Packet interleaving from theory to practice

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

Abstract

Multimedia streaming over the Internet is becoming more and more popular, and different technologies and protocols are currently being used. However, the bursty nature of losses over the Internet is constantly asking for effective solutions. Packet interleaving allows to cope with loss burstiness, improving the quality perceived by different kinds of applications at the cost of an additional delay. In this paper we tackle the problem of how to apply such a transmission schema in real scenarios. For this aim we firstly study the potential benefits of packet interleaving in simulation, which allows to understand its loss decorrelation power, and to determine the interleaving configuration most suited to different network conditions. We then present the design and prototype implementation of a packet interleaver, which can be used both for a specific traffic source (e.g. besides a video server) or a particular network path (e.g. before a wireless link). We describe our design choices and the main issues we had to solve during the implementation. Using the prototype we then perform an experimental measurement campaign in order to understand the pros and cons of packet interleaving in real networks. Achieved results allow to confirm findings obtained in simulation and to uncover issues related to real world scenarios. Finally, using an improved interleaving schema we discuss the interactions between packet interleaving and transport protocols with congestion control, and we show how it is possible to mitigate negative effects the interleaving has on such protocols.

1 Introduction

Packet loss degrades the performance of Internet applications, affecting the quality perceived by the users (the so called Quality of Experience). This is even more true with multimedia streaming applications. In facts, while such applications are less sensitive to the network delay, their performance is dominated by the loss of packets. On the other hand, Internet has being more and more used for audio and video streaming. Even more because current technology allows users to upload their multimedia contents both on- and off-line. YouTube as well as p2p streaming applications such as Soapcast or TvAnts are clear examples of such a trend. Several studies in literature reported on the characteristics of the loss process in different environments (backbone [YKT96] and

stub networks [CAK+04]), with different kinds of traffic (unicast [YMKT99] and multicast [BCVG95]), different protocols (UDP [YMKT99] and TCP [Pax99]), and at different time scales (sub-round trip time [WCL07] or larger [Pax99]). All these studies reported that losses on the Internet are not isolate but rather happen in bursts. The problem is that the performance of streaming applications, which often use predictive codecs such as MPEG, is more impacted by bursty losses than the by same quantity of isolated losses [LAG03]. To cope with this problem different techniques have been proposed which can roughly be classified as FEC- and ARQ-based. However, they 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 know as time diversity, represents a good candidate for these networking scenarios allowing to spread the losses over the multimedia stream without transmitting any additional information.

In this paper we present a simulation and experimental analysis of packet-level interleaving aimed at understanding if and how such technique can be exploited in real networks. After discussing the main assumptions at the base of our work (Section 3), we analyze the benefits of interleaving in simulation. To this end, we develop a simulation framework to discover the configurations most suited for different channel conditions (Section 4). We then present a prototype application we designed, developed and publicly released that allows to experiment with time diversity on real networks (Section 5). Possible application scenarios are discussed together with design and implementation issues. Thanks to such application, we perform an analysis of the performance of time diversity with real traffic (Section 6). Obtained results show how to deploy such technique on real networks. However, they also evidence that some important issues have to be addresses, especially when using transport protocols with congestion control algorithms such as TCP, SCTP, and DCCP both with CCID 2 and 3. We devise a simple mechanism to mitigate the effect of the buffering operation performed by our application, showing the obtainable improvements (Section 7). We believe that this work provides interesting insights and tools to increase the performance of multimedia streaming applications .

2 Related work

2.1 Measurement and modeling of packet loss

The work [BCVG95] presents a study of packet loss experienced by audio packets. The authors use an audio conferencing tool to send the packets in both unicast and multicast mode. In this second case they use the MBone facility. Looking at the sequence numbers of the received packets, the authors report information related to the number of packets that are consecutively lost on different Internet paths. In particular they show that the number of consecutive lost packets is small unless the network is highly congested. They then derive a model to represent this loss phenomenon, which is partly based on queuing

theory. The paper ends with a discussion on the use of the FEQ or ARQ for this kind of traffic.

In [YKT96] the authors report the results of a measurement campaign conducted on the MBone network. By monitoring the packets received from an Internet radio station by different hosts around the world, the authors study properties of packet loss such as correlation in space (different sites at the same time) and time (same sites at different times). The results show that the majority of the bursts have length smaller than 7 packets.

Paxon presents the results of a large scale TCP measurement in [Pax99]. To obtain the measures the author uses a set of measurement daemons that perform 100 Kbytes transfers using TCP while capturing the packet traces with tcpdump at sender and receiver sides. The obtained results show how some popular assumptions such as in-order packet delivery, FIFO bottleneck queuing, independent loss events, single congestion time scales, and path symmetry are violated in practice. As for the losses in particular, Paxon reports the average values as well as the conditional probability of having a loss after another, which shows that the loss events are not independent.

In [YMKT99] the authors present an analysis of the correlation properties of packet loss. They perform long-term measurements using both unicast and multicast from a host located at the University of Massachusetts at Amherst towards different hosts in USA and Sweden. The results show that: i) there is a high degree of non-stationarity in loss samples; ii) the minimum time after which the samples are uncorrelated is about 1 s; iii) the losses can be modeled by using a Bernoulli model (in 7 cases), a 2-state Markov chain (in 10 cases), or a higher order Markov chain (in 21 cases). They also discuss whether it is better to use an exponential smoothing rather than a sliding window to obtain a good estimate of the average loss rate.

In [CAK+04] the authors report passive measures of packet rate, bit rate, and loss from a campus network. With regard to this last parameter, the results show that while a high loss is observed on highly congested links, losses are also observed on very underutilized links (i.e. utilization of about 5%). The authors say that such losses are probably caused by spikes in the traffic rate that are not observable at the time resolution they used (larger than or equal to 1 second).

Wei et al. in [WCL07] report the results of an analysis of the behaviour of packet loss in sub-RTT scale performed with simulations, emulations, and on the Internet. They show that the loss process is bursty on such time-scale, and they show the implications for congestion control algorithms. They also report on the possible origins of such burtsiness. Moreover, they perform investigations on the fairness between congestion control algorithms based on packet windows and on bit rate.

2.2 Interleaving

2.2.1 Interleaving at packet level

In [SRPIP04] the authors present a simulation framework to study packet interleaving. Results are presented related to two channel models, one with uncorrelated random losses and the other one with correlated losses following a Markov chain. They assume the delays to be Gamma distributed, citing the work [Pax97], and they use a quality function to estimate the benefit of the packet interleaving.

In [YC97] the authors present a system that performs packet interleaving on MPEG audio. The work contains an interesting introduction to the problem of packet interleaving and a description of the system to both interleave the packets and recover from possible errors. The authors briefly discuss the problem of determining the optimal interleaving configuration, and they present the results of an evaluation performed both in simulation and on real networks, which show the loss decorrelating power of their scheme.

In [LAG02] a study of the effect of packet interleaving on real video sequences is performed. A loss model is used to determine the interleaving configuration most suited to the channel conditions. A delay-distortion optimization problem is set in order to choose the interleaving block sizes taking into account both the maximum acceptable delay constraint and the minimal distortion objective. Basically, knowing the maximum delay, they test all the possible configuration and choose the one providing the minimal average distortion according to the loss model.

2.2.2 Interleaving at frame/packet level

The work [CZ03] presents a research exploiting interleaving on MPEG encoded video frames. The proposed system acts just before the MPEG encoder applying the interleaving to the sequence of frames, i.e. frame position is altered before encoding. The resulting sequence is surely more robust to possible frame loss but the compression is less efficient because some temporal redundancy is lost due to the reordering. The system is evaluated using real videos ranked by real users. Results show that the average score is higher for interleaved frames and that an interleaving depth of 5 achieves in some cases better performance than an interleaving depth of 2.

In [LR05, LR06], the authors present an interleaving scheme that operates before a video encoder (the kind of encoder is not specified but the scheme is useful with predictive codes). Basically, the video stream is divided in two sub stream, which are then encoded separately. After encoding, the frames are rearranged in their original position. The fact that they have been coded by separate encoders, however, allows to be more resilient to burst losses as the reference and predicted frames are never consecutive. The decision on which frames to put into each sub stream is made using a Markov Decision Process, whose reward function takes into account the event that frames can not be decoded because a reference frame has been lost.

The authors of [CB01] perform a simulation study of interleaving applied to layered coding schemes. The proposed solution interleaves base information with enhancement one. The authors test also the usefulness of randomizing the obtained sequences. Simulations are conducted by using loss patterns from real traffic. Results are presented in terms of how many base packets are lost after applying both the interleaving only and the interleaving plus randomization. In most of the cases the interleaving and randomization are able to actually protect the base information from burst losses.

The authors of [CB01] refer to another work by them ([LAG03]) for what concerns the study of the channel and the derivation of the loss model. In [LAG03], in particular, they provide information about the effect of bursty rather than isolated losses on videos. They present a model to estimate the distortion caused by losses on videos. Simulation results show how a burst of losses actually causes more distortion than an equal number of isolated losses.

The work [WL99] is concerned with the design and verification of a technique to transmit voice over the Internet exploiting interleaving and a matrix-based transformation of the sent voice samples. The packet receiver continuously informs the sender about the status of the channel, i.e. the loss pattern. The sender then applies a matrix-based transformation to the voice samples in order to compensate the effect of lost samples. Moreover, the voice samples are interleaved by putting the odds ones in one packets and the even ones in another packet. Before explaining the details of the technique, the authors present the results of a measurement campaign aimed at estimating the loss pattern between 6 Internet hosts located in China, Japan, Italy, and USA.

The work [CC01] steps from the assumption (demonstrated in other papers according to the authors) that the MPEG encoder is more sensitive to isolated losses rather than burst losses. The authors then explain that performing an interleaving at symbol level before applying a channel coding scheme such as Reed-Solomon has the effect of protecting the bit stream from channel error, but results in isolated losses at the source (MPEG) level. Therefore they propose to apply interleaving at source level (to the bit stream produced by the source encoder), then to apply the FEC, and then to re-apply the interleaving on the obtained symbols. This will protect the stream from channel errors but will result in burst losses at the MPEG decoder. The reported results show that this scheme is able to increase the peak signal-to-noise ratio of about 5 dB.

2.2.3 Interleaving at bit/symbol level

In [SFLG00] the authors develop a model that allow to understand the achievable performance with a given channel and coding scheme. The model is also able to capture the effect of interleaving on the performance. Thanks to a theoretical analysis, the authors show how the interleaving is effective in decorrelating the losses, asymptotically approaching the memoryless case. They then conclude that interleaving is a very effective tool if the additional delay is acceptable.

With regards to wireless networks, an interleaving scheme to alter the order

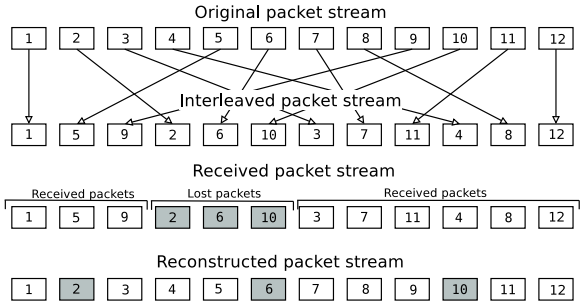


Figure 1: Packet level interleaving

of symbols before applying a FEC scheme is proposed in [CSSG04]. Taking into account Bluetooth networks, the authors present both analytical and simulation results that show how changing the order of the bits inside every single packets and then applying the standard Bluetooth FEC allows to obtain better performance. Simulation results related to TCP throughput are also presented.

2.3 Competing techniques

In [WtTAE02] the authors propose and evaluate a technique consisting in changing the order of packets in a streaming flow such that I and P frames are sent before the related B frames. In this way, even when the delay of the network is such that the I and P frames cannot be reproduced because they are too late, it can still be possible to reproduce the related B flows. This technique is particularly useful in networks with largely varying delays.

Kalman et al. [KSG02] propose a technique that combines adaptive media playout with rate-distortion optimized streaming. Basically, they suggest to employ a control strategy that works coordinating the media streaming server with the receiver. In such proposal the server uses an already proposed system to decide the rate of the streaming in order to optimize the total distortion. The client instead employs a technique to decide the playout speed (the frames are shown for a longer period if it has to slow down) as a function of the buffer status.

3 Time diversity at packet level

As shown in Section 2, the interleaving has been applied at different levels, which range from the bit/byte to an entire frame of a video stream. We decided to work at packet level for two reasons: i) the losses on the Internet happen at such level, and ii) this allows to exploit the flexibility of IP, which is not tight to a particular technology neither to a particular streaming application.

The basic idea to realize packet interleaving is to resequence the packets before transmission. The resequencing allows to space out originally close packets

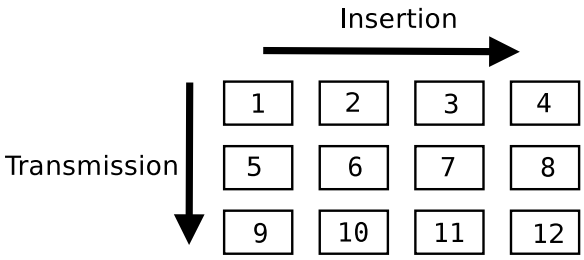


Figure 2: Block interleaving

such that a loss of consecutive packets is translated in a loss of distant ones. In Fig. 1 we report an example of such scenario. We can observe that using the interleaving on a sequence of 12 packets causes a loss of 3 consecutive packets (from the fourth to the sixth one) to be perceived at receiver side as the loss of packet number 2, 6, and 10. If interleaving was not used than the lost packets would have been the number 4, 5, and 6.

Interleaving techniques can be classified in periodic and pseudo-random. In the first case data is divided in sequence of equal length, using the same interleaving schema for all the sequences. In the second case, instead, pseudo-random sequences are used in which the position of data in the interleaved sequence is not the same for all the sequences. In practical realizations, periodic interleaving is almost always used due to its simplicity. For this reason in the following we will focus on this technique only.

The main disadvantage of packet interleaving is the delay introduced. In order to resequence a stream of k packets, it is necessary to wait for them. For example, looking at Fig. 1, packet number 4 has to wait until packet number 11 arrives before being transmitted. Such delay is however acceptable for certain kinds of applications such as audio and video streaming. Moreover, it is controllable through the length of the interleaving sequence.

Besides, even when the length of the sequence is chosen, determining the optimal permutation of the sequence is not an easy task. Up to now, the only solution for this problem has been an exhaustive search, which is clearly feasible only for small values of k. For this reason, interleaving is commonly implemented in blocks (in this case we speak about block interleaving). Block interleaving is made by inserting packets into a matrix by row, and picking them by column, as shown in Fig. 2. Basically, the flow of packets is divided in sequences of k packets. Each sequence is then placed into a *block*, a matrix of size $n \times m$ where: n, the number of rows, is called *interleaving depth*; and m, the number of columns, is called *block size*.

Considering a flow of packets $\mathrm{F} = \{P_a, P_b, P_c, ...\}$, and the example block of size 4×3 reported in Fig. 2. Then the packets will be inserted such that P_a will be in position $(1, 1)$, P_b will be in position $(1, 2)$, P_c will be in position $(1, 3)$, P_d will be in position $(1, 4)$, P_e will be in position $(2, 1)$, ... And the packets will be transmitted in the following order: $P_a, P_e, P_i, P_b, P_f, P_m, ...$

We can easily guess that the effectiveness of such transmission schema depends on the values of n and m[1], and on channel conditions. For example, a burst of length B will be converted in smaller bursts of length B/n. In the ideal case, 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, i.e. we can deal with bursts that are longer and longer. On the other hand, increasing n and m has two counter effects: 1) the number of bursts of small length increases; 2) the buffering delay, which is equal to $(n-1) \times (m-1)$, increases. This trade-off represents one of the main aspects to consider when we want to implement the interleaving on a real network, with real traffic. How to chose the parameters depends on different factors such as the channel conditions, the kind of application, etc. On the other hand, with this technique we are able to decorrelate losses in a simple way and with no additional bandwidth requirements.

3.1 Finding the optimal parameters

As explained before, an important challenge we have to fact if we want to deploy a time diversity technique is how to chose the block sizes n and m once we know the delay constraint. As proposed in [LAG02], we could proceed in a theoretical way as follows. Let us introduce the following notation.
$K_o = k_1, k_2, ...$ is the set of indexes of lost packets when the interleaving is not used,
$K_i = I(n, m, K_o)$ is the set of indexes of lost packets when the interleaving is used with a block of size $(n \times m)$,
$D[K_i] = D[I(n, m, K_o)]$ is the distortion associated with the decoded stream, which is a function of K_i and therefore of the parameters chosen for the interleaving. Such parameters are explicitly reported in the definition.
We can then set the problem of finding the optimal parameters in this way. Given the loss characteristics of the channel and the delay constraint (C_{delay}), we look for the couples (n_{opt}, m_{opt}) that minimize the distortion[2] D:

$$(n_{opt}, m_{opt}) = \underset{n,m:(n-1)\times(m-1)\leq C_{delay}}{\operatorname{argmin}} D[I(n, m, K_o)] \quad (1)$$

We could then think of an exhaustive search algorithm for solving this problem, which works as follows.

1: Estimate the loss pattern K_o using the channel characteristics.
2: Determine the possible couples (n, m) using the maximum allowed delay C_{delay}.
3: **for** all the (n, m) determined **do**
4: Determine K_i
5: Estimate $D[n, m, K_i]$
6: Determine (n_{opt}, m_{opt}) using (1)

[1] Once l and n are chosen, m is automatically determined.
[2] The distortion depends on the particular application considered.

7: **end for**

This algorithm requires the perfect knowledge of the channel characteristics, and, in particular, of the distribution of the burst length. As also stated in [ReST05], we can analytically obtain a closed form for such a distribution only for small interleaving delay and single path transmission. This poses a limit to the applicability of such an algorithm, which is therefore not effective to deal with real network scenarios. Moreover, performing an exhaustive search jeopardizes the simplicity of the interleaving as a methodology that can be used also on high-speed networks. For this reason, in order to determine the optimal parameter values we performed an extensive set of simulations, which are presented in the next section. Before presenting the simulations, in the following we explain the loss model we used.

3.2 Loss model

A typical characterization of the loss process over the Internet is provided by the continuous-time Gilbert model [G$^+$60], which is also referred as 2-state continuous-time Markov chain (2-MC) [ReST05, WCL07]. Such model is able to capture the potential correlation between consecutive losses, which is the common case on the Internet [YMKT99]. Due to FIFO queuing discipline used by routers, it is very common to discard a number of consecutive packets from the queue when a congestion event is detected. Moreover, routing decisions are taken at time intervals that are usually longer than flow durations. This implies that when a congestion happens, different packets from the same flows are discarded in most loss events. On the other hand, such model has also proven to be able to capture the loss dynamics on wireless networks [CSSG04]. This is because once the wireless channel turns bad, which happens because of attenuation, fading, scattering, or interference from other active sources, the errors will happen in bursts and becomes dependent on each other. It is also worth noting that 2-MC is not the only neither the fittest model for Internet traffic dynamics. It has actually been proved that Markov chains with more states are able to obtain higher modeling accuracy [YMKT99] in some cases. However, 2-MC represents the best compromise between complexity and accuracy.

Let $X = \{X(t) : t \geq 0\}$ the aleatory process of losses following the Gilbert model. The state X at time t can assume one of the following values: b or g (b = "bad" and g = "good"). Process $X(t)$ at a fixed time is characterized by the parameters μ_g and μ_b, which can be though as the rates at which the chain passes from state g to state b and vice versa. A diagram of the 2-state Markov chain is reported in Fig. 3.

In general, when $X(t)$ is in state b the probability to loose a packet is much larger than that in state g. To simplify the problem we assume that when a packet is transmitted and channel is in state b, then the packet is surely lost. The packet is surely received in the opposite case. For this reason we will refer to the two state as "loss" and "no-loss" beside "bad" and "good" respectively. The steady state probabilities to receive or loose a packet are respectively equal

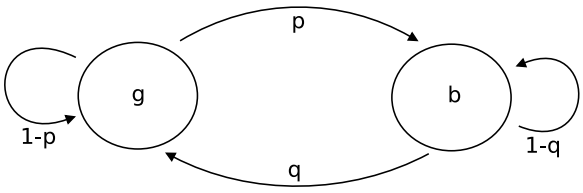


Figure 3: 2-state Markov chain

to: $\omega_b = \frac{\mu_b}{\mu_b+\mu_g}$ and $\omega_g = \frac{\mu_g}{\mu_b+\mu_g}$. It is worth noting that we consider a packet to be lost either when it is not delivered to the destination, or when it is delivered too late, and it is therefore not useful for the application (e.g. audio samples arriving after playback time). To further simplify the problem, let us substitute the continuous-time model with a discrete-time one. Such transformation is justified by the fact that we are only interested in the behaviour of the network when packets are sent. Therefore, once we fixed the sending rate to $S = 1/\tau$, with τ being the inter-packet time[3], we can express the transition probability in the discrete case as

$$p_{gg}(\tau) \equiv P(X_{n+1} = g|X_n = g) = \omega_g + \omega_b e^{-(\mu_g+\mu_b)\tau} \quad (2)$$

$$p_{gb}(\tau) \equiv P(X_{n+1} = g|X_n = b) = 1 - p_{gg}(\tau) \quad (3)$$

$$p_{bb}(\tau) \equiv P(X_{n+1} = b|X_n = b) = \omega_b + \omega_g e^{-(\mu_g+\mu_b)\tau} \quad (4)$$

$$p_{bg}(\tau) \equiv P(X_{n+1} = b|X_n = g) = 1 - p_{bb}(\tau) \quad (5)$$

In this way we can easily model the process of sending a flow of n packets on a link using a time-discrete Markov process that evolves in n steps. In particular, assuming the homogeneity of the Markov chain and choosing $\tau = 1$, the probabilistic description of such process can be obtained by using a 2-MC, see Fig. 3, having $p = p_{bg}(1)$, which is the probability that the next packet is lost given that the previous is not, and $q = p_{gb}(1)$, which is the probability that the next packet will be correctly received given that the previous was lost. In general, the following condition holds: $p + q \leq 1$. However, if $p + q = 1$ then the model becomes a Bernoulli one, for which losses are independent and happen with an average probability $\hat{p}$.

In the general case, the probability to loose n consecutive packets is equal to $(1-q)^{n-1}q$, according to the geometrical distribution. While the probability to receive n consecutive packets is equal to $(1-p)^{n-1}p$. Moreover, the steady-state probabilities for the two states are π_b, which is the probability of staying in state b, and π_g, which is the probability to stay in state g. They can be evaluated using the following

[3]Note that here we are implicitly assuming constant bit rate traffic. This is because the simulations presented in the following section are actually performed with this kind of traffic. Besides this, the model derivation can be extended for other kinds of traffic.

$$\pi_b = \frac{p}{p+q}, \quad \pi_g = \frac{q}{p+q} \tag{6}$$

In particular, π_b represents the average loss probability. Another important parameter, which is typically considered as the channel memory because it represents the correlation between losses, is ρ. It is defined as

$$\rho = \frac{1-q}{p} \tag{7}$$

Again, to obtain a Bernoulli model it is sufficient to impose $\rho = 1$, and the losses will be independent and identically distributed. This implies that a loss has the same probability to happen given that the previous state was b or g. On the other hand, $\rho > 1$ implies that a loss is more probable if the previous state was b than if it was g, i.e. if the previous packet was lost. For this reason, ρ is considered as an indicator of the channel memory.

A last important result is related to the probability of transition from state i to state j in l steps (with $l = 1, \ldots, n; \quad j, i \in \{b, g\}$, defined as

$$p_{ji}(l) \equiv P(X_{n+l} = j | X_n = i) \tag{8}$$

Starting from the 1-step transition matrix

$$\mathbf{Q} = \begin{pmatrix} (1-p) & p \\ q & (1-q) \end{pmatrix}$$

We have to calculate the l-step transition matrix $Q_l = Q^l$, and recall that [Rod74]

$$p_{bg}(l) = Q_l(1,2), \quad p_{gb}(l) = Q_l(2,1). \tag{9}$$

Otherwise we can use directly the following [VGS05]

$$p_{bg}(l) = \frac{p}{q+p}[1 - (1-q-p)^l], \tag{10}$$

$$p_{gb}(l) = \frac{q}{q+p}[1 - (1-q-p)^l]. \tag{11}$$

4 Time diversity in simulation

In order to understand the benefit of packet interleaving, and to determine the best trade-off values for its parameters (see Section 3.1), we conducted some simulations using Matlab^{TM}. In particular we develop some functions to simulate the behaviour of the interleaver and the network. Then we calculated some performance metrics for the received packet stream, evaluating the impact of the interleaving parameters on the obtained results. Details on the performed simulations and obtained results are presented in the next sections.

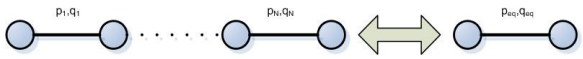


Figure 4: End-to-end equivalent channel

4.1 Network model

Let us consider a path between a source and a destination as a series of a number of links characterized by burst losses. If every link is modeled using a 2-MC, it can be easily demonstrated [San04] that the end-to-end path can still be modeled using a Gilbert model with equivalent parameters if the various links are independent.

In facts, given a series of N links of a path, modeled with 2-MC with parameters $(p_1, q_1), (p_2, q_2), \ldots, (p_N, q_N)$, it is possible to obtain an equivalent model with parameters (p_{eq}, q_{eq}), which represents the entire end-to-end path (or equivalent channel) as shown in Fig. 4.

It is worth underlining that the possibility to obtain such equivalent model is subject to the hypothesis that the parameters associated with the various links are related to the characteristics of the packet flow of interest. In other words, such parameters should be those that our flow of interested, or other flows with the same characteristics (packet sizes, inter-packet times, bit-rate, ...), would experiment. Such hypothesis is reasonable because the parameters $(p_1, q_1), (p_2, q_2), \ldots, (p_N, q_N)$, even if specifically related to the flow of interest, take into account also the effect of other flows constituting the background traffic [YMKT99].

Under such hypothesis, the Gilbert model for our equivalent channel will be characterized by the following probabilities (see Appendix).

$$p_{eq} = 1 - \prod_{i=1}^{N}(1 - p_i) \tag{12}$$

$$q_{eq} = \frac{\prod_{i=1}^{N} \pi_{gi}}{1 - \prod_{i=1}^{N} \pi_{gi}} p_{eq} \tag{13}$$

where p_{eq} is equal to the probability that the end-to-end path switches from state "no-loss" to state"loss", and q_{eq} equal to the probability that it passes from state "loss" to state "no-loss". This can be used for modeling a generic network topology based on the description of the single links of all the possible paths between a source and a destination using equivalent channels [JPF05]. In facts, if we have a topology in which the links are described using 2-MC, it is possible to obtain an equivalent representation using (12) and (13). On the other hand, if cannot obtain an equivalent channel mode by means of such equations, it is also possible to use the parameters of the worst link for the end-to-end path [GLT+02]. In this case, however, a lower accuracy will be obtained. Finally, thanks to an on-line estimation of the behaviour of the end-to-end paths between a source and a destination, it is possible to directly obtain the

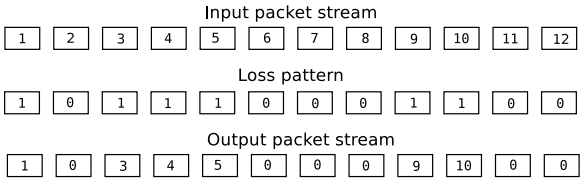


Figure 5: Simulation of packet loss

couples (p_{eq}, q_{eq}) associated to the various paths, and to continuously update them [TG04].

4.2 Running the simulations and collecting results

We model the packets generated by a source as an array of integers, each of which represents the sequence number of a packet. Such sequence is given as input to a function, called *path(I, ρ, π_b)*, which reproduces the behavior of the end-to-end path. The function therefore receives as input also the path characteristics in terms of π_b and ρ. It then calculates a pattern of losses using the model and the input parameters discussed above. After that, it applies the loss pattern to the input sequence in order to simulate the effect of the path traversal, substituting the sequence numbers of the packets that are lost with a 0. The output sequence will be therefore constituted by a series of positive numbers corresponding to the correctly received packets, and 0s corresponding to the lost packets (see Fig. 5).

When the interleaver is active, before being passed to the *path()* function, the input array is preprocessed by another function called *block_interleaver(I, l, n, m)*, which changes the order of the packets inside the input array in order to perform a block interleaving with parameters l, n, and m.

After the output sequence is obtained, we calculate some metrics to evaluate the effect of the interleaving. In particular, we evaluate the distribution of the length of the loss bursts, and the distribution of the distance between two loss event (i.e. the length of the no-loss sequences). While the importance of the length of the loss bursts is immediately clear, some considerations should be made about the no-loss sequences. Once the average loss rate is fixed, the loss bursts and the no-loss sequences represent two tightly linked aspects of the same phenomenon: 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 can cause similar effects to loss bursts, decreasing application performance. This happens, for instance, when certain types of codecs are used for videos and the distance between losses becomes lower than a threshold [LAG03]. In these cases no-loss sequence length plays an important role in order to understand if and how to use packet interleaving. For a increased readability of the results, in the following we report the average value and the variance of these two variables.

Table 1: Parameters of the simulations performed.

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

4.3 Results

We performed a number of simulations with different values for all the previously cited parameters. Such values are reported in Tab. 1. The simulations were conducted using all the possible combinations of the parameters reported in such table. As shown, we tested different values of average loss as well as different values of loss correlation. This allows to understand the impact of each single parameter.

For space constraints we cannot report all the obtained results. We therefore restrict our analysis to the most important ones, which are useful to assess the impact of the interleaving. The plots in Fig. 6 shows the average value of both the loss-burst length and the no-loss sequence length for four representative combinations of ρ and π_b, which are related to progressively degraded channel conditions. The plots in Fig. 7, on the other hand, present the same parameters but obtained using a channel with fixed π_b (i.e. fixed loss rate) and progressively increasing loss correlation (i.e. increasing burst-loss length).

As a first consideration, the results confirm the effectiveness of the interleaving in reducing the average loss-burst length. As shown in the two top plots of Fig. 6 and 7, in all the cases such parameter presents a decreasing trend when increasing the interleaving depth. We also observe that all the curves are steeper when the length of the loss-burst is high, i.e. the slope decreases when the interleaving depth increases. Which means that the more burstiness we have the more convenient is to use the interleaving. The same behaviour can be observed comparing the curves related to the different values of ρ: the higher is the correlation (i.e. higher burstiness) the more effective is the interleaving.

This translates in an asymptotic behaviour of the curves. Increasing the interleaving depth the curves tend to that achieved on a channel with no memory, which is a line parallel to the x-axis and determined only by the value of π_b. This happens because at a certain point the interleaving manages to make the losses perceived as uncorrelated, i.e. the channel behaves as a Bernoulli one. After such point, further increasing the interleaving depth provides no additional benefits. This behaviour is also confirmed by the plots related to the no-loss sequence length (bottom plots of Fig. 6 and 7). For this second parameter we observe how the interleaving actually decreases the no-loss sequence length (i.e.

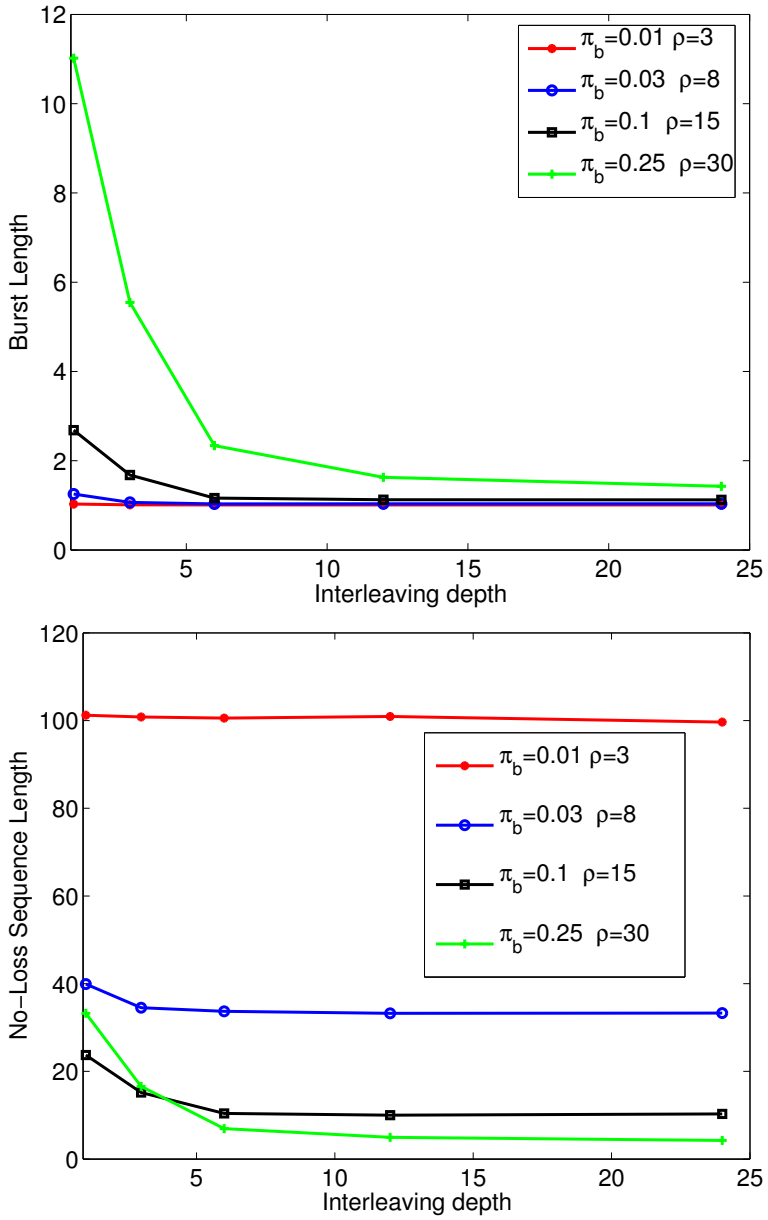


Figure 6: Simulation results obtained with different values of π_b and ρ.

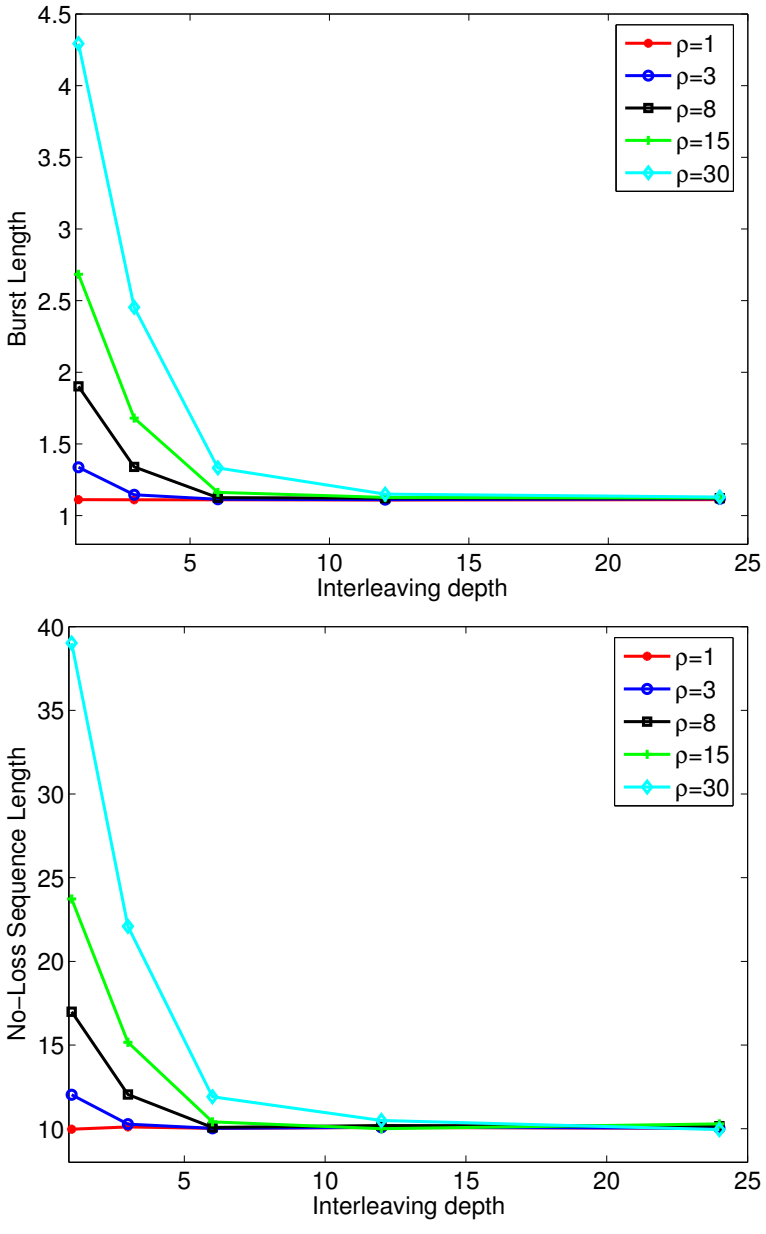
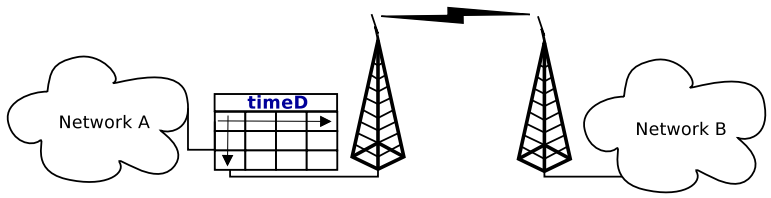
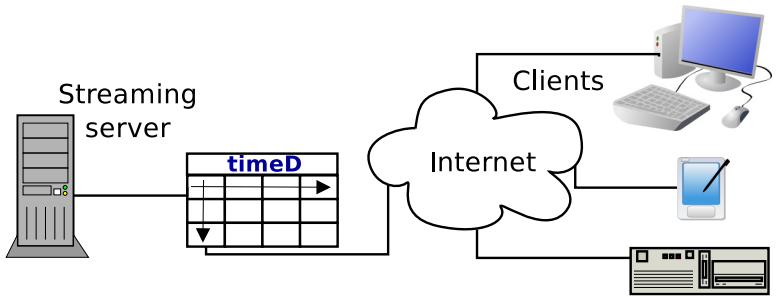


Figure 7: Simulation results obtained with different values of ρ and $\pi_b = 0.1$.



(a) Interleaving box before a wireless link.



(b) Interleaving box beside a streaming server.

Figure 8: Two possible application scenarios.

losses get closer), as anticipated in Section 4.2. Also, we observe that the trend of the plots is very close to that of the loss-burst length. This is also 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.

From this analysis we can devise a simple heuristic to determine the optimal parameters for the interleaving: the interleaving depth has to be chosen looking at the value for which the loss-burst length reaches the asymptotic value. Clearly, if we are interested to avoid that the no-loss sequence length goes below a certain value we have to take into account also the bottom plots of Fig. 6 and 7.

5 TimeD: deploying time diversity in real networks

We stepped from the idea of developing a solution to modify the order of packets traversing or generated by a host, working on every IP-based network, with every transport protocol and every application. For this reason our platform had to work at the IP level.

As a starting point, we thought about two possible application scenarios, which are depicted in Fig. 8. In the first scenario the interleaving (realized by the box called *TimeD* in such figures) operates on aggregated traffic just before it traverses a wireless link (or a particularly congested path). The second one sees the time diversity employed beside a streaming server to protect the video/audio traffic from the losses it will encounter on the Internet. While these are the two scenarios traditionally used to show the effectiveness of the interleaving, our application is actually able to work also in novel networking environments (see Section 7).

After deciding the initial operating scenarios, we passed to the identification of the implementation methodology and tools more appropriate for our aims. We decided to look at Linux operating system (OS) because it allows easier access to and modification of the networking stack with respect to other OS. Moreover, this OS provides more flexibility and allows the interleaving application to be easily deployed as an embedded system (we will call it interleaving box) in any operational network. We then considered the possibility to work both at kernel and user level. We opted for an user-space application for two reasons: i) it allows quicker and easier prototype development; ii) more importantly, it allows to interact with user-space monitoring applications, whose output is useful to tune the interleaving parameters. The second reason was particularly attractive for us, as explained in Section 8. After looking at different available possibilities for modifying the order of packets traversing a Linux host from user-space, we decided to use the *libipq*[4]: a mechanism provided by *netfilter*[5] for passing packets out of the kernel stack, queueing 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 we were able to modify the order of the packets that are enqueued into such *netfilter* queue, before re-injecting them into the kernel. As our application 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.

Passing packets from kernel- to user-space has surely an impact on the performance, which poses a limit on the maximum rate supported by *TimeD*. However, we were interested in developing a prototype to experiment with time diversity on real networks, having in mind the two initial application scenarios in Fig. 8. Therefore, we verified that the delay introduced by such a process is negligible with respect to the one of our experimentation scenarios, and that *TimeD* was able to sustain the rate used for the experiments. 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.

The implementation of *TimeD* was performed in C language. Fig. 9 shows a high-level view of the operations of *TimeD*. The first step is the initialization of the queue, creating a handle for it and defining the operation mode. After

[4] `https://svn.netfilter.org/netfilter/trunk/iptables/libipq`
[5] `http://www.netfilter.org`

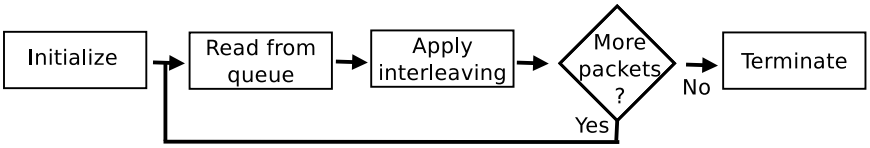


Figure 9: timeD high-level view

Table 2: Parameters of the experimentations.

Parameter	Values
Interleaving block size	12, 48
Interleaving block depth	[1, 2, 3, 6, 12, 24]
π_b ($loss$)	[0, 0.01, 0.03, 0.1, 0.25]
ρ (rho)	[0, 1, 3, 8, 15, 30]
Repetitions	20
Protocols	UDP, TCP, SCTP, DCCP 2, DCCP 3
Packet rate	10, 100, 1000 pps
Packet size	512 Bytes

that, the application starts a cycle in which it reads the packets from the queue, inserts them into the interleaving block, reorders the block according to the specified policy, and re-injects the packets into the kernel. At the end of the cycle, i.e. before exiting, *TimeD* destroys the handle of the queue releasing the resources. The current version of *TimeD* allows to define the size of the interleaving block (number of rows and columns), a timeout to wait for packets (see Section 7), and it uses a fixed policy for packet interleaving. An alpha version of *TimeD* is publicly available at [tra] under the terms of the GNU General Public Licence (GPL).

6 Testing and experimentation

We performed experimentations combining all the parameters reported in Table 2. In the following we first provide information about the test environment, and then discuss most important results we obtained.

6.1 Testbed and tools

In Fig. 10 we report the testbed we used for the experimentations. It is composed of two Linux hosts connected using a Fast Ethernet network interface to a hardware WAN emulator called Shunra Virtual Enterprise[6]. Such emulator is able to reproduce the behaviour of a WAN in terms of different parameters. For our experiments we used a feature called *WAN Cloud*, which allows to introduce arbitrary delay, jitter, and losses to the packets in transit. We set a loss pattern

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

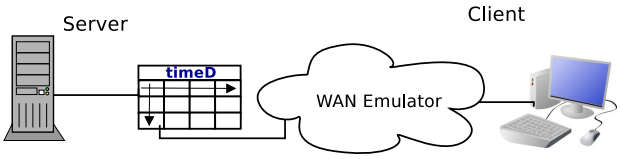


Figure 10: Testbed used for the experimentations.

equal to a 2-MC (the parameters are reported in Table 2), and left the delay and jitter unset (i.e. no delay or jitter is intentionally added). On the host named *Server* we run both the traffic sender application, *TimeD*, and the NTP client. On the host named *Client* we run the traffic receiver application and the NTP daemon. NTP was used to synchronize the clocks of the two hosts. While providing a good accuracy in LAN environments[7], the NTP daemon performs continuous adjustments to the clock in order to cope with clock skew (i.e. the difference between the frequencies of the two clocks). This can impact the results of the measurements as the clocks can change during an experiment. To avoid that, we instructed the NTP daemon on *Client* to only provide the clock on the network, and we manually launched an NTP client (i.e. ntpdate) on *Server* before each measurement cycle. To cope with the clock skew we performed a clock-skew detection and removal procedure on the acquired data [MST99].

For generating probe traffic we used D-ITG [BDP07], a highly customizable, packet-level traffic generator, which supports different transport-layer protocols (UDP, TCP, SCTP, and DCCP) and provides a large number of features for network measurement. Details about the configurations we considered in our experimentations are reported in Table 2.

6.2 Experiments and results

We first present a comparison in terms of loss decorrelation power between the results obtained in the experimentations and those obtained in simulation. After that, we present results related to the additional delay introduced by *TimeD*. This is to understand what are the problems we are going to face when deploying such an interleaving strategy on a real networks. For these experimentations we used UDP because we wanted to avoid interferences due to the activities performed by the other transport protocols.

Fig. 11 reports the average lengths of loss bursts and no-loss sequences in the four channel conditions analyzed in Section 4 (Fig. 6): $\rho = 3$ and $\pi_b = 0.01$, $\rho = 8$ and $\pi_b = 0.03$, $\rho = 15$ and $\pi_b = 0.1$, $\rho = 30$ and $\pi_b = 0.25$. For these tests we instructed the WAN emulator to reproduce the behaviour of such four channel conditions, and we let the probing traffic traverse the emulated WAN. We also instructed D-ITG to produce a log of the experiments. In such log D-ITG provides a sequence number for sent and received packets. Using such information we obtained the samples of the performance indicators

[7] `http://www.cis.udel.edu/~mills/ntp.html`

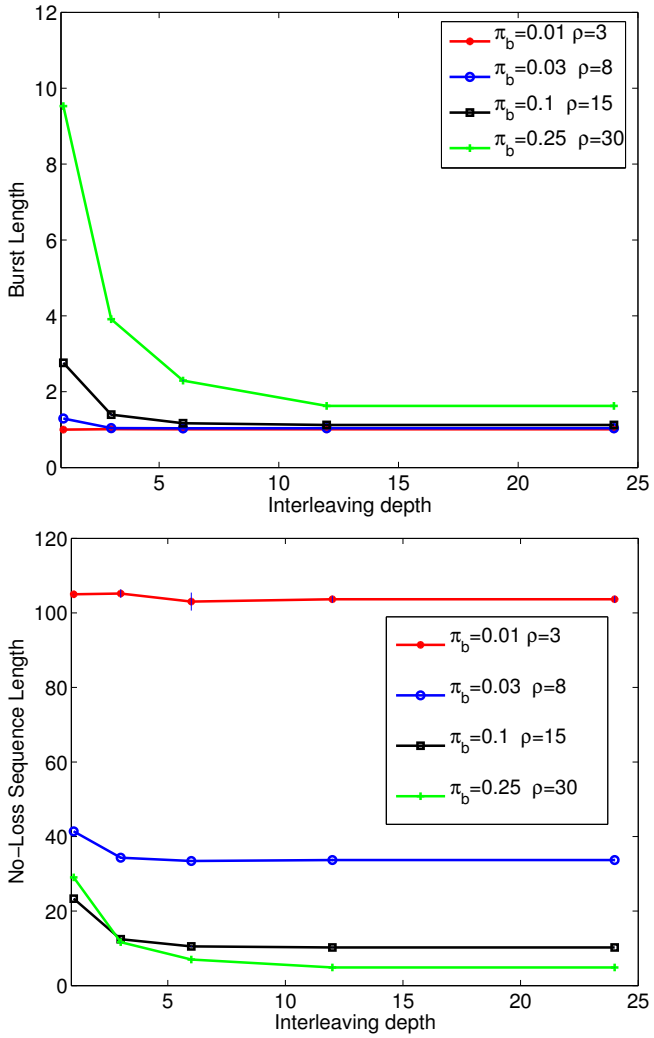


Figure 11: Experimental results obtained in four different channel conditions.

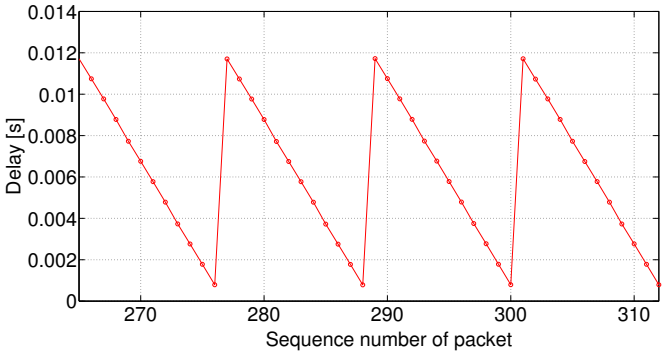


Figure 12: Packet delay of a UDP flow subject to a 3x4 block interleaving.

discussed before. After these experimentations we also modified D-ITG in order to directly report information about the loss bursts. The results in Fig. 11 have been obtained by sending UDP packets at a constant rate of 100 pps with a payload of 512 Bytes. From such figure 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. This shows the actual decorrelation power of the interleaving. Moreover, we observe that at a depth equal to 6 we achieve the asymptotic value for almost all the channel conditions. The results obtained in such experimentations are very close to those from the simulations. This means that the benefit of interleaving can actually be exploited in a real environment thanks to *TimeD*. It is worth stating that similar consideration can be done with the other traffic and channel conditions. The main differences have been obtained with the other transport-layer protocols. We discuss these issues in details in Section 7.

Beside the capability to decorrelate the losses we were also interested in understanding the additional effects that *TimeD* has on packet transmission. 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 interleaving on a real network we have to consider all the effects that we might expect. For these experiments we connected the two hosts in 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 in Table 2. 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. From this analysis we discovered that the overhead due to forwarding operations is in the order of few tens of microseconds at all the considered packet rates. We believe this delay can be considered negligible, at least for the two application scenarios considered.

For the buffering delay, we measured the transfer time experimented by UDP packets using all the rates and interleaving configurations reported in Table 2. Fig. 12 shows a zoom of the packet-by-packet delay of a UDP flow with rate of 1000 pps passing through *TimeD* instructed to perform a 3x4 block interleaving. The saw-tooth trend is due to the buffering performed by the interleaver, which waits for a block to be completed before re-injecting the packets into the network. The result is that the first packets are kept in the buffer for longer with respect to the last ones. In particular, the buffering delay experimented by the i-th packet of a block of size N is equal to

$$\delta_i = \sum_{j=i+1}^{N} IPT_j \quad \forall i = 1, \ldots, N-1 \tag{14}$$

where IPT_j is the time elapsed between the arrival of packet $j-1$ and j (i.e. the inter-packet time). The last packet will not experiment any buffering delay. In the CBR traffic case, IPT is constant and the (14) becomes

$$\delta_i = (N-i) \times IPT \quad \forall i = 1, \ldots, N \tag{15}$$

This explains the regular trend of Fig. 12. As already remarked before, such delay has to be carefully taken into account from the application point of view. For example, if we want to use interleaving for the streaming server in Fig. 8, we could assume that the stream has a constant packet rate of 720 pps (i.e. frame rate of 24 frames per second, 30 slices per frame, and a single slice per packet) and packet sizes following a normal distribution [GW94]. In this case, using an interleaving block of 48 packets implies a maximum buffering delay of about 65 ms, which is acceptable in most cases. However, such buffering delay will be a cause of problems for transport protocols implementing a congestion control algorithm. We discuss this issue in details in the following section.

Finally, we performed experiments aimed at understanding the maximum throughput that *TimeD* is able to sustain. From these experiments we learned that *TimeD* is able to work at full speed (i.e. 100Mbps) with all the interleaving configurations in Tab 2, even when running on the same host as the traffic generator.

7 Coping with congestion control

The results presented in the previous section allowed to understand that the interleaving in a real network behaves very closely to the simulation environment. However, it worth saying that part of the merit is surely of UDP, which does nothing more than adding a small (yet important) header to IP packets. In current Internet scenarios multimedia traffic is transported more and more frequently by protocols implementing at least a congestion control algorithm. This is due on the one hand to the fact that such protocols are better suited to adapt to varying network conditions, and on the other hand to the fact that multimedia contents are frequently accessed through the WEB, encapsulated

into HTTP. For this reason, it is important to understand the interactions between packet interleaving and congestion control. To this end we performed a set of experiments with different transport protocols implementing such a control: TCP, SCTP, and DCCP with both CCID 2 and CCID 3 (we simply call them DCCP 2 and DCCP 3 in the following).

In these experiments we completely disabled the WAN emulator and we connected the hosts back-to-back. This was to isolate the effect of the losses from that of the interleaving. The first condition we experimented was that the connection between the two hosts was not established with almost all the protocols. Only TCP was able in some cases to complete the 3-way handshake and to send some packets. This is due to the fact that the congestion control algorithms start sending packets at a very low rate, which is only increased when acknowledgments are received. The buffering operation performed by *TimeD* is clearly in contrast with such a behavior because the block has to be filled in order to release the packets. The reason why TCP was able to actually overcome such big obstacle is because it initially filled the buffer with several retransmissions of the first packets. DCCP instead, while implementing a congestion control algorithm, does not retransmit packets. So it was completely stuck in this situation. For SCTP, finally, the number of maximum allowed retransmissions for the first packets was set to a lower value with respect to TCP. Therefore, also for such protocol *TimeD* locked the connection. It is also worth noting that the performance of TCP in this situation was really low, and the end-to-end delay reached values up to 30 s.

To cope with this problem we modified *TimeD* introducing a timeout for the buffering operation. Basically, thanks to this modification, *TimeD* releases the packets when either the interleaving block is full or a certain amount of time has passed. This is already sufficient to avoid freezing the connections of the protocols with congestion control. After such modification we performed experimentations aimed at understanding the benefits of the timeout. In particular we repeated all the experiments of the previous section, setting a timeout value equal to $N \times IDT \times 1.1$. In this way UDP packets will never make a timeout expire, and they will therefore experiment the same conditions as before.

Table 3 shows the average delay experimented by the packets with different transport protocols and interleaving configurations. Remember that we are not inducing any loss on the path. Therefore we are only observing the effect of the interleaver. Recall also that the average buffering delay for a CBR flow subject to interleaving with block size N can be obtained using Eq. (15). In particular, the average buffering delay δ_{avg} is

$$\delta_{avg} = \frac{1}{N} \sum_{i=1}^{N} \delta_i = \frac{(N-1) \times IPT}{2} \tag{16}$$

If we look at the values reported in Table 3 for UDP we can observe that they are very close to the theoretical values. On the contrary, all the other protocols considered the buffering performed by *TimeD* as a congestion. As a result they reduced the sending rate, and their packets experiment a higher delay. It

is interesting to note that the average delay experimented by DCCP packets is higher than that of TCP and SCTP. This is due to the fact that the last protocols performed a lot of retransmissions. As a consequence the actual rate of packets in the network is higher, and the packets waited in the buffer for a shorter time.

Table 3: Average delay of the packets when sending at 100 pps rate.

Block size	Delay [s]				
	UDP	*TCP*	*SCTP*	*DCCP* 2	*DCCP* 3
3x4	0.059	0.123	0.197	0.280	0.277
6x2	0.059	0.123	0.146	0.280	0.276

Table 4 presents the number of packets received using the 4 different protocols when generating CBR traffic at 100 pps for 120 s, and using a 3x4 interleaving block. We observe that UDP, TCP, and SCTP were able to transfer all the packets injected by D-ITG, while DCCP was able to transfer less than the half of such packets. The behaviour of UDP is clearly due to the fact that it completely ignores the status of the network. TCP and SCTP were slowed down by *TimeD* in the first period. However, they were able to quickly recover thanks to the retransmissions. After a while, they reached a sending rate even higher than that requested by the application, and they managed to send all the requested packets.

We believe these considerations have to be carefully taken into account for deploying interleaving on a real network. Besides, it is also worth underlining that the timeout mechanism we implemented in *TimeD* is a simple way to solve the issues with congestion control algorithms. In the case of DCCP we also observed a decrease of the performance (less packets are delivered also when there are no losses) due to the fact that the buffering is seen as a congestion. To solve this problem we are now working on extending *TimeD* functionalities introducing a more effective buffer management.

Table 4: Percentage of received packets sending 100 pps for 120 s.

Block size	Received packets [%]				
	UDP	*TCP*	*SCTP*	*DCCP* 2	*DCCP* 3
3x4	100.0	99.9	99.8	40.8	41.4
6x2	100.0	99.9	99.9	41.5	41.6

8 Conclusion

In this paper we paved the way to the deployment of time diversity at packet level in real networks. In particular we provided the following contributions: i)

we developed a simulation environment for the study of time diversity in packet networks; ii) using such simulator, we studied the achievable performance of interleaving in terms of loss decorrelation, and we obtained the optimal configuration parameters for several network conditions; iii) we designed, implemented and publicly released a prototype application (called *TimeD*) that allows to experiment with time diversity in real networks; iv) we experimentally studied packet interleaving with different application scenarios and protocols (UDP, TCP, SCTP, and DCCP), showing how it it possible to obtain the same benefits observed in simulation; v) we estimated the overhead of a platform to perform packet interleaving in terms of buffering and forwarding time and achievable rate; vi) we analyzed the interactions between time diversity and congestion control protocols, and we proposed a simple modification to *TimeD* in order to cope with such scenarios, evaluating also the achievable performance.

Our ongoing work is concerned with extending the functionalities of *TimeD* in order to improve its performance especially when dealing with TCP, SCTP, and DCCP. Our idea is to complement it with network monitoring features in order to automatically and continuously tune the interleaving parameters. Moreover, we are performing a careful analysis of the interleaving applied on bidirectional traffic using *TimeD* on both ends of a network path.

References

[BCVG95] Jean-Chrysostome Bolot, Hugues Crépin, and Andrés Vega-García. Analysis of Audio Packet Loss in the Internet. In *NOSSDAV '95: Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 154–165, London, UK, 1995. Springer-Verlag.

[BDP07] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. Multi-protocol and multi-platform traffic generation and measurement. In *IEEE Conference on Computer Communications (INFOCOM07), Demo session*, 2007.

[CAK+04] S.H. Chung, D. Agrawal, M.S. Kim, J.W. Hong, and K. Park. Analysis of Bursty Packet Loss Characteristics on Underutilized Links Using SNMP. *IEEE/IFIP End-to-End Monitoring Techniques and Services (E2EMON)*, 2004.

[CB01] Suk Kim Chin and R. Braun. Improving video quality using packet interleaving, randomisation and redundancy. In *Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on*, pages 405–413, 2001.

[CC01] Jianfei Cai and Chang Wen Chen. Fec-based video streaming over packet loss networks with pre-interleaving. In *Information Technology: Coding and Computing, 2001. Proceedings. International Conference on*, pages 10–14, Apr 2001.

[CSSG04] Ling-Jyh Chen, T. Sun, M.Y. Sanadidi, and M. Gerla. Improving wireless link throughput via interleaved FEC. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 1, pages 539–544 Vol.1, June-1 July 2004.

[CZ03] Mark Claypool and Yali Zhu. Using Interleaving to Ameliorate the Effects of Packet Loss in a Video Stream. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 508, Washington, DC, USA, 2003. IEEE Computer Society.

[G+60] E.N. Gilbert et al. Capacity of a burst-noise channel. *Bell Syst. Tech. J*, 39(9):1253–1265, 1960.

[GLT+02] L. Golubchik, J. C. S. Lui, T. F. Tung, A. L. H. Chow, W.-J. Lee, G. Franceschinis, and C. Anglano. Multi-path continuous media streaming: what are the benefits? *Perform. Eval.*, 49(1-4):429–449, 2002.

[GW94] Mark W. Garrett and Walter Willinger. Analysis, modeling and generation of self-similar vbr video traffic. *SIGCOMM Comput. Commun. Rev.*, 24(4):269–280, 1994.

[JPF05] D. Jurca, S. Petrovic, and P. Frossard. Media aware routing in large scale networks with overlay. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 892–895, July 2005.

[KSG02] M. Kalman, E. Steinbach, and B. Girod. R-D optimized media streaming enhanced with adaptive media playout. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 869–872 vol.1, 2002.

[LAG02] Yi J. Liang, John G. Apostolopoulos, and Bernd Girod. Model-Based Delay-Distortion Optimization. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, 2002.

[LAG03] Y.J. Liang, J.G. Apostolopoulos, and B. Girod. Analysis of packet loss for compressed video: does burst-length matter? In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 5, pages V–684–7 vol.5, April 2003.

[LR05] Jin Young Lee and Hayder Radha. Interleaved source coding (ISC) for predictive video coded frames over the Internet. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 2, pages 1224–1228 Vol. 2, May 2005.

[LR06] Jin Young Lee and H. Radha. Evaluation of Interleaved Source Coding (ISC) over Channel with Memory. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 1152–1157, March 2006.

[MST99] S.B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 227–234 vol.1, Mar 1999.

[Pax97] Vern Paxson. End-to-end Internet packet dynamics. *SIGCOMM Comput. Commun. Rev.*, 27(4):139–152, 1997.

[Pax99] Vern Paxson. End-to-end internet packet dynamics. *IEEE/ACM Trans. Netw.*, 7(3):277–292, 1999.

[ReST05] B. Ribeiro, E.S. e Silva, and D. Towsley. On the efficiency of path diversity for continuous media applications. *UMass CMPSCI Technical Report 05-19*, 2005.

[Rod74] Coleman Rodney. *Stochastic Processes*. Allen & Unwin Ltd, London, 1974.

[San04] Petrovic Sanja. Analysis of packet loss process in multipath networks. *EPFL Technical Report*, 2004.

[SFLG00] K. Stuhlmuller, N. Farber, M. Link, and B. Girod. Analysis of video transmission over lossy channels. *Selected Areas in Communications, IEEE Journal on*, 18(6):1012–1032, Jun 2000.

[SRPIP04] P. Salvo Rossi, F. Palmieri, G. Iannello, and A.P. Petropulu. Packet interleaving over lossy channels. In *Signal Processing and Information Technology, 2004. Proceedings of the Fourth IEEE International Symposium on*, pages 476–479, Dec. 2004.

[TG04] S. Tao and R. Guerin. On-line estimation of internet path performance: an application perspective. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1774–1785 vol.3, March 2004.

[tra] `http://www.grid.unina.it/Traffic`.

[VGS05] E. Vergetis, R. Guérin, and S. Sarkar. Improving Performance Through Channel Diversity in the Presence of Bursty Losses. In *Proc. ITC-19*, Beijing, China, August 2005.

[WCL07] David X. Wei, Pei Cao, and Steven H. Low. Packet Loss Burstiness: Measurements and Implications for Distributed Applications. *Parallel and Distributed Processing Symposium, International*, 0:222, 2007.

[WL99] B.W. Web and Ding Lin. Transformation-based reconstruction for real-time voice transmissions over the internet. *Multimedia, IEEE Transactions on*, 1(4):342–351, Dec 1999.

[WtTAE02] S. Wee, Wai tian Tan, J. Apostolopoulos, and M. Etoh. Optimized video streaming for networks with varying delay. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 89–92 vol.2, 2002.

[YC97] Jey-Hsin Yao and Yao-Min Chen. Experiments of Real-Time MPEG Audio over the Internet. *Fujitsu Scientific and Technical Journal*, 33(2):138–144, 1997.

[YKT96] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the MBone multicast network. In *Global Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity*, pages 94–99, Nov 1996.

[YMKT99] M. Yajnik, Sue Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependence in packet loss. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 345–352 vol.1, Mar 1999.