Is TCP Packet Reordering Always Harmful?

Giovanni Neglia[(∗)], Vincenzo Falletta[(∗)], Giuseppe Bianchi[(+)]

(∗) Dipartimento di Ingegneria Elettrica
Università degli Studi di Palermo, viale delle Scienze, ed. 9
90128 Palermo, Italy
{giovanni.neglia,vincenzo.falletta}@tti.unipa.it

(+) Dipartimento di Ingegneria Elettronica
Università degli Studi di Roma - Tor Vergata, Via del Politecnico,1
00133 - Roma, Italy
bianchi@elet.polimi.it

Abstract

IP networks do not provide any guarantee that packets belonging to the same flow are delivered in the correct order. It can be argued that out-of-order reception of packets is limited to pathological network conditions (such as link failures, etc). However, it has been experimentally proven in the past that packet reordering is a phenomenon which do occurs even in normal network operation, due to a number of link-level and/or router-level implementation features such as local parallelism and load balancing. Packet reordering is intuitively considered as a negative phenomenon, which may severely affect TCP traffic performance since it is expected to cause inefficient usage of the available link bandwidth and is expected to induce bursty transmission behaviour. Instead, in this paper, we show that a limited amount of reordering can improve network performance. To the authors' knowledge, this is the first paper which claims that TCP packet reordering, rather than being harmful, may be a beneficial phenomenon in terms of overall network performance. In order to justify this perhaps counter-intuitive result, in addition to extensive simulation results, we present a theoretical justification, by providing an analogy with the performance improvements experienced when TCP flows encounter a small dropping probability.

Introduction: Packet Reordering

Packet reordering is a phenomenon which occurs when packets belonging to a same flow are received in a different order than the packet transmission one[1]. Packet reordering was originally considered to be an uncommon event, occurring only in pathological network conditions, e.g. caused by router malfunctioning, route flapping, etc. However, recent studies have shown that in present IP networks there are several causes of reordering under absolutely normal network operation.

Previous research work (Bennett & Partridge & Shectman, 1999) has identified two major causes for the occurrence of packet reordering: local parallelism and load balancing.

Local parallelism implies that packets may follow multiple paths within a device or logical link. Examples of local parallelism are link-level striping and switches which allow packets traveling between the same source and destination to take different paths through the switch. Local parallelism is an increasing phenomenon in modern Internet devices, because it leads to reduce the cost of equipments and trunks. It is often more cost effective to put two components in parallel than to use one component that has twice the speed. A concrete example of local parallelism at the data link layer is represented by the 802.3ad link aggregation control protocol, today extensively used in giga-ethernet switches.

Load balancing means distributing processing and communications activity evenly across a computer network so that no single device is overwhelmed. For example IS-IS, the Interior Gateway Protocol (IGP) used by Sprint provides a load balancing mechanism based on link weights (Iannaccone & Jaiswal & Diot, 2001) and can cause reordering.

The amount of reordering in IP networks is quite debated: early experimental results (Bennett et al., 1999) suggest that the probability of a session experiencing reordering is over 90%. More recent results (Iannaccone et al., 2001, Iannaccone, & Jaiswal, & Diot, & Kurose, & Towsley, 2003) state that the same probability appears to be below 3% and the number of packets that are reordered is no more than 2%. Anyway, whatever the amount of reordering is, reordering is usually considered harmful for TCP flows. In fact, in TCP, packets received out of order cause the transmission of a duplicate ACKs, which interfere with the normal operation of the TCP congestion control algorithm. Due to reordering, TCP flows may experience a great difficulty in opening their congestion windows and may end up in making inefficient usage of the available link bandwidth. Moreover, TCP flows may lose self-clocking and become highly bursty. The reader can refer to (Bennett et al., 1999) for a detailed overview of the effects caused by packet reordering. Here, we simply summarize that forward-path reordering has five side-effects: 1) it may trigger fast retransmission and cause unnecessary retransmissions; 2) it may trigger fast recovery and cause unnecessary slowdown of TCP window growth (cwnd and ssthresh); 3) it may obscure actual packet losses; 4) it may cause the round-trip estimator to poorly estimate the roundtrip time; and 5) it may reduce the efficiency of the receiving TCP. Reverse-path reordering has one major effect: a loss of self-clocking leading to highly bursty transmission patterns.

Notwithstanding these negative effects of reordering, the goal of this paper is to show, via extensive simulation

[1]Hence packet reordering refers to the disorder caused by the network and not to the packet resequencing problem considered often in queueing theory papers like (Kamoun & Kleinrock & Muntz, 1981).

results, that a limited amount of reordering may improve the network performance in terms of delay versus utilization. In order to explain these apparently surprising and counter-intuitive results, we carry out an analogy with a simplified system model where TCP sources experience an artificially induced steady-state dropping probability. This simplified scenario can be analytically evaluated, and leads us to conclude that, in the presence of a "small" (to be quantified in this paper) packet loss probability, an improvement in terms of network performance is expected.

The rest of this paper is organized as follows. In the next section, we present the simulation scenario and parameters, focusing in particular on the way reordering has been obtained. We then follow up by presenting simulation results which show that packet reordering may have a beneficial effect on the network performance. Then, we try to provide a theoretical justification for the obtained results by making an analogy with a system characterized by a given steady-state dropping probability. Finally, in the last section, conclusive remarks are drawn.

Simulation Scenario

We have run simulations using ns-2.1b9a (Network Simulator). In order to introduce packet reordering we have modified an existing ns module ("*hiccup*" by Morten Schlaeger, Technical University of Berlin). Hiccup provides functionalities to simulate link outages and segment reordering without loosing a packet. The hiccup class is derived form the ns queue class and integrated into the ns link object. The hiccup operates in four different modes. In HICCUP_IDLE mode, all packets are directly passed to the next link object. In HICCUP_DELAY mode packets are queued until the mode is changed back to HICCUP_IDLE, then all packets are passed to the neighbor object in a single, no time consuming, burst. In HICCUP_RESORT mode, the queueing of a single packet is delayed until `resort_len_` later packets are queued. The last mode, HICCUP_CONG, allows to drop packets. The hiccup module drops all packets until it is set to a different mode.

In our study we employed *hiccup* only in HICCUP_RESORT and HICCUP_CONG modes. The hiccup operation in HICCUP_RESORT mode allows one to obtain specific packet lag[2] patterns. Despite of this, one can shortly see that this behavior is not directly related to the physical phenomenon that causes packet reordering. Besides it can produce spurious timeouts when the hiccup module operates on a per-flow basis (as in our simulations), in fact the hiccup module could wait for new packets before transmitting old ones, while the TCP window do not allow the sender to transmit them.

In order to overcome this problem and to reproduce a more realistic behavior we modified the HICCUP_RESORT mode so that it introduces a tunable per-packet random delay. The random delay is the sum of an exponential random variable with expected value τ and a constant term equal to $T-\tau$. So τ can be changed, but the average delay introduced by hiccup is constant and equal to T. Finally we have also modified the way the module works in HICCUP_CONG: packet are dropped independently with a constant probability (p_s) selected during the simulation set-up.

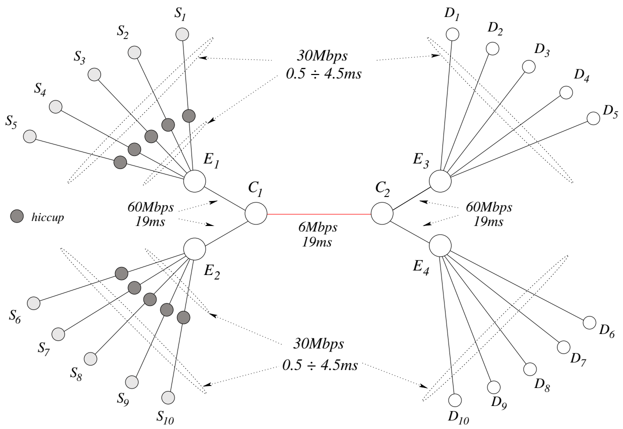

Figure 1: Network topology

Our analysis has been developed on the network topology shown in figure 1. The central part has a simple structure, with a 6Mbps single-bottleneck link between the core routers C_1 and C_2, where C_1 implements RED.

We have set up 10 sources ($S_i, i = 1, 2, ...10$), each one connected to one of the edges E_1 and E_2 by a 30Mbps link. We considered long-lived flows. The sources employ TCP Reno and have always data to transmit to destinations ($D_i, i = 1, 2, ...10$), so the throughput is determined only by the network conditions.

A *hiccup* module has been located between each source and the edge. It introduces an average delay $T = 6$ ms.

In order to avoid synchronization among the sources, each source starts to transmit randomly in the interval 0-1 s, and propagation delays of the access links are chosen so that Round Trip Time are different (from 124ms to 156ms, the average value is 140ms).

Each router deploys RED (Floyd & Jacobson, 1993) as Active Queue Management. RED gateway calculates the average queue size (avg), using a low-pass filter with an exponential weighted moving average of the instantaneous queue (x): $avg = w_q x + (1 - w_q)avg$. The average queue size is compared to two thresholds, a minimum threshold (min_{th}) and a maximum threshold (max_{th}). When the average queue size is less than the minimum threshold, no packets are dropped. When the average queue size is greater than the maximum threshold, every incoming packet is dropped. When the average queue size is between the minimum and the maximum thresholds, each arriving packet is dropped with probability p, where p is a linearly increasing function of the average queue size. RED configuration is hence specified through three parameters: the minimum and the maxi-

[2] In (Iannaccone et al., 2001) the packet lag of a reordered packet is defined as the number of packets with a sequence number greater than the one of the reordered packet, that are seen before the reordered packet itself; if the hiccup module operates on a per-flow basis, the packet lag coincides with the `resort_len_` parameter.

Table 1: RED thresholds settings

min_{th} (packets)	max_{th} (packets)
8	24
16	48
24	72
32	96
48	144
64	192
96	288

mum threshold and the maximum dropping probability in the region of random discard (P_{max}). The thresholds and P_{max} are chosen according to (Floyd, 1997), the filter coefficient w_q according to (Floyd & Gummadi & Shenker, 2001), i.e. $max_{th} = 3min_{th}$, $P_{max} = 0.1$ and $w_q = 1 - exp(-M/(10 * RTT * C)) = 0.0012$, where C is the link capacity, M is the packet size and RTT is the Round Trip Time.

RED configuration allows the network provider to trade off link utilization and delay performance: the higher the RED thresholds, the higher link utilization and delay. Different settings were considered and they are reported in table 1.

Lastly queue physical lengths were chosen so that packet losses occurred only in the core router C_1, due to RED (not to physical queue overflow).

For each configuration at least 10 simulations with different random seeds were run. Each simulation lasted 1000 simulated seconds, statistics were collected after 50 seconds.

Simulation Results

The thorough performance evaluation of TCP traffic in the presence of RED routers is by no means a simple task. In fact, the specific configuration of the RED thresholds is proven to strongly affect the TCP traffic performance in terms of throughput and delay. Rather than pre-selecting a given RED threshold configuration (and thus evaluate the TCP traffic performance for the very specific RED configuration chosen), we have found very useful in the past (Neglia & Bianchi & Sottile, 2003) to rely on the so-called concept of "performance frontier".

A performace frontier is a delay versus utilization plot, where different network utilization levels are obtained via different settings of the RED thresholds. As regards link utilization we have not considered the throughput of the TCP sources, but rather the goodput, i.e. the data amount delivered to the applications at the receivers, without losses and retransmissions, because it is more significant from the user point of view. Figure 2 shows the effect of reordering on the achievable performance frontiers. Each curve represents the average queueing delay versus the link utilization (goodput), for a specific

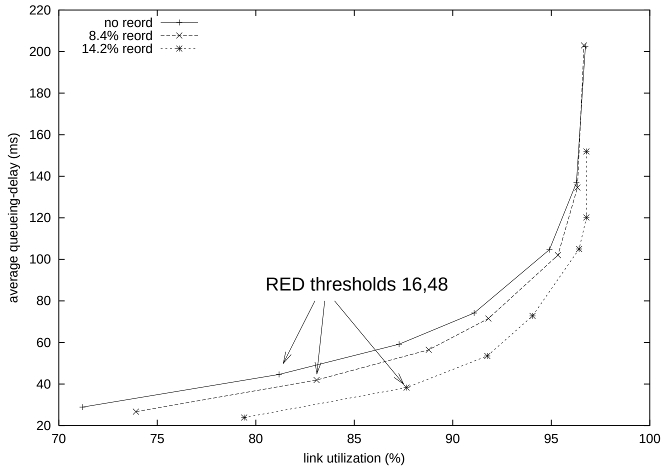

Figure 2: Delay vs link utilization with packet reordering

setting of the *hiccup* module. In particular $\tau = 0, 2, 4$ ms have been chosen respectively for the continuous curve, the dashed one and the dotted one. The resulting percentage of packet reordering is reported in the legends of the figure[3]. Each point in a given curve corresponds to a specific configuration of the RED thresholds. The arrows in the figure show the points corresponding to a particular RED configuration ($min_{th} = 16, max_{th} = 48$). The other points are obtained with the RED threshold configurations summarized in table 1.

We can note that, as reordering increases, each point moves towards lower queuing delay and higher link utilization. At first sight, we could be surprised seeing the frontiers moving right-down when we increase reordering, since this appears as an overall improvement of network performance. On the contrary we would expect a disturb like reordering to cause impairments to TCP operation.

An analogy: constant dropping probability

In order to explain this counter-intuitive result, we consider the effect of another form of disturb: a steady-state dropping probability independent from the network congestion status, as it could be introduced by a wireless link. It is advantageous because a lot of research has been done on the effect of random losses on TCP performance and different formulas are available which relate the throughput of a TCP source to the network dropping probability and the average Round Trip Time (RTT).

In order to introduce an additional dropping probability we configured the *hiccup* module in HICCUP_CONG mode. In figure 3, the performance frontiers for different values of the dropping probability p_s appear similar to those in figure 2.

[3] Actually, as τ increases, the packet reordering probability increases. Being τ constant, the reordering probability decreases slightly as we move from left (lower RED thresholds) to right (higher RED thresholds). Hence, to avoid complex notation, we have found simpler to label each curve with the average value of reordering probability.

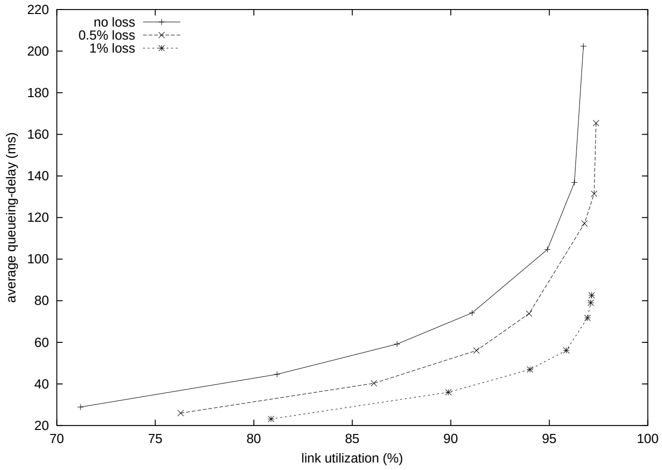

Figure 3: Delay vs Link Utilization with packet dropping

The improvement for high threshold values is essentially a decrease of the average queueing delay. This reduction can be easily explained by the relation between throughput, average buffer occupancy (q) and packet dropping probability (p), we indicate it with $T(q, p)$ (see for example (Padhye & Firoiu & Towsley & Kurose, 1998) for a common formula). $T()$ is a decreasing function of q and p. Usually packet discards are caused by routers, hence in well-configured AQM mechanism, dropping probability is a function of queue occupancy through the AQM law, i.e. $p = f_{AQM}(q)$ and $T = T(q, f_{AQM}(q)) = T(q)$. For high threshold values, the network equilibrium point can be found if we consider that TCP sources are able to achieve almost 100% utilization of the bottleneck bandwidth, i.e. $T \approx C$. The equilibrium point (p_i, q_i, T_i) satisfies the following relations: $T(q_i, f_{AQM}(q_i)) = C$, $p_i = f_{AQM}(q_i)$ and $T_i = C$.

When we add a steady dropping probability it holds: $p = p_s + f_{AQM}(q)$ and $T = T(q, p_s + f_{AQM}(q)) = T(q, p_s)$. If $p_s < p_i$, then we can assume that $T \approx C$ still holds and the new equilibrium point (p_f, q_f, T_f) satisfies the following relations: $T(q_f, p_s + f_{AQM}(q_f)) = C$, $p_f = f_{AQM}(q_f)$ and $T_f = C$. Being $T()$ a decreasing function of q and p, from the comparison of previous equations it follows $q_f < q_i$. Hence the queueing delay is reduced by the additional dropping probability. If $p_s > p_i$ no solution is admissible with $T = C$, in particular it will be $T_f < C$. These arguments are inspired to those presented in (Firoiu & Borden, 2000) and justify the curves behavior in figure 3 for high link utilization.

As regards low link utilization, i.e. low RED thresholds, this kind of justification is not adequate, and we have to proceed in a different way. Let us suppose that an analytical expression exists relating the average queue value to offered TCP traffic, i.e. a relation $q = q(T)$ similar to the relation for $M/M/1$ queue system $q = \frac{\rho}{1-\rho}$, where ρ is the normalized offered load. Using such relation and the previous one ($T = T(q)$) one would obtain the network equilibrium point EP_i, according to a fixed point approach. Even if such relation is not known, it is reasonable to assume that it is an increasing function of T and that q diverges as T approaches C. A qualitative curve for $q(T)$ is shown in figure 4 together with the curve $T = T(q)$ for RED settings (8,24) predicted by the formula in (Padhye et al., 1998). The intersection of the two curves identifies the equilibrium point EP_i. Note that for high values of link utilization the curve $q(T)$ approaches the line $T = C$, hence this approach recovers the above considerations. At the same time it predicts that the introduction of p_s would move the equilibrium point towards lower delay but also to lower throughput. In facts, in figure 4 the curve $T = T(q, p_s = 0.5\%)$ intersects the curve $q = q(T)$ in a new equilibrium point EP'_f with $q'_f < q_i$, $T'_f < T_i$, and $p'_f > p_i$. On the contrary figure 3 shows that for low link utilization the throughput increases and the queueing delay is almost constant (a bit smaller).

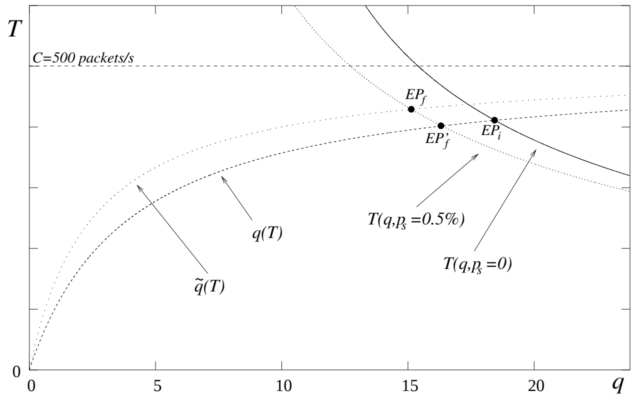

Figure 4: Determination of equilibrium points

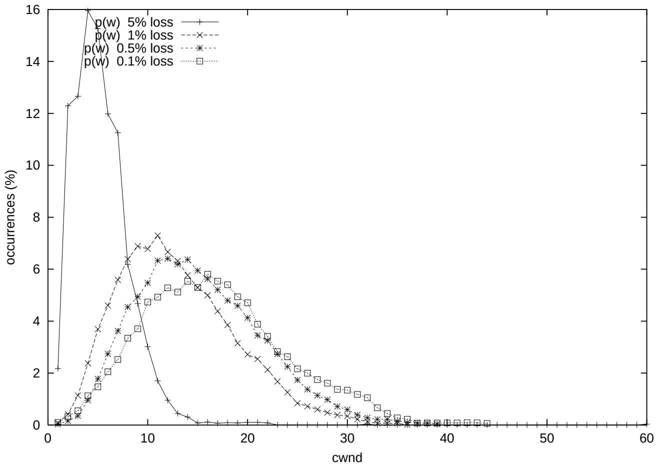

Figure 5: *cwnd* probability density with packet dropping

The error of the previous reasoning is that the average TCP throughput is not sufficient to characterize the stochastic arrival process at the queue -as it is for exponential arrival-, but more complex statistics are needed.

The purpose to adequately characterize the TCP packets arrival is out of the scope of this paper. Anyway we

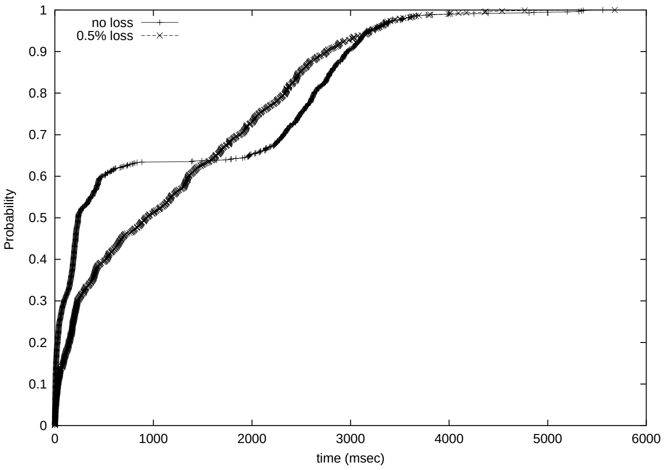

Figure 6: Probability distribution of interloss time intervals

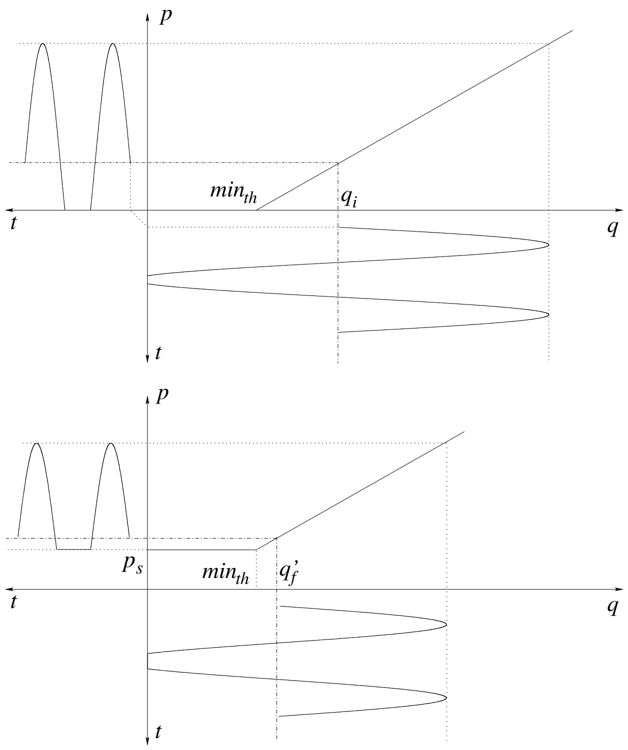

Figure 7: The effect of queue oscillation on dropping probability

assert that the additional dropping probability makes TCP throughput less variable, producing better network performance. In order to support such thesis, firstly, we present some results showing that TCP throughput is indeed more regular in the presence of the additional packet dropping probability; secondly we justify such results by RED operation; lastly we try to extend fixed-point argument to identify the new equilibrium point.

Figure 5 shows the congestion window *cwnd* probability density for a fixed RED setting and for different values of loss rate introduced by *hiccup*. This figure shows that increasing the loss rate results into a restraint of *cwnd* values so that the *cwnd* of each flow exhibits less variability and it has a lower average value. If the dropping probability exceeds a certain threshold the consequent average window reduction is excessive and network resources utilization is limited by the dropping probability, hence performance dramatically collapses. In particular, a loss rate of 5% is enough to heavily downgrade the functioning of the network, as is visible in the figure: the corresponding curve is extremely shrunk. We did not show the relative performance frontier in previous figure 3 since it is out of range (throughput below 70%).

The different *cwnd* behavior can be explained looking at the packet discards pattern. Figure 6 shows the probability distribution of the time intervals between two consequent losses for $p_s = 0$ and $p_s = 0.5\%$. It appears that for $p_s = 0.5\%$ the curve resembles that for an exponential process[4], while for $p_s = 0$ the curve suggests a bimodal dropping process. This can be explained by queue oscillation[5] between the region of null dropping probability ($q < min_{th}$) and the region of linear increasing dropping probability ($min_{th} < q < max_{th}$). Such oscillation causes the dropping probability oscillation and the consequent variability of the TCP window.

Let us observe that this oscillation is not due to an improper RED configuration: the parameters have been chosen according to commonly accepted heuristics (see previous section) and also control theoretic analysis developed in (Hollot & Misra & Towsley & Gong, 2001) predicts stable behavior. Simulations with $P_{max} = 0.05$, which increases stability margins according to (Hollot et al., 2001), show similar results.

Let us consider what happens for $p_s \neq 0$. In order to determine the new equilibrium point we can assume as a starting point for our consideration the equilibrium EP'_f shown in figure 4, where $q'_f < q_i$, $T'_f < T_i$ and $p'_f > p_i$. The queue is expected to assume lower values and to exceed min_{th} threshold less frequently. As a consequence dropping probability is more constant. Figure 7 illustrates qualitatively such behavior. In particular note how an equivalent RED curve has been considered, where the dropping probability is increased by p_s all over the range of queue values, with $q < max_{th}$. These considerations explain the quantitative results in figure 6.

As we said, the lower variability of the dropping probability produces a lower variability of the TCP *cwnd*. If the throughput offered to the network is more regular, then the queue occupancy is lower, i.e. in the terms of the previous fixed-point approach, we should consider a different relation $q = \tilde{q}(T)$, where $\tilde{q}(T) < q(T)$. An hypothetical curve for this new relation is shown in figure

---

[4]The exponential distribution may be viewed as a continuous counterpart of the geometric distribution originating from i.i.d. losses.

[5]For the sake of simplicity we do not distinguish between the instantaneous queue value and the filtered queue value calculated by RED in order to determine the dropping probability.

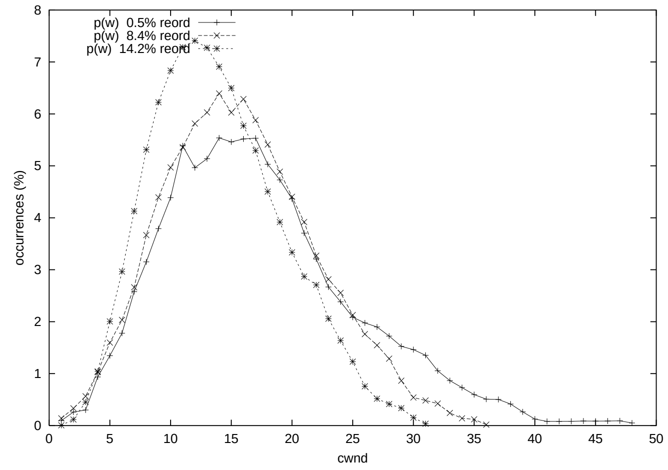

Figure 8: *cwnd* probability density with packet reordering

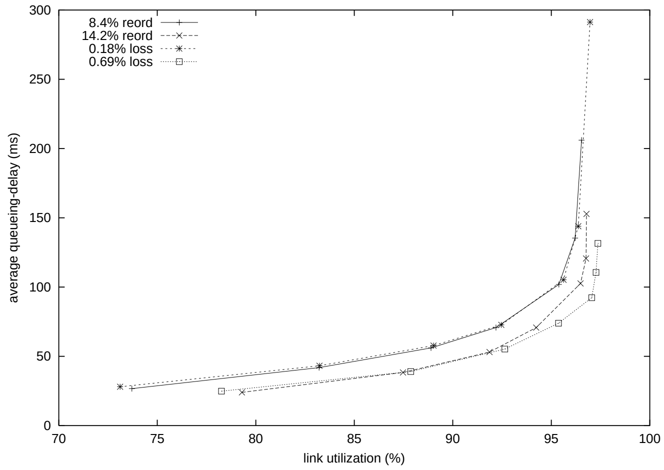

Figure 9: Reordering frontiers versus Losses frontiers

4, and it shows that the new equilibrium point EP_f is characterized by: $q_f < q'_f < q_i$ and $T_f > T'_f$[6].

In particular from figure 4 we note that T_f can even exceed T_i, as it appears from our simulative results.

In this subsection we have presented different results showing that an additional steady dropping probability can lead to better network performance, basically due to a better RED operation, and we have extended a fixed point approach in order to support such results.

Back to reordering

If we consider the introduction of reordering, we find similar results as regards the TCP *cwnd* and the inter-loss time interval. For example figure 8 shows how *cwnd* density function varies as we introduce reordering. These results confirm the validity of the parallel between reordering and dropping. Now we are going to justify it in details.

In TCP Reno fast retransmit and fast recovery algorithms are implemented (Stevens, 1997). According to these mechanisms if the sender receives three duplicate acknowledgements, it assumes that the data segment indicated by the acknowledgements is lost, it immediately retransmits the lost segment (fast retransmit) and it halves the congestion window (fast recovery). We note that if reordering causes a packet lag greater than or equal to three, the receiver sends three duplicate acknowledgements and the event for the TCP sources is indistinguishable from a loss event. This is the reason of analogous effects on network performance.

In order to verify such hypothesis we have run some simulations where p_s has been chosen equal to the probability of a reorder with packet lag greater than or equal to three, that has been measured in the simulations corresponding to figure 2. The resulting overlapping frontiers are shown in figure 9. We observe that corresponding performance frontiers overlap, with the exception of the utmost RED configurations for high reordering percentage, where the frontiers branch. In particular the points obtained in HICCUP_RESORT mode are above the ones found in HICCUP_CONG mode. This difference is mainly due to the particular performance metric chosen, as we are going to explain. Even if beyond the threshold of *packet lag* three the TCP sources behave as if a packet loss happened, yet this is not the case: packets are only shuffled, they arrive at the receiver and retransmissions are useless. In figure 9 we are considering goodput, hence such useless retransmissions do not appear, but we can note their effect on the delay. In facts, even if they constitute a small share of the total throughput (0.69%), for high link utilization, the delay is high dependent from the offered load, and they may produce a significantly increase of the delay. In order to support our thesis, we can plot the performance frontiers considering the traffic offered to the bottleneck. In the case of reordering the traffic offered to the bottleneck is the throughput of the TCP sources, while in the case of dropping losses due to hiccup have to be taken into account. For high link utilization losses due to AQM are negligible in comparison to additional losses, hence the traffic offered to the queue is almost equal to the goodput of the TCP sources. Figure 10 shows the delay versus throughput and versus goodput respectively for the reordering case and the dropping case. It appears that the two frontiers do not split.

In this subsection we have shown that the similarity of results for reordering and dropping is due to the fact that packet reordering with packet lag greater than or equal to three triggers fast retransmit and fast recovery algorithms. Performance frontiers overlap when the dropping probability and reordering probability are chosen according to this consideration.

[6] We should evaluate the new packet discards pattern in correspondence to EP_f, and repeat again the same reasoning to iteratively evaluate better approximations for the equilibrium point, anyway the final results should satisfies the relations indicated for EP_f.

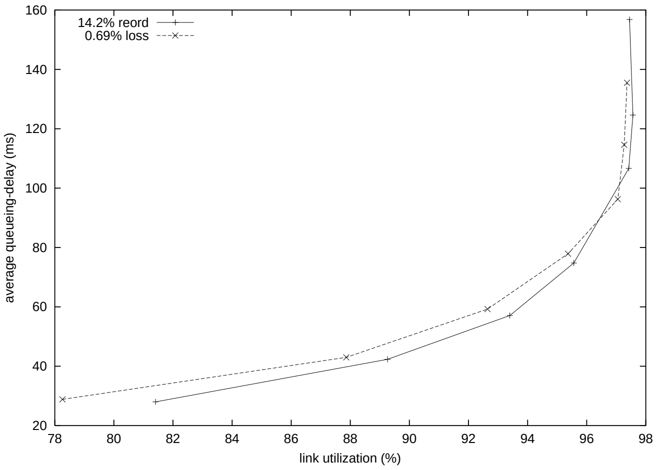

Figure 10: Performance frontiers in terms of throughput and goodput respectively for congestion and resort mode

Conclusions

The main contribution of this paper is that it shows that packet reordering is not necessarily a harmful effect in terms of network performance. In fact, our results show that a small amount of reordering can actually improve the network performance. *Small* is referred to the average dropping probability in absence of reordering.

In the attempt to find a justification for this result, we have made an analogy with a system characterized by a constant steady-state dropping probability not related to the congestion status of the network (as it happens in wireless links), and we have shown (via both simulation and theoretical analysis) that it produces the same beneficial effects.

Even if the result can appear counter-intuitive and somewhat suprising, in reality we note three strong limitations:

- the improvement is highly dependant from the uniformity of the reordering (or dropping) probability;
- we think that the improvement is highly reduced if short lived flows are considered, or if reverse traffic reordering is taken into account;
- the amount of helpful reordering (dropping) depends from the specific network scenario, the same probability may be harmful for a different configuration.

We remark that the improvement is essentially due to a better operation of RED: basically we have shown that a RED with a different configuration, i.e. with a small dropping probability for low queue values would have performed better for the specific network scenario.

Ongoing research work is dedicated to a more thorough understanding of the effect of reordering, in the presence of different traffic mixtures including short-lived TCP flows and reverse path reordering.

References

Bennett, J.C.R., & Partridge, C., & Shectman, N., *Packet Reordering is Not Pathological Network Behavior*, IEEE/ACM Transactions on Networking Volume 7, Issue 6, December 1999, Pages: 789 - 798

Firoiu V., & Borden M., *A Study of Active Queue Management for Congestion Control*, IEEE INFOCOM 2000, Tel-Aviv, March 26 - 30, 2000

Floyd S., *RED: Discussions of Setting Parameters*, email November 1997, http://www.icir.org/floyd/REDparameters.txt

Floyd S., & Gummadi R., & Shenker S., *Adaptive RED:An Algorithm for Increasing the Robustness of RED's Active Queue Management*, http://www.icir.org/floyd/red.html, August 1, 2001

Floyd S., & Jacobson V., *Random Early Detection gateways for Congestion Avoidance* IEEE/ACM Transactions on Networking Volume 1, Issue 4, August 1993, Pages: 397-413

Hollot C. V., & Misra V., & Towsley D., & Gong W. *A Control Theoretic Analysis of RED*, IEEE INFOCOM 2001, Anchorage, April 22-26, 2001

http://www.isi.edu/nsnam/ns/

http://www-tkn.ee.tu-berlin.de/∼morten/eifel/ns-eifel.html

Iannaccone G., & Jaiswal S., & Diot C., *Packet reordering inside the Sprint backbone*, Sprintlabs technical report TR01-ATL-062917, June 2001.

Iannaccone, G., & Jaiswal, S., & Diot, C., & Kurose, J., & Towsley, D. *Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone*, IEEE INFOCOM 2003, San Francisco, March 30-April 3, 2003

Kamoun, F., & Kleinrock, L., & Muntz, R. , *Queueing Analysis of the Ordering Issue in a Distributed Database Concurrency Control Mechanism*, Proceedings of the Second International Conference on Distributed Computing Systems, Paris, April 8-10, 1981

Neglia G., & Bianchi G., & Sottile M., *Performance evaluation of a new adaptive packet marking scheme for TCP over DiffServ networks*, GLOBECOM 2003, December 1-5, 2003

Padhye J., & Firoiu V., & Towsley D., & Kurose J., *Modeling TCP Throughput: A Simple Model and its Empirical Validation*, ACM SIGCOMM 1998, Vancouver, September 2-4, 1998

Stevens W., *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, RFC 2001, January 1997