POLITECNICO DI TORINO
Repository ISTITUZIONALE

Monitoring Web QoE in Satellite Networks from Passive Measurements

Original
Monitoring Web QoE in Satellite Networks from Passive Measurements / Perna, G., Trevisan, M., Giordano, D., Perdices, D., Mellia, M.. - (2024), pp. 669-676. (2024 IEEE 21st Consumer Communications & Networking Conference (CCNC) Las Vegas (USA) 06-09 January 2024) [10.1109/ccnc51664.2024.10454720].

Availability:
This version is available at: 11583/2987743 since: 2024-04-11T15:40:40Z

Publisher:
IEEE

Published
DOI:10.1109/ccnc51664.2024.10454720

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

18 November 2026

Monitoring Web QoE in Satellite Networks from Passive Measurements

Gianluca Perna†, Martino Trevisan‡, Danilo Giordano†, Daniel Perdices⋆, Marco Mellia†
†Politecnico di Torino, ‡University of Trieste, ⋆Universidad Autónoma de Madrid
first.last@{polito.it,dia.units.it,uam.es}

***Abstract*—Satellite Communication (SatCom) is the only choice to access the Internet in remote regions and is characterized by extreme latency and constrained capacity. For SatCom operators, it is thus fundamental to monitor the Quality of Experience (QoE) of subscribers, to measure their satisfaction, spot anomalies and optimize the peculiar network setup. The Web has become the primary source of Internet content, and Web browsing is the main activity of internauts. This paper addresses the challenge of monitoring Web QoE in SatCom environments, proposing a tailored system that employs a supervised approach to predict Web QoE using passive measurements. The system collects training data through Test Agents that mimic real subscribers' traffic patterns and uses them to build Machine Learning (ML) models that predict performance metrics. The findings demonstrate the feasibility of monitoring Web QoE in SatCom environments, with limitations on website applicability and temporal stability. The need for periodic data generation and the development of a general machine learning model for unseen websites remain open challenges. This research contributes to enhancing web browsing experiences in SatCom and expanding understanding of Web QoE monitoring in diverse network settings.**

***Index Terms*—Satellite Communications, QoE, Web Performance, SpeedIndex - OnLoad, Machine Learning, Regression.**

I. Introduction

Satellite Communication (SatCom) offers Internet connectivity where traditional infrastructures are too expensive to deploy, including rural areas and the territory of underdeveloped countries. Here, we focus on the GEO SatCom technology, which relies on satellites positioned in geostationary orbits. Propagation delay makes the Round Trip time (RTT) higher than 550ms [1] so that the substantial hurdle of high link latency significantly impairs traditional interactive browsing experiences [2], [3], [4]. For instance, the download of a webpage can take several seconds.

In this context, monitoring the Quality of Experience (QoE) subscribers obtain becomes a crucial factor [5], [6], [7], and it would allow the SatCom provider to detect anomalies, plan network upgrades, and optimize management policies. Because quantifying Web QoE remains a formidable challenge due to its subjective nature, proxy quality metrics like the "onLoad" (or Page Load Time) and "Speed Index" are used as indirect measures within web browsers [8]. Recent research efforts have focused on proposing unsupervised [9], [10] and supervised [11], [12], [13] approaches for measuring web browsing QoE in traditional networks. However, these approaches cannot be directly used for SatCom scenarios. This is especially true because operators deploy middleboxes

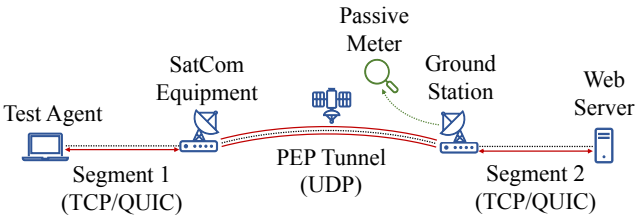


Fig. 1: Test Agent and Passive Meter deployment scenario.

to mitigate the impact of high latency, i.e., Performance Enhancing Proxies (PEPs) [14]. The impact of PEPs on QoE is not clear, and the modification to the traffic they cause challenges traditional QoE estimators.

This paper aims to fill this gap. We propose a system to monitor Web QoE specifically designed to address the challenges unique to SatCom. We employ a supervised Machine Learning (ML) approach to predict Web QoE from in-network passive measurements and explore the different factors that affect prediction accuracy. Our system gathers training data through Test Agents, which automatically visit the website to monitor. Ingenuity is needed to design such Test Agents, as they need to generate traffic patterns similar to real subscribers' traffic and we illustrate the most significant challenges to achieving this goal. In addition, the complex nature of Web traffic requires ad-hoc approaches to construct meaningful features. After designing a proper ML pipeline to predict the QoE of a given website, we explore the feasibility of using a single ML model for all websites. Our results are negative and show that a website-specific model must be considered. At last, we explore the model drift with time and propose a continuous learning solution to address this problem.

Our findings demonstrate the feasibility of monitoring Web QoE in SatCom environments, albeit with some limitations. Notably, this approach works well with the subset of websites whose pages include objects from multiple domains. On the contrary, for websites hosted on a single infrastructure, the prediction accuracy remains limited. Periodic generation of training data becomes essential to adapt to evolving web dynamics, and it remains an open challenge to build a general ML model able to predict the QoE of websites unseen at training time.

II. Problem Statement

Our goal is to design and implement a monitoring system that uses passively collected measurements to estimate

the Web QoE of SatCom users. We target GEO operators, which rely on one or a few satellites positioned at a fixed distance from Earth, providing continuous coverage over large geographic areas. In GEO SatCom, operators typically deploy PEPs to improve performance on the satellite segment. In particular, PEPs are designed to overcome TCP limitations in high-latency scenarios. For this, the PEP transparently manipulates TCP connections. Referring to Figure 1, the user's SatCom home gateway impersonates the server for all TCP connections initiated by the end-user devices (Segment 1 in the Figure). Acting as TCP proxy, the home gateway buffers the TCP data stream and forwards it to the operator's ground station via a bidirectional UDP tunnel (PEP Tunnel in the Figure). Complex reservation and scheduling algorithms decide how to share the SatCom link capacity among active users. The ground station receives the tunneled traffic and acts as a second TCP proxy. It opens a new TCP connection toward the actual server to download and store the responses before forwarding them to the home gateway through the shared satellite link (Segment 2 in the Figure). The PEP complicates the collection of passive in-network measurement, as it splits each TCP flow into two sections. A passive probe installed on the ground station will then only observe Segment 2 – i.e., the TCP connection the ground station PEP opens –, and not the actual traffic as received by the end-user device. As such, any QoS metric collected for TCP (such as packet loss, throughput, or RTT) might not be representative of the end-user experience. In the case of QUIC (over UDP), PEPs operate in a slightly different fashion: Packets are tunneled via UDP, but the home gateway cannot impersonate the server as QUIC header encryption prevents middleboxes from manipulating connections.

We depict our deployment scenario in Figure 1. We control a number of Test Agents that are connected as regular end-user devices. At the same time, a passive probe collects measurements at the ground station, where it observes the traffic of all subscribers. The passive monitor collects per TCP and per UDP flow summaries with rich statistics. Here we rely on Tstat [15] which exports more than 100 features per each TCP/UDP flow, including the server name (as recovered from Server Name Indication (SNI) in TLS Header), the RTT, the total number of packets and bytes exchanged from the server to the client and vice versa, the size and timing of each flow first packets, etc. Notice that the observed timings radically differ from the time at which packets are transmitted by/delivered to the subscriber's devices as packets are collected after/before they travel the satellite segment. We assume that each subscriber is uniquely identified by their subscriber IP address.

Using passive measurements, we design a system to estimate the Web QoE of subscribers using ML models that map flow-level and packet-level features to quantitative metrics. As target metrics, we consider well-established measures that are correlated with users' subjective QoE. We contemplate:

- **onLoad**: The time when the browser fires the `onLoad` event, i.e., when all objects of the page, (images, stylesheets, javascript, etc.), have been downloaded and processed;
- **Speed Index**: Proposed by Google[1], it represents the time at which the visible part of the page rendering is completed. It is computed by tracking the visual progress of the page rendering over time.

We use *Test Agents* to periodically collect QoE measurements resulting from automated visits to a list of monitored websites. The test agents access the internet via the Satellite modem offered by the operator with a standard subscription.

III. System Design

A. Test Agent Design

A Test Agent aims to emulate the behavior of a regular user by visiting a set of websites to gather the necessary QoE metrics for training the ML models. The operator chooses the list of websites to include websites of particular interest, such as business-related portals.

A Test Agent consists of dedicated physical machines connected as a regular end-user device. It uses a browsing automation suite (the dockerized version of Browsertime[2] in our case) to automatically visit webpages. We assume Test Agents continuously operate and seek to maximize (i) the diversity within collected data and (ii) the size of the data available to train ML models. Creating an accurate and realistic training set requires cleverness and involves overcoming various pitfalls, here detailed.

Our Test Agent takes into account three aspects that have been recently shown to largely impact automatic web experimentation: The need to include internal pages of a target website; The need to accept the cookie policies; The need to visit pages with warm browser caches.

In fact, recent literature [16], [13] has shown that it is necessary to include websites' internal pages when running any kind of web testing. Thus, for each website, Test Agents visit at least 10 internal pages chosen to maximize diversity. Notice that several approaches can be used to automate this task. Here, we opt for manually defining such a list.

Second, the presence of Privacy Banners (also known as Cookie Walls) impairs automated navigation[17]. If not specifically instrumented, the Test Agents will always visit a website as a "first visit" so that the website will show the privacy banner and will not download any third-party elements that require the user to accept the cookie policy first. Given 95% of users simply click on "accept all cookies and policies" [17], here we create a custom Javascript script to accept the Privacy Banner and continue the navigation just like a normal user would do.

Third, the presence of a browser cache radically impacts the resulting network traffic. Thus, we run repeated visits to webpages, so that subsequent visits occur with a populated browser cache.

For the experiments in this paper, we use two Test Agents, focusing on 10 websites. We chose them among the most visited ones by the operator's subscribers. These include top websites for e-commerce, news, search engines, and adult

[1]https://web.dev/speed-index/

[2]https://www.sitespeed.io/documentation/browsertime/

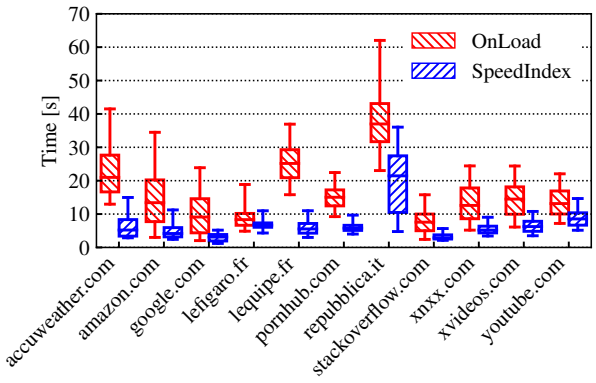


Fig. 2: BoxPlot OnLoad vs SpeedIndex

videos. In building such a list, we exclude those websites for which user login is needed to consume most of the website's content (e.g., online social networks). The website list can be derived from Figure 2. For each website, we select p=5 representative internal pages. We run a first measurement campaign in September 2022 and a second one from December 2022 to February 2023. Each campaign includes about 2 000 visits to each website and page. In Figure 2, we show the distribution of QoE metrics in the form of boxplots. The boxes represent the Inter-Quartile range, while the whiskers span from the 5^{th} to the 95^{th} percentile. The central stroke represents the median. Observe how different are the metrics on each website, and how the onLoad and SpeedIndex are in the order of tens of seconds and radically higher than those seen on wired or mobile networks [18]. This testifies to the peculiar SatCom extreme latency, which impairs the browsing experience.

B. Feature Engineering

Next, we propose to follow a domain expert-driven approach to engineer specific features that are highly correlated with the target QoE metrics. As done in the previous works [19], [12], we focus on time-related and volume-related metrics that we extract from the flow-level measurements. For this, to gather passive measurements, we deploy a Tstat passive probe in between the operator's ground station and the internet access, where we continuously collect flow-level data. Our system is designed to be scalable. Once data is collected at the probe, records of different subscribers can be processed independently, making our architecture amenable to parallelization on large computing infrastructures.

To build the training set for ML models, we first join the passive data (i.e., flow records collected at the operator's ground station) with Test Agent data (i.e., the target metrics onLoad and Speed Index). We join the Test Agent and passive data records using subscriber IP addresses as key.

Referring to Figure 3, we identify the beginning of a new webpage visit when a client contacts the domain of one of the target websites. To this end, we exploit the observation of a flow with the contacted domain name. We call this event *Trigger*. It serves as the observation initial point for building the features. We mark those with thick arrows in Figure 3. We refer to the *Trigger Flow* as f_0.

After the *Trigger*, we open an *Observation Window* of $\Delta T = 30$ seconds during which we extract the information on the first $n = 10$ packets of the first $k = 5$ flows. The rationale behind this choice is twofold. First, the webpage rendering process entails contacting different servers to download all webpage objects. Thus, we want ML models to leverage information from the group of *Related Flows* immediately subsequent to the visit start. Second, we want to minimize the impact of caching and persistent HTTP connection that allows the same flow to download multiple objects (i.e., when a user scrolls a page, the browser downloads new images and material using the same previously opened TCP connections). For this, we extract features only from the first $n = 10$ packets of a flow to include uniquely the first instants of the communication. In the following, we refer to Related Flows as $f_1, \ldots, f_k$.

We engineer features using the information available in the first n packets of the *Trigger* flow f_0 and *Related Flows* $f_1, \ldots, f_k$. Given a flow f, we denote with $c_{i,f}$ and $s_{i,f}$ the i^{th} packet on the client and server side, respectively. Each packet $c_{i,f}$ (or $s_{i,f}$) is characterized by its size $|c_{i,f}|$, its timestamp $t(c_{i,f})$ and the value of its Time-To-Live field $TTL(c_{i,f})$. From them, we build our features. Specifically, each dataset entry represents a webpage visit and is described by the following features:

- We build two sets including, respectively, the size of all client packets $|c_{i,f}|$ and the size of all server packets $|s_{i,f}|$ for *Trigger* and *Related* flows. Out of each set, we extract the 25^{th}, 50^{th}, 75^{th} and 90^{th} percentiles, as well as the mean, standard deviation, maximum, minimum values, and the occurrences of the missing samples[3].
- We compute the inter-arrival time of each packet as the time elapsed since the previous packet in the same direction. Given $c_{i,f}$ (or $s_{i,f}$), its inter-arrival time is defined as $t(c_{i,f}) - t(c_{i-1,f})$, with $i \geq 2$. We build two sets including all inter-arrival times for client and server packets, respectively. Out of them, we extract the same statistics as for packet sizes.
- For each *Related Flow* f_i, we compute its relative starting time with respect to the *Trigger* as $t(c_{1,f_i}) - t(c_{1,f_0})$ with $i \in \{1 \ldots k\}$.
- For each flow f_i, we compute the time between the first server and client packet $t(s_{1,f_i}) - t(c_{1,f_i})$. This time measures, in fact, the Round Trip Time between the operator's ground station and the actual server.
- For each flow, we compute the size of the first flight in each direction. A flight is the total size of the application payload contained in packets sent without any confirmation received from the counterpart. It provides an estimate of Client and Server Hello TLS messages. For the client side, given flow f_i, we compute it as $\sum_{j=1}^{j_{max}} |c_{j,f_i}|$ with j_{max} set to highest j that verifies $t(c_{j,f_i}) < t(s_{1,f_i})$. For the server side, we compute the same measure specularly.[4]

[3]Missing samples can occur if a flow has less then n packets, or if a website opens less than k flow in ΔT

[4]For the server side, we clearly neglect client packets related to the first flight while computing j_{max}.

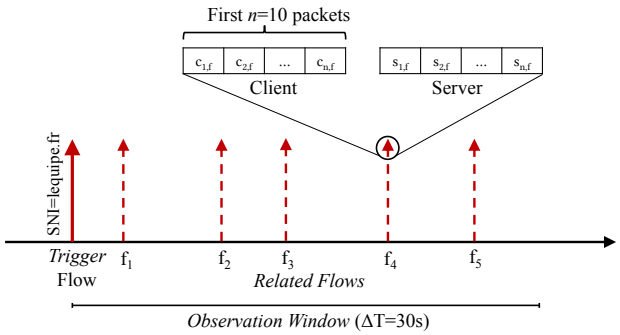


Fig. 3: Detection of *Related Flows* and corresponding features.

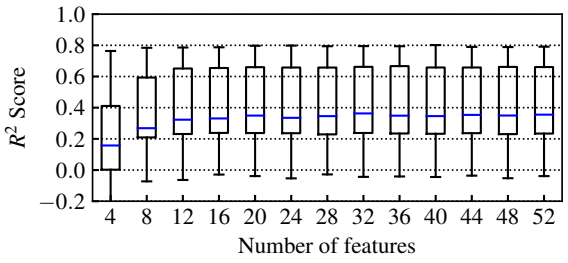


Fig. 4: Prediction performance for OnLoad with a different number of features.

- For each flow, we extract the Time-To-Live (TTL) value from the IP header of server packets. This is a coarse indication of the server's distance (in number of hops). As the TTL could vary within a single flow, we take the minimum value. For each flow f_i, we extract $\min_{j \in 1...n} TTL(s_{j,f_i})$.

In total, we have 52 features. By design, they are L4-agnostic, meaning that even if the communication is carried over UDP/QUIC, the system continues to operate seamlessly. At last, observe that, once the Starting Point is triggered, the first k Related Flows may include some flows generated by background traffic and applications. This may result in occasional uncorrelated flows that can confuse the classifier. For instance, if the user accesses multiple web pages at the same time, the flows each page generates gets multiplexed in the network, possibly impacting the feature extraction process.

C. ML Pipeline

To train the ML models, we adopt the typical pipeline for supervised tasks. We formulate the problem as a regression task, the goal being the prediction of the value of the target QoE-related metrics. Given a data point y_i, we aim at building a model $\hat{y}_i = f(\mathbf{x}_i)$, where $\mathbf{x}$ is the array of the input features described in the previous section.

To measure prediction performance, we use two established metrics: R^2 Score and Mean Absolute Percentage Error (MAPE). The R^2 coefficient serves to determine whether a linear regression can be used to describe the target variable. An R^2 score of 0 signifies a model $\mathbf{x}$ that is not able to predict correctly the actual values (i.e., $\mathbf{x}$ cannot explain y). Conversely, $R^2 = 1$ means $f(\mathbf{x})$ is a perfect predictor. In mathematical terms:

$$R^2 = 1 - \frac{\sum_{i=1}^{z}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{z}(\hat{y}_i - \bar{y})^2} \quad (1)$$

where $\hat{y}_i$ represents the predicted values of the actual sample y_i, $\bar{y}$ being the mean y_i and z the number of data points.

The MAPE is a metric used to assess the accuracy of a predictive model in percentage terms. It measures the average percentage error between the predicted and actual values:

$$MAPE = 100\frac{1}{z}\sum_{i=1}^{z}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \quad (2)$$

We apply a feature selection stage to identify the most relevant features from an initial array $\mathbf{x}$ of 52 variables. This reduces the complexity of the model by discarding either features that are redundant or uncorrelated with the target variable. To tackle this task, we utilize the Recursive Feature Elimination (RFE) algorithm [20] in conjunction with a Random Forest Regressor-based model. This approach involves systematically training the algorithm and iteratively discarding features deemed least important. The output **x'** is then used to build the model $f(\mathbf{x'})$

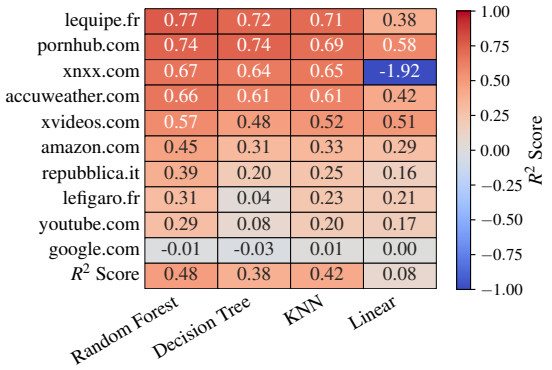


Fig. 5: Comparision of prediction performance for onLoad, using one model per site.

We train a specific ML model for each website using the selected features and adopting the standard Stratified K-Fold Cross-Validation methodology to mitigate the risk of overfitting, with 70% and 30% of data used for training and validation. As regression models $f(\mathbf{x'})$, we test Decision Tree Regressor (DTR), Random Forest Regressors (RFR), Linear Regressors (LR) and K-Nearest Neighbors Regression (KNNR).

IV. Experimental Results

In this section, we show and discuss the experimental results, delving into feature and ML model choice and evaluating the performance in different scenarios. To determine the most suitable ML model and training strategy, we undertake a systematic exploration of several ML models and alternatives on how to build them.

A. Feature and algorithm impact

We first discuss the impact of the feature selection and the choice of regression algorithm. As detailed in Section III-C,

TABLE I: List of 25 most important features according to RFE.

Measure	Direction	Statistics
Packet Inter-arrival Time	Client	Mean, Standard Deviation, Maximum, #(Zero Value), Percentiles 25, 50, 75, 90
Packet Inter-arrival Time	Server	Standard Deviation, Maximum, Percentiles 50, 75
Time-To-Live	Server	Minimum value of the first 3 flows
Time between first Client and Server packet	-	First 5 flows
Relative time of Related Flows	-	All Related Flows

we adopt the RFE feature selection method to find the most meaningful features among the 52 we extract. In Figure 4, we show how R^2 Score varies with models that consider only the top-k ranked features for the onLoad prediction. Boxplots represent the distribution across the 10 websites. We observe that the median R^2 score significantly increases up to 20 features. Including more than 25 features brings negligible improvement.

Looking at which are the most relevant features, we observe those related to packet inter-arrival time and other flow timings (relative starting time in particular), as well as some on the server Time-To-Live. For completeness, we list in Table I the set of 25 features that we use in all experiments.

We now compare the performance of different classification algorithms. For each one, we run a hyper-parameter tuning step using a coarse grid search. A Random Forest Regressor with 100 estimators provides the best performance in all cases, for both OnLoad and SpeedIndex and for both the R^2 and MAPE performance metrics. In Figure 5, we detail the average per-site R^2 scores of all algorithms in predicting the OnLoad. The last row presents the column-wise average. As said, the Random Forest model provides the best performance for all websites, followed by KNN. However, the latter entails large models, as they must contain the entire training set. Thus, we believe the Random Forest model represents the best choice. A Decision Tree provides modest performance for all websites, while the Linear regressors perform largely worse than other methods, hinting they are not well-suited for this kind of problem.

At last, we test the performance of Neural Network (NN) models. However, the need for large amounts and always recent data limits their applicability in this scenario. Our dataset in fact was too small to let the NN converge. In practice, the cost of obtaining thousands of samples for each target website increases the cost of data collection with a limited payoff.

B. Per-website Model Performance

We now dissect performance for different websites and target metrics. We build a website-specific model, an option suitable for our target deployment, where the operator can select the target websites. Here we consider the Random Forest Regressor, which provides the best results (see Sec. IV-A).

In Figure 6, we show per-website performance in terms of R^2 score (top plot) and MAPE (bottom plot). As expected, values of the R^2 score and MAPE are negatively correlated. Prediction accuracy radically varies across websites, with some exhibiting very good performance while others have unsatisfactory results. For onLoad, 6 websites have R^2 score > 0.5 and MAPE $< 25\%$, hinting that ML models are capable

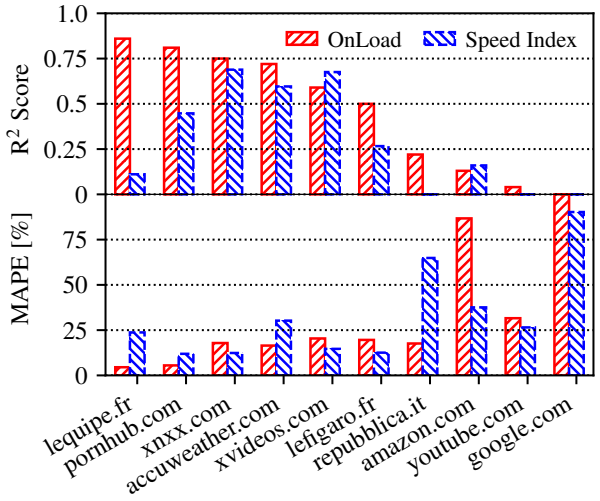


Fig. 6: Performance on different websites, measured using R^2 Score and MAPE.

of providing a reliable prediction for them. However, for the remaining 4 websites, the predictive power of the model is very poor, with an R^2 score below 0.25 or negative (not visible as the scale represents positive values only). For *amazon.com*, *youtube.com* and *google.com*, we link bad prediction to the fact that those websites include objects that are served by a few domains hosted in the same infrastructure owned by the same company. This limits the number and the diversity of related flows that are often below $k = 5$ impacting the set of meaningful features. Conversely, for *repubblica.it* we observe that all pages include a very large number of third-party objects. Some of them are advertisement banners extremely slow to load. This impairs the OnLoad time as observed in Figure 2, which exceeds 35 seconds most of the time. This makes the prediction for this website very unreliable. Similar considerations hold for the Speed Index, even if prediction accuracy is overall lower – only 3 websites present R^2 score above 0.5. This is somewhat expected, as the Speed Index value depends on how the webpage is rendered by the browser [8], [21] and, thus, network traffic has a more indirect impact.

We provide two examples to qualitatively illustrate predictions for a website with a high/poor R^2 score in Figure 7. We show the predicted and real values for the OnLoad metric using a scatter plot. Each point represents a different visit, and the color indicates the density of points. Ideally, in the case of a perfect regressor, all points should lie on the main diagonal.

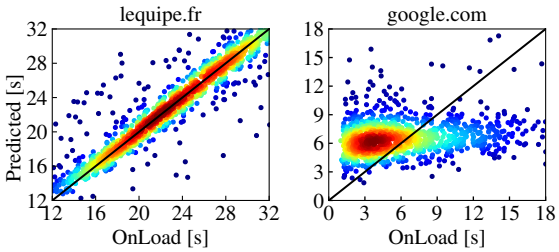


Fig. 7: Scatter plots of predicted and real OnLoad values for two websites.

This is what happens for *lequipe.fr* ($R^2 = 0.8$) where points lie on the diagonal stretched between 16 and 30 seconds, hinting that predictions are accurate both for slow and fast visits. Different is the picture for *google.com* ($R^2 < 0$). The model exhibits almost no prediction power, with predictions that are massed in a circular shape centered along 6 seconds, regardless of the real visit OnLoad time.

In summary, the different performance indicates that the operator shall carefully select the target websites and test the prediction performance of the regressor before deploying it.

C. One vs Many Models

We now discuss the possibility of using a single model to predict the QoE for multiple websites.

First, we consider the case of building a single model trained on all websites. The intuition is to create a model that generalizes the complexities and nuances present across multiple domains, resulting in a more robust and adaptable solution. For this, we create a single set containing all points for all websites. We use it for training (70%) and testing (30% split) a *single* model. We normalize the target metrics in a website-wise fashion to obtain values in the same order of magnitude. In Figure 8, we compare the R^2 score we obtain for different scenarios. The first two columns compare the prediction performance with a Specific Model for each website (first column) with the performance of the Single Model (second column). Overall, prediction accuracy, measured as R^2 score, decreases but never more than 0.2. For some websites, the drop is negligible (see for example *pornhub.com*), while for *accuweather.com*, we even observe a moderate increase.[5] Again, some websites present awful performance in general

We now evaluate the scenario in which the ML model is used to predict the QoE for websites unseen at training time. The intuition is to have a global single model that works for any website. To this end, we adopt a leave-one-out approach: we train a model using all data from all websites except the one under consideration for testing, i.e., systematically excluding data about one target domain at a time during the training phase. The resulting performance is depicted in the third column of Figure 8. It shows that almost no prediction capability is offered in such a scenario. In most cases, the R^2 score is negative, and above 0.5 only in two cases. This clearly demonstrates that we need to include samples during training

[5]The increase is moderate and not statistically significant.

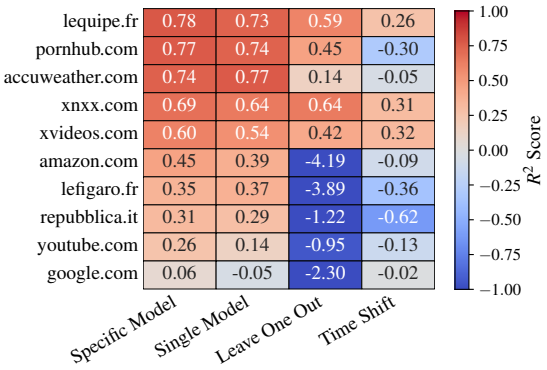


Fig. 8: Performance for onLoad in different scenarios.

for the specific target website the operator wants to monitor. In a nutshell, it is not possible to generalize a single model applicable to any website. This reflects the specificity of the traffic generated by each website and underlines the need to collect site-specific data for training.

We conclude that the best results are obtained by considering a site-specific model. It would be possible to train a single model on multiple websites, even with a moderate penalty. However, it is not possible to train a generic model to predict the QoE of a website unseen at training time.

D. Temporal Stability

We finally evaluate the impact of training data and model freshness. We want to quantify to what extent a model trained on a dataset collected at a given point in time can correctly predict the QoE later in time. To this end, we use the data collected in September 2022 to train models, while we use data from February 2023 to evaluate their prediction performance. We use the best-performing regressor, i.e. Random Forest regressor, trained for each website. We show results in the last column of Figure 8. With few exceptions, the predictive capabilities of the models are completely lost. For instance, the *lefigaro.com* R^2 drops from 0.78 to 0.26. For *xnxx.com* and *xvideos.com*, some prediction capability survives after three months. In the other cases, the prediction power is completely lost, with negative R^2 scores. This is due to changes in the website aspect, in the infrastructure that serves them, or/and in the network properties. For instance, manual inspection of available snapshots on the Wayback Machine[6] reveals that some websites incurred a graphical restyle in the last weeks of 2022. This clearly makes the previously collected training set totally outdated.

We simulate such a system using the data in the December 2022 - February 2023 dataset. In Figure 9, we show the evolution over time of the R^2 score for *pornhub.com*. The blue dashed line reports performance when using always the same model trained using only the first week of data (the penultimate week of December 2022), and then testing its performance on all visits occurring in each period of 5 days. Clearly, the frozen model becomes quickly obsolete over time. From the beginning of 2022, its prediction power vanishes. In

[6]https://web.archive.org/

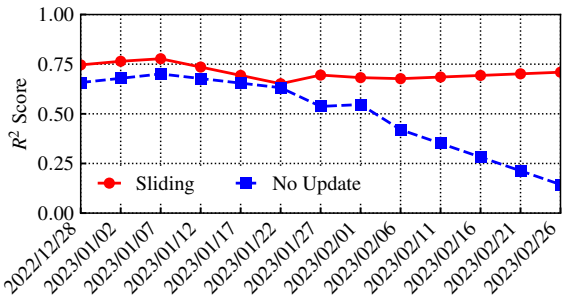


Fig. 9: Prediction performance for OnLoad of *pornhub.com* with different training strategies.

a nutshell, the regression model is becoming too outdated. We observe the same behavior for all websites: in the long term, any model must be updated.

To overcome the model aging, we propose to use the *walk-forward* approach typically used for time series prediction. Here, we assume test agents continuously collect data. For each target website, at the end of a period (e.g., a few days), we use the data collected from the past w days to train a new model. We follow a *sliding-windows* approach, i.e., we keep constant the size of the training set. We then use this freshly trained model to predict the performance during the next period of time.

We simulate a system that implements the walk-forward policy previously described. We show results with the red solid line in Figure 9. Here, we consider a period of w equal to 5 days for training a model. We then evaluate its performance for the following 5 days. Since we use a model that is trained always on the most recent dataset, the R^2 score remains stable at around 0.75.

We conclude that it is mandatory to continuously operate Test Agents to ensure the freshness of the training data which is key to maintaining model performance consistent in time.

V. Related Work

The QoE of Internet users is a wide and complex topic as it involves the subjectivity of users and has been extensively studied in the literature [22], [23]. Several works address the challenges of gathering meaningful metrics and the impact of different network conditions [24], [5], [6], [7], [25]. Commonly, Web QoE is measured through proxy metrics, which have been shown to be correlated with users' subjective experience. Notably, Page Load Time (also called OnLoad) and Speed Index are the most widely adopted, although Bocchi et al. [8] have shown that they do not necessarily model all factors influencing users' perceptions. More recently, "Above-The-Fold" metrics have been proposed as a more accurate estimation of users' QoE [21].

Regarding SatCom environments, the literature extensively explores the role of the physical layer on the seamless network operation and Quality of Service [26], [27]. Several works proposed approaches to enhance browsing performance in SatCom environments using the most diverse techniques, including Performance Enhancing Proxies and HTTP caching [2], [3], [4]. However, there is still a dearth of studies evaluating or measuring QoE in these kinds of networks. Recent efforts regard estimating video QoE [28] or measuring the performance of the QUIC novel protocol in the SatCom environment [29], [30]. Similar to us, these works follow leverage ML to estimate QoE in SatCom, but target different types of services.

Measuring QoE using passive network measurements poses additional challenges, as it is not trivial to derive meaningful metrics or features from network traffic, especially when encryption is in place. Recent research efforts have focused on proposing unsupervised [9], [10] and supervised [11], [12], [13] ML approaches to monitor browsing QoE. These techniques have in common careful feature engineering and the use of machine learning models to derive measures that correlate to users' perceived QoE or to well-known proxy metrics. However, none of them has been studied specifically for SatCom environments or even tested in such a scenario, and no existing literature comprehensively studies Web QoE metrics in SatCom networks. Our work aims to bridge this gap. We engineer a system based on supervised learning and exploit some of the intuitions proposed in [9], [11], specifically the link of different flows to the same user visit. We tailor our system to be deployed at the SatCom operator's premises, where the presence of complex middleboxes (i.e., PEPs) challenges the harvest of truthful metrics.

VI. Conclusions and Discussion

A. Lessons Learned

In this paper we explored the possibility of predicting website performance in a SatCom scenario. Using ingenuity to gather training data, it is possible to reach good prediction performance for a given subset of websites. However, the diverse content dynamics, resource loading patterns, and architectural variations across different websites prevent the construction of a general model able to operate on new websites unseen at training time. While a unified model encompassing multiple domains is conceivable, the ability to accurately predict the performance of unseen websites remains a daunting task.

Our results highlight the indispensable role of continuous model training. Given the dynamic nature of websites, characterized by evolving content and fluctuating infrastructure deployment, it turns mandatory to update and train models on a regular basis. An iterative training process as the walk-forward approach is fundamental to ensure that models adapt to the ever-changing nature of traffic patterns and maintain their efficacy over time.

Considering ML models, we showed that a simple random-forest algorithm suffices. While Neural Network models may outperform it, the need for large training data coupled with the need to continuously update the model with freshly collected data hinders their adoption.

B. Next Directions

Our results show the feasibility of the approach. They represent an initial step towards a large-scale QoE-monitoring infrastructure for SatCom operators. Aspects related to temporal stability and model generalization shall be further studied

using larger and more diverse datasets. In particular, the diverse and global coverage offered by SatCom networks that span entire continents mandates considering (i) the diversity of users' habits and interests in different websites, (ii) the subscriber setup (WiFi, wired, shared, etc.), and (iii) the backbone that connects the ground station to the servers and CDNs. Transfer learning and domain adaptation mechanisms shall be considered to adapt and generalize models so that they can be ported to different countries or continents.

Moreover, further ingenuity shall be used to design more accurate and realistic Test Agents. Additional dimensions to study include testing different browsers, emulating mobile devices, or accurately modeling users' web browsing behavior. Moreover, mobile applications are more and more used to access Internet services as an alternative to classical web browsers for smartphone users. Gathering data from native mobile applications is notably cumbersome and requires further work.

ACKNOWLEDGEMENT

The research leading to these results has been funded by the PRIN 2022 Project ACRE (AI-Based Causality and Reasoning for Deceptive Assets - 2022EP2L7H) and the PRIN 2022 Project COMPACT (Compressed features and representations for network traffic analysis in centralised and edge internet architectures - 2022M2Z728).

REFERENCES

[1] D. Perdices, G. Perna, M. Trevisan, D. Giordano, and M. Mellia, "When Satellite is All You Have: Watching the Internet from 550 Ms," in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 137–150.

[2] P. Davern, N. Nashid, C. J. Sreenan, and A. Zahran, "HTTPEP: A HTTP performance enhancing proxy for satellite systems," *International Journal of Next Generation Computing (IJNGC)*, vol. 2, pp. 242–256, 2011.

[3] I. Bisio, S. Delucchi, F. Lavagetto, and M. Marchese, "Transmission rate allocation over satellite networks with Quality of Experience-Based performance metrics," in *2014 7th advanced satellite multimedia systems conference and the 13th signal processing for space communications workshop (ASMS/SPSC)*. IEEE, 2014, pp. 419–423.

[4] A. Thibaud, J. Fasson, F. Arnal, D. Pradas, E. Dubois, and E. Chaput, "QoE enhancements on satellite networks through the use of caches," *International Journal of Satellite Communications and Networking*, vol. 36, no. 6, pp. 553–565, 2018.

[5] A. Sackl, P. Casas, R. Schatz, L. Janowski, and R. Irmer, "Quantifying the impact of network bandwidth fluctuations and outages on Web QoE," in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, 2015, pp. 1–6.

[6] T. Hoßfeld, F. Metzger, and D. Rossi, "Speed Index: Relating the Industrial Standard for User Perceived Web Performance to web QoE," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, 2018, pp. 1–6.

[7] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, ""Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics"," in *Passive and Active Measurement*, R. Beverly, G. Smaragdakis, and A. Feldmann, Eds. Cham: Springer International Publishing, 2018, pp. 31–43.

[8] E. Bocchi, L. De Cicco, and D. Rossi, "Measuring the quality of experience of web users," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 4, pp. 8–13, 2016.

[9] M. Trevisan, I. Drago, and M. Mellia, "PAIN: A Passive Web performance indicator for ISPs," *Computer Networks*, vol. 149, pp. 115–126, 2019.

[10] L. R. Jiménez, M. Solera, M. Toril, C. Gijón, and P. Casas, "Content matters: Clustering web pages for QoE analysis with WebCLUST," *IEEE Access*, vol. 9, pp. 123 873–123 888, 2021.

[11] A. Huet, A. Saverimoutou, Z. B. Houidi, H. Shi, S. Cai, J. Xu, B. Mathieu, and D. Rossi, "Revealing qoe of web users from encrypted network traffic," in *2020 IFIP Networking Conference (Networking)*. IEEE, 2020, pp. 28–36.

[12] P. Casas, S. Wassermann, N. Wehner, M. Seufert, J. Schüler, and T. Hossfeld, "Mobile Web and App QoE Monitoring for ISPs-from Encrypted Traffic to Speed Index through Machine Learning," in *2021 13th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 2021, pp. 40–47.

[13] P. Casas, S. Wassermann, N. Wehner, M. Seufert, and T. Hossfeld, "Not all Web Pages are Born the Same Content Tailored Learning for Web QoE Inference," in *2022 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE, 2022, pp. 1–6.

[14] J. Griner, J. Border, M. Kojo, Z. D. Shelby, and G. Montenegro, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135, Jun. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3135

[15] M. Trevisan, A. Finamore, M. Mellia, M. Munafo, and D. Rossi, "Traffic analysis with off-the-shelf hardware: Challenges and lessons learned," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 163–169, 2017.

[16] W. Aqeel, B. Chandrasekaran, A. Feldmann, and B. M. Maggs, "On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 680–695.

[17] N. Jha, M. Trevisan, L. Vassio, and M. Mellia, "The Internet with privacy policies: Measuring the Web upon consent," *ACM Transactions on the Web (TWEB)*, vol. 16, no. 3, pp. 1–24, 2022.

[18] M. Rajiullah, A. Lutu, A. S. Khatouni, M.-R. Fida, M. Mellia, A. Brunstrom, O. Alay, S. Alfredsson, and V. Mancuso, "Web experience in mobile networks: Lessons from two million page visits," in *The world wide web conference*, 2019, pp. 1532–1543.

[19] S. Wassermann, P. Casas, Z. B. Houidi, A. Huet, M. Seufert, N. Wehner, J. Schüler, S. Cai, H. Shi, J. Xu *et al.*, "Are you on mobile or desktop? on the impact of end-user device on web qoe inference from encrypted traffic," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–9.

[20] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[21] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between QoS metrics and Web QoE using Above-the-fold metrics," in *Passive and Active Measurement: 19th International Conference, PAM 2018, Berlin, Germany, March 26–27, 2018, Proceedings 19*. Springer, 2018, pp. 31–43.

[22] O. Kondratyeva, N. Kushik, A. Cavalli, and N. Yevtushenko, "Evaluating quality of web services: A short survey," in *2013 IEEE 20th International Conference on Web Services*. IEEE, 2013, pp. 587–594.

[23] S. Baraković and L. Skorin-Kapov, "Survey of research on Quality of Experience modelling for web browsing," *Quality and User Experience*, vol. 2, pp. 1–31, 2017.

[24] T. Hoßfeld, S. Biedermann, R. Schatz, A. Platzer, S. Egger, and M. Fiedler, "The memory effect and its implications on Web QoE modeling," in *2011 23rd International Teletraffic Congress (ITC)*, 2011, pp. 103–110.

[25] D. Giordano, S. Traverso, L. Grimaudo, M. Mellia, E. Baralis, A. Tongaonkar, and S. Saha, "Youlighter: A cognitive approach to unveil youtube cdn and changes," *IEEE Transactions on Cognitive Communications and Networking*, vol. 1, no. 2, pp. 161–174, 2015.

[26] C. Niephaus, M. Kretschmer, and G. Ghinea, "QoS provisioning in converged satellite and terrestrial networks: A survey of the state-of-the-art," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2415–2441, 2016.

[27] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff *et al.*, "Satellite communications in the new space era: A survey and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 70–109, 2020.

[28] M. Petrou, D. Pradas, M. Royer, and E. Lochin, "Forecasting YouTube QoE over SATCOM," in *The IEEE 97th Vehicular Technology Conference*, 2023.

[29] L. Thomas, E. Dubois, N. Kuhn, and E. Lochin, "Google quic performance over a public satcom access," *International Journal of Satellite Communications and Networking*, vol. 37, no. 6, pp. 601–611, 2019.

[30] N. Kuhn, F. Michel, L. Thomas, E. Dubois, E. Lochin, F. Simo, and D. Pradas, "Quic: Opportunities and threats in satcom," *International Journal of Satellite Communications and Networking*, vol. 40, no. 6, pp. 379–391, 2022.