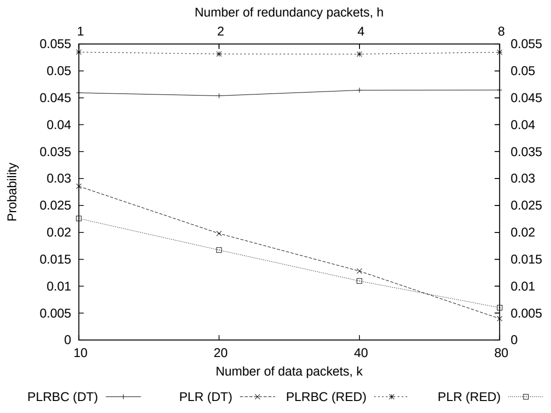
But even if the amount of redundancy is not large (10%), Figure 6(c) shows that if the number of TCP



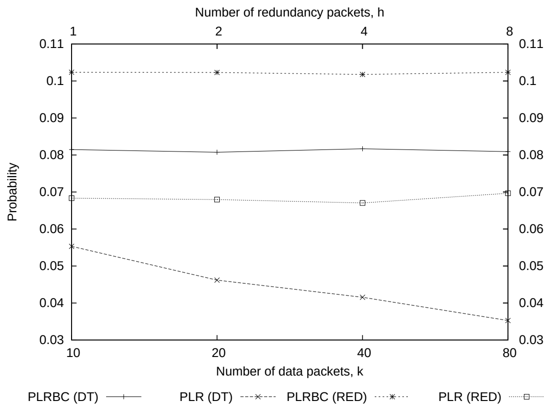
(a) Block size variation with constant rate $\frac{h}{k} = 0.25$ and with 50 TCP sources.



(b) Block size variation with constant rate $\frac{h}{k} = 0.25$ and with 100 TCP sources.



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



(d) Block size variation with constant rate $\frac{h}{k} = 0.1$ and with 100 TCP sources.

FIG. 6 – Influence of the block size variation.

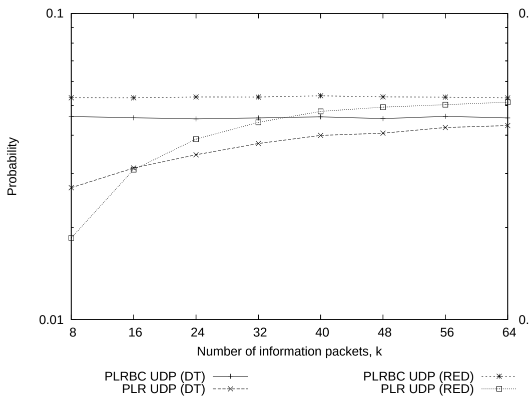
flows is not important (50 flows), then RED can be advantageous for a certain size of FEC block (approximatively for a block size smaller than a block containing $k = 40$ packets of information and $h = 4$ packets of redundancy).

To summarize, all these results suggest that in case of a constant UDP offered load, the larger the block size, the better correction rate, 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 size of FEC block and a relatively important amount of redundancy. If the cross traffic is large, RED/FEC loses its advantage over DT/FEC whatever the FEC block size and the amount of redundancy.

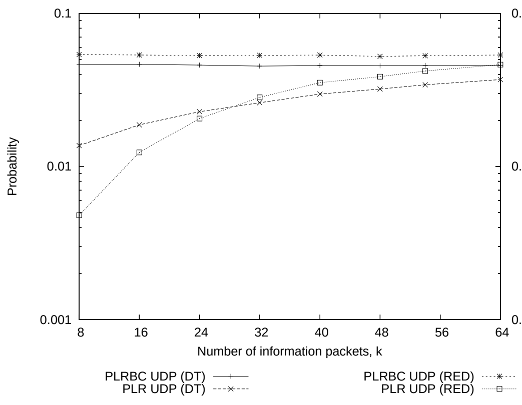
4.2.2 Variable UDP load increase

We now vary the size of the FEC block by increasing the number of data packets k and by keeping the number of redundancy packets h constant. Therefore, the FEC offered load ρ_{FEC} decreases when the number of data packets k increases (see Equation (1)). As expected, the offered load generated by the UDP flow decreases leading to a slight decrease of the PRLBC as illustrated by Figure 7. However, unlike the case presented in section 4.2.1, the PLR increases with k because as h remains constant and k increases, the FEC block becomes more vulnerable, *i.e.* the probability to repair all lost packets belonging to the same FEC block becomes lower.

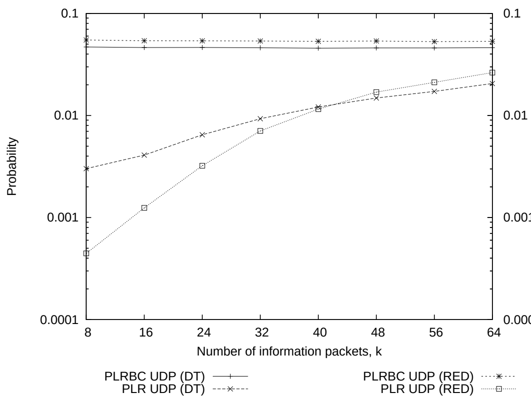
Figure 7 also shows that by increasing the FEC block size, RED is globally unable to offer an improved performance as compared to Drop Tail. However, for small sizes of FEC block that is for small number of data packets and fixed number of redundancy packets, RED does out-perform Drop Tail. In addition, we have noticed Figure 3 that there is an intersection point between the curves of the PLR of RED and the PLR of Drop Tail. This point moves when the number of redundancy packets increases. For instance, for $h = 1$ the intersection point is obtained for $k = 16$ which represents a tolerable UDP load increase of about 6.25%. On the other hand, for $h = 4$, the intersection point is obtained for $k = 40$ which represents a higher UDP load increase of about 10%.



(a) $h = 1$.



(b) $h = 2$.



(c) $h = 4$.

FIG. 7 – Packet loss rate before (PLRBC) and after (PLR) correction by FEC for $k = 16$.

Finally, these results showed that when the FEC block size increases, and the UDP offered load decreases, the PLR for RED and Drop Tail increases since the FEC blocks become more and more vulnerable. The results also show that RED reduces the PLR as compared to Drop Tail for small number of data packets belonging to a FEC block. The number of data packets below which RED out-performs Drop Tail can be increased when the number of redundancy packets is increased. Indeed, since RED drops packets earlier and increases the loss rate, as compared to DT, it needs a sufficient amount of redundancy in order to recover from the excess of losses. However, this addition of a large amount of redundancy packets increases the UDP source load and therefore may penalize the TCP sources.

5 Conclusions and perspectives

In this paper, we have studied the effect of a forward error correction (FEC) code on active queue management schemes like Drop Tail and RED. We have analyzed the situation where a small UDP traffic using FEC shares a link with a number of TCP sources. 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, *i.e.* RED has a higher probability of having a loss run length of size one. For this reason, one can assume that FEC would be more efficient combined with RED than with Drop Tail. Indeed, the results have shown that even though RED experiences more losses before FEC correction as compared to Drop Tail, RED can be more advantageous concerning the losses after correction. 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.

Using the PLR metric, we showed that RED/FEC is more efficient than DT/FEC for small number of TCP flows since a larger number of TCP flows increases the burstiness of the flows and also the network load, which in turn increases the loss rate for the UDP source implementing FEC. We also observed that if the loss rate before correction (PLRBC) is too high, RED/FEC gives worst performance as DT/FEC since it is unable to repair sufficiently many lost packets whatever the FEC block size and the amount of redundancy packets. The results also show that the number of TCP flows under which RED is advantageous increases with the amount of redundancy added in a FEC block. In addition, we checked that in the situation where RED is advantageous, the performance gains of RED such as queueing delay and delay jitter are maintained.

Moreover, our results show that for a fixed amount of redundancy packets in a FEC block, the advantage (respectively the disadvantage) of RED/FEC over DT/FEC is more (respectively less) important for small FEC block sizes than for large FEC block sizes.

In the case where RED/FEC is more advantageous than DT/FEC, the advantage of RED over Drop Tail decreases when the FEC block size becomes too large up to a point where DT/FEC becomes advantageous. This point is reached more or less quickly depending on the relative amount of redundancy contained in a FEC block.

In the case where RED/FEC gives worst performance as compared to Drop Tail, the performance difference between RED/FEC and DT/FEC increases with the FEC block size. The performance of RED/FEC degrades with the relative amount of redundancy contained in a FEC block.

All these results suggest that if the UDP flow implementing FEC has the knowledge of its PLRBC and if the traffic generated by this flow crosses a RED gateway, then it is preferable to conform to the following guidelines in order to obtain good performances of RED/FEC. For this purpose, we should be situated in the case where the PLRBC is not too high, that is in a case where the number of TCP flows is small. We can then increase reasonably the relative amount of redundancy in the FEC block without increasing the network load and penalizing TCP flows. In addition, increasing reasonably the FEC bloc size without increasing the delay of the reception of the block and its the time for coding and decoding improves further the performance of RED/FEC. We noticed that the amount of redundancy and the FEC block size above which RED loses its advantages is large. Since it is not practical to choose such large values, it is therefore preferable to opt for RED/FEC rather than for DT/FEC when the PRLBC is low.

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 make its 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 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 always isolated. We are currently developing analytical models as a way to determine the threshold values at which RED/FEC loses its advantage over DT/FEC.

Références

BIERSACK E. W. (1992). A simulation study of forward error correction in ATM networks. *Computer Communication Review*, **22**(1), 36–47.

BONALD T. & MAY M. (1999). Drop behavior of RED for bursty and smooth traffic. In *Proc. IEEE/IFIP IWQoS'99*.

BRADEN B. & AL. (1998). *Recommendations on Queue Management and Congestion Avoidance in the Internet*. Rapport interne RFC 2309, IETF.

BRANDAUER C., IANNACCONE G., DIOT C., ZIEGLER T., FDIDA S. & MAY M. (2001). Comparison of tail drop and active queue management performance for bulk-data and web-like Internet traffic. In *ISCC 2001*, Hammamet, Tunisia.

CALAS Y. & JEAN-MARIE A. (2003). *On the Efficiency of Forward Error Correction at the Packet Level*. Rapport interne 03-003, LIRMM, University of Montpellier II.

CIDON I., KHAMISY A. & SIDI M. (1993). Analysis of packet loss processes in high-speed networks. *IEEE Transactions on Information Theory*, **39**(1), 98–108.

FLOYD S. (1997). Discussions on setting RED parameters. http ://www.aciri.org/floyd/red.html.

FLOYD S. & JACOBSON V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, **1**(4), 397–413.

ISI (2004). Ns simulator homepage. `http://www.isi.edu/nsnam/ns`.

MAY M., BONALD T. & BOLOT J. (2000). Analytic evaluation of RED performance. In *Proc. IEEE INFOCOM'00*.

SANNECK H. & CARLE G. (2000). A framework model for packet loss metrics based on runlengths. In *Proc. Multimedia Computing and Networking Conference (MMCN)*, p. 177–187, San Jose, CA.