Assessment of Recovery Journal-based Packet Loss Concealment Techniques for Low Latency MIDI Streaming

Leonardo Severi, Antonio Cuccarese, Andrea Bianco and Cristina Rottondi[*]
(leonardo.severi@polito.it) (antonio.cuccarese@studenti.polito.it) (andrea.bianco@polito.it) (cristina.rottondi@polito.it)

Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy

In networked music performances, real-time Packet Loss Concealment (PLC) is a task of pivotal importance to compensate the detrimental impact of loss or late delivery of audio portions that often occur in low-latency audio-streaming scenarios.
This paper proposes an open-loop PLC method tailored for MIDI data and compares it to a closed-loop state-of-the-art benchmark in terms of effectiveness of audio recovery and communication overhead. Moreover, implementations aimed at reducing the computational overhead are proposed and compared for both approaches. Results show that the proposed open-loop policy achieves performances similar to those of the closed-loop one, while reducing the number of operations executed at the transmitter side.

0 Introduction

A Networked Music Performance (NMP) is a real-time musical interaction where musicians displaced in different geographical locations convey locally-generated sounds to their remote counterparts by means of low-latency audio streaming over a telecommunication network (typically the Internet) [1]. These interactive music sessions feature packet transmissions from a sender to one or multiple receivers. Packets may include digitalized audio (either raw or encoded) or, less frequently, MIDI data [2]. MIDI data do not carry sampled audio but rather digitally encode in a MIDI message an action performed by the player (e.g., a note activation/deactivation or a piano pedal pressure/release).

To provide a satisfactory Quality of Experience (QoE) during the performance and to ensure that musicians maintain the required synchronism for ensemble playing, data transmission must be both reliable and timely. Concerning latency constraints, several empirical studies (e.g. [3, 4]) show that, to ensure maintenance of a stable tempo, the mouth-to-ear latency experienced by NMP performers should not exceed a few tens of ms, depending on the musical genre, timbral characteristics of the instrument and leading/following role within the ensemble. For what concerns instead timeliness of delivery of audio data, high packet delay/latency, jitter, and losses are unfortunately unavoidable in the vast majority of real deployment scenarios. If such events occur, well-known packet retransmission techniques (e.g. those implemented by the TCP transport protocol [5, 6]) are typically avoided in NMP applications, as they would significantly increase the mouth-to-ear delay. As a consequence, most NMP systems rely on UDP [7], which does not implement retransmission mechanisms, thus not providing any guarantee of reliable data transfer. If the receiving application does not receive data in due time for sequential playback, due to losses, or excessive delays, or both, Packet Loss Concealment (PLC) mechanisms must be implemented to mitigate the impact of missing data during the playout of the received audio signal.

In this paper, we consider the RTP-MIDI protocol [8] over UDP. We propose an open-loop PLC method (i.e. a concealment mechanism not relying on feedback information exchanged among a receiver-sender pair) tailored for NMP applications and aimed at limiting the perceived impact of lost/delayed packets carrying MIDI messages. The basic principle of the proposed approach is to periodically integrate the differential information contained in MIDI messages with a representation of the current MIDI state at the sender side. The sender MIDI state can be constructed straightforwardly, without the need of any form of feedback from the remote counterpart. To validate the proposed approach, we simulate the streaming of MIDI data along a telecommunication network in presence of

[*]To whom correspondence should be addressed e-mail: cristina.rottondi@polito.it

packet losses and evaluate the similarity between the original MIDI stream and the one reproduced at the remote side after execution of the PLC mechanism. Moreover, we evaluate the network overhead introduced by our solution and compare it to the state-of-the-art default PLC method integrated into the RTP-MIDI protocol [8]. Results show that the proposed open-loop policy achieves similar performance to that of the RTP-MIDI benchmark , while reducing the number of operations executed at the transmitter side.

The rest of this paper is organized as follows: Sec. 1 provides basic background notions on the MIDI standard, whereas Sec. 2 briefly reviews the related literature. Sec. 3 describes the considered NMP scenario and Sec. 4 presents the proposed PLC method, the performance of which is assessed in Sec. 5. Some final remarks are summarized in Sec. 6.

1 Background on MIDI

According to [2], "The Musical Instrument Digital Interface (MIDI) protocol provides a standardized and efficient means of conveying musical performance information as electronic data." Fig.1 depicts the classical use of the MIDI protocol , whereby MIDI messages are transmitted over a MIDI cable to connect a MIDI instrument to a synthesizer that translates MIDI commands to audio.

A MIDI data stream is composed of MIDI Messages which are typically 2 or 3 bytes long.

Each MIDI Message affects and modifies the current state of the synthesizer: for example, a Note On message instructs the synthesizer to start playing the given note (see Fig. 2 for an example). The MIDI Standard also defines a way to store MIDI information in files. Messages are stored in tracks: each track contains the messages representing the events in the same order in which they are produced by the instrument. In each track, up to 16 independent channels (channels could be associated, for example, with different instruments) can be defined. In the track, messages are interleaved with time information: time is encoded as variable-length big-endian integers (1 to 4 bytes, for each byte 1 bit is used as prefix and 7 bits are used for value encoding) representing the difference in terms of ticks (fraction of quarter notes) between two subsequent events (0 for simultaneity). The first byte, also called "Status Byte", brings information about the type of musical event (e.g., note or command) and indicates which channel it belongs to. For each status byte, the count and the meaning of the successive bytes is documented in [2].

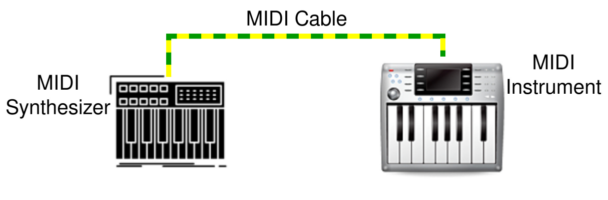


Fig. 1: Basic MIDI setup with a MIDI instrument sending commands to a hardware synthesizer.

MIDI instruments can be modelled as Finite State Machines (FSM), where the set of possible states includes all the possible combinations of values of their controllable parameters. In other words, every combination of parameter values corresponds to a distinct state. RFC4696 [9] provides examples of implementations of the MIDI state concept in the context of the RTP-MIDI protocol. In this paper, we define a MIDI system *state* $S(t)$, as the set of values assumed by MIDI parameters at a given time t, mainly focusing on *Note* and *Control Change* as classes of parameters that constitute a MIDI state. The MIDI state permits to identify a set of MIDI events that can lead the MIDI system to such state starting from any other state, including the default idle one, with no active events: in FSM jargon, this set of events represents a *transition*. Such events are those producing an active effect on the audio playback (e.g., an active note is a note which has been triggered by a Note On event generated at a time instant $t' < t$ and has not yet been stopped by a correspondent Note Off event). The system state can therefore be considered as a sort of "snapshot" of active/inactive MIDI events in the system, which, however, does not provide any information about the starting time or duration of any specific event.

Note that the MIDI standard can also be used in a networked environment, where the instrument and the synthesizer are connected via the Internet, as described in Sec.3.

2 Related Work

PLC in audio streaming is a mature topic, that has been widely addressed by the scientific community [10], especially for speech signals in VoIP applications [11, 12, 13]. Focusing on NMP systems, several efforts have already been dedicated to devising low-latency PLC techniques for audio streams: for example, robust audio codecs with integrated PLC have been proposed (such as OPUS [14] and [15]), as well as multiple-description audio coders, whereby audio frames are encoded into several redundant packets, so that a single correctly received packet provides a minimum acceptable quality, whereas the information contained in multiple packets can be combined to further improve the perceived quality level [16, 17, 18]. Moreover, predictive PLC methods specifically tailored for NMP purposes have been recently proposed, such as those based on Autoregressive Models [19] or on Machine Learning algorithms [20]. However, only a limited number of studies have focused on PLC techniques for MIDI-based NMP.

In [8], Lazzaro et al. proposed a protocol based on RTP [21] to encapsulate MIDI events, as well as the usage of a so-called "Recovery Journal" (RJ) to implement PLC. The RJ encodes the difference between two states of the system. Let us denote as sender the application that produces MIDI events and sends them over the network, and as receiver the application that receives, synthesizes, and reproduces them. Considering states S_1 and S_2, where S_1 is the state of the sender at t_1 and S_2 the one at t_2, with $t_1 < t_2$, the RJ for states S_1 and S_2 represents the set of differences

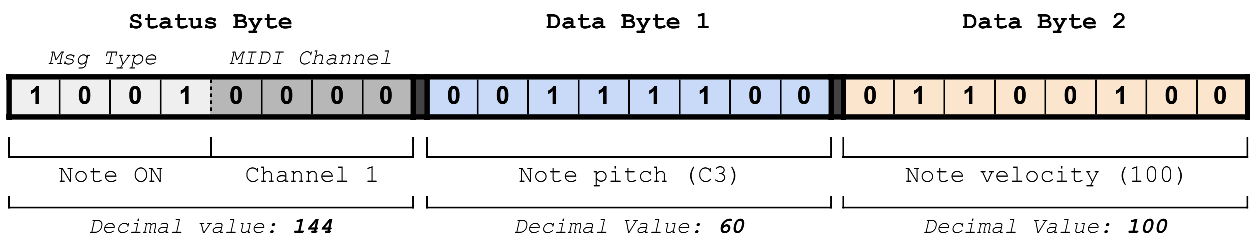


Fig. 2: Example of a MIDI message (Note On message referring to the C3 note onset with velocity 100 on MIDI channel 1).

between S_2 and S_1. In the remainder of this paper, such set of differences will be referred to as ΔS. Upon construction of an RTP packet, the RJ is calculated by considering S_2 to be the sender state just before the occurrence of the events carried in the packet under construction (which RJ is appended to), whereas S_1 is chosen according to a policy negotiated at the beginning of the session: the default policy is *closed-loop*, which implies that the sender expects the receiver to send an acknowledgment whenever it receives a packet. Upon reception of an acknowledgment related to a packet generated at time t_i, $S_1 := S_i$ applies. This method constitutes the core of the RTP-MIDI protocol described in RFC-4695. Several software solutions already handle MIDI, for instance SonoBus [22] or HPSJam [23], which use a custom protocol to deliver MIDI data over the network. Instead, other Digital Audio Workstations (DAWs) can route MIDI messages through the network thanks to the implementations of the RTP-MIDI protocol offered by some operating systems, according to RFC-4695 (e.g. the Ableton Live DAW can stream MIDI thanks to the Apple driver for MacOs which implements the RTP-MIDI protocol [24, 25]).

A different PLC method specifically tailored for NMP over wireless telecommunication networks is presented by Virolainen at al. in [26]. It consists in categorizing MIDI messages to be sent as critical or non-critical. Messages belonging to the critical category are transmitted using a reliable transmission protocol, whereas non-critical MIDI messages are transmitted using a non-reliable transmission protocol. Therefore, this approach suggests a classification of MIDI messages, based on their priority level: for example, a Note On message belongs to a non-critical category, while the corresponding Note Off message belongs to a critical category. Thereby, delays/losses of messages transmitted through the unreliable channel could lead to the non-reproduction of MIDI events (e.g., a note is not played), while errors on messages transmitted through the reliable channel could lead to a delayed action (e.g., the duration of a note could be altered).

A variation of the above-mentioned method is proposed in [27], which implements a severity assessment phase carried out by the receiver, upon detection of a packet loss. If the error is considered severe enough, a recovery phase

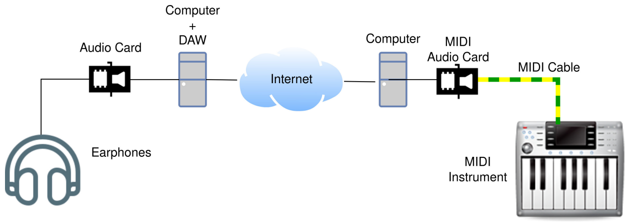


Fig. 3: Example setup for a MIDI-based NMP session in which MIDI commands are produced by the MIDI piano, received by a local computer through a MIDI audio card and sent over the Internet, then received by the remote musician's computer, synthesized as analog audio by a Digital Audio Workstation (DAW) and reproduced through a local audio card.

is activated, which involves replacing missing MIDI data with recovery data transmitted along an auxiliary channel. Recovery data is assumed to be generated by a dedicated server in real-time or pre-calculated.

Another approach for PLC of MIDI data specifically tailored for transmission over wireless networks is described in [28]. It leverages a packet acknowledgment mechanism that ensures retransmission of audio data that did not reach the receiver.

Differently, our proposed method operates in *open-loop*, i.e., it does not require the implementation of any acknowledgment mechanism, so that it can also be deployed on simplex channels. Moreover, it does not require two communication channels characterized by different transmission reliability guarantees nor the discrimination between critical and non-critical messages. In addition, the wireless communication channels considered in [26] adopt radio frequency or optical (e.g., infrared) transmission, whereas our proposed solution is agnostic to the transmission technology adopted at the physical layer.

3 NMP System Architecture

Access to an NMP session is typically negotiated between the software used by the musician and a server dedicated to NMP session management, which provides the

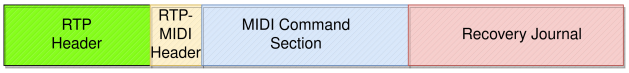


Fig. 4: High level packet structure, according to RFC4695. The RTP Header is the one described in RFC3550, without header extensions.

joining musician with the list of IP:port pairs of the devices of all the remote musicians participating in that session and manages the connection setup. An example of an audio architecture for a MIDI-based NMP session is depicted in Fig.3. For the sake of simplicity, in this description we limit the number of musicians to two and we only show a unidirectional flow from the right musician's instrument to the left musician's earphones. The instrument played by the local musician is connected to a device that runs a dedicated NMP transmitter application, which communicates through a telecommunication network with the NMP application running on the device of the remote musician that receives the audio data.

In a realistic scenario, two (or more) musicians acquire/transmit their own MIDI data and receive/reproduce the data of their remote counterpart(s). In such a case, the NMP application executed by each musician acts both as sender and receiver. The sender process is responsible for conveying the MIDI data generated by the musician's instrument to the remote player. Conversely, the receiver process obtains MIDI messages contained in the data stream generated by each of the remote musicians. The receiver application is responsible for concealing the effects of possible packet losses in each stream independently.

3.1 Communication Protocol and Message Format

We assume the usage of the RTP-MIDI application-level protocol running over UDP, as in RFC4695 [8]. RTP-MIDI deals with possible out-of-order datagrams by including a sequence number. This is provided by the RTP standard reported in RFC3550[21], which implements source identification (and multiplexing), packet ordering, and timing. To send MIDI messages over RTP, we will refer to the message format described in [8], which takes into account all the possible scenarios that may occur during a MIDI session.

The high-level structure of the application layer packet, including an RTP and an RTP-MIDI header, a MIDI Command section and an RJ section, is depicted in Fig.4. The sending algorithm [29] implemented by the NMP transmitter requires defining a grouping period Δt, which is the time interval between the generation of two consecutive packets. Straightforwardly, the longer the grouping period, the less the overhead generated by both the application and transport layers, but increasing the duration of the grouping period also leads to an increment of the perceived mouth-to-ear latency and to larger packet sizes (with the risk of exceeding the Maximum Transmission Unit). Practical values of the grouping period are in the order of few milliseconds [1], even though they are likely to imply the

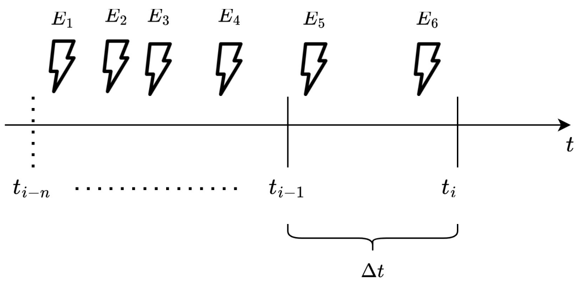


Fig. 5: An example of RJ construction: using Δt as grouping period and assuming that a new period starts at time t_i, the Command section will contain E_5 and E_6 whereas the RJ will encode information about the difference between the system state at t_{i-1} and the system state at time t_{i-n} (including events from E_1 to E_4)

transmission of several empty packets, since distinct musical events are usually temporally interleaved by at least some tens of milliseconds [30]. During each grouping period Δt, the sender process of the NMP application gathers all the MIDI messages representing the musical events generated by the musician's instrument, and populates the MIDI Command section of the packet, which follows the same data representation adopted for file tracks (see Sec. 1). Upon generation of a MIDI event by the user's instrument, the system state also gets updated. Conversely, the RJ Section of the packet implements a representation of the set of differences between system states (ΔS). Referring to Fig. 5, the RJ does not include events that are carried in the Command section of the packet it is appended to. Instead, it encodes the difference between states S_{i-1} and S_{i-m}, where S_{i-1} is the system state at time instant t_{i-1} (i.e., the beginning of the most recent grouping period) and S_{i-m} is the system state at time instant t_{i-m} (i.e., the beginning of m-th preceding grouping period). The choice of n determines the adopted policy. Ref. [8] focuses on the *closed-loop* policy, considering it as the default policy, and envisions as alternatives an *anchor* policy and an *open-loop* policy. The *closed-loop* policy expects the receiver to send back acknowledgments in forms of RTCP - Receiver Report (RR) about received packets. Whenever a RR for packet j is received, the RJ for packet i is computed as the difference between state S_{i-1} and state S_j. The *anchor* policy is the simplest one: every RJ is computed as a difference between states S_{i-1} and S_0. It is important to remark that, since in the *anchor* policy ΔS is computed as the difference between the sender's current state, S_{i-1}, and the initial (default) state, S_0, the receiver can adjust its current state (which we denote as S_c), even if such state is different from the default one. The steps necessary to change S_c to S_i, only knowing $\Delta S = S_{i-1} - S_0$ and the MIDI commands occurred in between time t_{i-1} and t_i, can be easily implemented as described in section 4.2. The *open-loop* policy lets the sender decide autonomously how to set m, which is then kept constant. This policy may lead to unrecoverable losses in the playback (for example, when more than m consecutive packets are lost): according to Ref. [8], this can be partially prevented in the initial negotiation phase by specifying a split for the set of MIDI Commands, according to which a subset of commands (e.g., Channel Volume (Controller 7)) must be sent in every RJ, as if the policy were *anchor*-based.

4 The Proposed PLC Policy for MIDI Streaming

In the following, we describe our proposed PLC policy, which enhances the *anchor* policy defined in Ref. [8] while avoiding the risk of introducing unrecoverable artifacts of the *open-loop* one (see again Ref. [8]), despite operating in open-loop fashion as well as the *anchor*-based one. Indeed, differently from the *closed-loop* policy in Ref. [8], our proposed approach does not require any acknowledgment mechanism.

4.1 Enhanced Anchor Approach

Ref. [8] recommends the usage of the *closed-loop* policy, as it is the least bandwidth-consuming one. Indeed, the *anchor* policy implies a much larger overhead, due to the fact that two states interleaved by one or a few grouping periods usually differ by a limited number of events, whereas the number of differing events between a generic state S_i and state S_0 might be significant. Conversely, the usage of the *open-loop* policy is discouraged since it guarantees the correct recovery of the system state only under further assumptions (see section C.2.2.3 of Ref. [8]).

To reduce the introduced overhead, we propose to make the sending of the RJ non-compulsory. In our proposed *enhanced anchor* approach, we define a negotiable value k called *refresh rate*, which generalizes the *anchor* policy by imposing to send one RJ every k packets. The canonical *anchor* policy can now be seen as a special case of the *enhanced anchor* policy where $k = 1$. This new policy can be easily implemented with the same packet structure defined in [8]. With the *enhanced anchor* policy, the RJ section in Fig. 4 becomes optional, as it is present only once every k packets. Let t_0 be the time at which the session starts. Packets are generated and sent to the remote musicians at times $t_i = i\Delta t + t_0$ (with i positive integer) but only packets where $i = nk$ (with n positive integer) shall include the RJ Section. Note that in our proposed approach the RJ is computed as in the *anchor* policy described in Ref. [8], i.e., it encodes the differences between S_{i-1} and S_0.

4.2 State Maintenance and Update Procedure

To enable the receiver process to perform any correction, both the sender process and the receiver process have to keep track of the current system state. As depicted in Fig. 6, at each RJ reception the receiver must verify the correctness of its current state S_c by comparing the *parameters* (where the word “parameter” is meant to be an umbrella term that encompasses note velocities, control values, etc.) contained in the RJ to those that characterize S_c and by updating them with the value contained in the RJ in case of mismatch. This procedure fits both the *enhanced-anchor* policy and the *closed-loop* policy. The transition

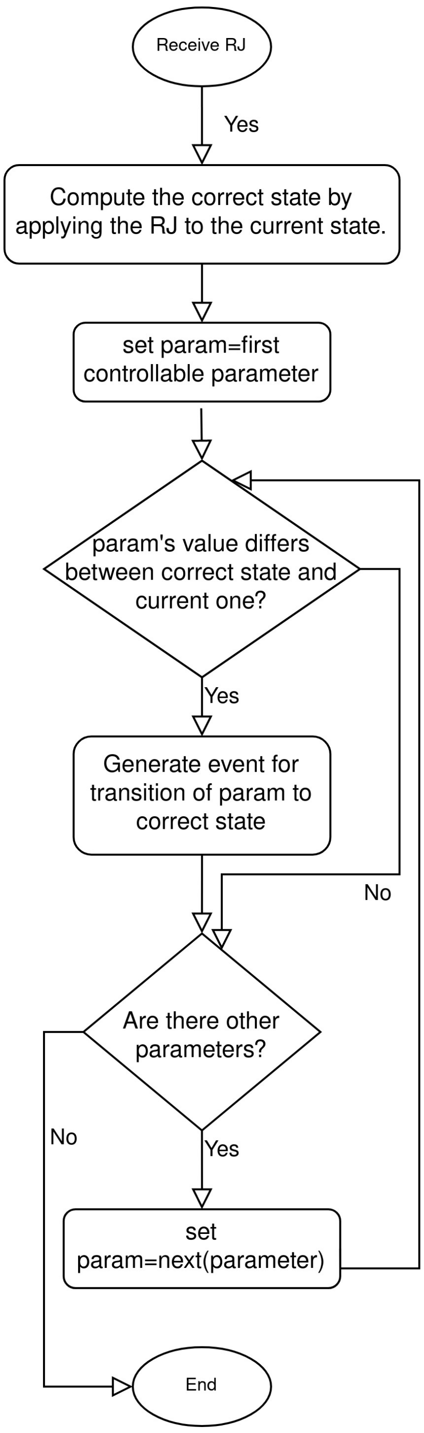

Receive RJ
Yes
Compute the correct state by applying the RJ to the current state.
set param=first controllable parameter
param's value differs between correct state and current one?
Yes
Generate event for transition of param to correct state
No
Are there other parameters?
No
Yes
set param=next(parameter)
End

Fig. 6: The algorithm to be implemented at the receiver side to perform packet loss concealment.

to the correct state is then performed by generating all the MIDI events that modify the system state from the “old” state S_c to the “new” correct one $S_{c'}$ (where$S_{c'} = S_{i-1}$), that will be considered as the current state from that moment on (i.e., $S_{c'}$ replaces S_c).

Conversely, the sender’s current state is stored in memory, and it is updated whenever a new MIDI message is generated by the musician’s instrument. Our proposed *enhanced anchor* policy introduces a simplification in the sender's implementation compared to the *closed-loop* policy, as in the latter case the sender is responsible for memorizing S_i and also the most recent acknowledged state S_{i-n} (i.e., the state at the beginning of the grouping period carried by the last acknowledged packet) for every known receiver. Indeed, different receivers may provide RRs at different times, thus a dedicated RJ must be produced for every receiver, and for each of them the most recent acknowledged state should be updated upon RR reception. This is a known drawback of the *closed-loop* policy, as reported in section C.2.2.2 of [8]. Differently, our proposed *enhanced anchor* policy only requires the sender to keep in memory exactly one state (i.e., S_i), regardless of the number of receivers.

Since several data structures to memorize a given state exist, Ref. [9] provides some guidelines for a possible implementation. From a high-level perspective, the state of the system should be a dictionary associating parameters (e.g. Notes) to their values (e.g. Note On with velocity = v, Note Off), although the suggested implementation is based on arrays for each parameter group (Notes, Controls, Programs...). This dictionary does not contain information about a parameter in case it has its default value. Straightforwardly, S_0 is represented by an empty dictionary. It is worth noticing that the sender's and receiver's software implementing the state-keeping cannot be agnostic about the specific semantics of MIDI messages. To motivate this statement, we provide the following example: let us assume we have a "dummy" software at the sender's side with no knowledge of the meaning of MIDI messages, and consider a basic assumption on Control change messages, i.e., that their default values equals 0. Assume that the MIDI Message with status byte = 121 is generated (its meaning is "all controllers off"). Our dummy software only understands that the status byte 121 is in the range of controls messages, and eventually modifies its state in the dictionary. This is not the right behavior, as it should reset to default all the others controllers that are in a non-default state.

The same holds at the receiver side. The receiver must, in fact, maintain the current state S_c of the local synthesizer. In case of lost packets, upon reception of an RJ, the receiver must be able to generate the messages necessary to adjust the synthesizer's state to match the received one, following the procedure reported in Fig. 6. The RJ representation proposed in Ref. [9] is meant to minimize the network bandwidth usage. It efficiently encodes the current state of all the parameters that changed their values. By doing so, if the starting state S_0 is the default one, the RJ brings all the information necessary to build the dictionary representing the current state.

4.3 Implementation Remarks

A simplified implementation guide for RJ-based PLC methods at the receiver side may be found in section 7 of RFC4696 [9], from which it emerges that the complexity of the PLC algorithm executed by the receiver grows linearly with the number of peers participating in the NMP sessions. Instead, for what concerns the implementation of our proposed enhanced anchor policy at the sender side, the sole task of the sender is to keep in memory a dictionary that reflects the MIDI instrument state. RFC4696 provides some hints regarding a possible representation in memory of such a dictionary, and how to build the associated RJ to be included in the packets that are required to contain it. As opposite to the closed-loop policy, which requires the keeping of a dedicated dictionary per each receiver, this data structure is the same for all the receivers. It follows that, at the transmitter side, the complexity of the proposed enhanced anchor policy does not depend on the number of peers.

Concerning network characterization, we identify three factors that may have an impact on the performance of our method: *i*) network latency, *ii*) jitter, *iii*) packet losses. In the following, we discuss such factors one by one.

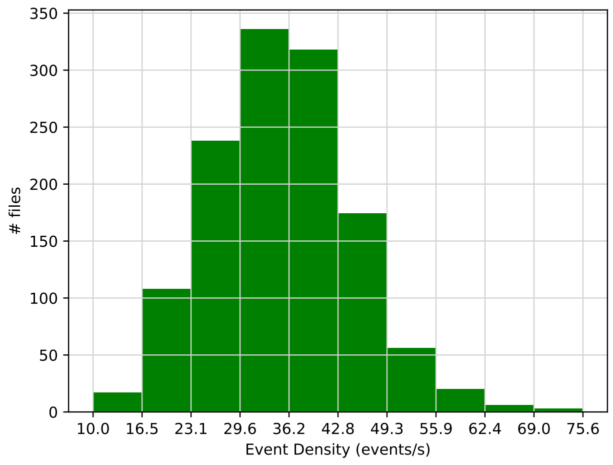
Network latency (i.e., the overall contribution to the mouth-to-ear delay due to transmission and propagation delays through communication link(s), as well as queuing and processing delays introduced by routers traversed by packets along their path from source to destination), does not have by itself a direct impact on the execution of the PLC algorithm: indeed, as reported in Fig. 6, the recovery journal is generated by the transmitter regardless of network conditions and compared by the receiver to the actual system state upon reception.

Differently, jitter may have an impact on the execution of the PLC algorithm, as the variation of the end-to-end delay experienced by consecutive packets may lead to an alteration of the packet sequencing (i.e., packets are received in a different order with respect to their generation sequence). If an out-of-order packet arrives too late to be reproduced by the receiver, it is practically equivalent to a lost packet. Therefore, by quantifying the packet loss probability, in our numerical assessment (see Section 5) we will capture the impact of both lost and late packets. To mitigate the impact of out-of-order packets, a buffer is typically maintained at the receiver to operate as "cushion": the larger the buffer, the higher the flexibility in re-ordering late packets, but the higher the increase of the mouth-to-ear delay.

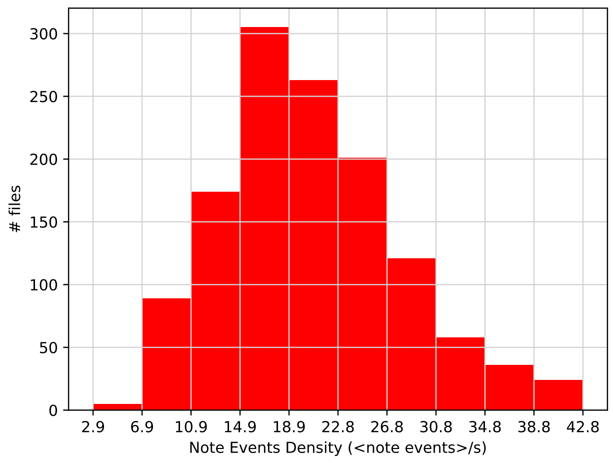
5 Performance Assessment

We provide a quantitative analysis of the performance of the proposed *enhanced anchor* PLC method in terms of similarity of the reproduced audio stream and network overhead, benchmarking it against the policies described in Ref. [8]. For the sake of reproducibility, the assessment is conduced on pre-recorded MIDI files instead of real-time generated MIDI streams. Our numerical assessment only takes into account the impact of packet losses and of delayed packets arrived too late to be reproduced. The impact of delayed but yet reproducible packets and of local clock drifts are not taken into account in this work since it is assumed that the MIDI streaming is implemented within a NMP system integrating an appropriate clock recovery mechanism (as proposed, e.g., in [31]) and adequate buffer size settings, capable of effectively managing the reordering of out-of-order packets. It is worth noting that the impact of clock drifting in MIDI streaming is milder than in audio streaming: in the latter case, clock misalignment may lead to buffer over/underruns that in turn generate audio glitches, whereas in the former case it may only cause alterations of the duration of active events, without introducing audio artifacts.

5.1 Dataset



(a) Empirical distribution of Events Density



(b) Empirical distribution of Note Events Density

Fig. 7: Empirical distribution of (note) event density values in the considered MIDI dataset

We considered the Maestro Dataset v3.0.0 [32], which includes about 200 hours of paired MIDI and Audio recordings from International Piano-e-Competition events within nine years. For the purpose of this study, only the MIDI dataset was considered, consisting in 1276 MIDI files, with a duration that varies between 45 seconds and 43 minutes, with an average of 9 minutes. Being a piano dataset, it is representative of a NMP session using a MIDI keyboard, as it contains Note messages, one Program Change at the beginning of the track and Control Changes messages affecting control 64 (Sustain pedal), control 67 (Soft pedal) and control 66 (Sostenuto pedal).

The dataset contains files with one track (namely one single track with MIDI messages representing MIDI events, hence not only meta-messages). Each file has a tempo signature of 120bpm. It can be divided in two subsets based on the timing information. One subset has a granularity of 384 ticks per beat, the other has a granularity of 480. For simplicity, network overhead measures have been conduced on the first subset, using as time units integer multiples of the tick duration.

To evaluate the effectiveness of the *enhanced anchor* policy, we identify some features related to the complexity of the performed musical pieces, according to which we split our dataset. In particular, we characterize the musical pieces by means of two metrics: the *event density* and the *note event density*. The proposed *event density* parameter (ED) aims at quantifying the rhythmic complexity of the piece being played and is an adaptation of the event density parameter defined in [3, 33] as the average number of event onsets per second, where the term onset refers to the initial transient of a musical note (or of any other sound). In this study, since we are considering MIDI data, an event onset refers to the beginning of a generic MIDI event. The *note event density* parameter (NED) has the same meaning as the *event density*, except for the fact that only Note events are considered (i.e., only Note On/Note Off messages). Fig. 7 reports the distribution of *event density* and *note event density* values among the MIDI files comprised in the dataset. As we can see, the distribution of the ED is well fitted by a normal distribution with mean 35.19 and variance 89.56 whereas the NED has mean 20.7 and variance 50.9.

5.2 Evaluation metrics for similarity assessment

The evaluation of the audio similarity achieved by the proposed *enhanced anchor* policy has been conducted by comparing the MIDI stream generated by the sender to the one played back by the receiver. For this assessment, the grouping time has been fixed to 3 ticks, which, in this dataset, correspond to $\approx 3ms$ or $\approx 4ms$, according to eq. (1), where τ is the duration of a tick in milliseconds, bpm is the time signature of the track in beats per minutes and tpb is the tick granularity in ticks per beats.

$$\tau = \frac{6 \cdot 10^4}{bpm \cdot tpb} \tag{1}$$

To perform such comparison, the content of the original MIDI file is streamed by the sender process and a new MIDI file is generated by the receiver process. Such MIDI file consists in the sequence of all the MIDI events that are played back, with their corresponding time offsets. In the absence of any transmission error, the file transmitted by the sender and the file reconstructed by the receiver must be identical. Conversely, if packet losses occur, the RJ section of a packet successfully received after one or multiple missing ones will be leveraged to conceal errors and to resume the correct system state. It follows that, in such a case, the file transmitted by the sender and the file reconstructed by the receiver will not be identical. For the sake

Table 1: Loss probability p and corresponding confidence interval of 95% of measured loss ratio (CI)

p	0.01	0.2	0.3
CI	[0.008,0.011]	[0.194,0.203]	[0.291,0.301]
p	0.5	0.8	0.9
CI	[0.492,0.507]	[0.800,0.812]	[0.900,0.907]

of our analysis, packet losses are modelled as statistically independent events, i.e., each packet is assumed to be unavailable for reproduction at the due time (due to either a loss or an excessive delay) with probability p. It is worth mentioning that, considering a single file, the longer the file duration, the more closely the actual ratio of lost packets (l) to generated ones (g) approximates probability p: Table 1 shows the confidence interval of $\frac{l}{g}$ compared to p. The discrepancies between the two files are then quantified[1].

The comparison algorithm consists in comparing the state of the two streams, when reproduced simultaneously, on per-tick basis. For such comparison, we define a very simple state similarity function $e(t)$. Note that, for the sake of our similarity analysis, we consider a back-to-back configuration that introduces negligible mouth-to-ear latency between the sender and the receiver processes to avoid the need for time realignment among the two streams.

$$test(S_a,S_b) : = \begin{cases} 1 \text{ if } S_a = S_b \\ 0 \text{ otherwise,} \end{cases} \quad (2)$$

$$e(t) : = test(S_r(t),S_s(t)) \quad (3)$$

where $S_r(t)$ (respectively $S_s(t)$) is the state of the receiver (respectively the sender) at time t.

The stream similarity value s is then defined as:

$$s = \frac{\int_0^T e(t)\,dt}{T} \quad (4)$$

Straightforwardly, formula (4) tends to 0 when the total time in which the two tracks produce the same musical output tends to 0, it tends to 1 when this time approaches the entire duration of the track.

We also define s_n, which follows the same definition as above except that we only consider note events (two states are equal if their set of active notes is the same, even if other events like control values may differ). This apparent simplification permits to exclude the impact of slow pedals variations. In other words, when a control command associated to a pedal (recall that this is the case for all the Control Changes in the reference dataset) is generated, the physical movement associated to the pedal pressure from up to down or vice versa spans, most of the time, multiple MIDI ticks (see eq.1), resulting in many intermediate Control Change events that do not significantly alter the reproduced audio stream (for example, giving an on-off interpretation to control change commands, by mapping the range of 128 possible values to a boolean variable which is set to 0 if the Control Change value is 0, to 1 otherwise, would imply no variation).

Note that the similarity evaluation analysis will focus only on the *enhanced anchor* and *anchor* policies. Indeed, as already mentioned, the *anchor* policy is obtained by setting $k = 1$ in the *enhanced anchor* policy, whereas the *closed-loop* policy behaves identically to the *anchor*-based one (in both cases, the receiver is capable of recovering the correct state of the system as soon as the RJ is received). The performance of the *open-loop* policy depends on the choice of m and on the burstiness of packet losses, but is surely upper-bounded by the performance of the *closed-loop* policy, so it is excluded from our analysis.

5.3 Evaluation metrics for traffic overhead

To quantify the impact of the considered PLC policies in terms of traffic overhead, we computed the amount of bytes transferred at transport layer during the simulated performance, considering the overhead bytes introduced by the RTP-MIDI protocol and the 8-byte-long UDP header at each packet. For this evaluation, we consider a unidirectional MIDI stream from a sender to a remote receiver process. The parameters influencing the performance of the *closed-loop* policy are the Round Trip Time (RTT - i.e., the time elapsed from the start of the transmission of an RTP-MIDI packet to the reception of the corresponding RR), the grouping period Δt and the packet loss probability p. The RTT affects the *closed-loop* policy because it can delay the pruning of the RJ structure [9], especially when it is higher than the grouping period (i.e., when RTT$>\Delta t$). Moreover, every time a packet or its corresponding RR gets lost, the size of the RJ may increase. Conversely, the performance of the *enhanced anchor* policy is influenced by the grouping period Δt and the refresh rate k, whereas the packet loss probability is not influential because the content of the RJ section does not depend on the acknowledgment mechanism.

We focus on the *closed-loop* policy and on our proposed *enhanced anchor* policy, since the behavior of the *anchor* policy in terms of traffic overhead corresponds to that of the *enhanced anchor* policy with $k = 1$, whereas the behavior of the *open-loop* policy is analogous to that of the *closed-loop* one, provided that parameter m in the *open-loop* policy is tuned so that $m \cdot \Delta t$ =RTT.

We compute the overhead by comparing S_{RJ} to S_{NRJ}, where S_{RJ} is the amount of bytes transmitted by the sender when the PLC policy under investigation is integrated in the RTP-MIDI protocol and S_{NRJ} is the amount of bytes transmitted by the sender when no PLC policy is adopted. It follows that S_{NRJ} depends only on the grouping period and on the amount of events in the track/performance. Depending on whether we include or ignore traffic generated by RRs sent by the receiver to the sender, the traffic over-

[1]Note that such evaluation method does not take into account the perceptual impact of transmission errors (i.e., any discrepancy between the two files is weighted equally). Future work will be devoted to the identification of more advanced performance evaluation metrics and subjective ratings.

p	0.2				0.5				0.8			
k	1	2	3	1000	1	2	3	1000	1	2	3	1000
$ED \leq 27.25$	98.14	97.08	96.10	21.54	93.14	89.03	85.76	5.68	78.87	68.87	61.90	1.43
$27.25 < ED \leq 32.49$	97.61	96.28	95.02	19.04	91.27	86.22	82.25	4.51	74.05	63.01	55.81	1.09
$32.49 < ED \leq 37.21$	97.23	95.69	94.25	17.37	89.95	84.27	79.91	4.04	70.93	59.31	51.97	0.95
$37.21 < ED \leq 42.84$	96.91	95.19	93.60	15.96	88.88	82.67	77.94	3.57	68.33	56.18	48.64	0.79
$ED > 42.84$	96.27	94.21	92.32	13.28	86.81	79.62	74.25	2.95	63.59	50.60	42.86	0.68

Table 2: Similarity (%) obtained with the enhanced anchor policy for different packet loss probability values, depending on the event density range, assuming a grouping period duration of 3 ticks.

p	0.2				0.5				0.8			
k	1	2	3	1000	1	2	3	1000	1	2	3	1000
$NED \leq 14.71$	98.98	98.39	97.82	25.84	96.19	93.68	91.48	7.71	87.31	79.46	73.40	2.42
$14.71 < NED \leq 18.17$	98.61	97.82	97.05	22.98	94.86	91.56	88.69	6.57	83.43	74.12	67.31	2.20
$18.17 < NED \leq 21.82$	98.36	97.44	96.53	19.87	94.01	90.23	86.97	5.37	81.07	70.93	63.76	1.70
$21.82 < NED \leq 26.32$	98.08	96.99	95.94	17.85	93.02	88.64	84.93	4.70	78.23	67.14	59.65	1.63
$NED > 26.32$	97.46	96.05	94.68	14.84	91.60	85.56	81.12	4.35	73.22	60.99	53.15	1.67

Table 3: Note similarity (%) obtained with the enhanced anchor policy for different packet loss probability values, depending on the note events density range, assuming a grouping period duration of 3 ticks.

head is defined as:

$$OH = \frac{S_{RJ}}{S_{NRJ}} \tag{5}$$

or as:

$$OH_{RR} = \frac{S_{RJ} + S_{RR}}{S_{NRJ}} \tag{6}$$

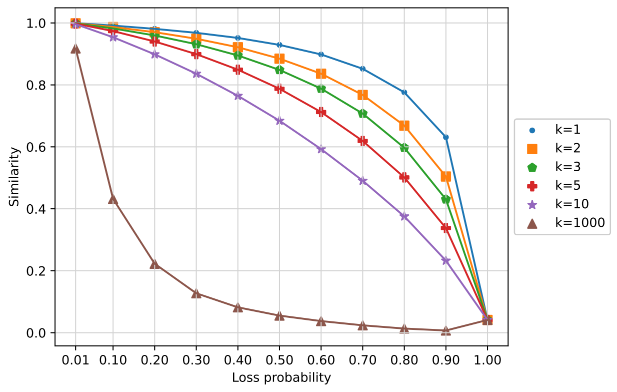
when backward traffic S_{RR} generated by the acknowledgment mechanism is taken into account ($S_{RR} = 0$ in the case of *enhanced anchor* policy).

5.4 Similarity Evaluation

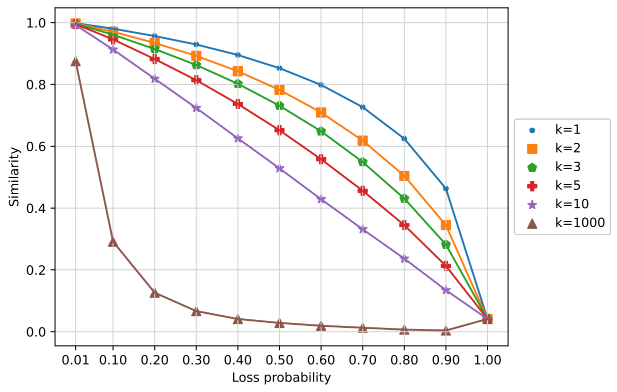
Fig. 8 reports the similarity results obtained for different values of the refresh rate k, depending on the packet loss probability p, and show that similarity is heavily influenced by the refresh rate value. In particular, with a refresh rate $k = 1$ (which means that every packet carries an RJ, thus making the behavior of the policy identical to that of the *anchor* policy), the protocol achieves about 90% similarity when $p = 0.5$. This means that, even in case of a communication where half of the transmitted data is lost, it is possible to reach a very high fidelity in the final audio playback. Moreover, even with $p = 0.8$, the proposed PLC method reaches on average 70% similarity.

When increasing k, the similarity inevitably worsens. The benchmark case of a very high refresh period is also reported: with $k = 1000$ the similarity is about 4% when $p = 0.5$. Moreover, a 0.2 packet loss probability is sufficient to make the similarity drop to less than 20%. Results with $p = 0$ and $p = 1$ require particular considerations: in both cases the achieved similarity does not depend on the refresh rate. This is because, for $p = 0$, there are no missing packets, while, for $p = 1$, all the packets are lost (no messages are received).

Note that, in the latter case, the similarity is higher than in the case with $p = 0.9$. This behavior is due to the fact that the absence of active events (i.e., silence) at both sender and receiver sides is considered as an identical state. Therefore, in the case of a reproduction characterized by total absence of events, the similarity represents the fraction of silence time among all the musical pieces. In cases with few packets received (e.g., when $p = 0.9$), a little amount of MIDI events is reproduced and, since the ex-



(a) With a grouping period of 2 ticks



(b) With a grouping period of 5 ticks

Fig. 8: Similarity (%) obtained with the enhanced anchor policy, depending on the refresh rate and on the packet loss probability (confidence intervals are below 1.5% and thus not reported for the sake of readability)

act correspondence of silence periods is not preserved any longer, the average similarity is lower.

To evaluate the impact of different event density and note event density values on the performance of the PLC method, the dataset has been subdivided into 5 subsets according to their ED (resp. NED), by splitting the whole ED (resp. NED) range in 5 equally sized sub-ranges. The global similarity trends reported in Tabs. 2 and 3 for $p = 0.2, 0.5, 0.8$ and $k = 1, 2, 3, 1000$ are similar to those reported in Fig. 8, showing decreasing similarity as k increases. However, for a given packet loss probability and refresh rate, higher values of ED or NED lead to lower similarity. The motivation lies in the fact that, with a higher average number of events per second, the number of packets not containing any MIDI event decreases. Therefore, it is more probable that a lost packet carries one or more MIDI events.

5.5 Overhead Evaluation

Fig. 9 shows the traffic overhead introduced by the usage of the RJ Section, when ignoring RR-related traffic (OH). Note that, in Fig. 9b, the number of boxplots progressively decreases when $\Delta t \geq 10$. This is due to the fact that a scenario with $RTT < \Delta t$ is equivalent to the ideal scenario where $RTT = 0$ (i.e., RR acknowledging packet i is received before packet $i + 1$ is ready for transmission).

In general, the *enhanced anchor* policy performs better when compared to the *closed-loop* one in terms of overhead, for $k \geq 3$. The lower bound of the overhead of the *closed-loop* policy is due to the fact that the Ref. [8] explicitly imposes that every packet must contain an RJ Section (even if empty). Note that the heaviest scenario occurs when $\Delta t = 1$ tick and $k = 1$ (*anchor* policy): in such situation, the highest average observed bitrate for a single stream was $38kB/s$, which is quite acceptable on modern networks, even compared to the worst simulated session with the *closed-loop* policy, which produced $22kB/s$ (obtained with loss probability $p = 0.8$, RTT= $200ticks$, grouping time= $1tick$).

However, as already mentioned, if the grouping period Δt is in the order of a few milliseconds, a significant amount of packets will have an empty Command section, since consecutive musical events are normally distanced by at least some tens of milliseconds. Therefore, with the aim of providing a more realistic estimation of the achievable bitrates, we also consider a more efficient sending policy by proposing an implementation that avoids unnecessary packet transmissions. Consequently, we repeat the computation of S_{RJ} and S_{NRJ} assuming that packets are transmitted only if non-empty. We define a non-empty packet as a packet which brings information useful for the playout at the receiver side. More specifically:

- for S_{NRJ}, non-empty packets are the ones that contain a non-empty MIDI command section;
- for S_{RJ} in the *closed-loop* policy, non-empty packets are those with either a non-empty RJ, or a non-empty Command section;
- for S_{RJ} in the *enhanced anchor* policy, non-empty packet packets are those either with a non-empty Command section or including an RJ (i.e., if, being i the sequential index of the packet, it holds that $i \, mod \, k = 0$).

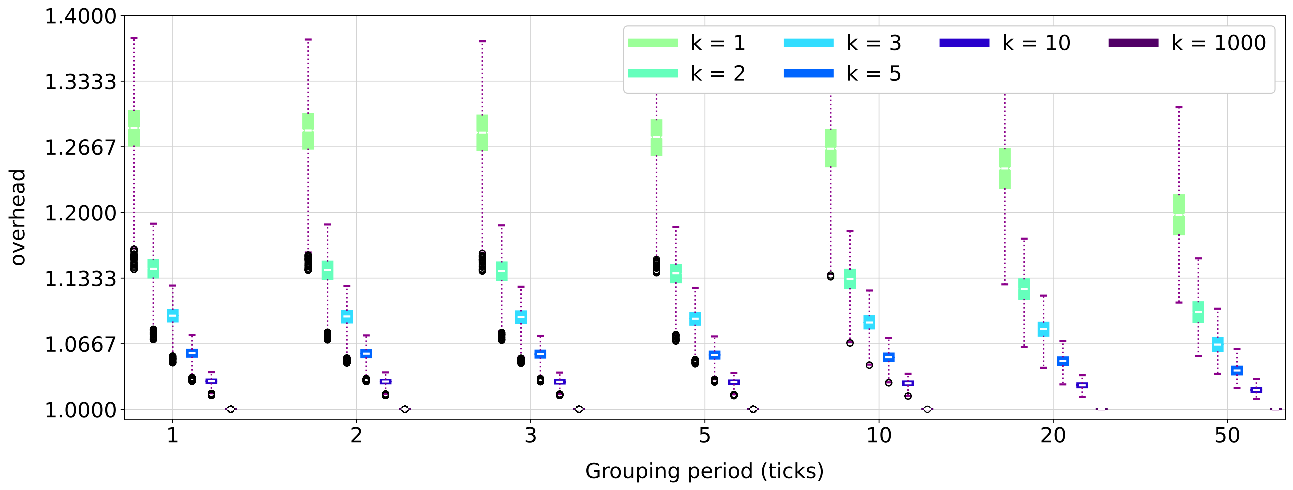
We underline that the definition of non-empty packet with respect to the RJ section differs between the two policies. The reason behind this is that, in the *closed-loop* policy case, the RRs act as an agreement between sender and receiver on the current state of the receiver. In such situation, an empty RJ is equivalent to an absent RJ as both mean that no events occurred at the sender side since the last acknowledged packet. It follows that, in the *closed-loop* policy, the variation of a parameter from a value different from the default one, to the default value, is a difference that must be encoded according to the rules of the protocol. Conversely, in the *enhanced anchor* policy, the sender is agnostic to the current state of the receiver. In this case, an empty RJ asserts that there are no differences between the sender state and the default state S_0, which is equivalent to saying that the sender is in the default state. When using the *enhanced anchor* policy, if a peer receives an empty RJ, it has to perform all the necessary actions (if any) to move to the default state. Thus, according to these observations, an empty RJ brings no information only in the *closed-loop* policy.

Moreover, in this new scenario we take into account the overhead generated by RRs (OH_{RR}) [2]. Results are reported in Fig. 10, which shows that with the aforementioned implementation the traffic overhead due to adoption of PLC policies substantially increases. In this case, the *enhanced-anchor* policy leads to higher overheads than the *closed-loop* one. Comparable or lower overheads are obtained by the *enhanced-anchor* policy for $k \geq 5$, when RTT$\geq 50ms$.

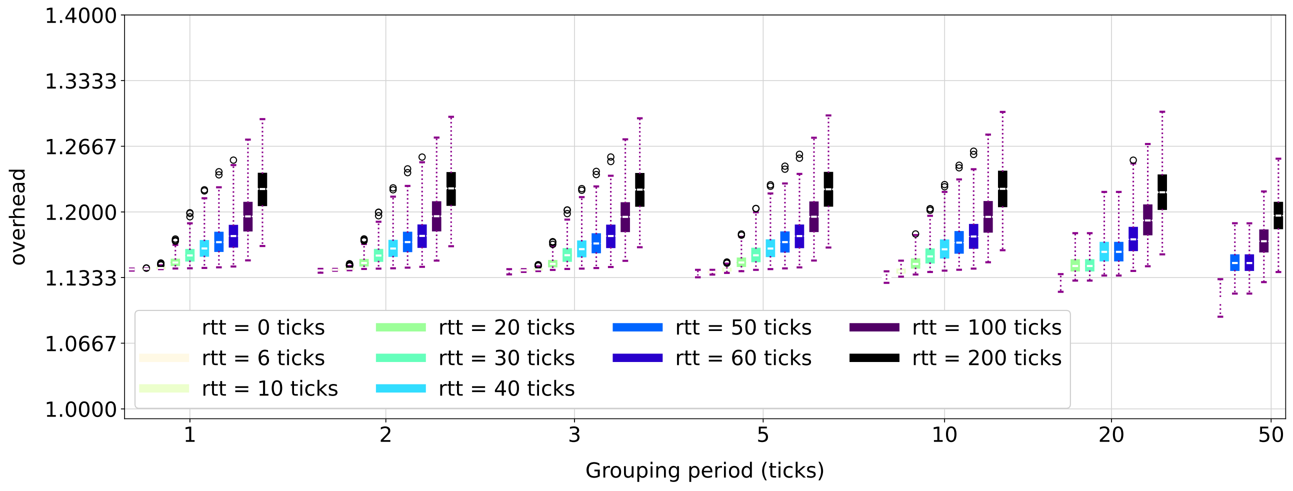
For a modern wired network connection, we can consider the following parameters for a plausible scenario:

- $\Delta t = 3ticks$ (≈ 4 ms assuming a tempo of 120 bpm and 480 ticks per beat), which is comparable to the time duration of a raw-audio segment (assuming stereo PCM audio with 16 bits encoding) conveyed by a packet of ≈ 700 bytes (i.e., approximately a half of Ethernet's Maximum Transmission Unit).
- $k = 3$, i.e., an RJ is sent approximately every 12 ms in the *enhanced anchor* policy. Scientific literature reports that the minimum time separation between two consecutive acoustic events ensuring that they are perceived as distinct is in the order of a few ms [34], and that a separation time of $15 - 20$ ms is required to make the listener able to report which of the two sounds preceded the other. Therefore, we speculate that setting $k = 3$ guaran-

[2]The acknowledgment procedure at receiver side is implementation-specific, however if RRs are sent at a too low rate, the size of the RJ at sender side increases. In our scenario, we suppose that a RR is sent upon reception of every packet. Under different assumptions S_{RR} may decrease, at the expense of a potential increase of S_{RJ}.



(a) Enhanced anchor policy ($k = 1$ represents the anchor policy).



(b) closed-loop policy, assuming a loss probability = 0.

Fig. 9: Transport level overhead with respect to a transmission with no RJ. Time is measured in MIDI clock ticks, $1\,tick \approx 1.3ms$.

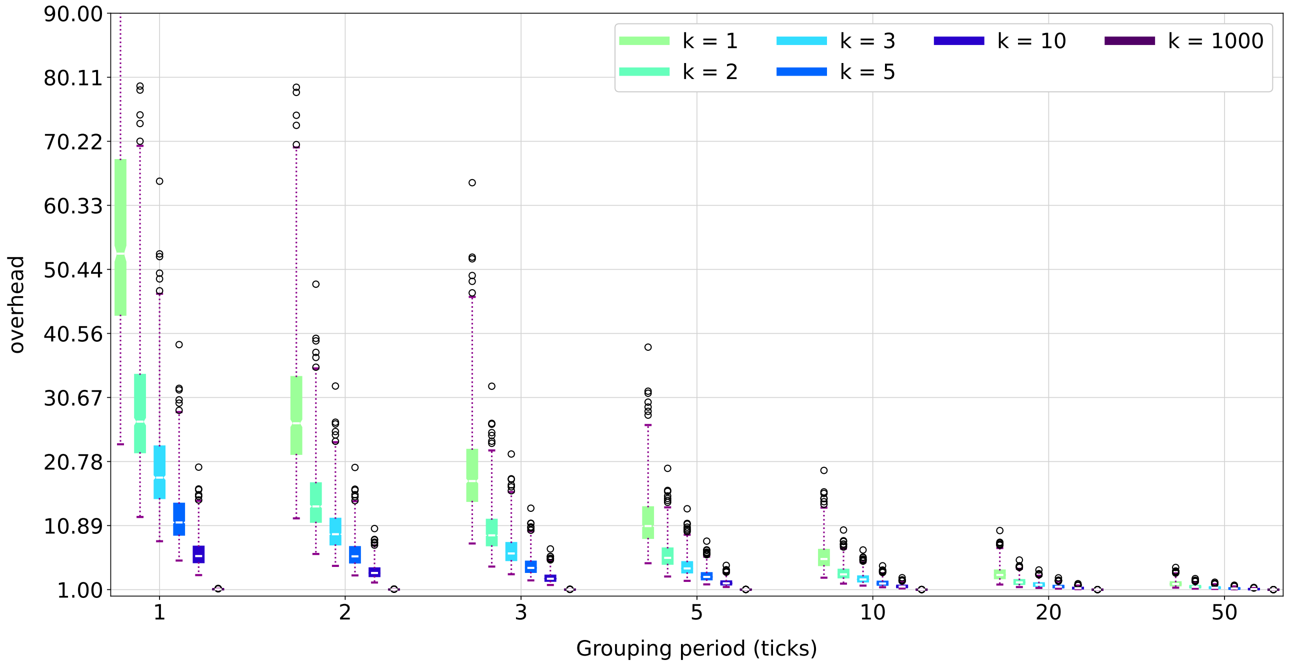
tees state recovery while making the perceptual impact of an incorrect state almost unnoticeable.

- RTT=30 ms, assuming a physical distance between remote players in the range of $500-1000$ km [35]: since the propagation time of a lightwave in an optical fiber is roughly 5 ms per 1000 km, and considering that queuing and processing delays at intermediate network nodes must be taken into account, as well as MIDI data acquisition and buffering delays at the two ends, we believe that such figure is representative of a wide range of practical NMP scenarios, with users placed at most some hundreds of km apart. It is worth remarking that the computational timings of our proposed PLC method are in the order of microseconds, thus they represent a negligible component of the mouth-to-ear latency.
- $p = 0.01$, i.e., we assume 1% of lost packets[3].

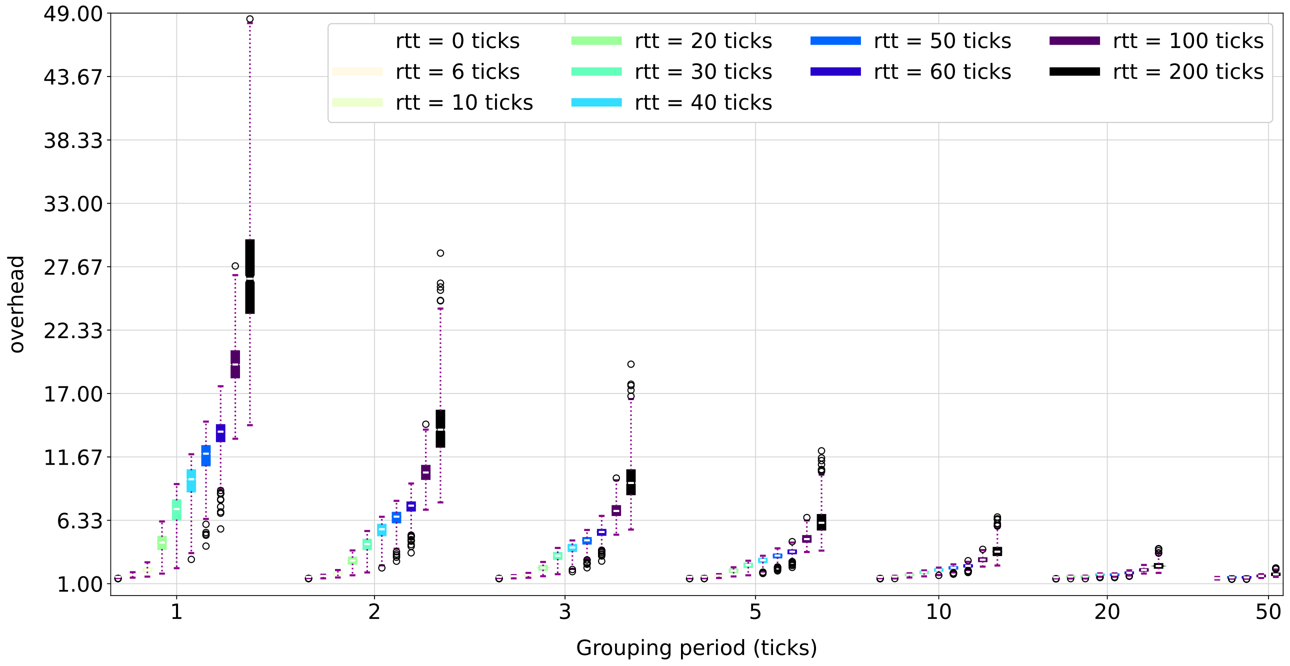
Under these assumptions, we measured an average bitrate of 2.62 kB/s (Confidence Interval: $(2.27, 2.89)$) for the *enhanced anchor* policy and 1.37 kB/s (Confidence Interval: $(0.48, 2.58)$) for the *closed-loop* policy[4]. Based on all the above reported results, it emerges that the proposed *enhanced anchor* policy achieves a good trade-off between traffic overhead and reduced operational complexity at the sender side. Therefore, its adoption is particularly suitable for large sessions where many musicians are involved, as its lower complexity and memory occupation at the transmitter side w.r.t. the *closed-loop* policy makes it more scalable. Its usage is also recommended for deployments with

[3]Note that a stable network, that is, a network exhibiting negligible packet loss probability, is considered a requirement for NMP. Anyway, higher loss probabilities have a small impact on the overhead of the *closed-loop* policy, especially with high RTTs. To prove that, we measured that $p = 0.2$ causes a $+1.45\%$ average bandwidth usage w.r.t. $p = 0.01$.

[4]It is worth reminding that the amount of participants impacts the overall data rates received/transmitted by a peer, which grows linearly with the number of peers, regardless of the PLC approach being implemented.



(a) Enhanced anchor policy ($k = 1$ represents the anchor policy)



(b) closed-loop policy, assuming $p = 0.01$

Fig. 10: Transport level overhead with respect to a transmission with no RJ, with the proposed efficient implementation. Time is measured in MIDI clock ticks, $1\,tick \approx 1.3ms$.

computationally-limited hardware (e.g., embedded processors). Moreover, the proposed policy offers higher flexibility, since the refresh rate k can be set and dynamically updated based on current network conditions to achieve the desired trade-off between transmission bitrates and fidelity of the reproduced MIDI stream.

6 Conclusion

This paper focuses on PLC methods for MIDI musical signals streamed via RTP-MIDI, using an unreliable transport protocol such as UDP, and proposes an enhancement of the anchor-based PLC method proposed in RFC4695. The proposed *enhanced anchor* policy relies on a representation of the MIDI system state and implements loss recovery without relying on feedback mechanisms (such as the Recovery Journal adopted in RTP). Thanks to the adoption of such a state representation, the proposed policy reduces the operational complexity of the data transmission algorithm implemented at the sender side, as it requires the keeping of a single state regardless of the number of receivers, whereas feedback-based policies require the construction of a dedicated RJ per receiver. This is particularly advantageous in Networked Music Performance applications to ensure scalability to scenarios where multiple participants are involved. Moreover, the proposed approach enables to strike a balance between incurred data transfer overhead and concealment capabilities, by adjusting the frequency of transmission of the system state, thus granting increased configuration flexibility w.r.t. feedback-based policies.

Results show that the proposed policy achieves transmission bitrates in the order of few tens of kB/s in realistic scenarios, while preserving the quality of the audio playback even in lossy network conditions. Future research will be devoted to the design of an algorithm to dynamically adjust the value of the grouping period, depending on actual network traffic and latency conditions, to ensure the best trade-off between bandwidth utilization and loss recovery capabilities.

7 Acknowledgments

Leonardo Severi's PhD Programme is funded by the European Union in the framework of the Resiliency and Recovery Plan (RRP), within the NextGenerationEU initiative.

The authors thank Prof. Alessandro Giusti and Prof. Christopher Chafe for the useful discussions.

8 REFERENCES

[1] C. Rottondi, C. Chafe, C. Allocchio, A. Sarti, "An Overview on Networked Music Performance Technologies," *IEEE Access*, vol. 4, pp. 8823–8843 (2016), URL https://doi.org/10.1109/ACCESS.2016.2628440.

[2] MIDI Manufacturers Association (MMA), "Complete MIDI 1.0 Detailed Specification Document," (1996), URL https://www.midi.org/specifications-old/item/the-midi-1-0-specification.

[3] C. Rottondi, M. Buccoli, M. Zanoni, D. Garao, G. Verticale, A. Sarti, "Feature-Based Analysis of the Effects of Packet Delay on Networked Musical Interactions," *Journal of the Audio Engineering Society*, vol. 63, no. 11, pp. 864–875 (2015), URL https://doi.org/10.1109/ACCESS.2016.2628440.

[4] A. Carôt, C. Werner, T. Fischinger, "Towards a comprehensive cognitive analysis of delay-influenced rhythmical interaction," presented at the *International Computer Music Conference, ICMC 2009, Montreal, Quebec, Canada* (2009 August).

[5] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control," RFC 2581, RFC Editor (1999 April).

[6] M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window," RFC 3390, RFC Editor (2002 October).

[7] J. Postel, "User Datagram Protocol," STD 6, RFC Editor (1980 August), URL http://www.rfc-editor.org/rfc/rfc768.txt, http://www.rfc-editor.org/rfc/rfc768.txt.

[8] J. Lazzaro, J. Wawrzynek, "RTP Payload Format for MIDI," RFC 4695, RFC Editor (2006 November), URL https://www.rfc-editor.org/info/rfc4695.

[9] J. Lazzaro, J. Wawrzynek, "An Implementation Guide for RTP MIDI," RFC 4696, RFC Editor (2006 November), URL https://www.rfc-editor.org/info/rfc4696.

[10] C. Perkins, O. Hodson, V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE network*, vol. 12, no. 5, pp. 40–48 (1998 September), URL https://doi.org/10.1109/65.730750.

[11] X. Chen, C. Wang, D. Xuan, Z. Li, Y. Min, W. Zhao, "Survey on QoS management of VoIP," presented at the *2003 International Conference on Computer Networks and Mobile Computing, 2003. ICCNMC 2003.*, pp. 69–77 (2003 October), URL https://doi.org/10.1109/iccnmc.2003.1243029.

[12] E. Thirunavukkarasu, E. Karthikeyan, "A survey on VoIP packet loss techniques," *International Journal of Communication Networks and Distributed Systems*, vol. 14, no. 1, pp. 106–116 (2015), URL https://doi.org/10.1504/ijcnds.2015.066029.

[13] M. M. Mohamed, M. A. Nessiem, B. W. Schuller, "On deep speech packet loss concealment: A mini-survey," *arXiv preprint arXiv:2005.07794* (2020).

[14] J.-M. Valin, K. Vos, T. B. Terriberry, "Definition of the Opus audio codec," *IETF* (2012 September), URL https://doi.org/10.17487/RFC6716.

[15] F. Pflug, T. Fingscheidt, "Robust Ultra-Low Latency Soft-Decision Decoding of Linear PCM Audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 11, pp. 2324–2336 (2013), URL https://doi.org/10.1109/TASL.2013.2273716.

[16] J. Østergaard, D. E. Quevedo, J. Jensen, "Real-Time Perceptual Moving-Horizon Multiple-Description Audio Coding," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4286–4299 (2011 June), URL https://doi.org/10.1109/TSP.2011.2159601.

[17] J. Leegaard, J. Østergaard, S. H. Jensen, R. Zamir, "Practical Design of Delta-Sigma Multiple Description Audio Coding," *Eurasip Journal on Audio, Speech, and Music Processing*, vol. 16 (2014 April), URL https://doi.org/10.1186/1687-4722-2014-16.

[18] J. Østergaard, "Low Delay Robust Audio Coding by Noise Shaping, Fractional Sampling, and Source Prediction," presented at the *2021 Data Compression Conference (DCC)*, pp. 273–282 (2021 May), URL https://doi.org/10.1109/DCC50243.2021.00035.

[19] M. Sacchetto, Y. Huang, A. Bianco, C. Rottondi, "Using Autoregressive Models for Real-Time Packet

Loss Concealment in Networked Music Performance Applications," presented at the *Proceedings of the 17th International Audio Mostly Conference*, pp. 203–210 (2022 October), URL https://doi.org/10.1145/3561212.3561226.

[20] P. Verma, A. I. Mezza, C. Chafe, C. Rottondi, "A Deep Learning Approach for Low-Latency Packet Loss Concealment of Audio Signals in Networked Music Performance Applications," presented at the *2020 27th Conference of Open Innovations Association (FRUCT)*, pp. 268–275 (2020 September), URL https://doi.org/10.23919/FRUCT49677.2020.9210988.

[21] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," STD 64, RFC Editor (2003 July), URL http://www.rfc-editor.org/rfc/rfc3550.txt.

[22] Sonosaurus LLC, "SonoBus," URL https://github.com/sonosaurus/sonobus.

[23] "HPSJAM," URL https://github.com/hselasky/hpsjam/.

[24] "Setting up a virtual MIDI network," URL https://help.ableton.com/hc/en-us/articles/209071169-Setting-up-a-virtual-MIDI-network.

[25] "RTP-MIDI or MIDI over Networks," URL https://www.midi.org/midi-articles/rtp-midi-or-midi-over-networks.

[26] J. Virolainen, P. Laine, "Methods and apparatus for transmitting MIDI data over a lossy communications channel," (2005 May 24), uS Patent 6,898,729.

[27] J. Virolainen, P. Laine, "Method and apparatus for enabling music error recovery over lossy channels," (2004 Aug. 12), uS Patent App. 10/359,809.

[28] J. R. Nelson, A. J. Heidorn, R. J. Cox, "Reliable real-time transmission of musical sound control data over wireless networks," (2017 Mar. 21), uS Patent 9,601,097.

[29] D. Fober, Y. Orlarey, S. Letz, "Real time musical events streaming over Internet," presented at the *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*, pp. 147–154 (2001), URL https://doi.org/10.1109/WDM.2001.990170.

[30] P. J. Fitzgibbons, S. Gordon-Salant, J. Barrett, "Age-related differences in discrimination of an interval separating onsets of successive tone bursts as a function of interval duration," *The Journal of the Acoustical Society of America*, vol. 122, no. 1, pp. 458–466 (2007 July), URL https://doi.org/10.1121/1.2739409.

[31] P. Ferguson, C. Chafe, S. Gapp, "Trans-Europe Express Audio: testing 1000 mile low-latency uncompressed audio between Edinburgh and Berlin using GPS-derived word clock, first with jacktrip then with Dante." presented at the *Audio Engineering Society Convention 148* (2020 May).

[32] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," *arXiv preprint arXiv:1810.12247* (2018).

[33] O. Lartillot, P. Toiviainen, "A Matlab toolbox for musical feature extraction from audio," presented at the *International conference on digital audio effects*, vol. 237, p. 244 (2007).

[34] I. J. Hirsh, "Auditory Perception of Temporal Order," *The Journal of the Acoustical Society of America*, vol. 31, no. 6, pp. 759–767 (2005 July), URL https://doi.org/10.1121/1.1907782.

[35] A. Carôt, C. Werner, "Fundamentals and principles of musical telepresence," *Journal of Science and Technology of the Arts*, vol. 1, no. 1, pp. 26–37 (2009 May), URL https://doi.org/10.7559/citarj.v1i1.6.

THE AUTHORS

Leonardo Severi

Antonio Cuccarese

Andrea Bianco

Cristina Rottondi

Leonardo Severi is Ph.D. student in Department of Electronics and Telecommunications of Politecnico di Torino (Italy). He obtained the B.Sc. in Politecnico di Torino in 2018 and the M.Sc. in the same university in 2021, both in Computer Engineering, he also obtained the Dipl. Ing.

in Grenoble INP - Ensimag in 2021. His current research activity is focused on Networked Music Performance.

●

Antonio Cuccarese graduated in Computer Engineering at the Politecnico di Torino in 2017 and got his M.Sc. degree in Computer Engineering in 2020 from the same university. During his Master's thesis work he focused on Real-Time Music Performance and, more specifically, on a low latency packet loss concealment method for MIDI signals. He is currently working as a Web Developer for TamTamy Reply. Music lover, he has played piano from its early years and he is studying Music Composition and Sound Design at the Scuola Internazionale di Comics.

●

Andrea Bianco is Full Professor and Head of the Department of Electronics and Telecommunications of Politecnico di Torino (Italy). He has co-authored over 250 papers published in international journals and presented in leading international conferences in the area of telecommunication networks. He was Area Editor for the IEEE JLT (Journal of Lightwave Technology) 2013-2018 and is Area Editor of the Elsevier Computer Communications journal. He was member of the HPSR steering committee in 2015. He was Technical Program Co-Chair for IEEE HPSR 2003 and 2008, DRCN (Design of Reliable Communication Networks) 2005, IEEE ICC 2010 (Optical Networks and Systems Symposium), IFIP Networking 2015 and IEEE GLOBECOM 2015 (Next Generation Networking Symposium) and received two best paper awards. His current research interests are in the fields of all-optical networks managment, switch architectures, SDN networks and NMP. Andrea Bianco is an IEEE Senior Member.

●

Cristina Rottondi is Associate Professor with the Department of Electronics and Telecommunications of Politecnico di Torino (Italy). Her research interests include optical networks planning and networked music performance. She received both Bachelor and Master Degrees "cum laude" in Telecommunications Engineering and a PhD in Information Engineering from Politecnico di Milano (Italy) in 2008, 2010 and 2014 respectively. From 2015 to 2018 she had a research appointment at the Dalle Molle Institute for Artificial Intelligence (IDSIA) in Lugano, Switzerland. She is co-author of more than 100 scientific publications in international journals and conferences. She served as Associate Editor for IEEE Access from 2016 to 2020 and is currently Associate Editor of the IEEE/OSA Journal of Optical Communications and Networking. She is co-recipient of the 2020 IEEE Charles Kao award, of the 2022 Journal of the Audio Engineering Society best paper award, of three conference best paper awards (FRUCT-IWIS 2020, DRCN 2017, GreenCom 2014), and of one excellent conference paper award (ICUFN2017).