Packet Loss in Real-Time Communications: Can ML Tame its Unpredictable Nature?

Packet Loss in Real-Time Communications: Can ML Tame its Unpredictable Nature?

Tailai Song, Gianluca Perna, Paolo Garza, Michela Meo, Maurizio Matteo Munafò
Politecnico di Torino
first.last@polito.it

***Abstract*—Due to the flourishing development of networks, and abetted by the Covid-19 pandemic, we have witnessed an exponential surge in the global proliferation of Real-Time Communications (RTC) applications in recent years. In light of this, the necessity for robust, scalable, and intelligent network infrastructures and technologies has become increasingly apparent. Among the principal challenges encountered in RTC lies the issue of packet loss. Indeed, the occurrence of losses leads to communication degradation and reallocation that adversely affect the Quality of Experience (QoE). In this paper, we investigate the feasibility of predicting packet loss phenomena through the utilization of machine learning techniques, solely based on statistics derived directly from packets. We provide different definitions of packet loss, subsequently focusing on the most critical scenario, which is defined as the first loss of a series. By delineating the concept of loss, we propose different problem formulations to determine whether there exists a mathematically advantageous scenario over others. To substantiate our analysis, we demonstrate that these phenomena can be correctly identified with a recall up to 66%, leveraging three ample datasets of RTC traffic, which were collected under distinct conditions at different times, further solidifying the validity of our findings.**

***Index Terms*—Real-time communications, RTP, packet loss, machine learning, prediction, classification.**

I. Introduction

Real-time communication (RTC) stands as an indispensable facet of contemporary society, enabling people to communicate instantly with a variety of networks and devices. In recent years, RTC applications, such as video-conferencing and streaming, have witnessed a remarkable surge in popularity, assuming a foundational role in both professional and recreational domains. The profound impact of RTC applications was unequivocally highlighted during the global COVID-19 pandemic in 2020, wherein lockdown measures compelled millions of individuals to rely heavily on these platforms. Consequently, the world experienced a momentous upswing in RTC traffic, exceeding a staggering 200% [1]. While the pandemic may have subsided, the ascendancy of RTC applications has endured, with remote work becoming the norm [2]. Nowadays, RTC services are rapidly evolving, with a myriad of competing applications on the market [3], which is spurred by the relentless expansion of network infrastructures worldwide and the proliferation of higher bandwidth availability, not to mention the advent of cutting-edge 5G technologies. While various applications employ different technical solutions, most still rely on the Real-Time Transport Protocol (RTP) [4] over the User Datagram Protocol (UDP). For web browsers, the widely adopted standard is the open-source WebRTC[1] framework atop RTP.

In light of these considerations, the assurance of reliability in RTC applications has assumed paramount importance, with a heightened focus on optimizing the Quality of Experience (QoE) for users. Possible improvements can originate from either the application layer or the network layer. At the application level, QoE can be influenced by factors such as hardware performance and software optimization. Conversely, at the network level, it hinges upon diverse elements, encompassing the quality of connections among participants, network topology, and network management. Of noteworthy concern, packet loss emerges as one of the primary detriments to achieving a satisfactory QoE [5], [6]. Such losses engender undesirable repercussions, including delays, jitter, audio or video distortions, and in severe cases, complete service disruption. The effects of packet loss can be particularly deleterious in RTC applications, where users expect near-instantaneous responses and uninterrupted service. Numerous techniques have been proposed in response to this challenge, such as error correction and congestion control [7], [8]. Nonetheless, these approaches possess inherent limitations, leaving packet loss as a formidable hurdle. Thus, the demand for innovative and effective methodologies to predict packet loss in RTC applications remains pressing. By accurately forecasting network conditions, including the occurrence of packet loss, on network devices, substantial enhancements in network management and system configuration can be realized, ultimately elevating the QoE for end-users.

In this paper, we delve deep into the packet loss phenomenon through comprehensive analyses across three datasets[2] of heterogeneous RTC traffic collected during various periods over three years. Intriguingly, our investigations unveil a compelling pattern wherein packet losses exhibit a propensity for consecutive occurrences, coalescing into aggregations of losses, which propels us to identify and tackle the underlying origins of loss eruptions so that the subsequent losses can be addressed effectively. To this end, we propose a novel system that leverages machine learning (ML) approaches and consists of two distinct optimized solutions, formulating the problem either as a time-series prediction task to forecast the onset of clustered losses or a classical classification task to discriminate between different types of losses. Our solutions capitalize on ML algorithms tailored to exploit carefully selected features from historical or contemporaneous statistical values of RTC traffic, such as inter-arrival times and packet sizes. Moreover, we explore both supervised and unsupervised learning techniques, and base our work on abundant traffic collected from multiple sources, including border routers and edge nodes.

To select the optimal ML algorithm and suitable configurations in our case, we evaluate a broad spectrum of models, ranging from conventional techniques to deep neural networks

[1]https://webrtc.org/

[2]Namely "Supervised", "Campus2020" and "Campus2023", and they will be fully described in Section IV.

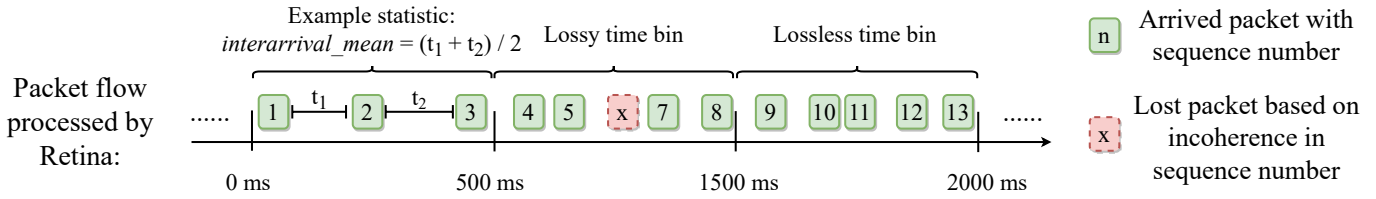


Fig. 1: How Retina works and the concept of time bins.

and anomaly detection algorithms. We employ the recall measure as a key performance metric and implement multiple simple yet effective techniques to address the class imbalance problem. Notably, our system is meticulously developed, taking into consideration the constraints prevalent in real networks. For instance, we discretely partition the dataset for training and testing, separating different RTP streams to avoid data infiltration, and for the prediction, we target further into the future, ensuring that the system has sufficient time to enact network policies and execute system management based on predictions.

Our proposed solution is envisioned to function as a software module in network devices, such as media servers, to establish a comprehensive, ML-based, RTC-aware, and proactive traffic management and monitoring system. The system provides application-level observability at the network control plane, empowering efficient and informed decision-making, and incorporates a feedback mechanism to notify deteriorating network conditions promptly. It consists of several offline-trained classifiers that predict in real-time various network conditions [9], [10], one of which is the possible packet loss in the near future. Subsequently, a controller (e.g., an SDN orchestrator) can apply specific network management techniques to alleviate the worsening conditions, such as rerouting flows along alternative paths or allocating additional bandwidth to them. Furthermore, as part of our commitment to advancing research and fostering reproducibility, we have made the processed and anonymized datasets referenced in this paper publicly available[3]. In summary, our contribution is delineated by the following four key points:

- A comprehensive data compilation with three independent datasets with ample traffic sourced from various vantage points at different times.
- A thorough investigation into the packet loss phenomenon, that unveils its tendency of sustained occurrences, and accentuates the onset of the repercussion.
- An array of analyses and developments for multiple ML models, aimed to predict or identify the initiations of loss eruptions.
- Two proposed ML models that address the complexities stemming from diverse traffic characteristics and data imbalance to yield commendable and generalized outcomes.

The remainder of this paper is organized as follows. We commence by delineating the problem we endeavor to resolve, articulating the formulation of the problem in Section II. In Section III, we characterize the loss events to elucidate the underlying rationale behind our objective, followed by an exploration of the motivation for our work. In Section IV, we complement the background by introducing the dataset employed in constructing our system. Subsequently, Section V describes our methodology for feature engineering and classification, Section VI presents and discusses our experimental results, Section VII examines the literature, highlighting existing proposed solutions, and finally, Section VIII concludes the paper and proposes future works.

[3]https://mplanestore.polito.it:5001/sharing/4YNjiRhxW

II. Problem Statement

In this section, we delve straight into the problem at hand, commencing with the essential background information that paves the way for a precise problem formulation. Moreover, we proffer a series of formidable challenges and the judiciously selected evaluation metric. Note that certain pertinent details are deliberately deferred to subsequent sections to focus on the current problem.

A. Background

We have collected a substantial volume of RTC traffic and utilized a highly configurable tool called Retina [11] to parse the packet captures, thereby generating three distinct datasets (which will be expounded in Section IV). The tool adopts a common RTP flow definition, i.e., a tuple composed of $(ip_{src},\ ip_{dst},\ port_{src},\ port_{dst},\ ssrc,\ type_{payload})$, and on a per-flow basis, it proficiently aggregates time-series packets into consecutive time bins, arranged in chronological order. Over each bin, a set of 73 statistics is computed, effectively encapsulating the patterns exhibited by the aggregated packets. In our study, we have opted for a time bin duration of 500 ms. The rationale behind selecting this specific duration stems from the delicate balance it strikes: it is neither excessively brief, ensuring the inclusion of an adequate number of packets to yield representative statistics, nor excessively protracted, including too many packets and thereby diluting the influence of losses. Besides the statistics, each time bin is further characterized by the presence or absence of packet loss: a **lossless time bin** denotes a bin devoid of any losses, while a **lossy time bin** signifies a bin in which at least one loss has occurred. A visual elucidation is presented in Figure 1. On top of that, we further define two types of lossy time bins:

- **Sparse loss** denotes a lossy time bin, while there are no losses in the 5 s intervals preceding and succeeding the lossy time bin. In other words, the 10 time bins in the past and future of a sparse loss are lossless time bins.
- **Concentrated loss** represents a part of a loss burst that occurs over a certain duration, and it is defined as a lossy time bin with at least one additional loss in its neighborhood of 5 s. That is to say, there is at least one lossy time bin in the 5 s in the past and future of a concentrated loss, and the additional lossy bin is, therefore, also a concentrated loss. Furthermore, multiple

consecutive concentrated losses form a group, in which the first lossy time bin denotes the starting-point loss, while the others represent group members.

A more straightforward visual representation is provided in Figure 2. It is important to note that there is no loss neither in the 5 s before the starting point nor in the 5 s after the last group member.

By analyzing and exploiting statistical features of past packets (past time bins), the goal of our work is twofold: i) predicting the occurrence of a starting-point loss in the near future, or ii) distinguishing between sparse and starting-point lossy time bins once we observe a loss. Specifically, the traffic of each RTP flow is divided into successive 500 ms time bins, and at a certain time instant, we aim at leveraging statistical information of time bins in possession (in the past) to perform two tasks as follows:

- **Problem 1, prediction of starting points of concentrated groups** – We want to utilize the previous 10 time bins (5 seconds of traffic) to predict if there will be a loss or losses in the subsequent time bin. In this case, the current goal may be possible but impractical and meaningless because if we were to approach the problem of predicting losses in the immediate next time bin starting from the present moment, even if the occurrence of loss was correctly predicted, we would not have enough time to manage and instrument the network to react to this event. Therefore, we maintain the target time bin but intentionally skip the exact previous one, and consequently, a 500 ms gap is reserved for introducing a certain degree of freedom to execute operations;
- **Problem 2, classification of different losses** – We want to utilize the observed lossy time bin as well as its past 5 time bins (6 time bins in total, 3 seconds of traffic) to determine whether the observed lossy bin is a sparse or starting-point lossy time bin.

As a result, we end up with two binary classification problems. Additionally, despite collecting a massive amount of data, only certain portions are useful. For problem 1, we exclude sparse losses[4], concentrated group members, and lossless time bins after them, not only because of their irrelevance but also for the potentially misleading information introduced during model training. For problem 2, all lossless bins and concentrated group members are unneeded.

B. Problem formulation

Formally, assuming at a time instant t, both problems can be represented as a function of traffic features as follows:

1) Prediction of starting points:

$$\begin{aligned} Y_t &= f_1(\overline{x}_{t-2\Delta t}, ..., \overline{x}_{t-w\Delta t}, ...) \\ \text{with } & w \in [2, W_1], \\ & \overline{x} = (x_1, .., x_N), \\ Y_t &= \begin{cases} 0, \text{No loss} \\ 1, \text{Loss (starting point)} \end{cases}, \end{aligned} \tag{1}$$

2) Classification of different losses:

$$\begin{aligned} T_{t-\Delta t} &= f_2(\overline{x}_{t-\Delta t}, ..., \overline{x}_{t-w\Delta t}, ...) \\ \text{with } & w \in [1, W_2], \\ & \overline{x} = (x_1, .., x_N) \\ T &= \begin{cases} 0, \text{Sparse loss} \\ 1, \text{Starting point} \end{cases}, \\ \text{s.t. } & Y_{t-\Delta t} = 1, \end{aligned} \tag{2}$$

where t is the current moment and Δt is the time bin size (500 ms). Note that a time instant (e.g., t and $t-\Delta t$) represents the initial timestamp of a time bin. In problem 1, Y_t is a binary variable (class label) that represents the presence of packet loss in the subsequent time bin starting from the current moment t, and it takes the value 1 if a loss/losses exist and 0 otherwise. In problem 2, $T_{t-\Delta t}$ is also a binary variable (class label) but represents the lossy bin type of the observed (current) lossy time bin starting from $t-\Delta t$ and ending at the current moment t, and it takes 0 for sparse lossy bins and 1 for starting points. In both cases, $\overline{x}$ represents the input traffic statistics (e.g., packet sizes or inter-arrival times) of the ML model, and particularly, $\overline{x}_{t-w\Delta t}$ is the features vector in the past time bin at $t - w\Delta t$. In general, N is the number of different types of statistics (features) considered and W is the window size - how many time bins we consider from the past. In our case, we have $N = 73$ statistics in total, and consider $W_1 = 10$ time bins (5 s in the past) for problem 1 and $W_1 = 6$ time bins (3 s including the present and the past) for problem 2. We aim at developing ML models to learn two individual functions $f_1(\cdot)$ and $f_2(\cdot)$, which map our input vectors $\overline{x}$ to the binary classification target variables Y or T.

Additionally, we adhere to a time interval of 5 s when defining the two loss types as well as formulating the problem. This choice of duration strikes an judicious trade-off between a prolonged interval that encompasses extraneous information beyond the target time bin and a shorter interval that might overlook potentially crucial information in close proximity to the aforementioned bin. In essence, a 5 s duration is an empirically ideal temporal extent in the context of RTC, encapsulating the advantages of both breadth and precision. Meanwhile, we opt to reduce the designated duration for problem 2 to a span of 3 s, which is primarily based on the possession of statistical features within the target time bin, rendering the inclusion of temporally distant bins unnecessary. Nonetheless, the selection of a specific duration for time bins is not deemed a critical aspect in our study, which is mainly due to our adoption of a technical approach that effectively selects informative features, thereby mitigating the inherent time constraints. To proffer supplementary material, we elaborate on the rationale behind our duration choices in the appendix (Section VIII-A).

C. Challenges

Indeed, we anticipate multiple challenges in our problem: i) For problem 1, we are obliged to predict the lossy bins

[4]We remove sparse losses during model training to avoid inadvertently learning pertaining patterns. However, the model may still predict a sparse loss and identify it as a starting point, which, nevertheless, presents no significant concern in practice, because (1) if a sparse loss is predicted as a starting point, it is not inherently erroneous, as a sparse loss indeed represents a lossy bin, and (2) if a sparse loss is not identified, its impact is likely negligible since there are no subsequent losses to compound the ramification. It is exactly due to the same reason that we also omit sparse losses during the testing phase without explicitly factoring them into the ultimate performance assessment.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

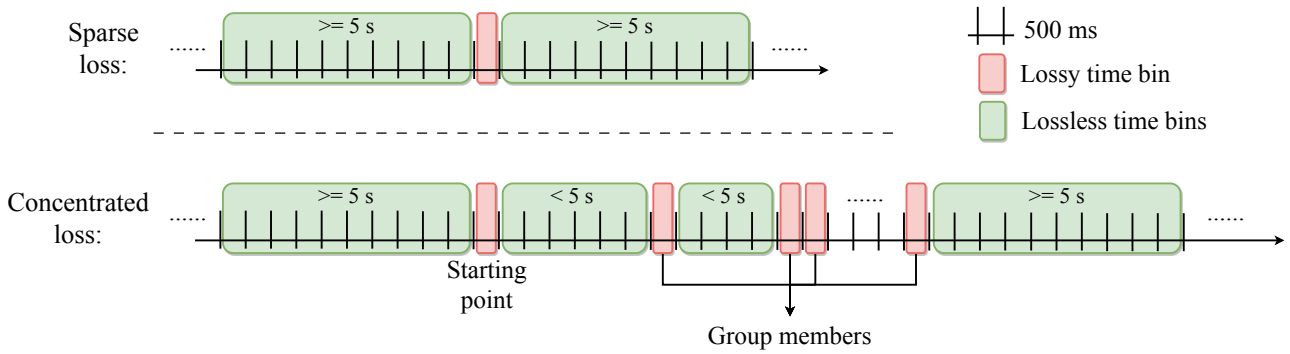


Fig. 2: The concept of sparse loss and concentrated loss.

further into the future, bypassing a time bin of 500 ms, which might contain significantly informative patterns, since it is the hithermost one to the target; ii) Our datasets are extremely imbalanced (as evidenced by Table I that will be explained in Section III), with lossy time bins representing only a small percentage (ranging from 0.15% to 0.80%), let alone the even less amount of our predicting target - starting points. In ML domain, models trained on such data typically exhibit a bias towards the majority class [12], [13]; iii) For problem 2, we have a limited number of available samples (also indicated by Table I), which are scarce for a data-driven problem [14]. Moreover, both sparse losses and starting points share the same behaviour of being lossless in the past 5 s, obfuscating the patterns and exacerbating the complexity of classification; iv) RTC is inherently dynamic and constantly evolving, influenced by extensive factors that generate diverse patterns that might not be captured at the same time and thus hamper the prediction. Meanwhile, instead of completely shuffling the dataset, we intentionally split the RTP flows for model training and testing to resemble the reality, where the incoming traffic has never been observed by the model. Not to mention that the training and testing are performed separately on different datasets spanning three years.

D. Evaluation metrics

In assessing the performance of our models, we face the challenges posed by the highly imbalanced nature of problem 1. In such a case, many evaluation metrics tend to favor the majority class, resulting in a distorted estimation of the model's true performance. To address this issue, we opt to use the recall metric as expressed by the formula: $\frac{True\ Positive}{True\ Positive\ +\ False\ Negative}$, which is particularly suitable for imbalanced classification problems. Recall, serving as the measure of accuracy for a particular class, allows for the evaluation of each individual class's performance independently, without any interference from other classes. It reflects the sensitivity of the model, i.e., its ability to correctly identify a sample as positive when it belongs to that class. Due to the same reason, despite the comparable quantities of both classes in problem 2, we still refer to recall as the evaluation metric.

Furthermore, most imbalanced problems devote to improving the performance of the minority class, leading to a balanced outcome with comparable recalls for both classes, in which the performance is optimized for the minority but may be degraded for the majority class [15]. However, our problem has a distinctive requirement that we want to avoid an excessive influx of false positive samples that could trigger unnecessary false alarms, squandering network resources. It is essential, therefore, to guarantee a commendable accuracy for class 0, while simultaneously maximizing the performance for class 1, which can only be explicitly assessed by the class recall. To summarize, the recall metric offers a more precise assessment of the model's performance in comparison to other metrics, particularly within the confines of our specific classification quandary.

III. Loss Characterization

In this section, we investigate the details of packet loss phenomenon, providing crucial information to rationalize our objective. Subsequently, we further justify our primary impetus.

A. Analysis and discussion

The quantities pertaining to different time bins are summarized in Table I, which demonstrates that the number of lossy time bins is rather scarce, indicating a class imbalance issue in all cases. More importantly, most lossy time bins are concentrated losses, exhibiting an inclination to aggregate regardless of datasets. This indicates a prevalent phenomenon of packet loss burst, suggesting a high likelihood of capturing and identifying more losses after encountering a loss. Another important observation is portrayed in Figure 3, which depicts the Cumulative Distribution Function (CDF) for the duration of individual concentrated-loss groups in each dataset. In general, all datasets have similar patterns: i) around 10% groups fall into a duration of 1 s with two adjacent lossy time bins, and ii) the majority, more than 80% of groups, last for less than 10 s. In other words, if a horizon of 10 s after any starting point is set, we are capable of allocating more than 80% concentrated groups for all concentrated losses. This, coupled with the fact that most lossy time bins are concentrated losses, implies that the starting-point bins are more valuable and crucial, because upon a successful recognition of such losses, most subsequent losses (concentrated group members) can be easily identified and thus addressed. The starting point indicates the onset of a potentially and relatively severe network issue, and the accurate prediction of it could enable timely and effective network management and optimization. In contrast, sparse losses are theoretically difficult to predict given their randomness, let alone the rare existence. They are also less detrimental with respect to concentrated losses, which might cause a continuous and lingering effect that keeps impeding the traffic and consequently impairing QoE. Therefore, instead

TABLE I: Summary of the quantity of time bins.

Dataset	Supervised	Campus2020	Campus2023
Total time bins	1537790	3100253	3897258
Lossy time bins	12330 (0.80%)	7009 (0.23%)	5745 (0.15%)
Sparse loss	4044 (32.80%)	1904 (27.17%)	2000 (34.81%)
Concentrated loss	8286 (67.20%)	5105 (72.83%)	3745 (65.19%)
Starting point	1741	815	814

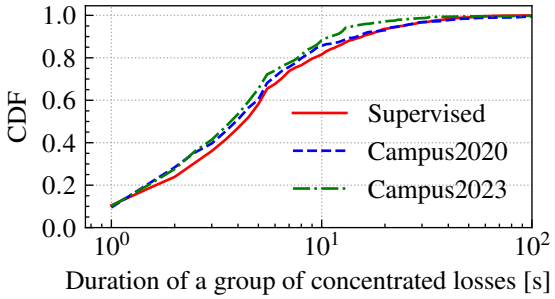


Fig. 3: CDF of the duration of concentrated losses.

of targeting any types of losses, which is ambiguous and unnecessary, we focus on the identification of starting-point lossy time bins. This choice is further motivated by a naive statistical observation based on the percentage of concentrated losses in Table I, i.e., the probability of observing another loss after a loss has occurred is approximately 65-73%. In this context, it is unnecessary to apply a complex classifier following a starting-point loss to predict whether we will encounter another loss in the future.

Furthermore, Figure 3 also provides an intuition about how the monitoring system could behave. For instance, upon successfully identifying a starting point, an alarm could be triggered for a duration of up to 10 s, since over 80% of concentrated groups are less than 10 s. In other words, the network problems that cause packet losses will most likely persist for a duration shorter than 10 s, and by adopting such a strategy, more than 80% of issues could potentially be addressed. However, this is a rather conservative approach since the potential future losses would probably not have occurred once the system management starts to intervene, which means that it is also feasible to program a radical reaction. Nevertheless, all of the aforementioned aspects highlight the practical importance of starting-point lossy time bins and emphasize the need of an effective solution for them. On the other hand, both sparse loss and starting point have the same concept of no loss in the past 5 s, meaning that it is hard to tell one apart from another to detect possible following losses when the system observes a lossy time bin for the first time. In fact, the quantity of both losses are comparable in all datasets according to Table I, resulting in a probability of around 30% to confront concentrated group members after a lossy bin without any losses in the past 5 s.

Therefore, we postulate an operating scenario in which the system starts to work from the beginning of an RTC stream and does not cease until observing a lossy time bin, or until being informed by a predicted starting-point lossy time bin. In both cases, no matter we detect a loss on the fly or we predict a loss in advance, an action persisting for a predetermined duration, e.g., 10 s, could be taken to optimize the traffic and mitigate further losses. Moreover, in the former case, we need to identify whether the observed lossy bin is a starting point to avoid unnecessary actions for sparse losses with no group members following it, and it is acceptable to make occasional mistakes, since we are dealing with only thousands out of millions samples. However, we are restricted by the fact that the system is not timely for the observed loss and may not be prompt for concentrated group members adjacent to the observed ones due to the time needed for further operations. Fortunately, such situations only account for around 10% of all concentrated groups according to Figure 3, verifying the effectiveness of the approach in most cases. In the latter case, we have relatively sufficient time to react, even for the predicted ones, but the prediction operates on any time bins and thus may introduce many errors and unnecessary reactions for lossless time bins. We propose both approaches in order to accommodate different network conditions and possible requirements, providing a trade-off between timeliness and preciseness.

To facilitate a successful prediction, we aim at identifying variations to distinguish lossy time bins from lossless ones by inspecting the patterns of statistics prior to target time bins. Figure 4 presents two examples of such statistics: the standard deviation and the minimum of inter-arrival time of all packets in a time bin. For each of them, the averaged value in each time bin in the 5 s before all starting points or the same amount of randomly selected lossless time bins is computed respectively. Despite several fluctuations, the former statistic produces an increasing trend and the latter one experiences a slight decline, regardless of the dataset. More importantly, the behaviors of statistics preceding lossless time bins are substantially stable, as indicated by the relatively flat lines and narrow confidence intervals. This suggests that there are indeed discernible variations in statistics before the actual occurrence of packet loss. While the averaging process may introduce certain degree of error and information loss, it nonetheless provides insight into the impact of packet loss and verifies the feasibility of distinguishing lossy and lossless time bins based on historical statistics.

Furthermore, differentiating between a starting point and a sparse loss also aligns with the objective of tackling concentrated group members, once the target time bin has been observed. However, this task poses even more challenges due to the minimal distinction between the two categories. The only differing factor is the presence of additional losses following the starting point, which may not significantly impact the statistics preceding the target time bin, particularly if the subsequent losses occur near the end of the 5 s time window. In such cases, we can only rely on ML models to detect subtle patterns embedded in the data.

B. *Underlying Motives*

The primary objective of this project is to develop a classifier capable of accurately predicting losses in RTP traffic flows. Several compelling reasons impel us to undertake this task, which will be thoroughly discussed below.

First and foremost, it is imperative to comprehend the limitations associated with exclusive reliance on the information contained in the RTP header. Despite being transmitted in plain text, it is often challenging to directly depend on this information. Delving deeper into the matter unveils that while web browser-based multimedia software implementations, employing WebRTC, furnish accurate data, other non-web applications hinge upon proprietary implementations, resulting

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

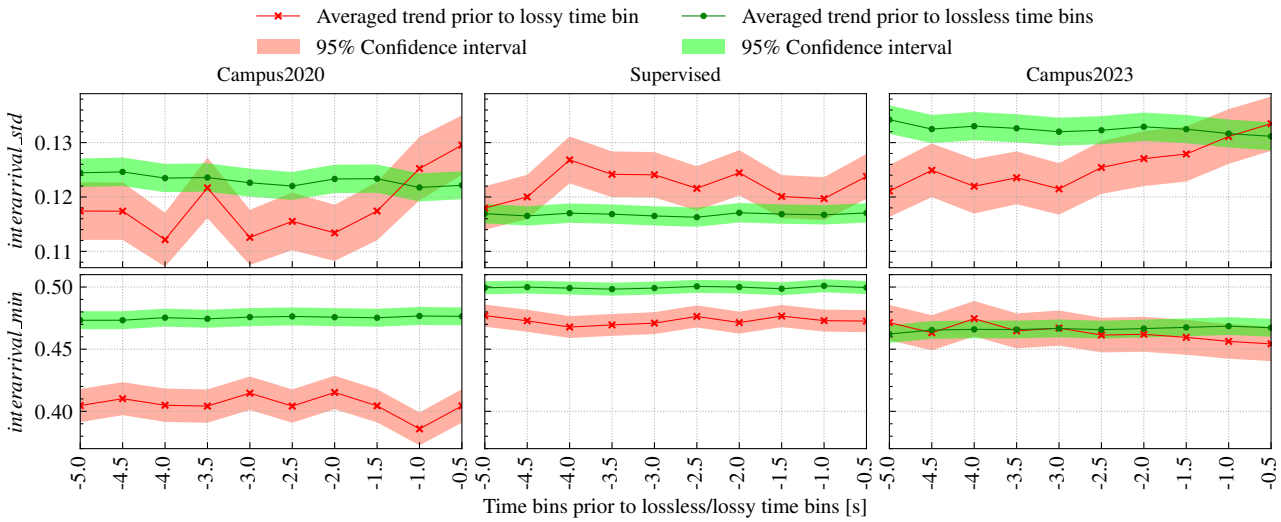


Fig. 4: The averaged patterns with 95% confidence interval of two example statistics (normalized value) in the past 5 seconds of starting-point lossy or lossless time bins.

in frequently inaccurate information. Thus, drawing reliable conclusions directly from the header becomes an arduous task. By focusing exclusively on network packet statistics for classification, we can effectively minimize the need for deep packet inspection. This approach allows us to concentrate on fields that are typically more difficult to manipulate, such as inter-arrival time and packet size.

The advantages of relying solely on packet statistics are manifold. Firstly, the acquisition of these statistics is seamless and convenience. As the packets reach the location of the classifier, relevant statistics can be swiftly calculated and stored. Moreover, this reliance on statistics for classification affords a degree of detachment from the complexities of underlying network elements, including devices with their queues, bandwidths, and available resources. While these elements are undoubtedly significant, attempting to collect and analyze all pertinent information at each network hop for predictive purposes would introduce considerable intricacy and overhead.

From an implementation perspective, integration of this classifier into existing network live monitoring software is achievable, thus providing valuable insights into the network's status. The optimal deployment entails situating this tool in close proximity to the border routers of expansive data centers or campuses. Notably, the training and testing data collected to develop the classifier align with this deployment scenario, as they have been gathered near the edge nodes, akin to these deployment points. Relocating the tool to a different location introduces the possibility of encountering more inaccurate predictions due to the phenomenon of data drift [16]. However, should the need arise, this effect can be mitigated by performing fine-tuning procedures, incorporating new data collected specifically from the desired deployment location.

On top of that, although it is relatively infeasible to evaluate the actual enhancement enabled by the predictions at this point, we still envisage the prospect of network performance improvement upon the loss identification, especially in light of the heightened likelihood of encountering extra incoming losses. For instance, Google Congestion Control (GCC) [7] could harness the prediction as an indicator to preemptively facilitate the bitrate management as well as mitigate network congestion, and the emergent Software Defined Networking (SDN) paradigm could intervene beforehand to effectively perform bandwidth allocation and traffic rerouting [17], [18], [19], [20]. Furthermore, despite of the added consumption induced by the model execution, we do not anticipate excessive overhead perplexing the network control strategy, given that packet loss per se stands as a fundamental performance metric, which is inherently factored into the majority of contemporary optimization technologies, and thus the prediction outcomes can be seamlessly integrated into the management system.

IV. Dataset

In this section, an exhaustive introduction of the datasets employed in our study is provided to supplement the information gaps in preceding sections.

A. Dataset description

The raw dataset takes the form of *pcap* files that contain all the packets captured during RTC sessions using the RTP protocol for information exchange. The overarching objective is to develop a supervised dataset where each entry is labeled to indicate the presence of packet loss. Although RTP traffic is transmitted over an unreliable protocol such as UDP, it is still possible to trace packet losses through information added to RTP packet headers in most cases. This is feasible because the sequence number is transmitted in plain text, enabling identification of missing packets and reconstruction of the original stream.

The study employs three datasets: "Supervised", "Campus2020", and "Campus2023". The "Supervised" dataset was collected using two videoconferencing applications, namely Webex[5] and Jitsi Meet[6], under different network scenarios (Wi-Fi, Mobile, Ethernet), and over different months between 2020 and 2021. It was intentionally collected during multiple real video conferences in a supervised manner, where rich logs are available for the purpose in [9]. The "Campus2020" dataset was gathered during a similar period, but from a different

[5]https://www.webex.com/
[6]https://meet.jit.si/

TABLE II: Dataset overview.

Dataset Name	File Size [GB]	Collection Period	Vantage Point
Supervised	66.54	2020-2021	Edge-node
Campus2020	68.33	2020-2021	Campus router
Campus2023	87.34	2023	Campus router

vantage point. Specifically, while the "Supervised" dataset was collected at the edge-node using tools like Wireshark, the "Campus2020" data was captured through TCP STatistic and Analysis Tool (Tstat)[7] [21] before the border router of a university campus after anonymization of the traffic using the Cryptography-based Prefix-preserving Anonymization (Crypto-PAn) algorithm [22], in compliance with user privacy. Additionally, "Campus2023" was captured in March 2023 from the same vantage point as "Campus2020".

Table II summarizes the characteristics of the raw datasets used in this study, including the size of the corresponding *pcap* files, collection period, and vantage point of data capture. The strategic placement of the vantage point at the exit node of the university campus proved advantageous for capturing a significant volume of traffic. This was primarily due to two key factors. Firstly, the campus's output link boasts a formidable bandwidth of 10 Gb/s, enabling the acquisition of a substantial amount of data in a relatively short period[8]. Secondly, unlike the "Supervised" dataset, the data collection approach for the campus scenario involved continuous collection using Tstat software, without further human intervention. It is important to emphasize that our analysis focuses exclusively on inbound streams, as they provide unique insights into network performance and behavior. These flows have traversed the entire network path, allowing us to examine factors such as congestion, packet loss, and latency experienced during transmission.

B. Dataset characterization

To provide a detailed characterization of the raw datasets, we present six Estimated Cumulative Distribution Function (ECDF) plots in Figure 5, each corresponding to a networking property of all datasets used in our study. These plots highlight the peculiarities and differences of the datasets.

In Figure 5a, 5b, and 5c, we present the most common networking statistics, including the number of packets, bitrate, and inter-arrival time, aggregated within one-second time frames. These statistics provide valuable insights into the network's performance. Notably, in the initial portion of the charts, all of these metrics exhibit a similar increasing trend, which will be further analyzed shortly. Moving on, Figure 5d demonstrates the inter-arrival time of the RTP header, a 16-bit record that plays a crucial role in synchronizing the source and destination within the application. Additionally, Figure 5e displays the average size of the UDP payload within one-second time frames. This information helps us understand the characteristics of data transmission size within the RTP flow. Lastly, in Figure 5f, we present the ECDF that allows us to examine the distribution of the duration for a given RTP flow. Notably, throughout the entire dataset, there is a significant probability of observing long-lasting flows, emphasizing the importance of developing a tool to monitor and maintain stable communication over extended periods.

Despite the variations in collection times, conditions, and applications, the datasets exhibit a notable degree of statistical similarity. Upon closer examination, we observe a prominent increase in the ECDF around 50 packets/s (as shown in Figure 5a) or an inter-arrival time of 20 ms (as shown in Figure 5c). This trend can be attributed to the presence of audio flows, which are packetized following the RTP RFC 3550 [23]. Given that the most common audio packetization implementation is 20 ms, and the widely used Opus codec operates at a frequency of 48 kHz [24], [25], the RTP timestamp between two consecutive audio packets is 960 (i.e., $48\text{kHz} \times 20\text{ms}$), as demonstrated by the ECDF in Figure 5d. Other commonly used packetization values are 10 ms, 40 ms, and 80 ms, from which the RTP inter-timestamp values can be derived.

Based on these observations, we can reasonably infer that a substantial portion of all three datasets is composed of audio flows. This is not surprising given that audio is a fundamental component of RTC applications. In fact, in many real-world scenarios, users may opt to disable the video component of a call, particularly if they are experiencing network congestion or have limited bandwidth. Consequently, audio traffic tends to constitute a significant proportion of the total traffic in such situations. Nevertheless, it is important to note that despite the dominance of audio flows, the datasets still possess a diverse range of flows, including a substantial representation of video traffic, given the vast amount of the overall traffic collected, totaling 222.18 GB. This comprehensive coverage ensures the datasets' relevance and representativeness in capturing the various aspects of RTC traffic.

C. Dataset construction

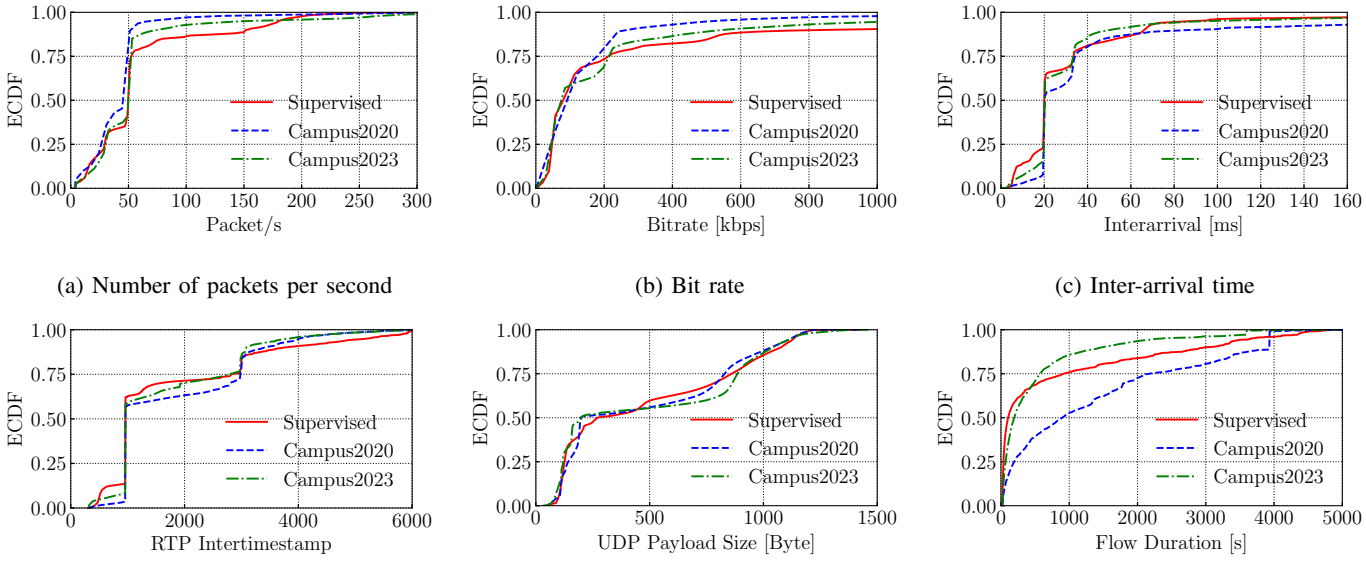
Herein, we present the procedure that converts the raw data from *pcap* files into a format suitable for ML algorithms. The initial stage entails employing Retina to process *pcap* files containing RTP traffic, as mentioned in Section II. However, the output generated by Retina serves as an intermediate step in our case. In order to construct the dataset for the ML pipeline (as elaborated in Section V-A), an additional stage is required, which involves reshaping and concatenating the intermediate dataset to incorporate the previous 5 seconds of history for each current time bin. By doing so, each data sample becomes 11 time bins long, encompassing the current bin and the 10 time bins from the past. Consequently, each data sample comprises 803 features (73 statistics $\times$ 11 time bins), and includes the corresponding class label indicating the presence or absence of losses in the current time bin. This manipulation process ensures that the dataset is appropriately structured as a time series, facilitating the subsequent procedures of our study.

Moreover, another crucial aspect is related to the definition of lossy time bins. Retina determines a lossy time bin by calculating the number of losses exiting within the bin based on the difference between RTP sequence numbers. However, many modern RTC applications introduce customized mechanisms for their own interests and deviate from the protocol that generates an increment of 1 for each packet produced. We manually inspect multiple *pcap* files and allocate packet flows with peculiar patterns as follows:

[7]http://tstat.polito.it/

[8]Throughout the day, the campus experiences a mean traffic flow of around 3.5 Gb/s.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616



(a) Number of packets per second (b) Bit rate (c) Inter-arrival time

(d) Inter-arrival time for RTP timestamp (e) Size of UDP payload (f) Duration of RTP flows

Fig. 5: Characterization of raw datasets: "Supervised" vs. "Campus2020" vs. "Campus2023".

- In some cases, a jump larger than 1 for sequence number is generated either after the packet labelled by RTP marker or after the last packet of an entire frame;
- Some traffic has dynamic RTP payload type and produces consecutive sequence numbers even the payload type is different. According to our flow definition, where the payload type is unique per flow, fake packet losses will be generated based on the difference between sequence numbers.

In other words, the ground truth of the presence of packet loss in a time bin becomes ambiguous in such cases. To address this, we set up thresholds and apply a filter to keep lossy time bins in the output of Retina satisfying two rules: i) The number of received packets in a time bin must be greater than the number of calculated losses; ii) The number of calculated losses must not exceed 5 packets[9]. Consequently, we should theoretically possess relatively more lossy time bins in the dataset, indicating potentially severer issues of packet loss in real networks. However, given the massive amount of traffic collected originally, we can reasonably consider the remaining lossy bins representative.

V. Methodology

In this section, we elucidate the details of the ML pipeline, including the selection of features, ML model, and sampling strategy. Specifically, we construct features via a deliberative technical feature selection process, and propose multiple ML models as well as sampling strategies with a primary emphasis on addressing the imbalanced problem.

[9]Superficially, a threshold of 5 packets seems to be overly cautious, but given the predominant audio traffic with fewer than 50 packets per second, equivalent to a mere 25 packets in 500 ms in our case, 5 losses, amounting to a loss rate exceeding 20%, already assumes a considerable magnitude. Although this is a stringent constraint on video traffic, our deliberate inclination towards conservatism arises from a paramount desire to avert data corruption, thereby guaranteeing the impeccable integrity of the model development.

A. Overview of the ML development pipeline

First, we illustrate the whole workflow following dataset construction, ML model development, model consolidation, and model deployment. A detailed description is in Figure 6. Initially, the raw data sourced from *pcap* files are parsed by Retina, and then preprocessed to construct the structured dataset. Afterwards, we perform a meticulous model development process, undertaking a two-step feature selection process, examining different ML models, and investigating sampling strategies. At last, we proceed to finalize the model performance, by evaluating class recalls across different datasets.

Crucially, the dynamic and ever-changing nature of networks and real-time communications has raised the need for a versatile ML model that can adapt to various scenarios, without being constrained to a specific application. In this study, our objective is to explore the feasibility of training a model on a critical endpoint, such as the incoming traffic in a campus, observed three years ago, that can continue to identify relevant patterns despite the passage of time. Instead of developing separate models for each dataset, our approach involves constructing models on the "Campus2020" dataset (or its subset) and assessing its performance on the other two datasets. By doing so, we aim to investigate the generalizability and transferability of the trained model and evaluate its capability of providing accurate insights without the need for retraining. In this context, "Campus2020" dataset serves as a basis (training and validation sets) for the model development where the common data splitting process in ML domain is needed, and the entire "Supervised" and "Campus2023" datasets (test set) are used to evaluate the ultimate performance. To ensure the independence of training and validation datasets, and to prevent data infiltration between RTP streams during the model development process whenever data splitting is needed, the "Campus2020" dataset is split in a way that the flows to which the data in training set belong to, would not present in the validation set. Specifically, we still refer to a random data partitioning approach, but shuffle the flows instead of individual

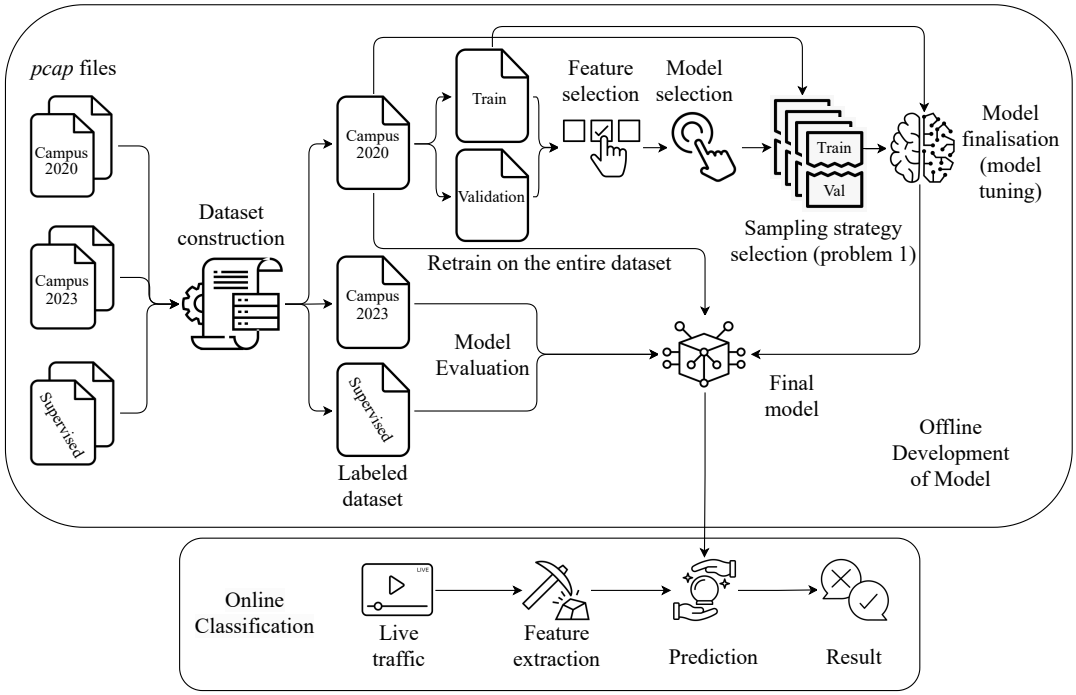


Fig. 6: Overview of the ML pipeline.

samples, and the dataset is split such that 70% of the data are allocated to the training set, while the remaining 30% are in the validation set. In cases when early stopping is needed, one-seventh of the training set, equivalent to 10% of the entire dataset, is reserved. With "Campus2020" dataset, as portrayed in Figure 6, our initial step involves conducting a preliminary data splitting to facilitate feature selection and machine learning model examination. Specifically tailored for problem 1, we then undertake a multiple-trial evaluation process to determine the optimum sampling strategy. Following this, we refine the best choice by tuning hyper-parameters of the selected model (with the selected sampling strategy) based on the performance metrics observed on the validation set. Upon achieving the optimal configuration, we retrain and finalize the model on the entire "Campus2020" dataset. Imperatively, both "Supervised" and "Campus2023" datasets are excluded from the model training and tuning phases, and are tested separately using the finalized model. All of these allow for the evaluation of the model's ability to generalize to different datasets, while also ensuring the independence and integrity of different datasets. By utilizing the "Campus2020" dataset, we aim to determine whether a ML model can be trained on historical data and still be effective in identifying patterns in any RTC traffic.

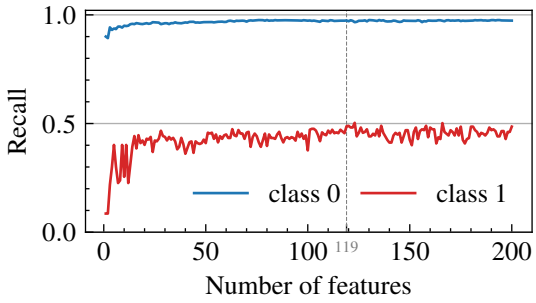
It is noteworthy that the model development is performed offline to fully exploit abundant data available, whereas the system is devised to construct features and perform tasks in real time. Additionally, all procedures can be optimally operated on a per-flow basis, and activities of features as well as predictions only require an individual stream, which enables our methodology to be readily amenable to seamless parallelization. Hence, it is viable to apply a multi-core parallel computing strategy, and possible to scale up to handle even larger volumes of traffic. Meanwhile, we postulate minimal bottlenecks for high-speed deployment. To achieve this objective, our methodology could rely on off-the-shelf hardware, specialized packet capture libraries such as DPDK[10], in addition to network cards that inherently support load balancing, such as Intel's Receive-Side Scaling (RSS). In the following, we focus on the two key technical aspects of the model development process.

B. Feature engineering

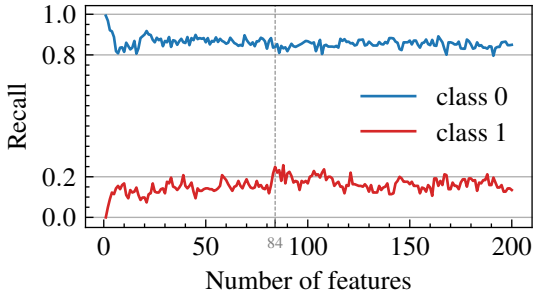
As explicated in Section IV-C, the preprocessing generates a dataset, in which each sample represents a time bin comprising $73 * 11 = 803$ features, corresponding to the 73 statistical values in the current time bin and each of the past 10 bins. Nevertheless, to effectuate the final dataset construction, two additional steps are necessary to be performed individually for each problem: i) For problem 1, in accordance with Section II, the current and the exact last time bins are removed, which requires an ultimate reshape of the intermediate dataset, transforming the number of features in each sample from 803 to $73 \times 9 = 657$. For problem 2, the targets are the current time bins that we already observed, and the unneeded time bins from the past 2.5 to 5 s (5 bins) are excluded, resulting in $73 \times 6 = 438$ features left; ii) The second step is feature engineering, which discards redundant information and decreases the overall number of features. Consequently, we initialize the model development process with a basic data splitting, and a two-step feature selection process is applied as follows:

1) Correlation analysis: A preliminary feature elimination process is executed, removing those that are highly correlated. In particular, we evaluate the correlation between each pair of features and remove one of them randomly, if the Pearson correlation coefficient is greater than 0.9 (in absolute value). Consequently, we keep 61 out of the 73 statistics, and we obtain $61 \times 9 = 549$ and $61 \times 6 = 366$ features for the next step.

[10]https://www.dpdk.org/



(a) Problem 1: prediction of starting points



(b) Problem 2: classification of different losses

Fig. 7: The class recalls of the result for RFE process.

2) Feature refinement: we utilize Recursive Feature Elimination (RFE) approach to systematically examine the importance of features and iteratively remove insignificant ones. In particular, we train on the training dataset an *eXtreme Gradient Boosting (XGB)* classifier, which inherently provides the feature importance scores, and afterwards, the least important one is eliminated. The whole procedure is recursively repeated to inspect the impact of number of features. Consequently, we refine the list of features derived in the previous stage and retain those that are most informative for the problems.

Figure 7 reports the result of feature selection process in terms of class recalls with respect to the number of features. For problem 1, class 0 experiences slight improvements initially and quickly stabilizes afterwards, reaching a plateau, while class 1 undergoes dramatic changes in the early stages, followed by marginal improvements with minor fluctuations. At first glance, the recall for class 1 appears to stabilize after 100 features, we still choose to include 119 features, which enables the recall for class 1 to reach 0.5 for the first time. For problem 2, both classes produce volatile variations even with more features. In particular, the recall of class 0 reaches its peak with 25 features and then decreases mildly. Therefore, we select 84 features, which yields the best performance for class 1. Although we do not observe a prominent inflection point, known as a "knee", beyond which adding more features becomes unproductive, we adopt a conservative approach and include a slightly higher number of features. Nevertheless, we effectively remove a significant portion, more than 80%, of the original features for both problems.

C. Model selection

The model selection is carried out for each problem separately. For problem 1, two different methodologies are considered. The first approach treats the problem as an unbalanced binary classification task, while the second approach views packet loss events as anomalies in comparison to lossless time bins, framing it as an anomaly detection problem (the data used to address the problem remain the same). Therefore, 3 tree-based classifiers: Decision Tree (DT), Random Forest (RF), XGB Gradient Boosting Decision Tree (GBDT), and 2 Deep Learning (DL) classifiers: Deep Neural Network (DNN) and Long Short-term Memory Neural Network (LSTM NN), are developed for classification in a supervised manner. And 2 algorithms: Isolation Forest (IF) and Autoencoder (AE), are used for anomaly detection in an unsupervised way. In all cases regarding the binary classification, the following two manipulation techniques of the model are attempted to handle and mitigate the problem caused by the highly imbalanced dataset:

- *Weighted sampling* is introduced during the model training in two ways: i) For RF, drawing bootstrap samples from the minority class and then randomly drawing the same number with replacement, from the majority class [26]. As a result, we construct the so-called Balanced Random Forest (BRF); ii) For deep learning approaches, the training batch is constructed by sampling elements with given probabilities, which are computed based on the class weight, and thus, the minority samples have a higher probability to be selected.
- *Assigning class weight* can adjust ML models so that misclassifying a sample from the minority class is more heavily penalized than misclassifying a majority-class sample. During the training phase, the weight of the positive class is set according to the multiplication between the majority and minority class in the dataset.

Instead, considering that the anomaly detection algorithms are unsupervised methods that are specifically designed to cope with rare events, no additional solutions are required. It is also important to note that, in the case of all DL methods, instead of using the subset of features derived earlier, we feed the network with all possible features, offering the model a chance to decide which the best representative subset is by exploiting the capability of NN to automatically extract useful features [27]. Additionally, for problem 2, we only implement the 3 tree-based models and the k-nearest neighbors (k-NN) algorithm, since it is a relatively simple classification problem with comparable class quantities, in contrast to the prediction of starting points.

Moreover, we introduce an additional step to further tackle the class imbalance in problem 1. After selecting the best model, we resort to artificial sampling, which is an approach that artificially manipulates the quantity of samples to mitigate imbalance between classes. This can be achieved through oversampling (increase the amount of minority class), undersampling (decrease the amount of majority class), or hybrid-sampling (combination of oversampling and undersampling). In our case, we refer to Synthetic Minority Oversampling Technique (SMOTE) [28] for oversampling and hybrid-sampling, and a random selection for undersampling. Particularly, for each sampling strategy, we perform 50 trials of training and validation, each featuring data from different randomly selected

TABLE III: Basic results of all ML models on "Campus2020" for problem 1.

Category	Model	Recall		Precision		F1-score
		Class 0 (Lossless bin)	Class 1 (Starting point)	Class 0	Class 1	Macro Average
Classification (Tree-based)	DT	0.988	0.045	0.999	0.001	0.498
	RF	0.996	0.051	0.999	0.001	0.499
	BRF	0.962	0.486	0.999	0.003	0.494
	XGB	0.973	0.522	0.999	0.005	0.498
Classification (DL-based)	DNN	0.856	0.537	0.999	0.001	0.462
	LSTM	0.875	0.652	0.999	0.001	0.468
Anomaly Detection	IF	0.973	0.020	0.999	0.001	0.493
	AE	0.999	0.001	0.999	0.001	0.500

RTP flows and yielding evaluation metrics. In essence, we conduct a 50-fold cross-validation-like process to evaluate the general performance without being affected by possible biases introduced by specific training data in certain flows.

VI. Experimental Results

In this section, we present the experimental results of each problem individually, and in order to comprehensively evaluate model performance and select the optimal configuration among different ML algorithms, hyper-parameters, settings addressing class imbalance, and sampling strategies, we divide the model development as well as evaluation process into two stages as follows:

1) **Preliminary model evaluation** focuses on the ML models themselves and aims to uncover the potential of each model, allowing the identification of the most promising algorithm. Notably, models are trained and validated exclusively on the "Campus2020" dataset.
2) **Model performance finalization** aims to finalize and present the ultimate model performance. To achieve this, we exploit the best model determined in the previous stage to train on the entire "Campus2020" dataset without segmentation so that we could include all informative samples. We then test the model on the other two datasets, deriving the class recalls regarding each dataset and all tested data.

The general procedure is similar for both problems, despite some extra operations introduced for the prediction of starting points. Subsequently, we discuss the results, offering valuable insights into the classifications.

A. Problem 1: prediction of starting points

1) Model performance:

a) Preliminary model evaluation: Table III showcases the class recalls of all models based on "Campus2020" for problem 1. Possible settings addressing class imbalance are implemented for each model, while no sampling strategy is utilized. Anomaly detection approaches, as evidenced by their negligible recalls for class 1, completely fail to achieve the objective. DT and RF models perform slightly better but still fall short of effectively addressing the problem, even with class weights. DL models, on the other hand, demonstrate the ability to identify a larger number of starting points but exhibit the worst performance for class 0. Notably, the sophisticated tree-based models, BRF and XGB, demonstrate substantially better results by successfully predicting approximately 50% of the target bins, without excessively penalizing class 0. Consequently, we select XGB for further analysis, due to its outstanding performance and ease of training and tuning. Additionally, Table III also presents class precision and macro average F1-score, providing supplementary insights into the performance evaluation. It is evident that all models exhibit a consistent behaviour, characterized by unexceptionable precision for class 0, inferior precision for class 1, and mediocre F1-score, thereby reinforcing our choice of the evaluation metric. Only XGB and BRF that manifest acceptable recalls for class 1 without significantly penalizing class 0, yield slightly higher precision for class 1. These outcomes are attributable to the dominance of the majority class (lossless bins), which overwhelms the minority (lossy bins) in terms of the predicted positives, and thus leads to a relatively constant F1-score of around 0.5. Notably, the diminished precision for class 1 does not represent a significant issue, as this is an inherent property for imbalanced ML [13], and the performance presented herein are merely from the preliminary model development, not to mention that a portion of the misclassifications are even advantageous, as elucidated in Section VI-A2a.

Regarding the impact of different sampling strategies, we employ the aforementioned multiple-trial examination with the best model (XGB) selected in the previous step on the "Campus2020" dataset, and test the model also on the entire "Campus2023" and "Supervised" datasets[11]. Figure 8 presents box plots depicting the class recalls of all experimental trials for each sampling strategy. Overall, all strategies exhibit similar patterns, with decent and stable performance for class 0 and varying degrees of overfitting for class 1 (higher recalls for the "Campus2020" dataset where the models were trained on). Moreover, the ranges of recalls for class 1 are wide regardless of the sampling strategies, meaning that patterns extracted from training samples in different flows have a huge impact on the final performance, which necessitates the fine-tuning of models and all samples in the entire "Campus2020" in the next step. Regarding each sampling strategy, we find that no sampling results in the best performance for class 0, while hybrid-sampling generates comparatively better performance for class 1. Therefore, we continue the process with both strategies.

b) Model performance finalization: We now consolidate the performance and finalize the result by training on the

[11]Notably, the "Campus2023" and "Supervised" datasets are presented as well to just showcase the unified pattern, and they are not considered for the selection of sampling strategy.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

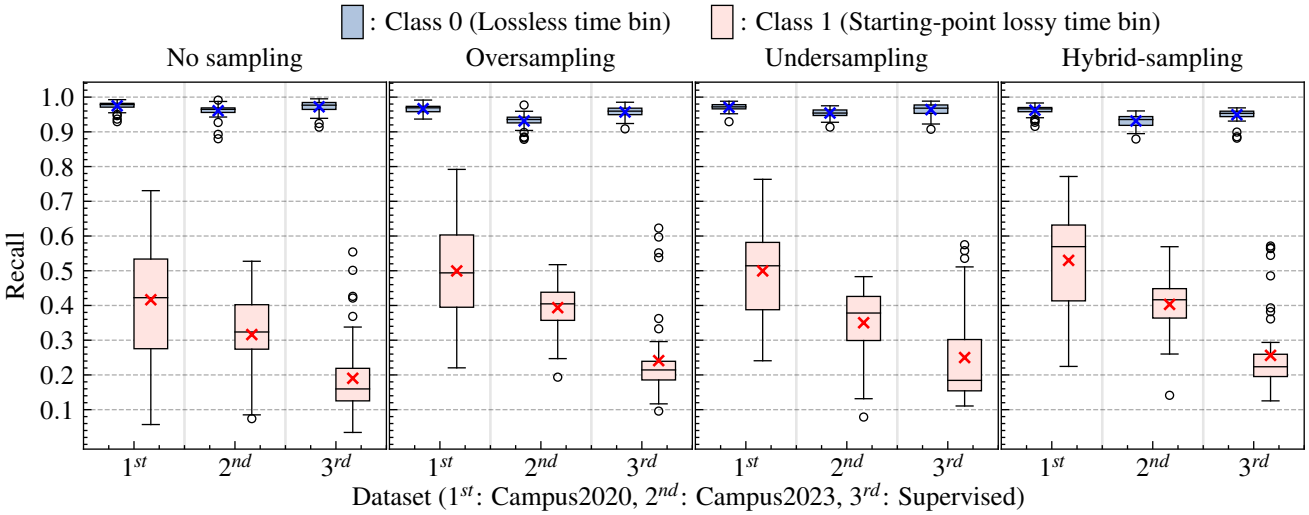


Fig. 8: Class recalls of results based on different sampling strategies for problem 1.

TABLE IV: Final result of models trained on the entire "Campus2020" dataset for problem 1.

Model		XGB (No sampling)		XGB (Hybrid-sampling)	
Test dataset		Campus2023	Supervised	Campus2023	Supervised
Recall for each dataset	**Class 0 (Lossless bin)**	0.960	0.956	0.903	0.915
	Class 1 (Starting point)	0.553	0.454	0.662	0.524
Recall for all test data	**Class 0 (Lossless bin)**	0.959		0.906	
	Class 1 (Starting point)	0.486		0.568	

TABLE V: Amount of correct classifications of lossless time bins in the 5 s before starting-point losses.

Model	XGB (No sampling)		XGB (Hybrid-sampling)	
Dataset	Campus2023	Supervised	Campus2023	Supervised
Occupation of correct prediction [%]	56.6	56.7	39.9	53.3

entire "Campus2020" dataset the models (XGB) with selected sampling strategies, as in Table IV. As expected, no sampling outperforms towards lossless bins by identifying around 96% of class 0, while hybrid sampling generates more balanced results with higher recalls for class 1, reaching up to around 66% of starting points. In both cases, "Campus2023" outperforms the "Supervised" dataset, which is reasonable since it was collected using the same tool at the same location as the training set. On top of that, by successfully predicting the starting point, we actually identify and tackle subsequent concentrated-loss groups of 49% to 57% across all datasets. Although the general prediction for lossy time bins is not ideal, we can reasonably consider the performance acceptable given the difficulty of the problem, let alone the prevention of discernible concentrated losses in the future, which could be more theoretically detrimental.

2) Discussion: Herein, we examine the classification results to gain insights into four aspects: (i) the extent to which we can utilize misclassifications of class 0, (ii) the reason behind our difficulties in predicting lossy time bins, (iii) the distribution of classifications for lossy time bins among RTP flows, and (iv) the performance of the models for sparse losses.

a) Classifications for lossless time bins: In the best-case scenario, we can reach a recall of 96.0% for class 0, but it still yields a significant number of false alarms due to the massive quantity of the majority class per se. However, it is reasonable to assume that identifying lossless time bins approaching packet loss is more challenging than detecting those that are distant from lossy time bins. This is because packet losses could be caused by network problems, e.g., network congestion, that are progressive and escalating instead of instantaneous or ephemeral. These issues could influence the network before the onset of losses and thus impose variations on traffic patterns, which are captured and reflected by the statistical features in our possession, and, in turn, mislead the model to make mistakes. Therefore, we attempt to investigate such phenomena by observing a time window of 5 s (10 lossless time bins) prior to any starting-point losses. Table V indicates the occupancy of correctly classified lossless time bins, i.e., the accuracy of class 0 with respect to all lossless time bins in any target time windows in each experimental result. From 45% to 60% lossless time bins within the defined time window are misclassified as losses. Compared to the overall recalls of class 0 in Table IV, we observe significant performance "drops", ranging from 40% to 50%, which highlights the difficulty of distinguishing between classes when approaching the loss. Nevertheless, these misclassifications could be an asset to some extent, because they could trigger alarms that are theoretically erroneous but unexpectedly effective and beneficial, given that they occur just couple of seconds ahead of the actual losses.

b) Classifications for lossy time bins: We now investigate the reason that accounts for the imperfect prediction for starting-point losses. In general, ML models are data-driven methods that rely on features to tell apart different classes. To facilitate the understanding of features, we aim to visualize their distribution through T-distributed Stochastic Neighbor Embedding (t-SNE) [29], which transforms high-dimensional data to low-dimensional embedding, that can be displayed in a 2D plot. Accordingly, for each model and each dataset finalized in section VI-A1b, we extract all features that correspond respectively to the starting-point losses and the same amount of samples randomly drew from lossless bins. We then convert

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

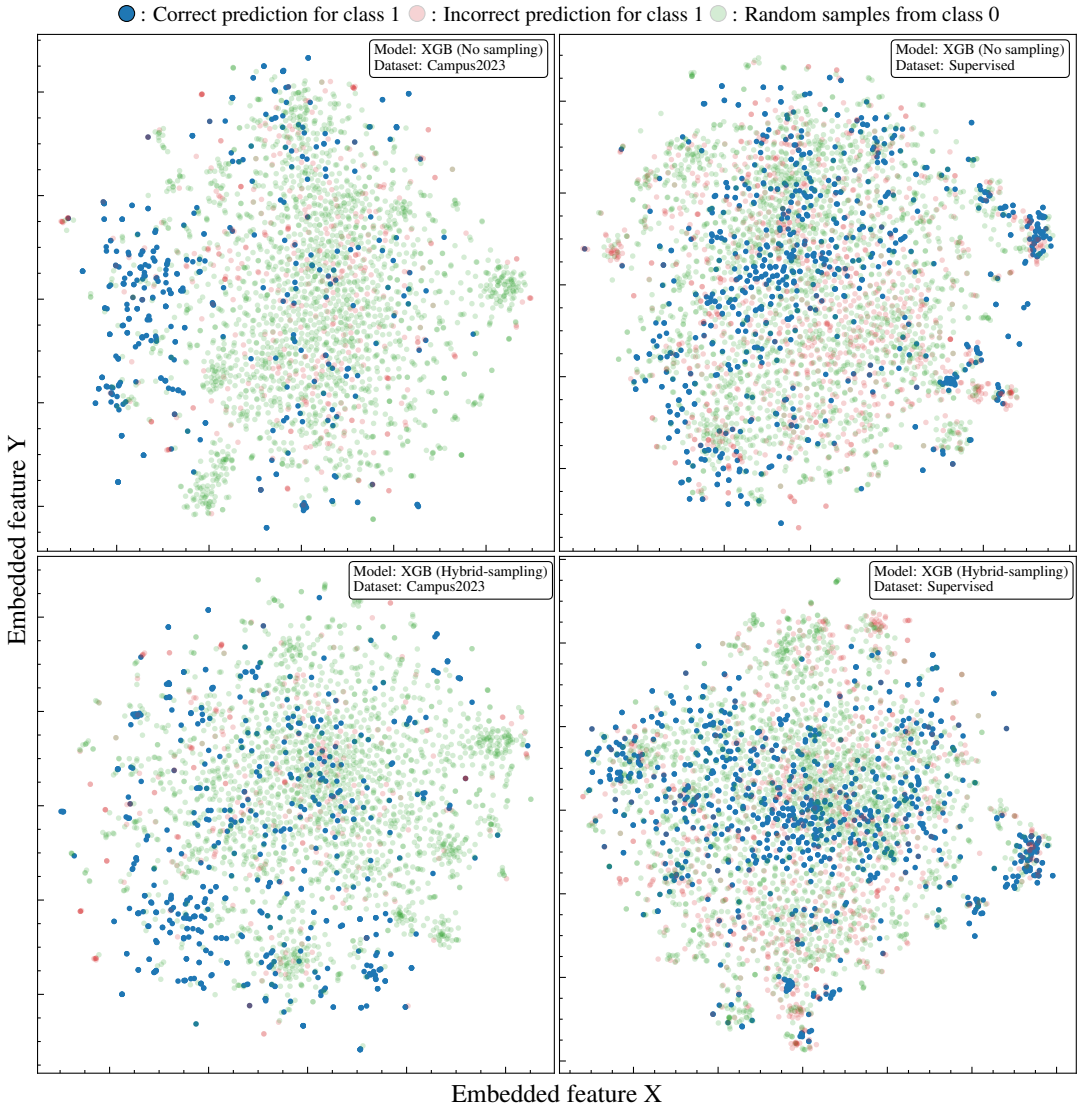


Fig. 9: Results of t-SNE for problem 1: the 2D plot for the embedding of correct prediction (blue dots) as well as incorrect prediction (light red dots) from class 1, and random samples from class 0 (light green dots).

these features into 2D values and plot them in Figure 9. It is important to acknowledge that t-SNE transcends mere naivety as a brutal dimensionality reduction technique, rather, it epitomizes a more sophisticated approach that strives to encapsulate and retain intricate patterns inherent in the original expanse of high-dimensional space. Consequently, the visual representation in a 2D plot serves as a seamless conduit, effortlessly reinstating the originality and authenticity of patterns, thereby facilitating an enhanced comprehension of different traffic features in a visually intuitive manner.

Overall, the light green (class 0) and red (incorrect predictions for class 1) data points exhibit a wide-ranging distribution across the plot, intertwining with one another. Conversely, the blue dots (correct predictions for class 1) are dispersedly positioned along the periphery, readily isolatable within the "Campus2023" dataset, yet displaying a tendency to coalesce into clusters within the "Supervised" dataset. On the one hand, the discernment between class 0 samples and misclassifications from class 1 proves arduous due to their overlapping nature, further compounded by the inclusion of only a limited quantity of lossless time bins, thereby accentuating the exacerbation of overlap with a greater abundance of class 0 samples. On the other hand, the relatively separated agglomerations of class 1 samples, despite several scattered distribution, elucidate the successful predictive capabilities therein. Moreover, the isolation of such samples of "Campus2023" dataset surpasses that of "Supervised" dataset, irrespective of the model employed, as evidenced by the elevated recalls for class 1 in Table IV. Interestingly, a subset of green and red data points appear proximate to the blue clusters, signifying that they are supposed to be either misclassified or correctly classified as class 1. This may stem from our conservative approach in ensuring decent performance for class 0, as opposed to fervently prioritizing the prediction of class 1, thereby fostering the model's sensitivity to subtle discrepancies during classification. Each of the aforementioned facets collectively underscore the inherent challenges posed by this problem, where RTP traffic manifests as dynamic and multifarious, engendering various nearly indistinguishable patterns.

c) Analysis of RTP flows: We proceed to examine the distribution of correct and incorrect classifications across various RTP flows, as depicted in Figure 10, which illustrates the ratio of accurately predicted instances to the total number of starting-point lossy bins for each individual RTP flow. It

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

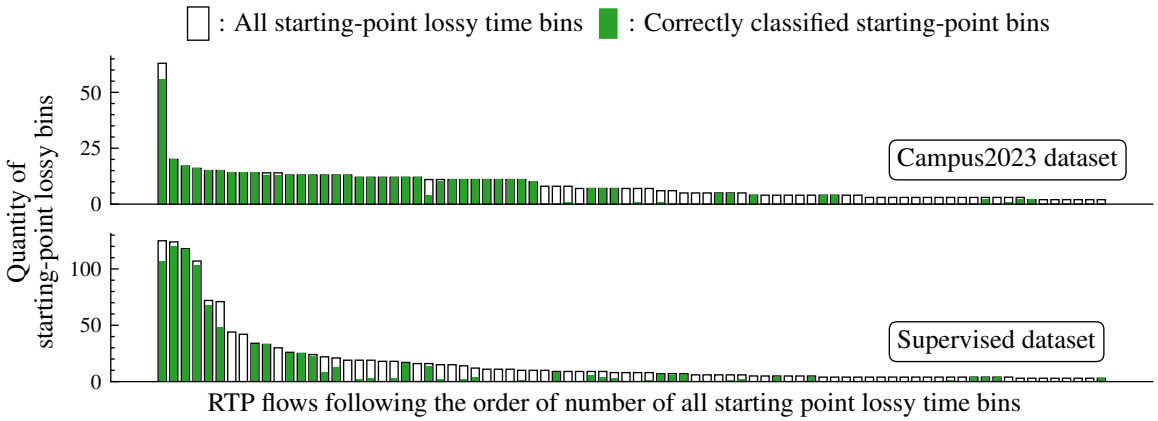


Fig. 10: Quantity of starting-point lossy time bins in each individual RTP flow for problem 1: the overall amount (white bars) and the correctly predicted amount (green bars). The flows (bars with their corresponding flow names omitted to be displayed on the x-axis) are ordered according to their amount of lossy time bins. For simplicity, only the two datasets examined by XGB with hybrid-sampling are presented and only the first 80 flows with relatively more losses are presented. We have noticed that the other model and flows exhibit similar behaviour.

is crucial to highlight that the prediction distribution exhibits a substantial bias for both datasets. The model excels in predicting some flows while performing poorly for others, while only a handful of flows yield mediocre results. This observation implies a unified behavior of packet loss within certain RTP flows, which the model can effectively capture and leverage to achieve reliable predictions consistently. On top of that, this particular phenomenon can facilitate a management strategy that terminates predictions completely in the presence of multiple initial errors. Intriguingly, the first several flows in both datasets, characterized by a significant number of starting-point losses, experience exceptional performance, which could signify a peculiar capability of the model to efficaciously predict RTP streams heavily affected by packet loss.

d) Performance of sparse losses: Previously, sparse lossy time bins were excluded from both model training and testing due to their random and innocuous nature, rendering predictions for them as dispensable. However, to optimize network resources and uphold the current traffic quality, it is preferable to treat sparse lossy bins as lossless bins and refrain from explicitly identifying them. In other words, the model is susceptible to misclassifying sparse losses as starting points, and it is desirable to minimize such misclassifications, aligning with our expectation originated from the strategy of removing sparse losses from the dataset. With this objective in mind, we employ our models to exclusively evaluate sparse losses and present the corresponding accuracy (i.e., $n_{\text{predicted as starting points}}/n_{\text{sparse losses}} \times 100\%$) in Table VI. When comparing these results to the accuracy of starting points (i.e., recall of class 1) presented in Table IV, we observe a discernible decrease in performance of approximately 30%. This implies that our models possess a certain degree of proficiency in discriminating between sparse and concentrated loss patterns. Consequently, it is reasonable to posit that our approaches exhibit greater robustness in predicting starting points that herald concentrated losses, as opposed to sparse losses.

B. Problem 2: classification of different losses

1) Model performance:

TABLE VI: Accuracy of sparse lossy time bins tested by models derived in section VI-A1b.

Model	XGB (No sampling)		XGB (Hybrid-sampling)	
Dataset	Campus2023	Supervised	Campus2023	Supervised
Accuracy of sparse loss [%]	26.8	19.4	34.4	27.8

TABLE VII: Basic results of all ML models for problem 2.

Category	**Model**	**Recall**	
		Class 0 (Sparse loss)	**Class 1 (Starting point)**
Classification (Tree-based)	DT	0.647	0.363
	RF	0.934	0.061
	XGB	0.825	0.184
Classification (Instance-based)	k-NN	0.762	0.310

a) Preliminary model evaluation: Table VII elucidates the class recalls of each model in problem 2. In general, there is a prevailing bias towards class 0 in all models, with their performances exhibiting an inverse relationship: higher recall for class 0 correspond to lower recall for class 1, and vice versa. To this end, we select XGB due to its intermediate performance and flexible configurability in model tuning to achieve varying levels of performance.

b) Model performance finalization: The principal goal entails discerning starting points to the fullest extent feasible, whilst ensuring a satisfactory performance for class 0. However, considering the relatively limited number of samples for class 0 across all datasets, misclassifications are moderately tolerable. This affords us the opportunity to elaborately tune the model to achieve varying levels of performance, either striking a balance between both classes or prioritizing class 1. Consequently, we deliberately train the model to target different optimization directions and present the finalized performance in Table VIII. Focusing on the main objective, the model effectively identifies the majority of sparse lossy time bins while successfully recognizing 24% to 40% of starting points.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

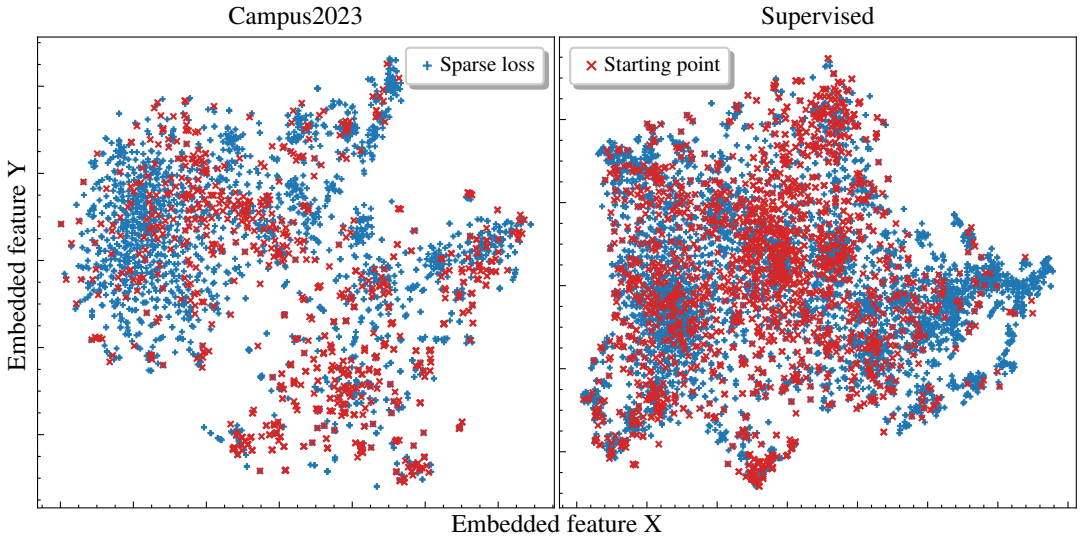


Fig. 11: Results of t-SNE for problem 2: the 2D plot for the embedding of class 0 (sparse lossy time bins, blue plus marker) and class 1 (starting-point lossy time bin, red cross marker).

TABLE VIII: Final result of models trained on the entire "Campus2020" dataset for problem 2.

Model	Test dataset	Recall for each dataset	
		Class 0 (Sparse loss)	Class 1 (Starting point)
Main Objective			
XGB	Campus2023	0.818	0.236
	Supervised	0.849	0.400
Different Optimisation direction (Balanced performance)			
XGB	Campus2023	0.529	0.502
	Supervised	0.620	0.610
Different Optimisation direction (Better performance for class 1)			
XGB	Campus2023	0.263	0.824
	Supervised	0.336	0.862

When striving for balanced performance, the model achieves a 60% recall for both classes in the "Supervised" dataset, but only surpasses random guess performance for the "Campus2023" dataset. In the last case, the performance is exactly opposite to the main objective, but the overall accuracy is below 50% due to the larger number of samples in class 0. Across all cases, the performance on the "Supervised" dataset is substantially superior to that on the "Campus2023" dataset, whose overall performance remains subpar with recalls barely reaching 0.8 on one end but merely achieving around 0.25 on the other end. In contrast, the results of the "Supervised" dataset are sufficiently respectable, i.e., the model can identify 40% of starting points, while only yielding 15% misclassifications. To sum up, our approach provides the possibility to meet different system requirements and demonstrates effectiveness in some specific scenarios.

2) Discussion: Similar to problem 1, we perform an in-depth analysis to inspect the characteristics of different classes and the distribution of correct classifications.

a) Sparse losses vs. Starting points: We now employ t-SNE to visualize the feature distributions of both classes in Figure 11. For both classes in both datasets, the samples are scattered extensively across the plot, resulting in a highly overlapped pattern, which could originate from the shared characteristic of being lossless in the past for both classes. In this vein, the aforementioned model tuning resembles the manipulation of decision boundary in the 2D plot (or hyperplane in multidimensional space) to separate different samples, and due to the mixed nature of both classes, it is barely possible to accurately classify one class while effectively isolating the other one. All of these indicate the difficulty in the distinction between classes and shed light on the inherent challenges in improving the performance.

b) Analysis of RTP flows: Figure 12 shows the distributions of correctly identified starting points in each flow for both datasets. Despite the first 4 flows in "Supervised" dataset with highly accurate classification, the overall distribution is relatively homogeneous, particularly in the case of "Campus2023" dataset. Being different from what have been observed in Figure 10 for problem 1, where flows with more losses are favored, the models in problem 2 are able to generalize across flows. This could also indicate unified patterns of sparse or starting-point losses throughout the traffic. Additionally, the extraordinary behaviour for the first 4 flows of "Supervised" dataset could be attributed to the comparatively decent performance for class 1 with a recall of 0.4.

C. Summary

The finalized results in both problems represent the upper limit of the performance achieved for both classes. However, there is still room for further optimization by fine-tuning the model to attain an intermediate level of performance that can accommodate various network conditions. Nonetheless, it is struggling to improve the performance for both classes simultaneously, regardless of the ML models, techniques addressing imbalance, sampling methods used, and optimisation directions. Based on different model tuning and data manipulation, we can intentionally optimize the performance towards two directions:

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

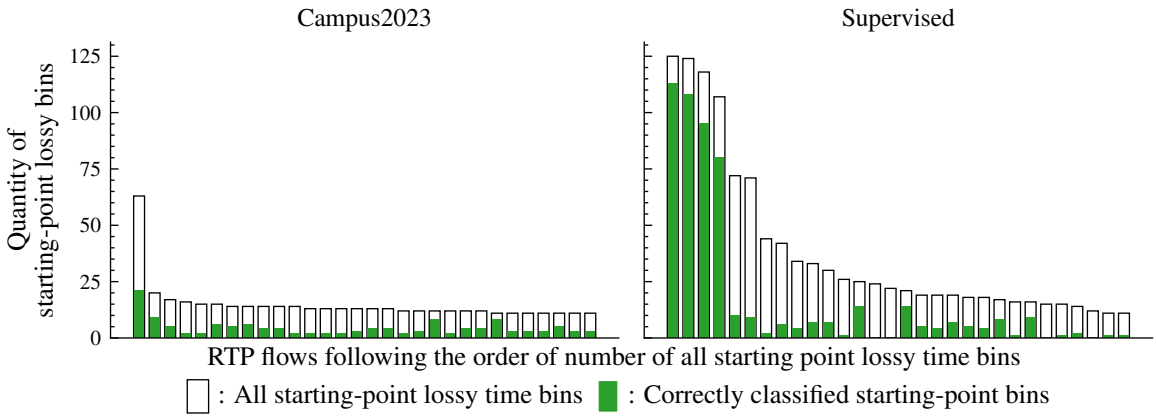


Fig. 12: Quantity of starting-point lossy time bins in each individual RTP flow for problem 2. Note that, only the two datasets regarding the main objective of problem 2 and the first 30 flows are presented.

i) avoiding excessive misclassifications of class 0 (lossless or sparse lossy bins), thereby reducing unnecessary alarms and avoiding wastage of network resources; ii) prioritizing the identification of starting-point bins to unreservedly alleviate the impact of losses and maximize QoE.

Furthermore, it is possible to draw certain comparisons between both problems, as they share the common objective of identifying starting-point losses. Obviously, the performance of class 1 in problem 1 is better even in case of no sampling strategy where the recall is the lowest, let along the promptness of the prediction. However, this approach carries the risk of generating more misclassifications, triggering erroneous events and deteriorating the network observability. In this vein, it is also feasible to adjust the models to achieve an extremely precise performance for class 0 with a recall of 0.99, which, without a doubt, subdues significantly the recognition of losses to a recall lower than 10%. Additionally, we do not restrict the probability of combining both problems, which could parallelize or serialize both approaches to further enhance the network monitoring. By leveraging the strengths of models in each problem, we can potentially achieve more comprehensive and effective network monitoring and management.

VII. Related Work

Numerous studies have investigated the packet loss in RTC and analyzed its impact on QoE [30], [31], [32], [33]. Packet loss can significantly degrade the perceived quality of multimedia streaming, and users are more sensitive to packet loss than to delay variation. In order to mitigate the effects of packet loss, various techniques have been proposed in the literature. Traditionally, RTC applications address packet loss relying on Forward Error Correction (FEC), which is a straightforward and effective method that appends redundant data to each packet, allowing the receiver to recover lost packets. However, it requires additional bandwidth to transmit the surplus data and can introduce additional latency. Furthermore, packet loss concealment (PLC), that endeavors to mask the effects of packet loss and reconstruct the missing ones based on the information gleaned from the preceding packets, has been introduced for real-time audio transmission. It can be supported by mathematical approaches or ML models [34], [35], but the reconstructed audio may still exhibit artifacts or distortions. Holistic systems that cope with packet losses also exist in the literature. For example, [36] proposed a technique known as early packet loss feedback (EPLF), where the MAC layer of the local WiFi link sends a spoofed NACK to the RTP layer. In [37], the authors proposed an adaptive hybrid NACK/FEC method with temporal layers to handle packet loss in WebRTC. Notwithstanding the efficacy of their proposed solutions, packet losses still endure. Moreover, the measurement of QoE metrics concerning packet loss also plays an pivotal role. Conventional methods of active or passive measurement for losses can estimate the loss rate using probe packets or Management Information Base (MIB) [38], [39], but they cannot disclose actual losses within particular end-to-end RTP flows and exhibit inadequate accuracy in multiple scenarios. The authors of [40], [41], [42], [43], [44] have developed various sophisticated technologies to access QoE metrics for prevalent videoconferencing applications. Nonetheless, they have merely touched upon or obscured the aspect of packet loss measurement, declaring this limitation in their respective works. On top of that, measuring packet loss serves as a monitoring tool, operating untimely post the incidence of a loss. To this end, there is a growing interest in the prediction and preemptive avoidance of packet loss for RTC.

To the best of our knowledge, only a limited number of studies have attempted to predict packet loss. Some of them apply mathematical methods that leverage various network metrics. For instance, [45] focused on the measurements of end-to-end delay variation, and developed an approach to predict packet loss and congestion for audio streams. Moreover, the authors of [46] derived a mathematical model with parameters defined by linear regression, based on simulated data. Their objective was to improve video codec and perform video loss prediction in wireless networks. In contrast to our real-time system, a lightweight ML model that estimates the quantity of retransmitted packets in bulky TCP flows, was proposed to predict retransmissions a-posteriori on a per-flow basis in [47]. On a different note, the authors of [48] adopted a Decision Tree and Logistic Regression models to predict packet losses for Call Admission Control (CAC) systems. Based on per-flow traffic metrics, they sought to predict a binary variable denoting "high" or "low" packet loss. Diverging from the prediction the occurrence of packet loss, authors of [49] developed a hidden Markov model to predict the loss rate in a short-term future, by initially estimating the distribution of losses, and then regarding

the expected value as the prediction target of rate. Unlike our approach, which leverages various network features, they solely relied on loss distributions.

In this context, it appears that packet loss prediction remains relatively understudied, with scant endeavors in the literature demonstrating the viability of network loss prediction solely based on network statistics, not to mention that our work represents the pioneering effort that sheds light on the onset of loss bursts. Throughout our study, we focused exclusively on the characteristics extracted from the network packets to perform the prediction, thus not taking into account all additional factors such as: the resources of the involved machines, queue occupancy, bandwidth, etc. The primary motivation driving this work is, therefore, related in the first instance to a feasibility test, where, secondly, an eventually high accuracy in loss prediction, would open up the potential to develop an extension for a monitoring software capable of actively scanning the network condition. While numerous works employ simulated data, our research is built upon extensive datasets comprising enormous amount of real RTC traffic gathered from various sources in multiple scenarios, lending our work a heightened level of robustness and practicality. Meanwhile, our approach only demands the statistics of received packets collected on one end, either in network infrastructures or on application level of user sides, empowering a highly versatile and implementable system. More importantly, most existing works forecast packet loss on a per-flow basis, whereas our proposed solution operates at a fine-grained granularity based on 500 ms time bins per flow, enabling the implementation of our system in real-time and yielding enhanced temporal resolution. Due to the substantial disparity in problem formulation and data management between our work and others, a direct numerical comparison of results is unfortunately not feasible.

VIII. Conclusion and Future Work

In this paper, we investigate the phenomena of packet loss in RTC traffic through a comprehensive analysis based on three real-world datasets. We reveal a common pattern of packet loss aggregation that allows for the identification of the initial loss of a burst, thereby addressing subsequent ones. To this end, we propose a ML-based system with two operational scenarios that either predicts the occurrence of starting points or distinguishes from sparse losses on a time-bin level, using traffic statistics. Furthermore, through extensive experimentation with various ML algorithms and specific techniques, as well as conducting in-depth analyses, we confirm that classifying packet loss is an exceptionally challenging task, and it is barely possible to improve both classes simultaneously. However, we emphasize the generalization capabilities of our ML models, regardless of time periods and scenarios. Finally, we demonstrate the flexibility of our solutions to accommodate different network requirements through the possibility of paying a trade-off between minimizing errors and prioritizing predictions, which partially conquers the relatively unpredictable nature of packet loss in RTC. Overall, our findings shed light on the intricacies of packet loss in RTC and provide ML-based solutions that offers insights and potential strategies for mitigating its impact.

As future work, we plan to specify and distinguish different scenarios by collecting data using various applications from different channels, such as mobile networks. Additionally, we aim to eliminate the constraint of time bins and shift our focus towards a per-packet level. In summary, our work serves as a crucial stepping stone towards the development of a sophisticated network management system, designed to gather fine-grained information and enable optimization policies to improve QoE perceived by users.

Acknowledgements

This work was supported by Cisco Systems Inc., the SmartData@PoliTO center on Big Data and Data Science and the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART", Focused Project R4R).

References

[1] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, O. Hohlfeld, and G. Smaragdakis, "The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic," in *Proceedings of the ACM Internet Measurement Conference*, IMC '20, (New York, NY, USA), p. 1–18, Association for Computing Machinery, 2020.

[2] C. Athanasiadou and G. Theriou, "Telework: systematic literature review and future research agenda," *Heliyon*, vol. 7, no. 10, p. e08165, 2021.

[3] A. Nistico, D. Markudova, M. Trevisan, M. Meo, and G. Carofiglio, "A comparative study of RTC applications," in *2020 IEEE International Symposium on Multimedia (ISM)*, pp. 1–8, IEEE, 2020.

[4] R. Frederick, S. L. Casner, V. Jacobson, and H. Schulzrinne, "RTP: A Transport Protocol for Real-Time Applications." RFC 1889, Jan. 1996.

[5] D. Vucic and L. Skorin-Kapov, "The impact of packet loss and google congestion control on QoE for WebRTC-based mobile multiparty audiovisual telemeetings," in *International Conference on Multimedia Modeling*, pp. 459–470, Springer, 2019.

[6] G. Carofiglio, G. Grassi, E. Loparco, L. Muscariello, M. Papalini, and J. Samain, "Characterizing the relationship between application qoe and network qos for real-time services," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Network-Application Integration*, pp. 20–25, 2021.

[7] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (WebRTC)," in *Proceedings of the 7th International Conference on Multimedia Systems*, pp. 1–12, 2016.

[8] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "Forward error correction (FEC) building block," tech. rep., 2002.

[9] G. Perna, D. Markudova, M. Trevisan, P. Garza, M. Meo, M. M. Munafò, and G. Carofiglio, "Real-time classification of real-time communications," *IEEE Transactions on Network and Service Management*, 2022.

[10] D. Markudova, M. Trevisan, P. Garza, M. Meo, M. M. Munafò, and G. Carofiglio, "What's my App?: ML-based classification of RTC applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 48, no. 4, pp. 41–44, 2021.

[11] G. Perna, D. Markudova, M. Trevisan, P. Garza, M. Meo, and M. M. Munafò, "Retina: An open-source tool for flexible analysis of rtc traffic," *Computer Networks*, vol. 202, p. 108637, 2022.

[12] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[13] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–36, 2019.

[14] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

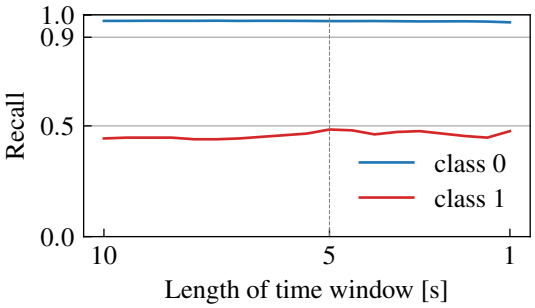
[15] Z. Liu, W. Cao, Z. Gao, J. Bian, H. Chen, Y. Chang, and T.-Y. Liu, "Self-paced ensemble for highly imbalanced massive data classification," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 841–852, IEEE, 2020.

[16] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A comparative study on online machine learning techniques for network traffic streams analysis," *Computer Networks*, vol. 207, p. 108836, 2022.

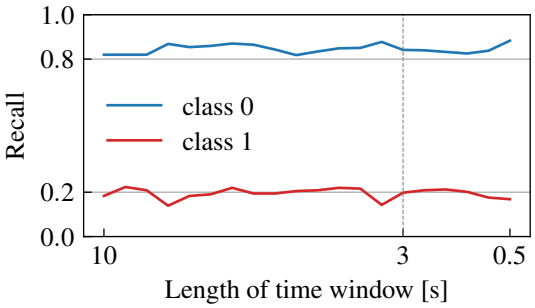
[17] B.-H. Oh, S. Vural, N. Wang, and R. Tafazolli, "Priority-based flow control for dynamic and reliable flow management in sdn," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1720–1732, 2018.

[18] N. Thazin, K. M. Nwe, and Y. Ishibashi, "End-to-end dynamic bandwidth resource allocation based on qos demand in sdn," in *2019 25th Asia-Pacific Conference on Communications (APCC)*, pp. 244–249, IEEE, 2019.
[19] R. F. Ghani and L. Al-Jobouri, "Packet loss optimization in router forwarding tasks based on the particle swarm algorithm," *Electronics*, vol. 12, no. 2, p. 462, 2023.
[20] A. Kannan, S. Vijayan, M. Narayanan, and M. Reddiar, "Adaptive routing mechanism in sdn to limit congestion," in *Information Systems Design and Intelligent Applications: Proceedings of Fifth International Conference INDIA 2018 Volume 1*, pp. 245–253, Springer, 2019.
[21] M. Mellia, A. Carpani, and R. Lo Cigno, "Tstat: Tcp statistic and analysis tool," in *Quality of Service in Multiservice IP Networks: Second International Workshop, QoS-IP 2003 Milano, Italy, February 24–26, 2003 Proceedings*, pp. 145–157, Springer, 2003.
[22] J. Fan, J. Xu, M. Ammar, and S. Moon, "Cryptography-based prefix-preserving anonymization," *URl: http://www. cc. gatech. edu/computing/Telecomm/projects/cryptopan/(cit. on p. 58)*.
[23] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Rtp: A transport protocol for real-time applications," July 2003. RFC 3550.
[24] M. Maruschke, O. Jokisch, M. Meszaros, and V. Iaroshenko, "Review of the opus codec in a webrtc scenario for audio and speech communication," in *Speech and Computer: 17th International Conference, SPECOM 2015, Athens, Greece, September 20-24, 2015, Proceedings 17*, pp. 348–355, Springer, 2015.
[25] J.-M. Valin, K. Vos, and T. Terriberry, "Definition of the opus audio codec," tech. rep., 2012.
[26] C. Chen and L. Breiman, "Using random forest to learn imbalanced data," *University of California, Berkeley*, 01 2004.
[27] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.
[28] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
[29] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.
[30] A. Laghari, R. Laghari, A. Wagan, and A. Umrani, "Effect of packet loss and reorder on quality of audio streaming," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 7, no. 24, 2019.
[31] Š. Mrvelj and M. Matulin, "Impact of packet loss on the perceived quality of udp-based multimedia streaming: a study of user quality of experience in real-life environments," *Multimedia systems*, vol. 24, no. 1, pp. 33–53, 2018.
[32] J. Frnda, M. Voznak, and L. Sevcik, "Impact of packet loss and delay variation on the quality of real-time video streaming," *Telecommunication Systems*, vol. 62, no. 2, pp. 265–275, 2016.
[33] Q. Dai and R. Lehnert, "Impact of packet loss on the perceived video quality," in *2010 2nd International Conference on Evolving Internet*, pp. 206–209, IEEE, 2010.
[34] C. A. Rodbro, M. N. Murthi, S. V. Andersen, and S. H. Jensen, "Hidden markov model-based packet loss concealment for voice over ip," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1609–1623, 2006.
[35] B.-K. Lee and J.-H. Chang, "Packet loss concealment based on deep neural networks for digital speech transmission," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 2, pp. 378–387, 2015.
[36] L. Ma, W. Chen, D. Veer, G. Sternberg, W. Liu, and Y. Reznik, "Early packet loss feedback for webrtc-based mobile video telephony over wi-fi," 2015.
[37] S. Holmer, M. Shemer, and M. Paniconi, "Handling packet loss in WebRTC," 2013.
[38] A. Miyamoto, K. Watanabe, and K. Ikeda, "Packet loss rate estimation with active and passive measurements," in *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1–4, IEEE, 2012.
[39] P. Barford and J. Sommers, "A comparison of probe-based and router-based methods for measuring packet loss," *submission,(see http://www. cs. wisc. edu/pb/publications. html)*, 2003.
[40] T. Sharma, T. Mangla, A. Gupta, J. Jiang, and N. Feamster, "Estimating webrtc video qoe metrics without using application headers," *arXiv preprint arXiv:2306.01194*, 2023.
[41] A. Choi, M. Karamollahi, C. Williamson, and M. Arlitt, "Zoom session quality: A network-level view," in *International Conference on Passive and Active Network Measurement*, pp. 555–572, Springer, 2022.
[42] O. Michel, S. Sengupta, H. Kim, R. Netravali, and J. Rexford, "Enabling passive measurement of zoom performance in production networks," in *Proceedings of the 22nd ACM Internet Measurement Conference*, pp. 244–260, 2022.
[43] K. MacMillan, T. Mangla, J. Saxon, and N. Feamster, "Measuring the performance and network utilization of popular video conferencing applications," in *Proceedings of the 21st ACM Internet Measurement Conference*, pp. 229–244, 2021.
[44] H. Chang, M. Varvello, F. Hao, and S. Mukherjee, "Can you see me now? a measurement study of zoom, webex, and meet," in *Proceedings of the 21st ACM Internet Measurement Conference*, pp. 216–228, 2021.
[45] L. Roychoudhuri and E. S. Al-Shaer, "Real-time analysis of delay variation for packet loss prediction," 2004.
[46] J. V. C. Carmona, E. M. C. de Matos, B. S. L. Castro, F. J. B. Barros, M. C. de Alcântara Neto, and E. G. Pelaes, "Video loss prediction model in wireless networks," 2019.
[47] A. Giannakou, D. Dwivedi, and S. Peisert, "A machine learning approach for packet loss prediction in science flows," *Future Generation Computer Systems*, vol. 102, pp. 190–197, 2020.
[48] D. K. Saha and T. Ghosh, "Study of packet loss prediction using machine learning," 2020.
[49] F. S. Filho and E. de Souza e Silva, "A method for predicting packet losses with applications to continuous media streaming," 2006.

APPENDIX



(a) Problem 1: prediction of starting points



(b) Problem 2: classification of different losses

Fig. 13: The class recalls regarding different durations of considered time window in terms of "Campus2020" dataset.

A. Information of time window

We also offer insights into different lengths of the time window for both problems. In particular, we construct the dataset of "Campus2020" in the same way, but instead of confining to 11 time bins within the preceding 5 s and the present moment for both problems, we extend our temporal consideration to include an additional 10 time bins in the more distant 5-second interval. Subsequently, we leverage all available statistical features in each time bin, investigating the evolution of performance as we decrease the time window length, by training an XGB model iteratively. Notably, we still adhere to the original arrangement of skipping one time bin for problem 1, and utilizing the current bin for problem 2. Such a process can be regarded as a preliminary assessment for determining time window durations.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3442616

Figure 13 illustrates the performance variation concerning class recalls as duration decreases (i.e., fewer features). For problem 1, the general behaviour is stable, with a marginal enhancement observed for class 1 initially and a slight decrement for class 0 throughout the process (longer durations favor class 0, albeit with negligible performance gains depicted in the plot). Evidently, the 5-s time window yields an enhanced recall for starting points without compromising lossless bins. Despite that certain shorter durations (e.g., 4.5 s) produce similar recalls for class 1, we still advocate for 5 s with superior performance, which possesses higher potential for the model development. As for problem 2, the pattern appears more erratic, lacking discernible trends. Consequently, we opt for the relatively shorter 3-s time window with a balance between the performance of both classes. Note that problem 2 aims to distinguish between starting points and sparse losses, and unlike problem 1 where a gap is reserved to consider only remote features, the current (target) time bin, in which the loss is present, is available for the prediction. Thus, we envision insignificant contributions from time bins on the far side. To sum up, both analyses demonstrate that the selection of time window duration holds no paramount importance, particularly given our employment of an automatic feature selection process.

Tailai Song obtained the B.Sc. in Automotive Engineering at Politecnico di Torino in 2020 and the M.Sc. in ICT for Smart Societies at Politecnico di Torino in 2022. Currently, he is a PhD student in the Telecommunication Networks Group (TNG) from Department of Electronics and Telecommunications (DET) at Politecnico di Torino (PoliTO), Italy, and also a member of the SmartData@Polito research centre. His research focuses on machine learning techniques applied to real-time communications to improve Quality of Experience (QoE) and the objective of full-stack observability through end-to-end telemetry.

Gianluca Perna is a PhD student at Politecnico di Torino and member of SmartData@Polito research center for Big Data technologies. He obtained a Bachelor's degree in Telecommunication engineering and a Master's degree in ICT For Smart Societies with an excellent grade, both in the same University. Thanks to his passion and expertise in Machine Learning, Internet of Things and Data analysis he obtained a six months research grant before starting his PhD in the SmartData group. He is also pursuing a patent in the field of Building Design and participating in an international project with the leading IT company Cisco Systems.

Paolo Garza received the master's and PhD degrees in computer engineering from the Politecnico di Torino. He has been an associate professor at the Dipartimento di Automatica e Informatica, Politecnico di Torino, since December 2018. He spent three years as an assistant professor at Politecnico di Milano. He coauthored about 100 papers in the areas of data mining and machine learning. His current research interests are in the fields of data mining, database systems, and big data analytics. He has worked on classification, clustering, itemset mining and scalable algorithms.

Michela Meo is a Professor of Telecommunication Engineering with the Politecnico di Torino. She coauthored about 200 papers, 80 of which on international journals. She edited a book Green Communications (Wiley) and several special issues of international journals. Her research interests include green networking, energy-efficient mobile networks and data centers, Internet traffic classification, and characterization. Prof. Meo was an Associate Editor of ACM/IEEE Transactions of Networking, Green Series of the IEEE Journal on Selected Areas of Communications Networking and IEEE Communication Surveys and Tutorials. She is a Senior Editor of IEEE Transactions on Green Communications. In the role of a General or Technical Chair, she has led the organization of several conferences, including ITC, ICC symposia, ISCC. She chairs the International Advisory Council of the International Teletraffic Congress. She was the Deputy Rector of Politecnico di Torino from March 2017 to March 2018.

Maurizio Matteo Munafò is Assistant Professor at the Department of Electronics and Telecommunications of Politecnico di Torino. He holds a Dr.Ing. degree in Electronic Engineering since 1991 and a Ph.D. in Telecommunications Engineering since 1994, both from Politecnico di Torino. He has coauthored about 80 journal and conference papers in the area of communication networks and systems. His current research interests are in simulation and performance analysis of communication systems and traffic modeling, measurement, and classification.