PLC Testbench: a modular tool for the study and comparison of audio Packet Loss Concealment algorithms

Luca Vignati
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
luca.vignati@unitn.it

Stefano Dallona
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
stefano.dallona@studenti.unitn.it

Luca Turchet
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
luca.turchet@unitn.it

***Abstract*—Whenever time-critical applications supporting remote interactions are deployed on the Internet they may face the issue of packet loss. This is particularly true for remote interactions that are audio-based and require low latency and high reliability, such as those enabled by networked music performance systems, where packet losses are detrimental to the quality of experience of the users. Packet Loss Concealment (PLC) for audio signals is then mandatory to cope with such an issue. PLC is a technique that aims at masking the effect of packet loss on human auditory perception. Many PLC algorithms have been developed over the last three decades, initially focusing on speech and more recently addressing broadband audio. However, to date, developers of PLC methods face the issue of the lack of tools supporting the design, comparison, and testing of such methods. To bridge this gap, this paper introduces PLC Testbench, a software tool that enables researchers to study, compare, and evaluate various PLC algorithms for audio. The software is modular and comprises a Python backend and a web user interface. It is open-source and is conveniently distributed as a docker container.**

***Index Terms*—Packet loss concealment methods, networked music performances, Internet of Sounds.**

I. Introduction

The digitalization process of the last two decades had many impacts on society. It revolutionized many areas of our everyday lives, from social interactions through the rise of social media to entertainment through on-demand video streaming services. This ongoing process is fueled by the rise of new technologies and services and the continuous improvement of the worldwide Internet infrastructure. Currently, one of the main efforts of tech giants like Meta and Microsoft is towards the achievement of the so-called "social presence" during remote interactions among people [1], i.e., the sense of "being there" in a digital environment together with other connected users. To date, videoconferencing software like Zoom and Skype are the most adopted tools for remote interactions, however, the user experience they deliver is very different from that of an actual real-life meeting. The rise of Virtual Reality and Augmented Reality is set to drastically improve the experience of remote interactions with the introduction of visual depth perception, spatial audio, and eye/face tracking that aim at enabling a high degree of immersion in the shared virtual/augmented environment.

A particular case in this space is represented by musical interactions. Today musicians can benefit from dedicated tools for remote interactions, called Networked Music Performance (NMP) systems [2], [3], which tackle the specific needs of low-latency and high audio quality that musical interactions require. These include latency of a maximum of 20–30 ms, a low and constant jitter (i.e., the variation of the latency), and a minimal amount of packet losses that hamper the perceived audio quality. Notable examples of such systems, at the commercial or experimental level, are Elk Live [4], LoLa [5], jacktrip [6], fast-music [7], Jamulus, and JamKazam.

When using remote interaction tools, it is important to remember that they typically rely on the public Internet. While we might think of the Internet as reliable, the truth is that the protocol at its core, the Internet Protocol (IP), is designed with a "best effort" approach to service quality, meaning that it does not guarantee reliable communication. Reliable communication can be achieved thanks to higher-level protocols like TCP at the cost of introducing unbounded latency. Since that is not acceptable for real-time applications, UDP must be adopted instead of TCP, effectively doing nothing to improve the reliability of the communication. This can cause packet loss, which occurs when network packets are not available when they are needed, resulting in perceivable (and typically not tolerable) glitches and artifacts in the received signal. For remote interaction tools that aim to provide a sense of social presence, this represents a serious problem because it drastically impacts the user experience. While redundancy and retransmission mechanisms can help mitigate some packet loss, an effective Packet Loss Concealment (PLC) algorithm is necessary to prevent or reduce the effect of perceivable loss.

Packet Loss Concealment is a technique to mask the effects of packet loss, mainly developed for Voice Over IP (VoIP) applications [8]. In real-time data transmission scenarios, PLC algorithms are necessary to conceal the loss of one or more

consecutive packets. The purpose is to ensure that the negative effects on the quality of service of the application consuming the data stream are limited. For a PLC algorithm to be effective, it must meet two main requirements [9]:

1) the output of the PLC algorithm should be perceptually indistinguishable from the original audio stream, even though it does not need to be an exact match;
2) its execution time should be shorter than the interval between packet consumption.

There are several ways to perform PLC on audio signals, ranging from simpler waveform-based solutions like waveform repetition and WSOLA [10] to more advanced autoregressive models [9], [11]. Recently, Deep Learning-based PLC has emerged in the literature, with some significant examples in speech and music [12]. Regardless of the type of PLC used, they all rely on the audio data received immediately before the lost packet as input to the algorithm. The algorithm then outputs the number of samples contained in the lost packet.

However, to date, developers of PLC techniques face the issue of the lack of tools supporting the design, comparison, and testing of such techniques. To bridge this gap, this paper introduces *PLC Testbench*, a software tool that enables researchers to study, compare, and evaluate various PLC algorithms for audio. The software is freely available and open source[1].

The remainder of this paper is organized as follows: Section II provides some background notions on the core topic of this paper and presents the related literature, in Section III we present the testbench with all of its modules and in Section IV we present its user interface. Section V concludes the discussion with some closing remarks and discusses future works.

II. Related work

This section covers the main topics discussed in the paper and introduces the most relevant research related to them.

A. Packet Loss Simulation

There are many ways to simulate packet loss. The simplest consists in drawing a realization from a uniform distribution. This model is governed by a single parameter called the Packet Error Ratio (PER), which determines the probability of losing any packet. A more realistic model is the Gilbert-Elliot model [13], which consists of a two-state finite-state machine. The states represent the good and bad behavior of the transmission channel. This model is governed by four parameters: the two probabilities of transitioning from one state to the other, and the probabilities of losing packets in each state. A more detailed description, along with other models, is provided in [14].

B. PLC

Several contributions to the field of PLC have been made over the last three decades. In the beginning, the focus was on VoIP (Speech) [8], mainly due to the premature state of the Internet infrastructure that could not sustain real-time communication of broadband audio. Over the past decade, however, the research community has begun working on PLC for audio too.

1) Waveform substitution: The simplest form of PLC is **zeros substitution**, where silence is used in place of the missing packet. **Waveform repetition** is and consists of using the last correctly received packet in place of the missing one. It is as simple as zeros substitution but it can perform better or worse depending on the type of audio it is used on. A more complex example of waveform substitution is the **WSOLA algorithm**. It performs time-scaling on several of the frames preceding the lost one to extend their time duration in order to make them span across the gap of the lost frame [15]. The **low-computational-cost algorithm** presented in [10] is a more heuristic oriented approach, but is still working on waveforms of previous packets in the time domain. It leverages simple techniques like zero-crossing, derivatives and cross-fading to produce a very low-computational-cost algorithm that is able to outperform the zeros substitution technique.

2) Autoregressive models: A different approach to PLC is to model the signal as an autoregressive process. This works especially well on the human voice because it can be described with the source-filter model, which in turn is usually autoregressive. While not as good, its effectiveness on broadband musical audio is still better than waveform substitution techniques in most cases. Notable implementations of autoregressive models for PLC on real-time audio are reported in [9] and [11].

3) Deep Learning models: Deep Learning is the current frontier of research in virtually any computer science field. A few Deep-Learning-based PLC implementations have been proposed so far. Most of them target speech [16]–[18], but there are a few implementations targetting broadband audio, like [12]. However, while speech can be represented using the simple and compact source-filter model, broadband audio is more complex and challenging to model. Therefore, developing a deep-learning model that is both small enough to be computed in real-time and capable of generalizing on any audio is a difficult task. For this reason, studies that develop a deep-learning model for broadband audio usually restrict the scope of the algorithm to just the audio that it has been trained on (e.g., same instrument, same room, or even same performer).

C. Metrics

There are many ways to compare the similarity of two signals, the classic examples are the Mean Square Error (MSE) and the Mean Absolute Error (MAE). However, the space of solutions to the PLC problem is bigger than just the single solution matching the original signal. Since the target of PLC is just to "conceal" the packet loss from the listener's ears, the problem is of perceptual nature. As such, perceptual metrics should be used. The most widely used are the Perceptual Evaluation of Speech Quality (PESQ) [19] for speech and

[1] https://github.com/cimil/plc-testbench

the Perceptual Evaluation of Audio Quality (PEAQ) [20] for broadband audio. However, these metrics have been developed in the early 2000s to evaluate audio codec artifacts. So, despite being widely adopted as the only available perceptual metrics, there is no consensus on whether they are actually representative of human perception when it comes to the extremely short and glitchy noises produced by packet loss [21]. Nonetheless, we decided to include the PEAQ metric in the PLC Testbench as it is the de-facto perceptual metric used on broadband audio PLC.

III. PLC Testbench

The purpose of the proposed testbench is that of providing developers with a platform supporting the efficient comparison of different PLC algorithms across different metrics. The performance of these algorithms may differ based on the distribution of lost packets and the type of musical audio they are used on. Therefore, it is important to measure the performance of each algorithm by varying both these factors. Moreover, the metric utilized to measure the performance of PLC algorithms can also impact the comparison, so it is essential to use multiple metrics and determine the most pertinent ones.

A. Architecture

Our tool was designed to be highly adaptable. It can be fully customized, with each execution receiving a complete set of inputs that specify which audio files to consider, which packet loss distribution to generate, which PLC algorithm to use, and which metrics to compute on the outputs. These inputs also allow for configurable settings of the related components. Figure 1 illustrates the module types along with a representation of the function. A NoSQL database has been used to manage the persistence of the data and avoid unnecessary computations of already computed data.

1) Inputs: Before conducting a measurement, the testbench requires the following pieces of information. A list of audio files describes which files to consider and where to find them. The list of packet loss simulators informs the software on what probability models to use to compute realizations of packet-loss time series. Similarly, a list of PLC algorithms specifies which of them can be selected for use and finally, a list of output metrics contains the metrics that will be used to evaluate the reconstruction quality of the PLC algorithms. Attached to each element of the input lists there are settings that provide essential information about the configurable options of the related module. For example, other than specifying which probability model to use for packet loss, it is possible to change the packet error ratio or the seed of the random function. This allows to have multiple entries of the same type that differ in at least one of the configurable parameters.

Once this information is provided, the testbench will generate a tree of depth 4 for each audio file with the audio file as the root node and the metrics as the leaves. The packet loss simulators and PLC algorithms are the intermediate nodes of the tree (see Fig. 2).

Fig. 1. The four module types used to carry out all the computations.

2) Outputs: After completing all the combinations, the testbench generates various types of results. The final outcome includes performance evaluation metrics calculated for every combination. Nonetheless, intermediate results may also interest researchers who are studying and developing a PLC algorithm. These intermediate results include the realizations drawn from the chosen statistical distributions of lost packets and the audio files to which the PLC algorithm was applied. Moreover, the unmodified original audio files can also be considered an interesting output of this testbench as they serve as the ground truth for the PLC algorithms.

B. Modules

The tool is designed to be modular, with the execution pipeline separated from the units performing the computations. There are various types of modules, namely, *OriginalAudio*, *PacketLossSimulator*, *PLCAlgorithm*, and *OutputAnalyser*. Each module type has an abstract base class, and any implementation of these classes can be easily added by inheriting from the base class. The following provides a summary of the module types.

OriginalAudio

For this type, we built just one simple module that loads the audio content into memory and extracts metadata information (i.e., Sampling Frequency).

PacketLossSimulator

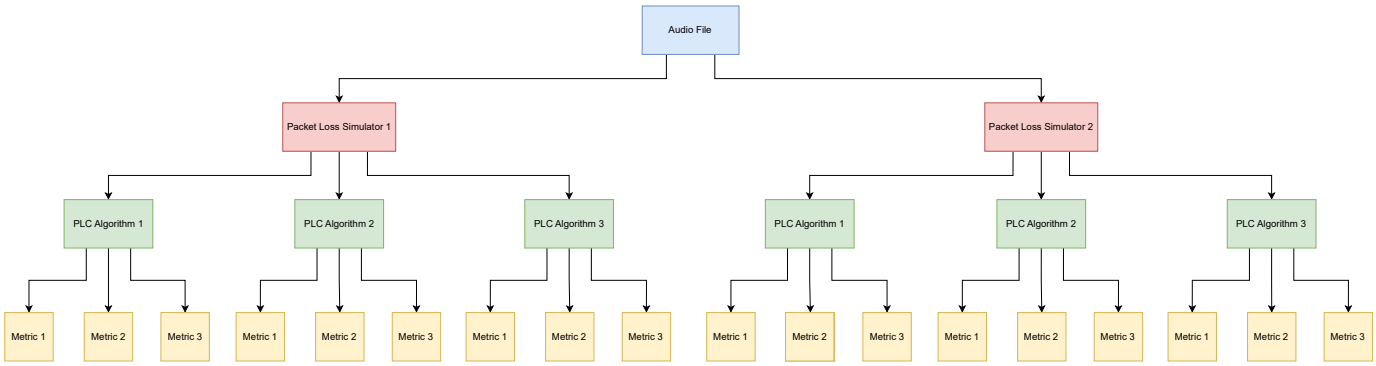

Fig. 2. The diagram shows an example of the computation tree that is generated on each run based on the provided input. In this case, there is one audio file, two packet loss simulators, three PLC algorithms, and three metrics.

This family of modules represents all the available ways of generating packet loss sequences.

- *BinomialPLS*: This is the simplest model for packet loss simulation as it is the implementation of a uniform distribution. Its only parameters are the Packet Error Ratio (PER) and the seed of the random function.
- *GilbertElliotPLS*: A more practical approach to packet loss simulation involves acknowledging that packet loss is not always present in a network. However, it is possible for a lost packet to trigger a series of lost packets, known as a packet loss burst. The Gilbert-Elliot model addresses this issue through a two-state machine. One state represents when the channel is functioning properly, while the other state signifies when a packet loss burst occurs. The model is governed by four configurable probabilities that determine the transitions between the two states. Each state has a probability of remaining in itself or transitioning to the other state and a probability of producing a valid packet. These probabilities dictate the behavior of the model.

PLCAlgorithm

This family of modules represents all the available PLC algorithms.

- *ZerosPLC*: This is the simplest form of PLC consisting in just setting every sample of the lost packet to zero. It represents the baseline of "doing nothing", below which no PLC algorithm can be considered useful.
- *LastPacketPLC*: This is another simple PLC algorithm consisting in filling the lost packet with the content of the last valid packet.
- *LowCostPLC*: This is an implementation of the low-delay with low computational overhead PLC algorithm described in [10]. It represents the baseline of a very fast and computationally efficient algorithm that can beat the ZerosPLC most of the time but not by a lot.
- *BurgPLC*: This is the implementation found in [22] of an autoregressive PLC with Burg estimation described in [9]. It represents the baseline for autoregressive PLC algorithms. Python bindings for C++ were used to integrate this algorithm into the testbench.
- *DeepLearningPLC*: This is an implementation of the Deep Learning-based PLC algorithm described in [12]. It represents the baseline for Deep Learning-based PLC algorithms.
- *ExternalPLC*: This is a generic implementation of Python bindings for C++. Thanks to this module, a PLC algorithm written in C++ (and adhering to the specified API) can be easily integrated into the testbench.

OutputAnalyser

This family of modules represents all the available output metrics.

- *MSECalculator*: This module computes the Mean Square Error (MSE) of the original signal against the reconstructed one. Its parameters are the window size and the hop size.
- *MAECalculator*: This module computes the Mean Absolute Error (MAE) of the original signal against the reconstructed one. Its parameters are the window size and the hop size.
- *SpectralEnergyCalculator*: This module calculates the difference between the magnitude of the DFT of the original and the reconstructed signals. Its parameters are the window size and the hop size.
- *PEAQCalculator*: This module runs the PEAQ implementation found in the GSTREAMER suite [23] as a command line utility. PEAQ is a perceptual measure of the quality of the reconstructed signal. It was originally conceived to quantify the perceived degradation of audio signals due to codec compression artifacts. Even though its effectiveness in assessing PLC quality has been debated, we include it as it currently represents the only perceptual metric available for PLC [9], [21].

C. Technology

This project is developed entirely in Python 3.10. The modules used are:

- *numpy* for fast and efficient mathematical calculations.
- *soundfile* for convenient management of audio files.
- *anytree* for the simple and flexible implementation of a tree structure on which the entire project is built upon.

- *pybind11* for easier interaction with the Python bindings for C++
- *pymongo* for the interaction with a MongoDB database.
- *tqdm* for displaying progress bars in Jupyter Notebook, terminals, and the GUI.
- *matplotlib* for plotting results as images.
- *tensorflow* for running Deep Learning-based PLC algorithms.

The NoSQL database of choice is MongoDB 6.0.8 due to its flexibility, scalability, and overall ease of use and deployment.

IV. UI

The purpose of the user interface for the Testbench PLC is manifold: i) to make it easier to use; ii) to reduce the time required to consult results; and iii) to increase the quantity and quality of information obtained from the analysis of results.

Before the implementation of this user interface, the only available kind of interaction took place directly in a Jupyter Notebook, where inputs are given directly by modifying the code. There was no aid in the interpretation of results, so the only way was to consult the raw output in the form of audio and image files. This type of interaction is suboptimal, especially when consulting the results, because the user is forced to consider one element at a time, precluding the possibility of increasing the quality of the analysis by considering multiple aspects at once.

The user interface can be conceptually divided into three parts:

1) Input Selection: Input selection is the set of all the interactions occurring before the program execution. As detailed in the previous discussion, the program inputs are a collection of modules with the associated set of settings. Fig. 3 shows the interface for choosing the PLC algorithms and setting their parameters. User interaction in this section is intended to simplify and speed up the input process and ends with the start of program execution.

2) Progress Monitoring: This part represents the program feedback aimed at informing the user about the status of the execution of the various modules of the testbench. The layout of this interface is strongly related to the choices described in the previous section and is mainly focused on the loading bars.

3) Result Analysis: Results consultation is the most relevant part of this project in terms of its impact on the user experience of this program. At the same time, it is also the most challenging part in terms of design and implementation. The results consultation groups all the interactions that occur once the program execution is finished and is characterized by browsing waveforms, series of lost packets, and listening to the audio files involved. Since for each input audio file, there are potentially many results to show, it would be ideal to be able to overlay as many results as possible on the same graph. This is made possible by the fact that all results for the same audio file share the same time axis.

A. Architecture

The user interface application is made of a Web application which is composed of two layers: one managing the user interface representation and the interaction with the user and the other providing the backend services to be consumed by the graphical components. The application, therefore, can be used both as a standalone application by deploying it on the user machine, or in a multi-user environment where the application is deployed on a remote server. The layers exchange information via a RESTFul API which, if needed, can be exploited also by third-party services. The API is secured by using OAuth 2 protocol.

In the presentation layer, modularity has been achieved through the creation of specialized but highly customizable components, which can easily be reused across different pages. In the backend layer, modularity is provided by the extensibility of the RESTFul API where new functions can be "plugged in" by simply adding new endpoints. This layer is also hiding the details of the interaction with the PLC Testbench from the presentation layer, providing an abstract and stable view of the library functions.

Persistence is based on a NoSQL database, where the data is stored in JSON format in order to provide higher flexibility to accommodate the future evolution of the schema. Both on-premise and cloud databases are supported. The GUI adapts automatically to the PLC Testbench library by using introspection to retrieve the list of the algorithms used for packet loss, PLC, and output analysis. This way any extension made to the underlying tool does not require any manual change to the UI code.

B. Technology

The product frontend is entirely written in the Javascript language, while the backend is entirely written in Python to allow for better integration with the PLC Testbench, as well as to avoid the unneeded complexity of the technological stack. The UI project can be installed as an ordinary Python application or through a docker image available on DockerHub[2].

C. Scalability

The application architecture has been designed to be as stateless as possible and thus supports very well horizontal scalability. You can split the workload among as many instances of the application as you need by providing proper load balancing in front of them. Scalability together with containerized distribution makes the application suitable for deployment in an orchestrated environment.

D. User Experience

In order to improve the user experience the GUI has been made responsive so that it can automatically adjust to different devices (PC, tablet, mobile). The interface has been designed to be user-friendly, intuitive, and as fast as possible, considering the large dataset inherently involved in audio processing

[2]https://hub.docker.com/r/cimil/plc-testbench-ui

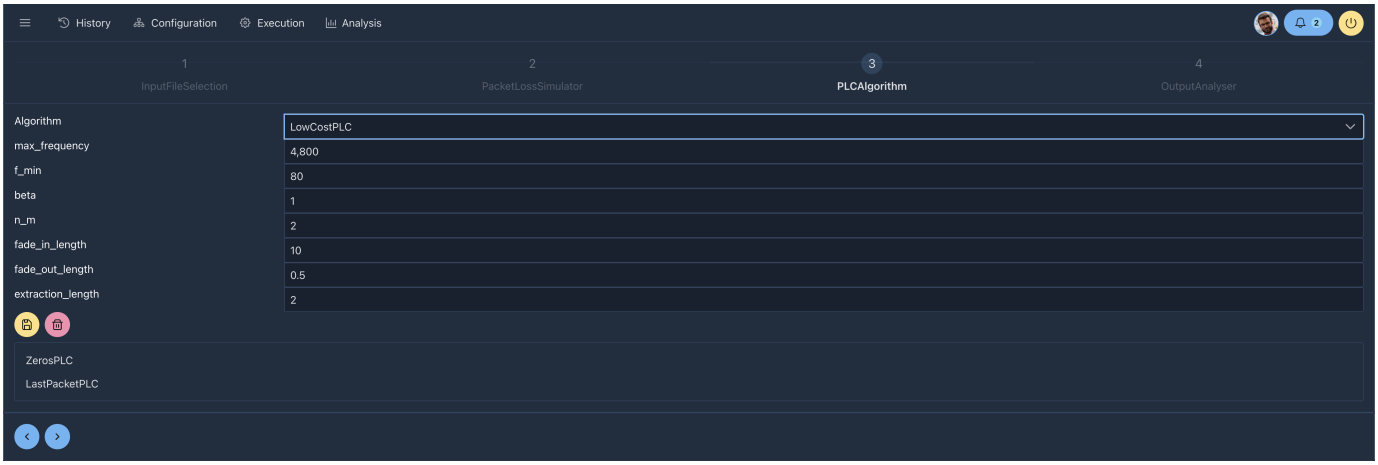

Fig. 3. The input section of the UI allows the user to choose which PLC algorithm to use. For each chosen algorithm, the related parameters can be changed.

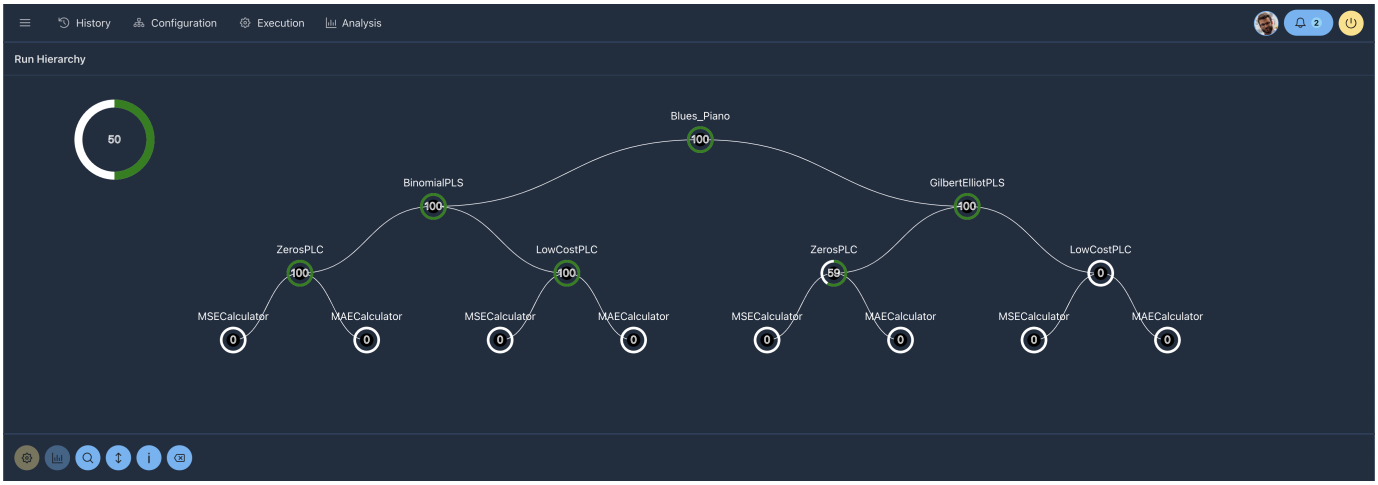

Fig. 4. The progress monitoring section of the UI provides feedback on the status of the execution to the user.

applications and the overhead related to charting functions. Whenever possible subsampling or caching techniques have been applied to minimize latency and network bandwidth waste.

V. Conclusion and Future Work

In this paper, we presented a modular and configurable software tool for the analysis and comparison of packet loss distributions and PLC algorithms. Such a tool is meant to be a companion of the researcher developing new PLC algorithms, so it features a web GUI for easy and convenient interaction. The project is fully open-source and modular making it easy to add new packet loss distributions, PLC algorithms, or metrics. It is also available as a Docker container for a zero-setup configuration and as an online version (no data persistence) for a quick look.

The output quality of a PLC algorithm is arguably the more difficult aspect of its evaluation but, as stated in the introduction, it is not the only one. Execution time is equally as important but has not yet been included in this testbench. Therefore, this aspect is the main point in the future roadmap of this project.

Acknowledgment

This work has been supported by the PRIN grant of the Italian Ministry for University and Research.

References

[1] C. S. Oh, J. N. Bailenson, and G. F. Welch, "A systematic review of social presence: Definition, antecedents, and implications," *Frontiers in Robotics and AI*, p. 114, 2018.

[2] C. Rottondi, C. Chafe, C. Allocchio, and A. Sarti, "An overview on networked music performance technologies," *IEEE Access*, vol. 4, pp. 8823–8843, 2016.

[3] L. Gabrielli and S. Squartini, *Wireless Networked Music Performance*. Springer, 2016.

[4] L. Turchet and C. Fischione, "Elk Audio OS: an open source operating system for the Internet of Musical Things," *ACM Transactions on the Internet of Things*, vol. 2, no. 2, pp. 1–18, 2021.

[5] C. Drioli, C. Allocchio, and N. Buso, "Networked performances and natural interaction via lola: Low latency high quality a/v streaming system," in *International Conference on Information Technologies for Performing Arts, Media Access, and Entertainment*. Springer, 2013, pp. 240–250.

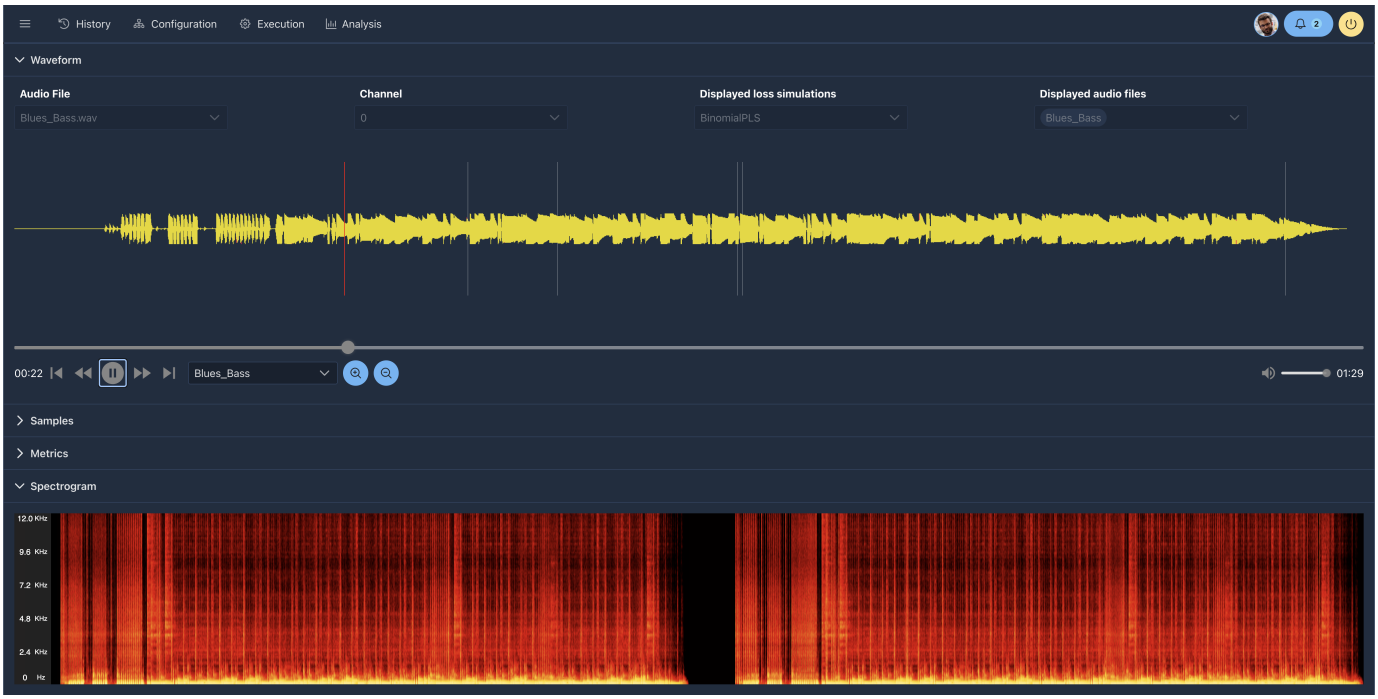

Fig. 5. In the results analysis section the user can interact with the results. The waveform is fully interactive and audio can be played back. Additionally, there is a spectrogram to provide spectral energy information to the user.

Fig. 6. The waveform can be zoomed in to the sample level. Additionally, there is a convenient view below the main waveform that is centered on the lost packet. Both views can display multiple waveforms at a time, allowing the comparison of different PLC algorithms.

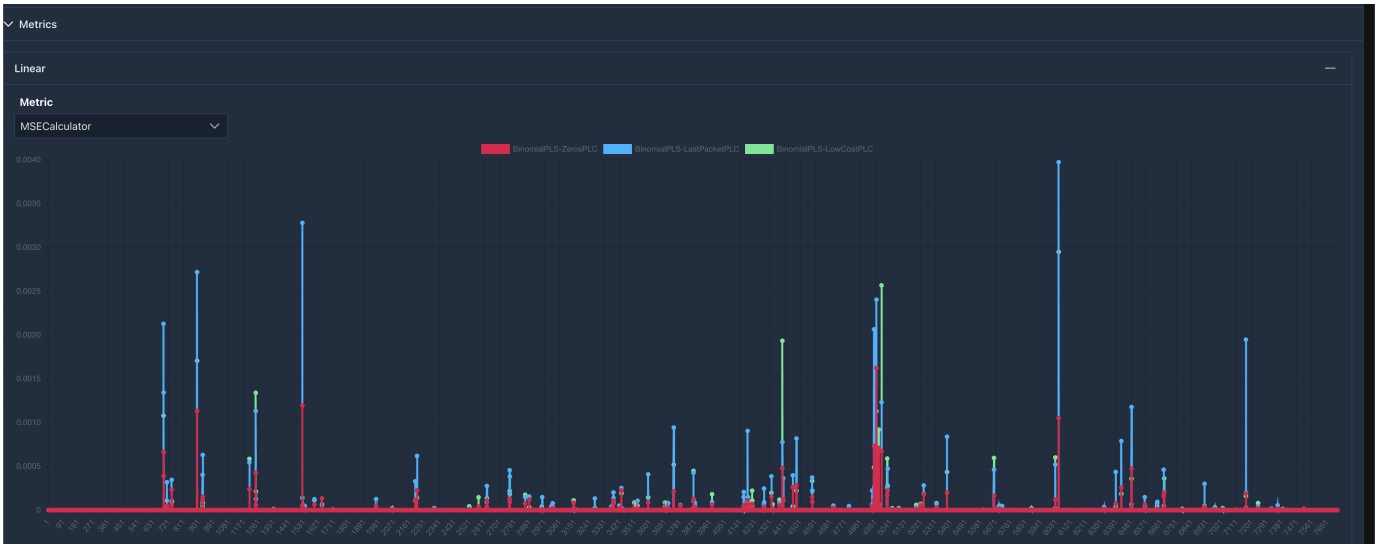

Fig. 7. The metrics view is zoomable and time aligned with the other time-based views (waveform, spectrogram ecc.). It displays multiple versions of the same metric at the same time allowing to compare between different PLC algorithms.

[6] J. Cáceres and C. Chafe, "Jacktrip: Under the hood of an engine for network audio," *Journal of New Music Research*, vol. 39, no. 3, pp. 183–187, 2010.

[7] A. Carôt, C. Hoene, H. Busse, and C. Kuhr, "Results of the fast-music project—five contributions to the domain of distributed music," *IEEE Access*, vol. 8, pp. 47 925–47 951, 2020.

[8] H. Sanneck, A. Stenger, K. B. Younes, and B. Girod, "A new technique for audio packet loss concealment," in *Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference*. IEEE, 1996, pp. 48–52.

[9] M. Fink, "Enhancements for networked music performances," Ph.D. dissertation, Universitätsbibliothek der HSU/UniBwH, 2018.

[10] M. Fink and U. Zölzer, "Low-delay error concealment with low computational overhead for audio over ip applications." in *Proceedings of the International Conference on Digital Audio Effects*, 2014, pp. 309–316.

[11] M. Sacchetto, Y. Huang, A. Bianco, and C. Rottondi, "Using autoregressive models for real-time packet loss concealment in networked music performance applications," in *Proceedings of the International Conference Audio Mostly*, 2022, pp. 203–210.

[12] P. Verma, A. Mezza, C. Chafe, and C. Rottondi, "A deep learning approach for low-latency packet loss concealment of audio signals in networked music performance applications," in *2020 27th Conference of Open Innovations Association (FRUCT)*. IEEE, 2020, pp. 268–275.

[13] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *The Bell System Technical Journal*, vol. 42, no. 5, pp. 1977–1997, 1963.

[14] C. A. G. Da Silva and C. M. Pedroso, "Mac-layer packet loss models for wi-fi networks: A survey," *IEEE Access*, vol. 7, pp. 180 512–180 531, 2019.

[15] M. Li, M. Wu, D. Wu, L. Wang, and C. Xu, "Packet loss concealment using enhanced waveform similarity overlap-and-add technique with management of gains," in *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2009, pp. 1–4.

[16] S. Pascual, J. Serrà, and J. Pons, "Adversarial auto-encoding for packet loss concealment," in *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2021, pp. 71–75.

[17] R. Lotfidereshgi and P. Gournay, "Speech prediction using an adaptive recurrent neural network with application to packet loss concealment," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5394–5398.

[18] Q. Ji, C. Bao, and Z. Cui, "Packet loss concealment based on phase correction and deep neural network," *Applied Sciences*, vol. 12, no. 19, p. 9721, 2022.

[19] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.

[20] T. Thiede, W. C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, and C. Colomes, "Peaq-the itu standard for objective measurement of perceived audio quality," *Journal of the Audio Engineering Society*, vol. 48, no. 1/2, pp. 3–29, 2000.

[21] A. F. Khalifeh, A.-K. Al-Tamimi, and K. A. Darabkh, "Perceptual evaluation of audio quality under lossy networks," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2017, pp. 939–943.

[22] M. Sacchetto, "Implementations of various autoregressive plc algorithms," 2023. [Online]. Available: https://github.com/matteosacchetto/burg-implementation-experiments

[23] M. Holters, "Gstpeaq - a gstreamer plugin for perceptual evaluation of audio quality (peaq)," 2015. [Online]. Available: https://github.com/HSU-ANT/gstpeaq