IP packet interleaving: bridging the gap between theory and practice

Alessio Botta and Antonio Pescapé
University of Napoli Federico II (Italy) – Email: {a.botta, pescape}@unina.it

***Abstract*—The bursty nature of losses over the Internet is constantly asking for effective solutions. *Packet interleaving* or *time diversity* allows to cope with loss burstiness, at the cost of an additional delay. In this work, after determining the loss burstiness degree of real networks, we implement a real *interleaver* (we called *TimeD*), and we tackle the problem of how to apply such a transmission schema to UDP flows in real networks. For this aim, we propose a methodology composed of the following steps: (i) firstly, we develop a simulator to study the potential benefits of *TimeD*, understanding its loss decorrelation power and determining the interleaving configurations most suited to different network conditions and loss burstiness degree; (ii) then, using an emulated network, we validate *TimeD* and derive important operating parameters; (iii) finally, we study *TimeD* over a real satellite network. Thanks to this multifarious analysis we are able to move step-by-step from the theory to the practice, showing how it is possible – using *TimeD* – to actually decorrelate bursty losses and increase the performance of real applications.**

I. Introduction and motivation

Packet loss, especially when happening in bursts, degrades the performance of Internet applications, affecting the quality perceived by the users (the so called Quality of Experience). Several studies in literature report the characteristics of the loss process in different environments (backbone [1] and stub networks [2]), with different kinds of traffic (unicast [3] and multicast [4]), different protocols (UDP [3] and TCP [5]), and at different time scales (sub-round trip time [6] or larger [5]). They found, and we also experimentally confirm these findings (see Sec. V-A), that losses on the Internet are not isolate but rather happen in bursts. The problem is that the performance of the applications (e.g. streaming using predictive codecs such as MPEG) is more impacted by bursty losses than by the same quantity of isolated losses [7], [8].

To cope with this problem, different techniques have been proposed, which can roughly be classified as FEC- and ARQ-based. Such techniques normally imply overhead in terms of error-correction information, which have to be added before transmission (FEC) or retransmitted in case of loss (ARQ). *Interleaving*, also known as *time diversity*, represents a good candidate for network scenarios with bursty losses, allowing to spread the losses without transmitting additional information. Different approaches have been proposed for applying interleaving in TCP/IP networks. They are typically based only on theoretical and simulative works, or, when presenting real implementations, they are tight to specific applications.

In this paper we propose, for the first time in literature, a practical approach for studying the time diversity at packet level in real networks affected by bursty losses. Even if the approach is not tight to a particular transport protocol, in this work we explicitly target UDP flows. Our choice is motivated by the expected rise of UDP traffic volume [9], [10], [11], which steps from the momentum of applications (e.g. VoIP, streaming, p2p IPTV, etc.) and new protocols (e.g. uTP) for p2p applications that deeply rely on UDP, also to avoid traffic shaping techniques. As reported in [10], in the period 2002-2009, UDP has gained popularity, especially considering the number of flows: UDP allows efficient establishment and maintenance of p2p overlay networks, while the use of random ports evades detection by port-based traffic engineering or filtering techniques. This increasing trend in UDP usage has been seen all over the world, with a peak in data from China where UDP-based p2p IPTV traffic is already common.

To the best of our knowledge, there are no platforms implementing packet interleaving in real networks. Also, no methodologies are provided to study pros and cons of a real packet-level interleaver. With the methodology presented in this paper, we show how the proposed interleaver can be used in real networks without being dependent on a particular application, codec, or protocol. Our work extends the literature in that: (i) we study the burstiness of losses over real networks; (ii) we present a tool (called *TimeD*) implementing time diversity at packet level; (iii) we propose a complete and repeatable, three-steps methodology (i.e. based on simulation, emulation, and experimentation) to study packet-level time diversity in real networks; (iv) we analyze, discuss, and propose practical and effective solutions for several issues arising from the use of an interleaver in real networks; (v) we publicly release the implemented tools (both the simulator and *TimeD*); (vi) we provide results on the use of *TimeD* in a real satellite network, demonstrating its positive impact on loss burstiness and video application performance.

II. TimeD: time diversity at packet level

In this section we discuss the loss model we consider for the analysis, and we describe our approach and the tool in which it is implemented.

A. Loss model

In this work we assume that the loss process we are dealing with follows a 2-state Markov chain (2-MC) [12], [6]. Such model is used for the analyses performed in both simulation (see Sec. III) and emulation (see Sec. IV). In Sec. V-A we also evaluate the characteristics of the loss process over a satellite

network. The loss model based on 2-MC is also known as Gilbert-Elliott model [13], [14]. Such model is able to capture the potential correlation between consecutive losses both on the Internet [3] and on wireless networks [15]. Two important parameters of this model are: π_b, which is the average loss probability (i.e. the steady-state probability of staying in state *bad*); and ρ, which represents the correlation between losses (i.e. the higher ρ, the more losses are bursty). It is also worth noting that 2-MC is not the only or the fittest model for packet loss. It has actually been shown that Markov chains with more states are able to obtain higher modeling accuracy in some cases [3]. However, 2-MC represents the best compromise between complexity and accuracy.

B. The approach

We focus our attention on *time diversity* at packet level mainly for two reasons: (i) the losses on the Internet happen at such level, and (ii) to exploit the flexibility of IP, being not tied to a particular technology, transport protocol or application. The basic idea to realize packet interleaving is to resequence packets before transmission. Resequencing allows to space out originally close packets, so that a loss of consecutive packets is translated in a loss of distant ones. The main drawback of packet interleaving is the delay introduced. In order to resequence k packets, it is necessary to wait for them. Such delay is however acceptable for several applications (e.g. audio and video streaming, p2p file sharing, etc.) and, more importantly, it is controllable through the length of the interleaving sequence. When the length of the sequence is chosen, determining the optimal permutation of such sequence is not an easy task. As exhaustive searching the optimal permutation is usually not feasible, interleaving is commonly implemented in blocks (we refer to it as *block interleaving*). Block interleaving is made inserting packets into a matrix by rows, and picking them by columns. Basically, the traffic flow is divided in sequences of k packets. Each sequence is then placed into a *block*: a matrix of size $k = n \times m$ where n, the number of rows, is called *interleaving depth*. The effectiveness of such transmission schema depends on the values of n and m [1] as well as on network conditions. A burst of length B will be converted in smaller bursts of length B/n. When $n \geq B$ we are able to convert the burst in an equivalent number of isolated losses, spaced of m or $m-1$ packets. Therefore, increasing n and m, the capacity of converting bursts into isolated losses increases. On the other hand, there are two counter effects: (i) the number of smaller bursts increases; (ii) the buffering delay increases. We implemented time diversity at packet level, using *block interleaving*, in a platform we called *TimeD*, and the trade-off described before represents one of the main aspects we considered in order to deploy *TimeD* in real networks.

C. TimeD design and implementation

TimeD is a user-space application that interleaves IP packets generated by or traversing a Linux host. *TimeD* is written in C language and works over Linux platforms. Besides other interesting features, Linux provides more flexibility and allows the interleaving application to be easily deployed as an embedded system in any operational network. Moreover, *TimeD* is a user-space application, which allows to interact with user-space monitoring applications, whose output is useful to tune the interleaving parameters (see Sec. II-D). *TimeD* uses the *libipq*[2]: a mechanism provided by *netfilter*[3] for passing packets out of the kernel stack, queuing them to user-space, and receiving them back into the kernel with a verdict specifying what to do (e.g. ACCEPT or DROP). The queued packets may also be modified in user-space, prior to re-injection back into the kernel. With such mechanism, *TimeD* modifies the order of the packets as required by time diversity. As *TimeD* is based on *iptables*, the packets to be interleaved can be selected using several criteria such as the destination host, transport protocol and ports, etc.

At a very high level, *TimeD* operates in a cyclic fashion. During each iteration, it performs the following operations: (i) if present, read a packet from the queue and put it in a buffer; (ii) if the block is full or the timeout is expired (see Sec. II-D2), evaluate some statistics (see Sec. II-D1), interleave the packets, and mark them as ready for release; (iii) if present, release a packet from those ready for release, using the packet release strategy described in Sec. II-D3. Moreover, an additional module is activated periodically to estimate the status of the network and configure the interleaving block accordingly (see Sec. II-D1). An alpha version of *TimeD* is publicly available at [16] under the terms of GPL.

D. Most important operating features

We considered several issues in order to deploy packet-level interleaving in real IP networks.

1) Block size: As anticipated in beginning of the section, one of the most important problems is related to the fact that the block size has to be tuned according to the loss pattern on the network. And, in order to avoid unnecessarily large buffering delay, we have always to choose the smallest possible block size (mainly in terms of interleaving depth) that provides the required loss decorrelation. To this end, we integrated an active probing tool in *TimeD*. Thanks to it, *TimeD* periodically performs active measurements, estimates the loss pattern on the network in terms of π_b and ρ (see Sec. II-A) from the received traffic, and adjusts the block size accordingly. In order to effectively perform these tasks, several important aspects have to be considered and deeply investigated:

- *the block size to be used as a function of the loss pattern*: we set up a simulator to study this problem in a controlled scenario. Our simulator allows to highly abstract the problem and to only focus on the relation between the loss patterns before and after the interleaving, as a function of the block configuration. In Sec. III we present the simulator and the main results obtained thanks to it;

[1] Once k and n are chosen, m is automatically determined.

[2] https://svn.netfilter.org/netfilter/trunk/iptables/libipq

[3] http://www.netfilter.org

- *the packet size and rate of the probing traffic*: to choose these parameters, we performed a large set of experiments in an emulated scenario. We decided to perform this analysis in emulation because, in such conditions, we can use real operating systems, applications, and traffic on top of an emulated network, for which we can completely control the loss behavior. This allows on the one side, to use the real implementation of the tool, and therefore to also validate it, and on the other side, to have a reliable reference value for the variables to be estimated. In Sec. IV we present the emulation environment and the main results obtained;
- *how frequently to estimate the status of the network (i.e. the measurement period)*: the measurement period is in strict relation with how frequently the loss pattern on the network changes. To tune this parameter we both studied the literature and performed a set of experiments over real networks, as described in Sec. V-A.

2) Buffering timeout configuration: TimeD buffers incoming packets until the block is full and the interleaving can take place. In order to limit the maximum delay that a packet can experiment, we introduced a timeout mechanism for the buffering operation: the packets are released if the timeout fires. The value of the timeout, can be either specified by the user or autonomously determined by *TimeD*. To achieve this, *TimeD* estimates the average rate of the received packets, and then sets the buffering timeout as a function of this rate. The idea is that, knowing the rate and the block size, we can reasonably predict the time needed to fill the block. The estimation of the application rate is also used to release the packets with the same rate they have been received by *TimeD*, as explained in the next section. To estimate the rate, *TimeD* works as follows. Let us suppose that we want to operate with a block of size $n \times m = k$, on a flow of packets $p_1, p_2, ..., p_n, ...$, whose arrival times are $t_1, t_2, ..., t_n, ...$ Every time we release the current block, because either the block is full or the timeout (TO) is expired after receiving p packets (with $p < k$), we calculate a new rate sample $r(j)$:

$$r(j) = \frac{k}{t_n - t_{n-k}} \text{ (if block is filled)}$$
$$r(j) = \frac{p}{TO} \text{ (if timeout is expired)}$$

We calculate the average and standard deviation of the flow rate using the last α rate samples[4]:

$$r_{avg} = \frac{1}{\alpha}\sum_{i=0}^{\alpha-1} r(j-i); \quad r_{std} = \sqrt{\frac{1}{\alpha}\sum_{i=0}^{\alpha-1}(r(j-i) - r_{avg})^2} \tag{1}$$

And, we set the timeout as follows:

$$TO = \frac{k}{r_{avg} + 4 * r_{std}} \tag{2}$$

3) Packet release strategy: Once the packets are buffered and interleaved, *TimeD* releases them with the same rate they have been received. This is to preserve as much as possible the original profile of the traffic, and to avoid artificial spikes due, for example, to sudden release of all the packets. Besides distorting the application rate, such sudden release can also introduce higher packet loss, as we experimentally verified on satellite networks.

[4]We empirically found that a value of α equal to 8 is enough for our aims.

TABLE I
PARAMETERS USED IN BOTH SIMULATION AND EMULATION.

Parameter	Values
Interleaving block size	[12, 24, 48]
Interleaving block depth	[1, 2, 6, 12, 24]
π_b	[0.01, 0.03, 0.1, 0.25]
ρ	[1, 3, 8, 15, 30]
repetitions	1000 (simulation) / 20 (emulation)

As explained in the previous section, *TimeD* estimates the rate of the packets received in every block ($r(j)$). This value, is stored together with the packets in the queue of the packets ready for release. Using this value, *TimeD* performs the following operations continuously: (i) pull the next packet from the queue and release it; (ii) wait for a time equal to $1/r(j)$, while receiving the incoming packets and performing the related operations; (iii) verify if the time elapsed is larger than the expected. This last condition is extremely important. The receiving-related operations can sometimes take longer than expected (e.g. when the process is descheduled from the cpu). This would cause an additional delay in releasing the packets. In some cases, such as with high-rate CBR traffic, this can also imply a monotonically increasing queue (i.e. packets waiting for longer and longer). To avoid that, we continuously monitor the elapsed time, and, in case we are in late, we recover the delay during the next cycles. The result is that some packets can leave the buffer before their scheduled time. However, they compensate for those which left the buffer in late, with the final result that the original rate is respected in average, and no blocks are buffered more than necessarily.

4) Performance: Passing packets from kernel- to user-space impacts the performance and poses a limit on the maximum rate supported by *TimeD*. We verified that the delay introduced by *TimeD* is negligible with respect to the one of our experimentation scenarios (Sec. IV-C1), and that the tool was able to sustain the rate considered in the experiments (Sec. IV-C2). We believe that performance issues have to be carefully taken into account before the deployment of *TimeD* in operational environments. We left this as a future work, in which we also consider whether it would be beneficial (and at which cost) to make packet interleaving in the kernel space instead of user space.

III. TimeD analysis

According to what we discussed in Sec. II-D1, we implemented a simulator (using Matlab™) to understand the benefits of packet interleaving and to determine the best trade-off values for its parameters. Due to space constraints we briefly discuss here the most important simulation results. More details are reported in [17]. An alpha version of the simulator is publicly available at [16] under the GPL terms.

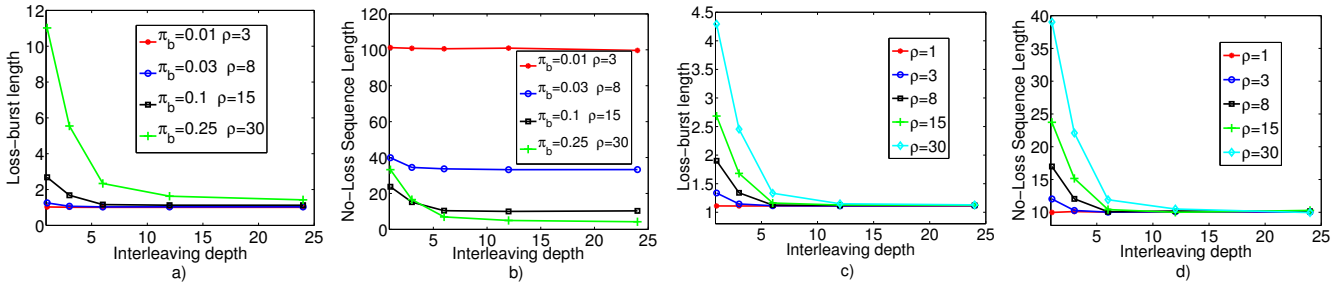

Fig. 1. Simulation results obtained with different values of π_b and ρ.

We evaluate the distribution of the length of the loss bursts and the distribution of the distance between two loss event (we call it *no-loss sequences length*). It is worth noting that once the average loss rate is fixed, the loss bursts and the no-loss sequences are tightly linked: the more we reduce the loss-burst length, the more we bring losses closer to each other. For this reason, the no-loss sequence length may seem a redundant parameter. However, in some cases, too close loss events may cause effects similar to loss bursts, decreasing application performance (e.g. video applications using H.264/AVC codecs, when the distance between losses becomes lower than a threshold [7]).

To understand the impact of each single parameter, we performed simulations using all the possible combinations of the values reported in Tab. I. In the following we report and discuss the results that are more useful to assess the impact of the interleaving with different channel conditions. More details and complete results of the simulations are reported in [17]. Fig. 1 shows the loss-burst length (1.a and 1.c) and the no-loss sequence length (1.b and 1.d) as a function of the interleaving depth, when using a block size of 48. Similar considerations can be done for the other block sizes. The two left plots of Fig. 1 are related to four representative combinations of ρ and π_b, with progressively degraded channel conditions. The two right plots of Fig. 1 are obtained using $\pi_b = 0.1$ and increasing ρ. As a first consideration, the results allow to understand the effectiveness of the interleaving in reducing the average loss-burst length (i.e. the loss decorrelation power). As shown in Fig. 1.a and 1.c, in all the cases such parameter presents a decreasing trend when increasing the interleaving depth. We also observe that the curves are steeper when the length of the loss bursts is high, i.e. the slope decreases when the interleaving depth increases. This means that the more burstiness we have, the more convenient is to use the interleaving (see also Sec. V). The same behavior can be observed comparing the different plots in Fig. 1.c: the higher the correlation, the more effective the interleaving. This translates in an asymptotic behavior of the curves, which tend to a line parallel to the x-axis, corresponding to a memoryless channel. This happens because after a certain depth, that depends on the channel conditions, the interleaving manages to make the losses perceived as uncorrelated, and the channel behaves as the Bernoulli one. Further increasing the interleaving depth provides no benefit. This behavior is also confirmed by the plots of the no-loss sequence length (Fig. 1.b and 1.d). According to the theoretical results (Sec. II-B), when the block size is fixed, pushing the interleaving depth (n) decreases (m), and consequently the no-loss sequence length. Also, we observe that the trend of the plots is very close to that of the loss-burst length (especially visible in the right plots). This is due to the fact that the interleaving does not alter the average loss rate, which can be roughly seen as the ratio between the average loss-burst length and the sum of the average no-loss sequence length and the average loss-burst length.

The simulations performed allow to understand the block size and the interleaving depth necessary for a given loss pattern. As for the block size, we decided to use a value of 48 because it provides the best trade-off between buffering delay (see Sec. IV-C1) and loss decorrelation. Tab. II shows the interleaving depth sufficient to obtain uncorrelated losses for different loss pattern and block size of 48. This table has been obtained looking at all the results of the simulation analysis. A depth value equal to 1 in this table means that the interleaving is not necessary. The table has been provided as input to *TimeD*. Looking at the values reported in this table, *TimeD* automatically configures the block depth most suited to the loss pattern measured on the network.

IV. TimeD validation

According to the discussion we provided in Sec. II-D1, we used an emulative approach for validating *TimeD* and checking its implementation over (controlled) real networks. In order to validate *TimeD* behavior, we first compared the results in terms of loss decorrelation obtained with the tool and that obtained in simulation. Then, we analyzed how the probing flow characteristics allow *TimeD* to effectively estimate the loss characteristics also in time-varying conditions. Finally, we discussed *TimeD* performance in terms of additional delay introduced and achievable throughput.

The testbed we used is composed of Linux hosts connected (using a Fast Ethernet network) to the Shunra Virtual Enterprise WAN emulator[5]. Such hardware emulator is able to reproduce the behavior of a WAN in terms of different parameters. For our experiments, we set a loss pattern equal to a 2-MC (the parameters are reported in Tab. I), and left the delay and jitter unset (i.e. no delay or jitter is intentionally

[5]http://www.shunra.com/shunra-ve-overview.php

TABLE II
INTERLEAVING DEPTH AS A FUNCTION OF THE LOSS PATTERN.

	$\pi_b < 0.01$	$0.01 <= \pi_b < 0.02$	$0.02 <= \pi_b < 0.05$	$0.05 <= \pi_b < 0.15$	$0.15 <= \pi_b < 0.25$	$\pi_b > 0.25$
$\rho = 1$	1	1	1	1	1	1
$1 < \rho <= 5$	1	6	6	12	24	24
$11 < \rho <= 22$	6	6	12	12	24	24
$22 < \rho <= 30$	12	12	12	24	24	24
$\rho > 30$	24	24	24	24	24	24

added). For generating probe traffic we used D-ITG, which emulates a real network application generating UDP flows with several traffic profiles mixing different packet rates (10, 100, 1000 pps) and packet sizes (64, 128, 256, 512, 1024, 1472 Bytes) and exploiting the benefits of packet interleaving. Further details and complete results of this stage are reported in [17].

A. Loss decorrelation

Fig. 2 reports the average lengths of loss bursts and no-loss sequences in the four channel conditions analyzed in Sec. III, which are characterized by $(\pi_b, \rho) \in \{(0.01, 3); (0.03, 8); (0.1, 15); (0.25, 30)\}$. For these tests we instructed the WAN emulator to reproduce the behavior of such four channel conditions, and we let the test traffic (generated with D-ITG) traverse the emulated WAN, after being interleaved by *TimeD*. We also instructed D-ITG to produce a log of the experiments. Using such information we obtained the samples of the loss bursts and no-loss sequences. The results in Fig. 2 have been obtained by sending UDP packets at a constant rate of 100 pps with a payload of 512 Bytes. We observe that the average values of the length of the loss bursts and no-loss sequences is decreasing with the increase of interleaving depth, as done in simulation. This shows the actual decorrelation power of *TimeD*. The results obtained in such experimentations are very close to those from the simulations (see left plots of Fig. 1), confirming the proper behavior of *TimeD*. It is worth stating that similar consideration can be made with the other traffic profiles and channel conditions (see [17]).

B. Probing flow characteristics

As said, *TimeD* performs active probing on the network to estimate the loss characteristics and set the proper block size. Therefore, it is important to understand how to perform such probing, in order to obtain a sufficiently accurate estimate for this aim. Different contrasting interests have to be considered, that are the *measurement intrusiveness* and the *accuracy*. In fact, the more accurate we want our measurements to be, the more samples we have to collect. And, once the measurement period is fixed, to collect more samples we have to probe the network with a higher packet rate, which implies an higher intrusiveness. To choose the parameters of the probing flow, we have performed a large set of measurements, varying the probing traffic profile (packet rate and size) and varying the loss pattern on the emulated WAN in a controlled scenario. We then looked at the values of loss rate and correlation estimated by *TimeD*. The results showed that the loss rate is easily estimated with a small probing rate: a probing rate of 10 pps was already enough to obtain a relative error in the order of 10^{-1}, which is sufficient for our aim. As for the rate correlation, instead, we observed a different situation. Fig. 3 shows the relative error obtained in two experiments performed using $\pi_b = 0.01$ and $\rho \in \{3; 30\}$. As shown, at low probing rates (e.g. 10 pps) the relative error reaches values up to 2. However, the results of the simulations performed in Sec. III indicate that it is necessary to estimate the loss correlation with an absolute error in the order of few units (i.e. 10^0). It is also worth saying that, in order to observe the same channel conditions experimented by the application, it is never necessary to probe the network with a rate higher than that of the application. For these reasons, we chose to probe the network at a packet rate equal to the minimum between that of the application and 50 pps. This guarantees both a sufficient accuracy and a low intrusiveness of the probing traffic (50 pps $\times$50 Bytes/pkt = 20 Kbps).

To validate the achieved accuracy, we performed a set of experiments in which we continuously varied the channel conditions imposed by the network emulator every $5 \rightarrow 35$ minutes. During the experiments, we run both *TimeD* and D-ITG: the former interleaved the packets generated by the

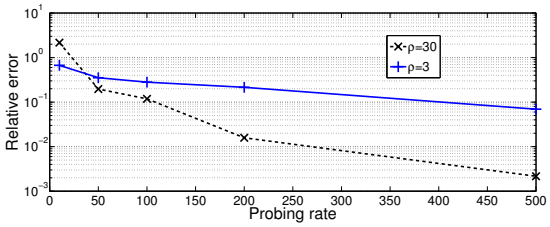

Fig. 3. Relative error of loss correlation estimation.

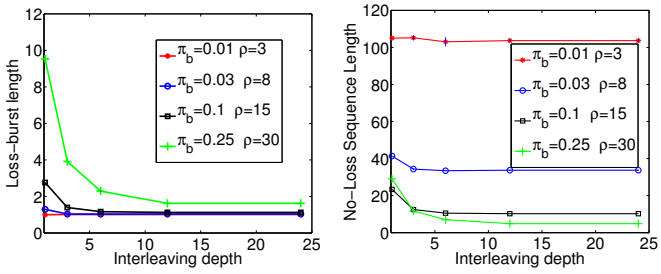

Fig. 2. Results obtained with *TimeD* in four different channel conditions.

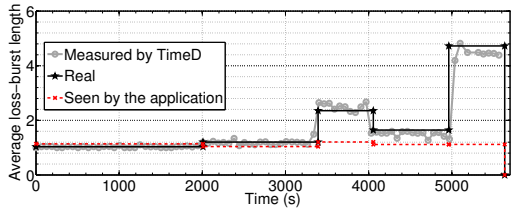

Fig. 4. Results obtained with *TimeD* with time-varying channel conditions.

latter. Fig. 4 shows the results in terms of loss-burst length over time, obtained during a period of about 1.5 hours. As we can see, the channel changes behavior 4 times during the experiment, and *TimeD* accurately estimates the time-varying channel conditions (gray bold line), being always able to set the proper block size. Consequently, the application whose traffic is interleaved experiments a channel with non-bursty losses during the entire experiment (red dashed line).

C. TimeD performance

1) Additional delay introduced: In simulation we neglected the delay issues because we wanted to understand the potential benefit and the optimal configuration. However, if we want to deploy the interleaving on a real network we have to consider all the effects that we might expect. For these experiments we connected the hosts of the testbed back-to-back, disabling the WAN emulator. We then performed two different kinds of measurements to estimate the two main delay components: the forwarding delay and the buffering delay. To estimate the forwarding delay we performed experimentations with *TimeD* configured to use a block depth $= 1$ and we injected UDP packets at all the rates reported above. With such a block depth, *TimeD* was forwarding the packets as soon as they entered the queue, and no buffering was performed. We then performed the same experiments without *TimeD* and compared the delays obtained. We discovered that the forwarding delay is in the order of 10 μs at all the considered packet rates, we believe this delay can be considered negligible, at least for the operating scenarios we consider.

For the buffering delay, we measured the transfer time experimented by UDP packets using all the rates and interleaving configurations reported in Tab. I. The experiments evidenced that, thanks to the packet release strategy presented in Sec. II-D3, the delay is typically constant (except few spikes), and in average, it is equal to:

$$\delta_{avg} = (N-1) \times IPT_{avg} \tag{3}$$

where N is the block size and IPT_{avg} is the average inter-packet time. As already remarked before, such delay has to be carefully taken into account from the application point of view. To provide a real example, if we use *TimeD* with a streaming server – assuming that the stream has a constant packet rate of 720 pps (i.e. frame rate of 24 frames/s, 30 slices per frame, and a single slice per packet) and that packet sizes follow a normal distribution [18] – and with an interleaving block of 48 packets, we obtain an average buffering delay of about 65 ms, which is acceptable in most cases.

2) Achievable throughput: Even if we believe that performance issues have to be carefully taken into account before the deployment of *TimeD* in operational environments, we verified that *TimeD* was able to sustain the rate used for the experiments. In particular, we performed experiments aimed at understanding the maximum throughput that *TimeD* is able to sustain: *TimeD* is able to work at full speed (i.e. 100Mbps) with all the interleaving configurations in Tab. I, even when running on the same host as D-ITG.

V. Experimentation over real networks

To provide evidences of *TimeD* benefits in real networks, we focus our attention on satellite networks. Satellite networks are being more and more used for Internet access (for rural areas, in-flight and over-sea connectivity, etc.), while being particularly exposed to bursty losses. For these experiments we used a commercial satellite connection from one of the largest providers in Europe. The satellite connection is bidirectional and has nominal bandwidth of about 3 Mbps in downlink and 300 Kbps in uplink. Our testbed is composed of hosts connected to the Internet through the satellite connection, and servers located in our University (connected to the Internet though the GARR[6]). We generated UDP flows with different CBR and VBR traffic profiles, with average packet sizes $\in \{64; 128; 256; 512; 1024; 1472\}$ Bytes and rates $\in \{10; 100; 1000\}$ pps. It is worth noting that with packet size > 512 Bytes and rate of 1000 pps the link was completely saturated. Flows had duration of $3 \rightarrow 10$ minutes and every experiment was repeated 10 times. Before deploying *TimeD* on this infrastructure, we performed a set of measurements to understand the loss characteristics of the satellite channel.

A. Losses over satellite networks

As anticipated in Sec. II-D1, to choose the measurement period we studied the literature and we analyzed the time behavior of the losses over the satellite network. In [19] the authors perform a long-term measurement campaign, and report the long-term delay and loss rate measured over different paths connecting the same hosts. The results show that there is high variability in the loss rate, and, if observed with a resolution of 1 minute (i.e. computing the average loss rate over 1 minute intervals), the paths look quite different. They also show that the average value of the loss-variation period is $1 \rightarrow 4$ minutes. Fig. 5 shows a zoom of the samples of the packet loss over the satellite network, during a period of 10 minutes, sending packets of 128 Bytes at 1000 pps. As we can see, there are periods (e.g. between 350 s and 450 s, and between 500 s and 600 s) in which the losses are higher than in the other periods (e.g. between 0 s and 80 s). We also observed that the duration of such *good* and *bad* periods is generally between 50 and 100 s. Tests performed with other traffic profiles evidenced the same behavior. This result, confirming previous results in literature [19], motivates our choice of evaluating the status of the network every 1 minute.

[6]http://www.garr.it

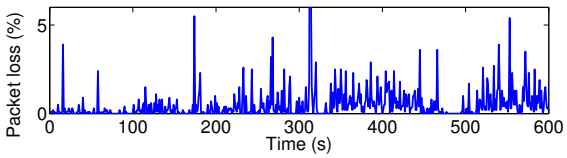

Fig. 5. Time behavior of the losses over the satellite network.

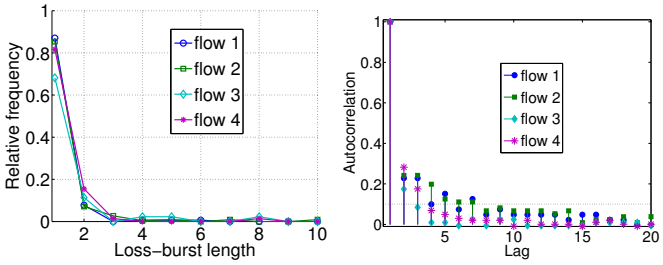

Fig. 6. Loss burst distribution and autocorrelation over the satellite network.

Regarding the burstiness of the losses, 50% of the flows experimented losses over this network scenario, and they can be divided in 4 classes. Fig. 6 shows the typical loss-burst length distribution (left) and loss autocorrelation (right) of flows – in the downlink direction – belonging to the 4 classes. To study the autocorrelation of the loss process, we follow the methodology proposed in [3]. As shown in the left plot of Fig. 6, for all the flows the loss burst lengths range from 1 to 5 packets. On the other hand, as shown in the right plot of Fig. 6, the loss process presents a high autocorrelation, which means that the losses are indeed bursty. For example, fitting the loss process of such flows with a 2-MC[7] gives π_b of $0.002 \rightarrow 0.005$ and ρ of $30 \rightarrow 400$. Such results indicate that the losses over the satellite link are not frequent but highly correlated. This means that in this scenario *TimeD* can provide high benefits, as shown in the next section.

B. TimeD results

Firstly, we analyzed the effectiveness of the packet release strategy described in Sec. II-D3. Fig. 7 shows the time behavior of the bitrate of VBR traffic flows with exponentially distributed packet rate (average value is 100 pps) and constant packet size of 512 Bytes. The different lines in the figure show the profiles of the injected traffic and of the received traffic when *TimeD* enables and disables the packet release strategy. The continuous spikes in the bitrate of the flow received when such strategy is disabled are due to the sudden buffer emptying. On the contrary, when the packet release strategy is enabled, *TimeD* closely follows the original traffic profile.

Secondly, we experimentally verified the improvements achieved with *TimeD* in terms of average loss-burst length reduction. The left plot of Fig. 8 is related to CBR traffic having packet size $\in \{64, 128, 256\}$ Bytes and packet rate of 1000 pps. We can observe that, when not using *TimeD*, the average loss-burst length on the link is between 1.5 and 3 packets, with a very high variance (see the vertical segments). The use of *TimeD* allows to reduce both the average loss-burst length and the related variance. As a result, the loss-burst length is always smaller than 1.12, and the loss correlation at lag 2 (not shown here for space constraints) is always smaller than 0.01. The experiments performed with the other traffic profiles evidenced similar results. This means that, thanks to its features, *TimeD* is able to decorrelate the losses in real networks, allowing the applications to experiment a channel that is close to the Bernoulli one.

Finally, we analyzed the benefits achievable by a real application. We used the results from [7], which presents a model to study the effect of packet loss on video communications, taking into specific account the loss pattern. Using such model, we analyzed the benefits in terms of video distortion reduction, achievable by using *TimeD* on the satellite network. The right plot of Fig. 8 shows the video distortion estimated by the model for two test video sequences - named *Foreman* and *Claire* - subject to the losses reported in the left plot of the same figure (with and without *TimeD*). As we can see, in accordance with the model, the distortion increases with the loss-burst length. Consequently, the video flows whose packets are interleaved by *TimeD* experimented lower average and standard deviation values of the video distortion. Where the losses are more correlated (i.e. for packet size of 256 Bytes) *TimeD* improves the distortion of more than 60% (i.e. from 450 to 150). Summarizing, we can say that *TimeD* highly improves the performance of applications such as those for video communications, providing higher Quality of Experience.

VI. Related work

Here we provide a quick review of the literature, for a more extensive discussion refer to [17]. In [20] the authors present a simulation framework to study packet interleaving. Results are presented using two channel models, one with uncorrelated random losses and the other one following a Markov chain. In [21] the authors propose a system that performs packet interleaving only on MPEG audio, and they show the loss decorrelation power of their scheme both in simulation and on real networks. In [22] a study of the effect of packet interleaving on real video sequences is presented, whereases [23] presents a research exploiting interleaving on MPEG encoded video frames. The system is evaluated using real videos ranked by real users. In [24], [25] the authors present an interleaving scheme that operates before a generic video predictive encoder. The authors of [26] perform a simulation study of interleaving applied to layered coding schemes. Simulations are conducted by using loss patterns from real traffic, and they show how interleaving and randomization are able to actually protect the base information from bursty losses. The work [27] is concerned with the design and verification of a VoIP technique exploiting interleaving and a matrix-based transformation. The receiver continuously informs the sender about the status of the channel, i.e. the loss pattern. The sender then applies

[7]As the correlation persists also at lags > 2, fitting the process with a Markov Chain with a higher number of states is also possible. However, 2-MC represents the best compromise between accuracy and complexity.

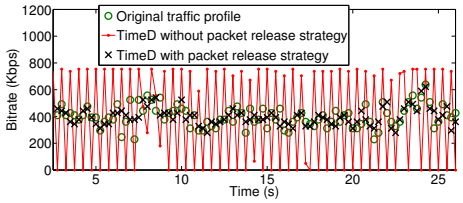

Fig. 7. Benefit of the packet release strategy.

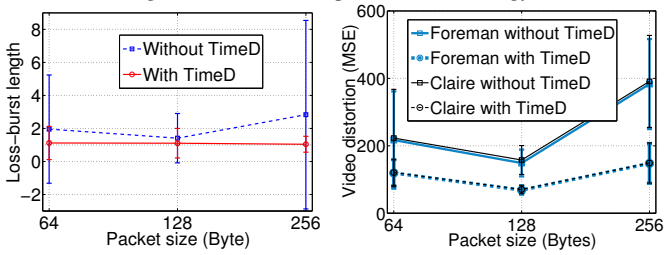

Fig. 8. Results of *TimeD* on satellite network.

interleaving to the voice samples in order to compensate the effect of samples that will be lost.

The analysis of the literature reports mostly theoretical and simulative works, and when real implementations have been proposed, they are tight to a specific streaming application/codec (e.g. [21]). In this paper, we presented – for the first time in literature – a real IP packet interleaver and a methodology to move step-by-step from the theory to the practice.

VII. Conclusion

In this work we paved the way to the deployment of time diversity at packet level in real networks. In particular, we verified the loss behaviour over real networks, and we presented a platform (called *TimeD*) to provide loss decorrelation and targeted to UDP flows. To move from theory to practice, we proposed a 3-steps approach (based on simulation, emulation and real experimentation), studying pros and cons of a real packet interleaver. Using *TimeD* on a real satellite network, we also showed the benefits of the proposed interleaver for real applications. In our ongoing work, we are studying the interactions between *TimeD* and transport protocols adopting congestion control strategies (SCTP, DCCP, and TCP) and we are performing a careful analysis of the interleaving applied on bidirectional traffic using *TimeD* on both ends of network paths.

References

[1] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the MBone multicast network," in *IEEE GLOBECOM*, 1996.

[2] S. Chung, D. Agrawal, M. Kim, J. Hong, and K. Park, "Analysis of Bursty Packet Loss Characteristics on Underutilized Links Using SNMP," *IEEE/IFIP E2EMON*, 2004.

[3] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *IEEE INFOCOM*, 1999.

[4] J.-C. Bolot, H. Crépin, and A. Vega-García, "Analysis of Audio Packet Loss in the Internet," in *NOSSDAV*, 1995.

[5] V. Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, pp. 277–292, 1999.

[6] D. X. Wei, P. Cao, and S. H. Low, "Packet Loss Burstiness: Measurements and Implications for Distributed Applications," *IEEE IPDPS*, 2007.

[7] Y. Liang, J. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: does burst-length matter?" in *IEEE ICASSP*, 2003.

[8] T. Liu, Y. Wang, J. Boyce, H. Yang, and Z. Wu, "A novel video quality metric for low bit-rate video considering both coding and packet-loss artifacts," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 2, pp. 280 –293, april 2009.

[9] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "Kiss: Stochastic packet inspection classifier for udp traffic," in *IEEE/ACM Transactions on Networking*, To appear, 2010.

[10] M. Zhang, M. Dusi, W. John, and C. Chen, "Analysis of udp traffic usage on internet backbone links," in *SAINT*, 2009.

[11] T. D. Dang, M. Perényi, A. Gefferth, and S. Molnár, "On the identification and analysis of p2p traffic aggregation," in *Networking*, 2006.

[12] B. Ribeiro, E. e Silva, and D. Towsley, "On the efficiency of path diversity for continuous media applications," *UMass CMPSCI Technical Report 05-19*, 2005.

[13] E. Gilbert *et al.*, "Capacity of a burst-noise channel," *Bell Syst. Tech. J*, vol. 39, no. 9, pp. 1253–1265, 1960.

[14] E. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. J*, vol. 42, no. 9, pp. 1977–1997, 1963.

[15] L.-J. Chen, T. Sun, M. Sanadidi, and M. Gerla, "Improving wireless link throughput via interleaved FEC," in *IEEE ISCC*, 2004.

[16] http://www.grid.unina.it/Traffic.

[17] A. Botta, "Towards informed diversity in IP networks: techniques, issues, and solutions," Ph.D. dissertation, University of Napoli Federico II, December 2009, http://wpage.unina.it/a.botta.

[18] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar vbr video traffic," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 269–280, 1994.

[19] S. Tao, K. Xu, Y. Xu, T. Fei, L. Gao, R. Guerin, J. Kurose, D. Towsley, and Z.-L. Zhang, "Exploring the performance benefits of end-to-end path switching," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 418–419, 2004.

[20] P. Salvo Rossi, F. Palmieri, G. Iannello, and A. Petropulu, "Packet interleaving over lossy channels," in *IEEE International Symposium on Signal Processing and Information Technology*, 2004.

[21] J.-H. Yao and Y.-M. Chen, "Experiments of Real-Time MPEG Audio over the Internet," *Fujitsu Scientific and Technical Journal*, vol. 33, no. 2, pp. 138–144, 1997.

[22] Y. J. Liang, J. G. Apostolopoulos, and B. Girod, "Model-Based Delay-Distortion Optimization," in *Asilomar Conference on Signals, Systems, and Computers*, 2002.

[23] M. Claypool and Y. Zhu, "Using Interleaving to Ameliorate the Effects of Packet Loss in a Video Stream," in *IEEE ICDCSW*, 2003.

[24] J. Y. Lee and H. Radha, "Interleaved source coding (ISC) for predictive video coded frames over the Internet," in *IEEE ICC*, 2005.

[25] ——, "Evaluation of Interleaved Source Coding (ISC) over Channel with Memory," in *Conference on Information Sciences and Systems*, 2006.

[26] S. K. Chin and R. Braun, "Improving video quality using packet interleaving, randomisation and redundancy," in *IEEE LCN*, 2001.

[27] B. Web and D. Lin, "Transformation-based reconstruction for real-time voice transmissions over the internet," *Multimedia, IEEE Transactions on*, vol. 1, no. 4, pp. 342–351, Dec 1999.