ABSTRACT

RTC traffic has been increasing by a lot in the recent years, especially during the Covid-19 pandemic when it has reached its peak due to the social distancing and the need of people around the world to keep being connected with each other for various reasons such as remote working, social interactions etc. Consequently, this implies the importance of monitoring and inferring of the quality of experience (QoE) at the end users' side by the network providers in order to guarantee high quality perceived by the clients. However, the definition of the QoE by the network providers is not that trivial since there isn't any tool at the user side capable of measuring QoE metrics for the app, and reporting this information to the entity (OS, ISP, or application server) implementing the QoE-aware mechanisms. This means that the information regarding QoE is only visible to the end user or application, however the network operator instead has access only to network level measurements such as throughput, packet loss or jitter etc., which are all QoS metrics. Therefore, this is the main point which this study focusses on, it aims at closing the gap between QoS and QoE metrics so that network operators by looking at the QoS parameters, they can translate them into QoE and accordingly apply traffic engineering mechanisms to be able to guarantee high quality experience.

In order to close the gap between the QoE and QoS, we aim to develop a Machine Learning approach. The whole procedure followed in order to build this approach is briefly explained in the Figure 1 below:

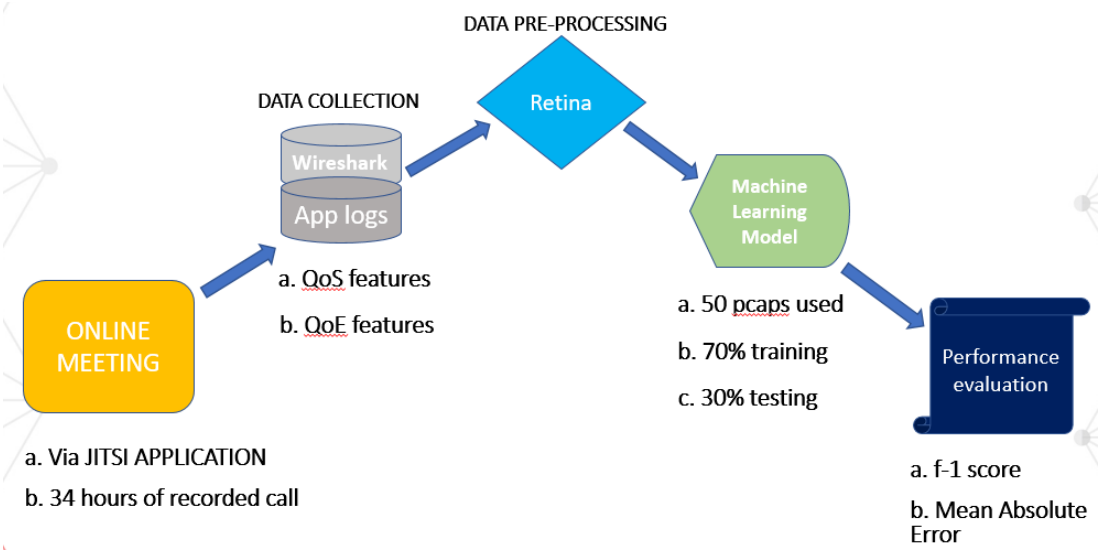

Figure 1: *Thesis workflow*

Translating the figure 1 into words, there are basically a considerable amount of dataset collected from an online meeting scenario where there is a 34 hours of recorded call and 50 pcaps in total. The data collection is applied in two ways, the QoS parameters which network operators have access to, are collected via the famous Wireshark tool whereas the QoE parameters (not accessible from network operators) from application logs, both at the end user side. After having the data collected, we prepare them via a very useful tool developed by the "SmartData@poliTO" research group called "Retina", which produces a rich log of statistics on observed streams. So basically it prepares the dataset to be processed by putting the QoS and QoE features according to the time of receiving and puts them to a table ready to be processed. Once having this file prepared with all the information needed, now we can process and experiment on them using various ML algorithms such as classification or regression, as well as different tools such as random forest, decision tree etc. In this way, after each experimentation we measure the model's performance and try to improve it as much as we can by using techniques such as feature selection, outliers detection and processing, and so on. Finally, we choose the model which performs the best and discuss on it. At the end, we test our model also with another dataset type, collected from another kind of RTC traffic scenario which is "Cloud gaming" in order to see how scalable the model is.

Basically, so far we have briefly introduced the problem, objectives and the methodology being used in this study, whereas now we'll present an overview of the results obtained after the whole process introduced a while ago. Note that we rely on 3 QoE metrics which are used as target features such as "*resolution*" defined as the size in pixels of the received video frames, "*smoothness*" which is the number of frames per second displayed to the user and "*concealment*" which represents the time spent by the decoder rendering frames that are partially received. In our dataset the equivalent features of those 3 metrics are "***framesPerSecond***" representing *smoothness*, "***frameWidth***" representing *resolution* and "***concealmentEvents***" and "***framesDropped***" representing the *concealment for audio and video* respectively. The "Table-1" below briefly presents the results obtained for the predictions of each of those fields:

Predicted metric	Model	Algorithm	Feature selection Method.	F-1 score	R^2 / MAE
Resolution	XG Boost	Classification	Annova f-value	0.91	-
Smoothness	Random Forest	Regression	Correlation coeff.	-	1.43
Audio conceal.	Random Forest	Classification	Annova f-value	0.64	-
Video conceal.	Decision Tree	Classification	Annova f-value	0.60	-

Table 1: *Results for each prediction*

Briefly we can say that the performance of the model when predicting the resolution is above 90% which can be considered a very promising result. We should note that there are 4 classes of resolution to be predicted here where each of them represents the quality of the video, where 1 is the lowest one and 4 the highest.

Then for smoothness, we use regression and we obtain a Mean Absolute Error of 1.43 which compared to the distribution of the values for this parameters (mostly around 30 frames per second) can be considered as a good result.

Finally regarding the audio and video concealments, in this scenario we predict only whether there is any concealment or not, so we have to do only with the prediction of 2 classes. However, although we have 2 classes to predict, the reason why the obtain such low results is because the gap between the number of supports for each class is quite high. Saying that the case where concealments are not present consist of almost 97% of the whole dataset which means that the gap in the representation of the 2 classes is really significant and this leads to such low f-1 scores.

As a conclusion, we can say that for the first 2 metrics we have such good results whereas in the concealment case there is the problem with the number of supports especially for the case where concealments are present. So, there is the need to increase the dataset where the cases for concealments being present should increase by a lot in order to close the gap between the 2 classes. Only in this way we can pretend to see the results we desire to.