Predicting the effect of home Wi-Fi quality on QoE

Diego da Hora, Karel van Doorselaer, Koen van Oost, Renata Teixeira

Predicting the effect of home Wi-Fi quality on QoE
Extended Technical Report

Diego da Hora
Telecom Paristech
Paris, France

Karel van Doorselaer, Koen van Oost
Technicolor
Edegem, Belgium

Renata Teixeira
INRIA
Paris, France

***Abstract*—Poor Wi-Fi quality can disrupt home users' internet experience, or the Quality of Experience (QoE). Detecting when Wi-Fi degrades QoE is extremely valuable for residential Internet Service Providers (ISPs) as home users often hold the ISP responsible whenever QoE degrades. Yet, ISPs have little visibility within the home to assist users. Our goal is to develop a system that runs on commodity access points (APs) to assist ISPs in detecting when Wi-Fi degrades QoE. Our first contribution is to develop a method to detect instances of poor QoE based on the passive observation of Wi-Fi quality metrics available in commodity APs (e.g., PHY rate). We use support vector regression to build predictors of QoE given Wi-Fi quality for popular internet applications. We then use K-means clustering to combine per-application predictors to identify regions of Wi-Fi quality where QoE is poor across applications. We call samples in these regions as poor QoE samples. Our second contribution is to apply our predictors to Wi-Fi metrics collected over one month from 3479 APs of customers of a large residential ISP. Our results show that QoE is good on the vast majority of samples of the deployment, still we find 11.6% of poor QoE samples. Worse, approximately 21% of stations have more than 25% poor QoE samples. In some cases, we estimate that Wi-Fi quality causes poor QoE for many hours, though in most cases poor QoE events are short.**

I. Introduction

Wi-Fi is the preferred technology for accessing the internet from home. Home Wi-Fi networks can, however, disrupt end-to-end application performance. For example, in dense urban neighborhoods it is typical to see tens of competing Wi-Fi networks [22], alongside many non Wi-Fi RF devices [24]. Also, a poorly located AP will leave stations with weak signal. In these cases, Wi-Fi quality can degrade users' internet experience, or the Quality of Experience (QoE).

When users' internet experience is poor, they are often helpless and just call their residential ISP for support. In fact, the "administrators" of home networks often have limited to no network management expertise [8]. The problem is that the ISP has little visibility into what is happening within the home WiFi, which leads to painful helpdesk calls or worse customers who simply change providers. Our discussions with ISPs revealed that technical support calls represent a considerable fraction of their operations cost and they claim that often the root cause of the problems is the home Wi-Fi (recent results confirm that the home Wi-Fi often bottlenecks end-to-end performance [23], [29]).

Our goal is to build a system for ISPs to detect when home Wi-Fi quality degrades QoE *before* users call. Such a system would enable ISPs to proactively fix recurrent problems and consequently reduce customer churn. This system can also help to reduce the length of helpdesk calls, which in turn would help reduce operational costs. In many cases, the ISP provides and controls the home AP, so we leverage the home AP as monitoring point within the home. For such a system to scale to all the customers of an ISP, it must work with commodity APs already deployed by ISPs. This restriction excludes solutions that rely on APs with specialized hardware or multiple Wi-Fi interfaces [2], [19], [21], [24]. In addition, the system cannot disrupt user activities, so we exclude solutions that require installation on end users' devices [6] and that employ active measurements [17], [30], since continuos active measurements may disrupt user traffic and drain devices' battery. Instead, we rely on periodic polling of Wi-Fi metrics typically available on commodity APs (e.g., PHY rate and RSSI). Wi-Fi quality, however, is highly variable [2] and reporting every single Wi-Fi degradation would lead to too many alarms; most often for events imperceptible to users.

The first contribution of this paper is to **develop a method to detect instances of poor QoE from Wi-Fi quality metrics**. Detecting when Wi-Fi quality degrades QoE is challenging as we can only obtain a limited number of Wi-Fi metrics in commodity APs and we have no information about the applications that end users are running at a given time.[1] We rely on regression models to build predictors that estimate the effect of Wi-Fi quality on QoE for four popular applications: web browsing, YouTube, audio and video real time communication (RTC) (§IV). We generate training samples in a controlled environment, where we seep a large set of Wi-Fi conditions, while observing application behavior (§II). To reduce the tests with real users, we measure application quality metrics and use state of the art methods to translate these into QoE as captured by the degradation of mean opinion scores (DMOS), in range [1-5](§III). Finally, we develop a method to combine the per-application predictors into a detector of poor QoE samples. For each possible combination of Wi-Fi quality parameters, we generate a sample with the predicted DMOS for each application and then we apply K-means clustering in the resulting dataset. Our analysis of the resulting clusters identifies ranges of Wi-Fi quality where we predict poor QoE for most applications, which we define as *poor QoE samples*.

Our second contribution is to **characterize the effect of**

[1]ISPs avoid running per-packet capture on APs due to privacy concerns and to avoid overloading the AP.

Wi-Fi quality on QoE in the wild. We apply our detection method on Wi-Fi metrics collected from 3479 APs of customers of a large Asian-Pacific residential ISP over a period of one month (§V). Our results show that Wi-Fi quality is often good enough for the four applications we study. We classify 11.6% of deployment samples as poor QoE samples, and we find that 21% of stations have more than 25% poor QoE samples. Then, we group consecutive poor QoE samples from a single station into *poor QoE events*, which allow us to distinguish among short, intermittent, and consistent poor QoE events. We find that over 78% of poor QoE events are short, although we observe poor QoE events stretching over many hours. Finally, we diagnose poor QoE events by verifying the underlying Wi-Fi metrics. We find that the majority of poor QoE events have average PHY rate $<$ 15Mbps. In the other cases, we observe an indication of interference. In particular, poor QoE events caused by non Wi-Fi interference are more likely intermittent than consistent.

II. Experimental setup

To understand how Wi-Fi quality affects user QoE, we perform controlled experiments with four applications: web browsing, YouTube streaming, and audio / video RTC. These are popular applications that represent a large fraction of users' traffic [2]. The typical method to assess QoE is to request users' explicit feedback and then average individual users' scores into a mean opinion score (MOS) [25]. This approach, however, is impractical for us given the wide range of Wi-Fi configurations we must cover. For example, our tests generate over ten thousands data points across the four applications and we require multiple users to rate each point. Instead, during each experiment we measure application-specific QoS metrics and use state-of-the-art models that map application QoS to QoE. This section describes our testbed, the Wi-Fi parameter space we test, and our method to generate traffic and collect per-application QoS.

A. Testbed

We emulate a Wi-Fi home network on the testbed shown in Figure 1. We use an AP with a Broadcom BCM6362 NIC with 802.11n 2x2 technology, and two MacBook pro stations (STA1 and STA2) with 802.11n 2x2 technology. We choose an AP with a Broadcom NIC because these are often deployed by residential ISPs. In particular, this model of AP is used in the deployment we study in §V. We only use metrics that are commonly exposed by Wi-Fi drivers and our predictors should generalize well for other 802.11n NICs.

The Wi-Fi testbed is located in Belgium, and it connects to the internet through an Ubuntu 12 gateway using TC to emulate bandwidth restrictions of a home network access link, with 16 Mbps available bandwidth for download traffic. Our Web and iperf servers are Ubuntu 12 computers. We also use an Android tablet with 802.11n 1x1 technology, using iperf in server mode to receive traffic that interferes with the AP under test. We use the default AP Wi-Fi configurations: 20MHz channel width and long guard interval PHY rates only. We restrict Wi-Fi physical layer (PHY) rates to only use one-spatial-stream, due to the difficulty of reliably enabling MIMO communication in shielded boxes. Although we only study PHY rates up to 65 Mbps, results for higher PHY rates should be similar to those where PHY rate is 65 Mbps, since in both cases the transmission rate is unlikely to degrade QoE.

B. Experimental parameters

We vary the Wi-Fi conditions over two axis:

Link speed. We vary the link speed by introducing 6, 12, 15, 18, 19, and 20 dBs of attenuation in the path between AP and STA using a programmable attenuator.

Medium availability. We vary the medium availability at the AP by introducing interference from Wi-Fi or non-Wi-Fi sources. To introduce Wi-Fi interference, we use the iperf client to generate competing Wi-Fi traffic in the "interfering link", as shown in Figure 1. The AP from the interfering link is configured to only use PHY rate 5.5 Mbps, which blocks the medium longer per medium access. We found that by generating constant bit rate UDP traffic of 0.7 Mbps, 1.1 Mbps, 2.1 Mbps and 2.9 Mbps we obtain, respectively, 70%, 50%, 30% and 15% medium availability, measured by the clear channel assessment (CCA) counters on the AP.

To generate interference from a non Wi-Fi source, we use a signal generator to inject a narrowband sinewave to block the AP's CCA. We sweep the interfering sinewave in and out of the spectral CCA range of the AP every 200ms. We vary the percentage of time the sinewave is inside and outside this range to create scenarios with medium availability of 100%, 75%, 50%, 25%, and 12.5%. AP's Wi-Fi counters confirm that we obtained the intended availability with less than 5% error.

C. Applications

For each Wi-Fi scenario, we execute automated tests with the following applications, while monitoring Wi-FI and application QoS metrics.

Web browsing. STA1 uses PhantomJS, a headless browser, to sequentially access 10 times a set of 10 pages on the internet. We chose a mix of pages figuring in the Alexa TOP 20 pages in Belgium. We access the front pages of the following domains: bing.com, google.be, twitter.com, live.com, wikipedia.org, facebook.com, yahoo.com, amazon.fr, nieuwsblad.be, and hln.be. We use JavaScript to record the page load time (PLT) triggered by the *onload* event.

YouTube. STA1 uses Google Chrome to access an instrumented YouTube page, which downloads and plays each YouTube video using DASH, during two minutes. We access three YouTube videos per experiment: a politician speech, a movie trailer, and a music clip. We choose these clips to obtain diversity in video content. We use JavaScript to record join time, buffering events, and video resolution changes.

Audio and video RTC. We use WebRTC to perform audio and video RTC tests. We implemented a simple WebRTC application, stored on the web server, that allows two peers to communicate in an audio and/or video call. We use the default OPUS plugin on WebRTC to encode audio and the

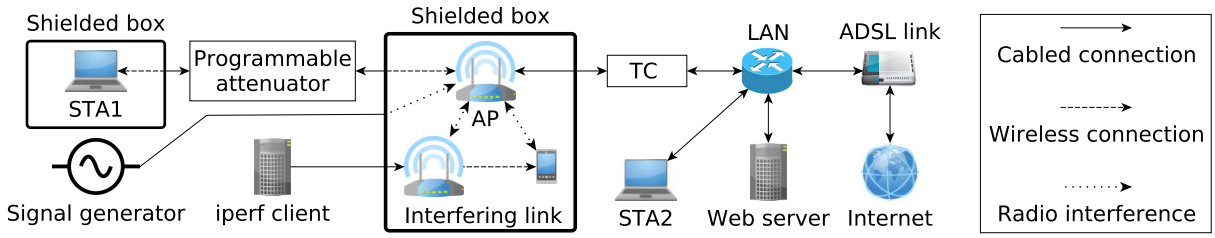


Fig. 1: Wi-Fi testbed used in the study.

VP8 plugin to encode video with default options. We use the *–use-fake-device-for-media-stream* command line option to perform RTC tests with a pre-recorded audio/video sample. This closely resembles a real user environment, since the audio/video processing will go through the same processing stack as a real microphone/camera. We record the received audio/video streams by each WebRTC peer.

We use a set of 20 different audio samples recommended by ITU-T for speech quality assessment [14], each with an approximate duration of eight seconds. We create a single audio file with all the audio samples, which is then sent to Google Chrome's fake audio input device. To prevent echo canceling from interfering with the audio samples, we only examine cases where STA2 sends audio and STA1 is silent. After the experiment, we manually extract the individual audio samples from the received audio feed. For video, we use standard reference videos featuring human subjects from the Xiph collection [37], namely: *FourPeople*, *Johnny*, *KristenAndSara*, *Vidyo1*, *Vidyo3* and *Vidyo4*. We downscaled the original video format to 640×480, 30 fps to match common webcam capabilities. We merged all the video samples, each with 300 frames (10 seconds), into a single video file, separated by 15 "black" frames to mark the transition between samples. After the experiment, we extract individual video samples from the received video using the "black" frames to detect sample transition, as well as manual verification (in rare cases, none of the "black" frames were received).

III. Training Set

We use the testbed and controlled experiments presented in the previous section to build a training set where each sample is a vector that contains Wi-Fi metrics we passively collect from the AP labeled with the estimated QoE for each application. This section describes our approach to map application QoS metrics into an estimated QoE and how we build the training set.

A. QoE metric: Degradation MOS

For each application, we denote the application QoS to QoE model as a function, $f_{app}^{MOS} : (a, x_a) \rightarrow y^{MOS}$, where x_a is a vector of application QoS metrics for an application a and y^{MOS} is an absolute MOS score. Although the models in the literature output an estimate of the absolute MOS, our goal is to estimate Wi-Fi's contribution to MOS degradation. For example, the simple act of encoding a video for real-time transmission will reduce its quality and the absolute MOS will capture this degradation. Yet, users today are used to the quality loss due to encoding. Hence, we measure QoE with the *degradation MOS* as the normalized output of f_{app}^{MOS} in range $[1-5]$, using Equation 1. Note that we call this function f_{app}^{DMOS} to emphasize that it estimates DMOS based on application QoS metrics, in contrast to the models we introduce in §IV, denoted by f_{Wi-Fi}^{DMOS}, which estimate DMOS from Wi-Fi quality metrics.

$$f_{app}^{DMOS}(a, x_a) = 4.0 \times \frac{f_{app}^{MOS}(a, x_a) - MOS_{min}(a)}{MOS_{max}(a) - MOS_{min}(a)} + 1.0 \quad (1)$$

We set $MOS_{min}(a)$ as $\min\{f_{app}^{MOS}(a, x_a)\}$ for all x_a in the experiments, and $MOS_{max}(a)$ as the average $f_{app}^{MOS}(a, x_a)$ for x_a during baseline experiments (i.e, no Wi-Fi impairment). The estimated DMOS is always 5 in baseline scenarios, and always 1 in the worst scenario. It is possible to find $MOS_{max}(a)$ by maximizing $f_{app}^{MOS}(a, x_a)$ for the set of application QoS parameters expected to be found in the target scenario, and $MOS_{min}(a)$ by minimizing $f_{app}^{MOS}(a, x_a)$ instead. We use Equation 1 to obtain estimated DMOS for each application model. We interpret DMOS quality impairments using the degradation category rating scale [12], where: 5, Imperceptible; 4, Perceptible but not annoying; 3, Slightly annoying; 2, Annoying; and 1, Very annoying.

We select state-of-the-art QoS-to-QoE models to implement $f_{app}^{MOS}(a, x_a)$ for each application, a. We validate each model through user studies as shown below.

B. Video RTC QoE model

There is extensive work on objective video quality assessment using three types of metrics to capture video quality: full reference, reduced reference, and no reference. Since we have access to the original video signal in our testbed, we use full reference metrics, which are the most accurate. Popular full reference metrics are PSNR, SSIM, and VQM. We opt to use Structural Similarity (SSIM), since it was shown many times to have strong correlation with human perception.

We compare each video clip received by STA1 (*transmitted* sample), with the original sent by STA2 (*reference* sample), and compute per-frame SSIM. We calculate the video's average SSIM (s) and map it to the video MOS score using Equation 2, proposed by Wang et. al for video quality assessment [36].

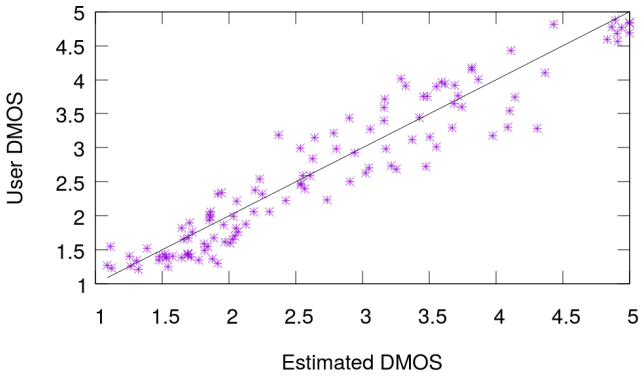


Fig. 2: Validation of video RTC QoE model

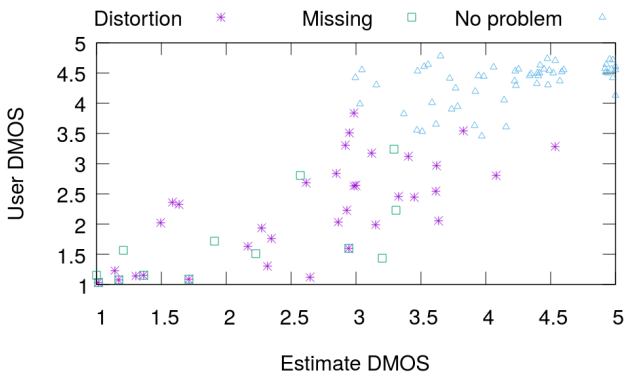


Fig. 3: Validation of audio RTC QoE model and manual classification of audio impairments.

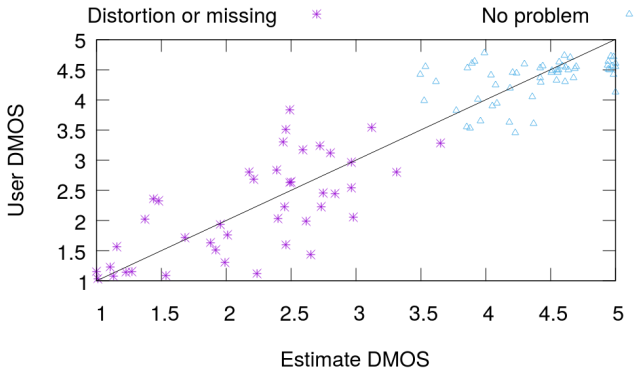
$$f_{app}^{MOS}(video, s) = 129.25s^3 - 64.76s^2 + 22.08s \quad (2)$$

We noticed that under low network capacity, WebRTC often suppresses frames to reduce bandwidth, sending as little as 10 frames per second in extreme conditions. We deal with skipped frames using the strategy employed by Zinner et. al [38], by comparing reference skipped frames to the last received frame. This reduces SSIM whenever the transmitted sample presents skipped frames or freezes.

On some experiments, we observed instances where STA1 was able to associate to the AP and communicate to STA2 through WebRTC, but could not sustain the video call due to high packet loss. Since WebRTC prematurely terminates the call, we could not obtain transmitted samples. For every experiment where we observe a WebRTC connection for 10 second but no video traffic, we generate a sample with DMOS = 1.0, since there is no service.

Model validation. We conduct a small user study to see how DMOS correlates with user opinion. For each one of the six reference samples, we select 20 different transmitted samples with different levels of video quality. We explain to users that they should consider a scenario of a real-time internet video call, show them the reference sample, and ask them to rate the quality of five transmitted samples. We give them the degradation category rating scale [12], to describe the perceived degradation: 1, very annoying; 2, annoying; 3, slightly annoying; 4, perceptible but not annoying; and 5, imperceptible. Each user rates a total of 30 samples, so tests require between 6 and 10 minutes to complete.

We requested for volunteers for this study through friends, at our company, and our lab mailing list, with answers from 40 users. We use a score normalization procedure similar to Wang et. al [35], to deal with users which do not use the whole range of grading options. First, we apply normalization by standard deviation per user. Then, we re-scale all normalized scores into range [1-5], obtaining user DMOS. Figure 2 compares user DMOS and estimated DMOS, with a strong correlation of 0.9547.



‘‘

Fig. 4: Audio RTC QoE model using adjusted PESQ MOS with $\delta = 0.5$

C. Audio RTC QoE model

There are many methods to estimate the perceived quality of a voice service. Signal based methods analyze the distorted signal to estimate listening quality, by either searching for known distortion patterns (e.g., ITU-T P.563) or by comparing it to the undistorted signal (e.g., Perceptual Speech Quality Measure (PESQ) [13]). Parametric models, such as the E-model (ITU-T G.107), estimate listening quality based on several impairment factors such as one way delay, codec distortions, and others [25]. Here, we combine PESQ and E-model to account for both audio distortions and one way delays, following the approach of Lingfen et. al [28]. We obtain PESQ MOS by comparing the reference and received test samples, and one way delay by monitoring the median latency of WebRTC packets.

Similarly to video WebRTC experiments, we observed instances where STA1 was able to associate to the AP and communicate to STA2 through WebRTC, but could not sustain the audio call. For every experiment where we observe a WebRTC connection for 10 second but no audio traffic, we generate a sample with DMOS = 1.0, since there is no service.

Model validation. To understand how audio quality correlates with user opinion, we conduct a small user study. We

select 12 reference audio clips, and 10 transmitted samples for each audio clip with different levels of audio quality. We explain users that they should consider a scenario of a real-time internet audio call, give them the reference sample and ask them to rate the quality of five transmitted samples using the degradation category rating scale. Each user rates a total of 60 samples, so tests require between 10 and 15 minutes to complete.

We requested native Dutch speaking volunteers (as the original audio clips are in Dutch) for this study among friends and at our company, with answers from 20 users. We apply the raw score normalization procedure described on §III-B to obtain user DMOS. Figure 3 compares user DMOS and estimated DMOS per audio sample. We see that while for the majority of cases estimated DMOS is in line with user DMOS, there are cases where user DMOS is significantly higher than estimated DMOS. We listened to all degraded samples from this study and found two main impairments: voice distortion, such as when the voice "digitalizes"; and missing audio, such as when a word or phrase is partially or completely absent, as shown by the different markers on Figure 3. We analyze all instances with no distortion or missing audio and where user DMOS ¿ estimated DMOS. This analysis shows that the vast majority of transmitted samples are slightly fast/slower than original, which is imperceptible for users. In other words, PESQ is too sensitive and may bias results if the application alters playback rate as part of its adaptation mechanism.

To account for this issue, we alter raw PESQ MOS values using a small δ parameter, as shown in Equation 3:

$$f_{app}^{MOS}(audio, < r, t >) = PESQ(r, t) + \delta \times g(t) \quad (3)$$

where r is the reference audio sample, t is the transmitted audio sample, and g is a function that outputs 1 when there is no voice distortion nor missing audio, or -1 otherwise. We listen to all transmitted audio samples to manually compute $g(t)$. We find that $\delta = 0.5$ minimizes the RMSE between user and estimated DMOS. Figure 4 shows estimated and user DMOS on the adjusted audio QoE model, with a correlation of 0.9191.

D. Web browsing QoE model

While there is extensive work showing that web browsing QoE is a result of many influence factors, end user waiting time is consistently identified as the main system influence factor for user QoE. There are different QoE models based on the type of web activity: browsing through an extended session, opening a page, reading e-mails, or downloading a file [25]. Here, we consider the simple case of opening a single web page.

Most single-page web-QoE models consider the page load time (PLT) as the main application QoS metric. The PLT, however, is hard to measure since it is not always clear when users consider the page "ready". Here we use the PLT as measured by the *onload* event, since for the majority of pages, the *onload* event coincides with the user perception

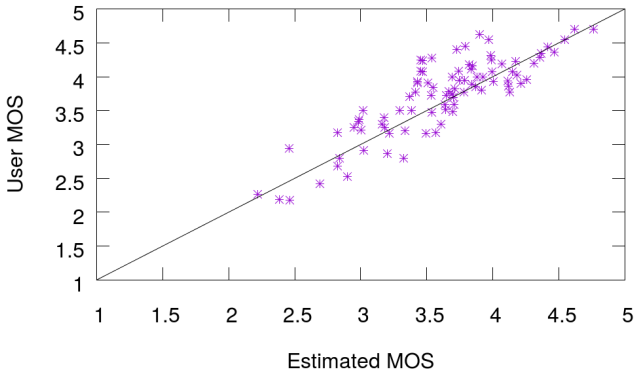


Fig. 5: Validation of Web QoE model.

of the page being ready [32]. There is work suggesting that expectation [11] and page aesthetics [31] significantly influence web QoE. Since Wi-Fi quality cannot directly affect those, we remain confident on modeling MOS as a function of PLT.

We employ the ITU-T G 1030 single-page web-QoE model [15], which maps PLT into MOS for fast, medium, or slow networks contexts. The network context expresses the user expectation of network speed, with users more tolerant to delays on slower network contexts. The ITU-T G 1030 proposes a logarithmic relationship between PLT and MOS, which is in line with most recent web QoE studies [9], [27]. We map PLT to MOS using Equation 4. We consider the medium network context, where $Min = 0.395$ and $Max = 38$, since home internet speed is usually lower than backbones or enterprise networks.

$$f_{app}^{MOS}(web, PLT) = 4.0 \times \frac{log(PLT/Min)}{log(Min/Max)} + 5.0 \quad (4)$$

During web browsing experiments, we limit PLT to a maximum of 10s. When the web page does not finish loading within 10s, we do not have the precise PLT. In these cases, if the main HTML finished loading, we consider PLT = 10s. Otherwise, we consider that the service failed and set DMOS = 1.0.

Model validation. We validate the proposed model with data from a user study on Web QoE made by Bocchi et. al [3], which contains over 4000 PLT and user scores from 146 users across 41 websites and 8 network scenarios. We only consider users with at least 10 scores, and only pages with at least 10 scores. We aggregate results for each combination of page and network scenario to calculate MOS and mean PLT, which we use to obtain estimated MOS. Figure 5 shows the relationship between user and estimated MOS, with a correlation of 0.8529. This result suggests that PLT has a strong correlation with user opinion, even though it is not the only influence factor. Indeed, we verify that our model under-estimates MOS for pages with content "below the fold", not visible until you scroll the page.

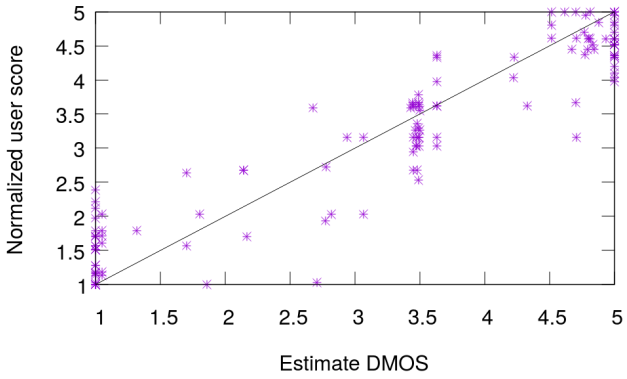


Fig. 6: Validation of YouTube QoE model.

TABLE I: a_i, b_i, c_i parameters for Equation 5

i	λ	a_i	b_i	c_i
1	$\lambda < 0.05$	2.97	0.74	2.03
2	$0.05 < \lambda < 0.10$	3.07	0.96	1.93
3	$0.10 < \lambda < 0.20$	3.17	1.55	1.83
4	$0.20 < \lambda < 0.50$	3.21	1.66	1.79
5	$\lambda > 0.50$	3.24	1.79	1.76

E. YouTube QoE model

Video streaming is a popular internet application and YouTube is the most used video streaming service on the internet. While network impairments may cause frame losses on video RTC, YouTube uses TCP to reliably transmit content. YouTube uses a playback buffer to smoothly play the video and executions typically occur smoothly as long as video download rate is faster than playback rate. *Buffering events* occur when the playback buffer empties, when the player stops to refill the buffer. YouTube uses adaptive bitrate streaming in default configuration. In this scenario, the video player dynamically selects video bitrate during playback to avoid buffering events. There are three main factors influencing QoE: average bitrate, join time, and buffering events [1], [33]. Most studies agree that join time does not drastically affect QoE, as users are more tolerant to initial delays. Therefore, we model YouTube QoE as a function of buffering events and video bitrate.

Buffering events. We use the model proposed by Wamser et. al, which estimates MOS values given the number and duration of buffering events [33], shown in Equation 5. It considers n as the number of buffering events, λ as the ratio between total buffering time and video elapsed time, and a_i, b_i, c_i parameters according to Table I.

$$f_{app}^{MOS}(youtube, < n, i >) = a_i \times e^{-b_i \times n} + c_i, \forall i \in [1-5] \quad (5)$$

Video bitrate. Similarly to Zinner et. al [38], we use SSIM to obtain estimated MOS scores to quantify the impact of video resolution. We compare the video at each resolution with the maximum resolution available ($1080hd$) to obtain video SSIM per resolution. For a video playback, we obtain a weighted average SSIM given by the fraction of time played at each resolution. We estimate MOS using Equation 2 and normalize it with Equation 1.

TABLE II: Wi-Fi metrics measured on the access point.

Metric	Description	Period
BUSY	% of time the medium is busy	2 s
WiFi	% of time busy due to Wi-Fi traffic	2 s
nonWiFi	% of time busy due to non Wi-Fi traffic	2 s
TxPhy	PHY rate of last frame sent	1 s
FDR	Frames sent / retransmitted to STA	1 s
RSSI	Received signal strength indicator	1 s

We model YouTube QoE as the minimum MOS between the estimated MOS of the models. By using the minimum MOS, we remain faithful to each model when only one type of impairment is present. We note that both types of impairment happen together in less than 10% of samples in our testbed dataset.

Model validation. We validate the proposed model using a small user study on YouTube quality, made by Katsarakis et. al [18], where 16 users rated a total of 128 YouTube video executions under different network impairments. We apply the raw score normalization procedure described on §III-B, and compare normalized user scores and estimated DMOS in Figure 6. We find a correlation of 0.9268 using our proposed model, in comparison to 0.7272 and 0.8323 when using only video bitrate or buffering events respectively. This result suggests that it is necessary to account for both types of video impairments.

F. Building the training set

During each experiment, we passively measure the Wi-Fi metrics shown in Table II on the AP. We calculate features describing the Wi-Fi metrics considering Wi-Fi samples measured over an interval T during the application execution. The training set consists of one feature vector per application sample containing the mean, std, min, max, 25%-ile, and 75%-ile for each metric in Table II, labeled with $f_{app}^{DMOS}(a, x_a)$, where x_a captures a's application QoS measured directly from each application.

We choose T based on the duration of the application execution. For audio and video experiments, we consider one application sample per audio / video sample, with $T = 10s$. For web browsing, we create one sample per web page access, using $T = 10s$ since this is the maximum page load time. For YouTube, we create one sample per video playback, using $T = 120s$ since we play each video for two minutes. We obtain a total of 3175, 3062, 153, and 5370 training samples for Video, Audio, YouTube, and Web respectively.

IV. Predicting the effect of Wi-Fi quality on QoE

This section shows how we predict the effect of Wi-Fi quality on QoE. We must learn a function, $f_{Wi-Fi}^{DMOS} : (a, x_{Wi-Fi}) \rightarrow y^{DMOS}$ that for each application, a, predicts DMOS based on a feature vector computed from measured Wi-Fi quality, x_{Wi-Fi}. We formulate the problem of predicting the effect of Wi-Fi quality on QoE as a regression problem.

TABLE III: Best subset of features per application.

Application	Feature vector		
Video	$\text{TxPhy}_{25\%}$	$\text{BUSY}_{25\%}$	BUSY_{max}
	RSSI_{mean}	$\text{RSSI}_{75\%}$	$\text{WiFi}_{25\%}$
Audio	TxPhy_{min}	RSSI_{std}	$\text{WiFi}_{25\%}$,
	WiFi_{max}	nonWiFi_{max}	FDR_{mean}
YouTube	TxPhy_{mean}	$\text{BUSY}_{75\%}$	RSSI_{mean}
	$\text{RSSI}_{25\%}$	$\text{WiFi}_{25\%}$	nonWiFi_{min}
Web	TxPhy_{max}	BUSY_{std}	RSSI_{min}
	WiFi_{max}	nonWiFi_{max}	FDR_{mean}

A. Building the predictor

We select support vector regression (SVR) as regression algorithm as it outperformed linear regression, gaussian naive bayes, and decision tree regression in terms of prediction accuracy and model generality during our preliminary tests. Since SVR is not scale invariant, we normalize features based on minimum / maximum values from the dataset. We evaluate prediction accuracy using root mean squared errors (RMSE), as it is common practice on regression models.

Feature selection. We perform feature selection with stepwise regression, a method which iteratively adds features to the feature vector in order to minimize prediction error. Figure 7 shows prediction accuracy as we increase the number of features. We find that using more than six features is unecessary, as prediction accuracy flattens out (or reduces in the case of YouTube). Table III shows the best feature subset per application. We show results for predictors using two set of features: $f^{DMOS}_{Wi-Fi,best}$, a predictor that uses the features from Table III; and $f^{DMOS}_{Wi-Fi,means}$, a predictor that considers the features available in the deployment dataset (§V), namely the means of TxPHY, RSSI, BUSY, WiFi, and nonWiFi.

Parameter selection. We configure three SVR parameters. SVR uses the C parameter to penalize misclassified samples and γ parameter to set the range of influence of each support vector in the model. High values of C and γ are known to over-fit the dataset. The ϵ parameter regulates the error margin over which predictions incur into no penalty. We find the best combination of parameter through grid optimization, with tests where $C \in [1, 1000]$, $\epsilon \in [0.01, 1]$, and $\gamma \in [0.1, 100]$ at regular intervals. First, we use RMSE from ten-fold cross-validation to track which combination of SVR parameters works best for each predictor. We found that $\epsilon = 0.3$ works well for all applications, but several combinations of γ and C produce low RMSE values. Since SVR models with high γ tend to generate high-variance models, we choose a low γ parameter (in our case, $\gamma = 3$) alongside the C which minimizes RMSE.

Web page complexity. For web browsing, we build one predictor per page, because the baseline PLT of different web pages vary significantly (e.g., 0.287s for *bing.com*, 2.849s for *facebook.com*). For simplicity, we summarize web QoE results using three different pages that vary significantly in terms of complexity: google.be (web simple), facebook.com (web average), and amazon.fr (web complex).

Model visualization. To understand how Wi-Fi metrics influence the decision of predictors, we show the decision curves for SVR models using two features: TxPhy_{mean} and BUSY_{mean} in Figure 10. The SVR models learn similar boundary conditions. When Wi-Fi conditions are perfect, with BUSY_{mean} near 0 and TxPhy_{mean} at maximum, we observe a predicted DMOS above 4.5 on all predictors. Similarly, when either TxPhy_{mean} is close to 0 or BUSY_{mean} close to 100%, we observe a predicted DMOS below 1.5.

Figures 10a and 10b shows the SVR models for video RTC and YouTube QoE. We observe that both models predict reduced QoE when TxPhy_{mean} is below 15 Mbps. This indicates that these applications are sensitive to Wi-Fi bottlenecks. We also see that YouTube is more resilient to Wi-Fi impairments. This is due to the adaptive bitrate selection, which dynamically adjusts video bitrate to match available bandwidth, avoiding buffering events. Figure 10c shows the SVR model for predicting audio QoE. Audio impairments only happen when Wi-Fi conditions are extremely poor. We also observe less cases where predicted QoE is between 1.5 and 4.5. This is because the audio RTC application has very little room for adaptation, so either Wi-Fi quality supports audio, in which case it works well, or it does not and audio quality is very poor or even the audio call drops.

B. Validation

To validate our predictors, we generate the "Office" dataset in an uncontrolled environment (i.e. our lab), over four weeks in 2016. We vary the position of STA1 according to Figure 9 and automatically execute experiments every 30 minutes, from 7am to 10pm, in order to observe different levels of link speed and interference. Each experiment consists of multiple runs as described in §II-C. Then, we select 22 experiments where we observe the most diverse set of Wi-Fi parameters. We follow the same procedure as to generate the testbed dataset to obtain the estimated DMOS per application, f^{DMOS}_{app}.

We evaluate prediction errors of the two predictors, $f^{DMOS}_{Wi-Fi,best}$ and $f^{DMOS}_{Wi-Fi,means}$ on the Wi-Fi testbed dataset through ten-fold cross-validation. We also evaluate $f^{DMOS}_{Wi-Fi,means}$ using the Office dataset, i.e., in these results we learned the predictor using the testbed dataset, and evaluate the accuracy using the office dataset.

Figure 8 shows prediction errors for each of these cases for each application. We observe that the error of $f^{DMOS}_{Wi-Fi,means}$ is only slightly higher than that of $f^{DMOS}_{Wi-Fi,best}$. This result indicates that $f^{DMOS}_{Wi-Fi,means}$, which use simple means, is sufficient to estimate the impact of Wi-Fi quality on QoE. We observe higher prediction errors on the Office dataset. These errors are closer to what we expect in a real user environment, given that the predictor was not learned using this dataset.

C. Detection of Poor QoE

Our predictors so far work on a per-application basis, but we have no information on the specific applications that home users are running. Instead, our goal is to focus on the worst cases, i.e., when we predict that multiple applications experience poor QoE. To identify these poor QoE samples in our Wi-Fi quality parameter space, we generate a synthetic set by sweeping all combinations of Wi-Fi parameters. For

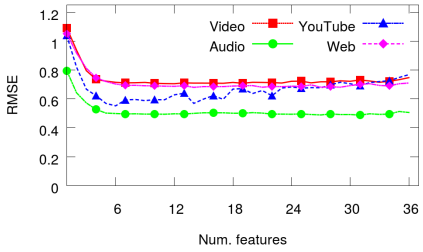


Fig. 7: Error for predictors with different number of features.

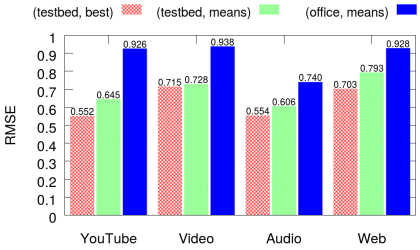


Fig. 8: Predictor evaluation.

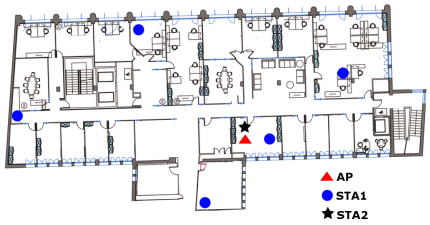
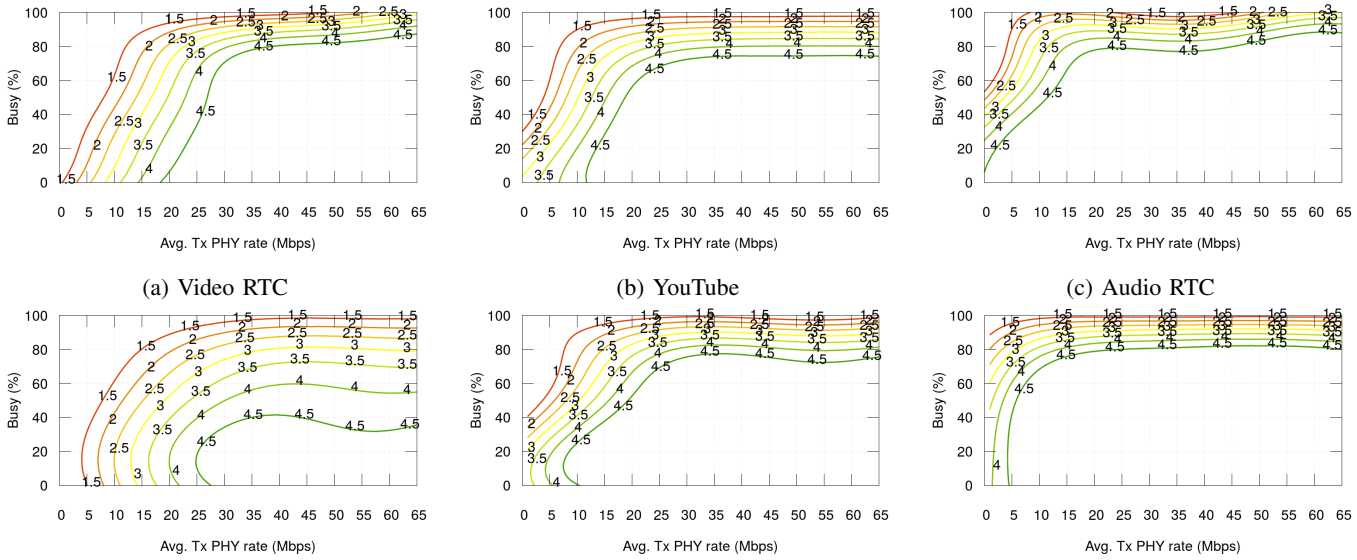


Fig. 9: AP / STA1 position on validation dataset



(a) Video RTC (b) YouTube (c) Audio RTC

(d) Web complex (e) Web average (f) Web simple

Fig. 10: Visualization of SVR models with two features: TxPhy$_{mean}$ and BUSY$_{mean}$.

this analysis, we consider the predictor $f^{DMOS}_{Wi-Fi,means}$. We only generate samples where BUSY < 100 and BUSY = WiFi + nonWiFi. For each Wi-Fi sample, x_{Wi-Fi}, we obtain a six-tuple where each element is the output of one of our per-application predictors (YouTube, Video, Audio, and Web complex, average, and simple), $f^{DMOS}_{Wi-Fi,means}(a, x_{Wi-Fi})$. Then, we apply a clustering algorithm to discover groups of related samples.

We find clusters using the K-means algorithm, which is a simple and efficient clustering algorithm. One challenge with K-means is to select the number of clusters, K. We do not expect to find rigid borders between clusters, since the QoE predictors produce continuous values between 1 and 5. We test K values between 2 and 30, and find that the fraction of explained variance, used to quantify intra-cluster similarity, increases very slowly for $K > 6$. Furthermore, a small number of clusters eases our manual analysis to identify their meaning of clusters.

Table IV shows cluster centroids for K=6, ordered by the Euclidean norm. We manually analyze each cluster. Clusters C5 and C4 generally present high DMOS on most applications, with DMOS ≥ 4.0 on all but two application. Clusters C0, C1, and C2 present DMOS < 3.0 on the majority of application. Figures 11 and 12 show cluster disposition on Wi-Fi QoS space, considering that only Wi-Fi interference or non Wi-Fi interference generates medium occupancy, respectively. It's interesting to note that cluster are cohesive in the Wi-Fi QoS. Also, clusters associated with low DMOS (e.g. C0) are located in regions of known poor Wi-Fi configurations (e.g. low PHY rate and high medium occupation).

TABLE IV: Cluster centroids in QoE space.

	C0	C1	C2	C3	C4	C5
Video	1.42	1.42	2.98	3.63	4.55	4.71
Audio	1.62	2.62	3.61	4.27	4.73	4.83
YouTube	1.35	1.49	2.51	3.37	4.43	4.87
Web complex	1.14	1.36	1.55	2.18	2.98	4.29
Web average	1.19	1.82	1.52	2.74	3.90	4.91
Web simple	1.74	4.48	2.70	4.30	4.74	4.98

Poor QoE samples. Clusters C3, C2, C1, and C0 contain samples where predicted DMOS < 4.0 on multiple applications. Therefore, we classify samples from clusters C3, C2, C1 and C0 as *poor QoE* samples. We call samples from cluster C4 and C5 as *good QoE* samples.

V. Effect of Wi-Fi Quality

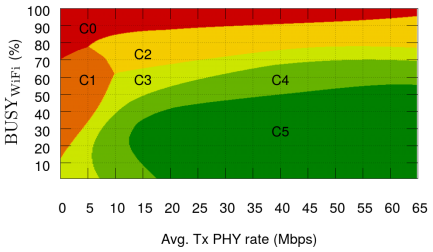


Fig. 11: Cluster classification: Wi-Fi interference

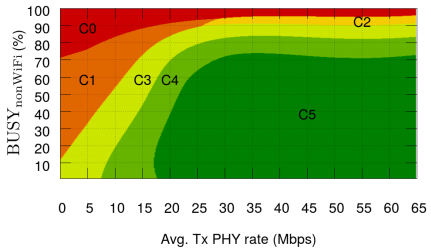


Fig. 12: Cluster classification: non Wi-Fi interference

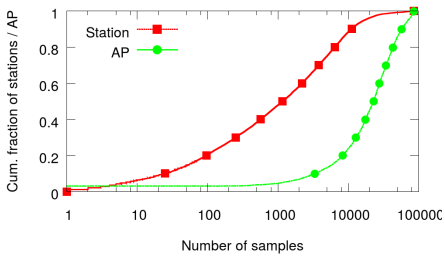


Fig. 13: Number of samples per station and per AP.

TABLE V: Station Wi-Fi technology on deployment.

Technology	Before filtering		After filtering	
	stations	samples	stations	samples
.11n 1x1	17784	58.5M	13496	58.1M
.11n 2x2	12272	72.1M	9729	71.8M
.11g	964	3.3M	636	3.3M
Total	**31020**	**133.9M**	**23861**	**133.2M**

ON QOE IN THE WILD

In this section, we analyze the effects of Wi-Fi quality on QoE in the wild. We collect Wi-Fi measurements from 3479 APs of customers of a large Asian-Pacific residential ISP, with 31020 stations connected to these APs, during the month of September, 2016. We analyze 133.9 million station samples.

A. Deployment dataset

The AP logs Wi-Fi performance every second for all stations connected to it. This AP collects all the metrics from Table II except for FDR. Our backend system polls the APs to collect the mean metrics every 30s. Since station metrics such as TxPhy need traffic to reflect the link quality, we filter out samples from inactive stations.

First, we look at deployment characteristics regarding station usage and technology. Figure 13 shows the number of samples per stations and per AP, where one AP sample contains one or more valid station samples. We find some stations with very little activity: we have less than 10 samples on 6.5% of stations. This is because some home Wi-Fi appliances seldom transfer data, such as "smart scales", and there is not enough traffic to provide valid samples. There are also APs with a small number of samples (2.99% had less than one hour of data). Figure 14 shows the cumulative fraction of the number of active days per stations/APs. We consider one station *active* if we observe at least 5 minutes of activity over 24 hours (similarly for the AP). 23.6% of stations have one or less active days, which indicates a device belonging to visitors or sporadically used.

We focus our study on APs and stations that actively use the Wi-Fi network. We filter stations with less than two active days per week (leaving 76.9% of stations for our analysis) and APs with less than five active days per week (we analyze 92.29% of APs). Table V shows the number of stations observed per technology before and after filtering, as well as the number of samples. Notice that we retain 99.4% of all samples, although we filter 23.1% of stations. Over 34% of .11g stations have less than 2 active days per week, which suggests that stations with legacy technology are more often infrequently used.

TABLE VI: Cluster frequency in deployment.

	C0	C1	C2	C3	C4	C5
Frequency	0.26%	0.59%	0.68%	9.93%	5.47%	83.1%

B. QoE predictions per application

Figure 15 shows the cumulative fraction of predicted DMOS per application in the deployment. We observe that over 68% of samples contain predicted DMOS above 4.5 for all applications. This is reassuring as most of the time, Wi-Fi works well for all applications. We observe instances where predicted DMOS < 3.0 for all applications. We predict DMOS below 3.0 for web complex and video RTC on 14.8% and 5.8% of samples, for YouTube and web average on 3.3% and 2.5% of samples, and for audio RTC and web simple on 1.2% and 1.1% of samples. The difference in frequency between applications is due to the application sensitivity to the Wi-Fi impairments. All samples with mean TxPHY up to 6.5 Mbps show predicted DMOS below 3.0 for web complex, on a total of 9.14% of all deployment samples. On other predictors, this only occurs when we also observe interference.

C. QoE across applications

Some applications require better Wi-Fi quality than others, with a larger fraction of samples where predicted DMOS $<$ 3.0. Now, we would like to investigate how the same Wi-Fi quality affects QoE for different applications. We rely on the clusters from §IV-C. Table VI shows the fraction of the samples in the deployment dataset in each cluster. Fortunately, we find that nearly 88.4% of samples in the deployment fall in Clusters C4 and C5, which have good QoE. We classify 11.6% of deployment samples as poor QoE samples.

Figure 16 shows the fraction of poor QoE samples per station. ISPs can use the fraction of poor QoE samples to identify stations with frequent Wi-Fi-related QoE issues. The majority of stations (53%) have less than 5% poor QoE samples, therefore, Wi-Fi is unlikely to impair user experience in those cases. 21% of stations, however, have more than 25% poor QoE samples, which represent frequent Wi-Fi impairments. For the rest of this analysis, we focus on stations with at least 5% poor QoE samples.

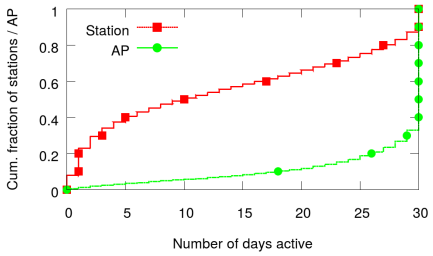


Fig. 14: Number of days with collected data per stations / AP.

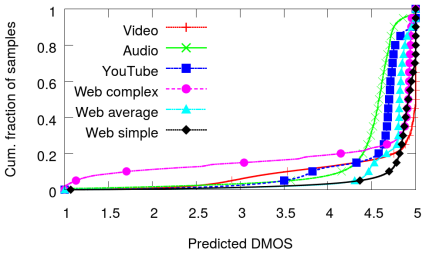


Fig. 15: Predicted DMOS frequency in the wild.

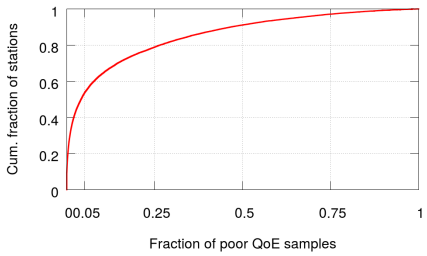


Fig. 16: Fraction of poor QoE samples per station

D. Characterization of poor QoE events

We define a *poor QoE event* as a series of poor QoE samples spaced by no more than a time threshold T. We select T based on the distribution of the inter-arrival times of samples (not shown for brevity). We use $T = 2$ minutes, because it represents the knee of the curve; 74% of all samples have inter-arrival time ≤ 2 minutes. Notice that our definition of poor QoE events allow for some good QoE samples to occur in-between poor QoE samples. Even in these cases, users are likely to experience poor quality during the poor QoE event, since Wi-Fi quality is intermittent.

Figure 17 shows the duration of poor QoE events, for T = 2 minutes. The majority of poor QoE events are short: 78% have duration of two minutes or lower. Mobile stations going in or out of Wi-Fi range or temporary Wi-Fi impairments, such as those caused by a peak of Wi-Fi interference, are some of many possible reasons for short-lived poor QoE events. We also observe a number of long poor QoE events, some stretching over several hours. These more persistent Wi-Fi problems can happen when either the station is poorly located or the AP is using a busy Wi-Fi channel.

We categorize poor QoE events into three distinct classes. Short-lived poor QoE events are those with duration below or equal 2 min. We divide other events between consistent and intermittent. Intermittent events have good QoE samples in-between poor QoE samples. If the fraction of poor QoE samples during a poor QoE event is greater or equal to 80%, we call it a consistent poor QoE event, otherwise it is an intermittent poor QoE event.

Figure 18 shows the division between short, intermittent, and consistent poor QoE events. We group stations based on their fraction of poor QoE samples. For stations with fraction of poor QoE samples below 5%, 51% of poor QoE samples belong to short or intermittent poor QoE events. Intermittent and short-lived poor QoE events are difficult to detect and diagnose, since when we detect a good QoE samples after a series of poor QoE samples it is unclear if the root cause was fixed or is only temporarily absent. These stations require a long-term monitoring approach to detect and diagnose Wi-Fi problems. We observe that stations with a larger fraction of poor QoE samples have less short and intermittent events, and more consistent poor QoE events. For stations with fraction of poor QoE samples above 50%, we observe, on average, 69% of poor QoE samples on consistent poor QoE events. While this holds true for the majority of stations, there are some exceptions. Some stations with a very high fraction of poor QoE samples have nearly all samples on short poor QoE events. In these cases, we found that this occurs because they generate traffic for very short periods, with long inactive intervals in-between. This prevents us from classifying poor QoE samples into long poor QoE events.

TABLE VII: Summary of the diagnosis of poor QoE events.

TxPHY < 15	**BUSY** > 60		**Consistent**	**Intermittent**
	↑ WiFi	**↑ nonWiFi**		
✓	✓	✗	4.0%	0.9%
✓	✗	✓	4.3%	0.4%
✓	✗	✗	49.9%	19.7%
✗	✓	✗	5.0%	3.3%
✗	✗	✓	3.9%	4.3%
✗	✗	✗	1.4%	2.9%
Total			**68.5%**	**31.5%**

Diagnosis of poor QoE events. Next, we investigate what is the most likely cause for intermittent and consistent poor QoE events. We use simple thresholds on the Wi-Fi metrics to diagnose the cause of poor QoE events.

Figure 19 shows TxPHY_{mean} and BUSY_{mean} for consistent and intermittent poor QoE events. We look at mean Wi-Fi metrics along all samples of a poor QoE event to identify its most likely cause. We find that, in general, poor QoE events show either $\text{TxPHY}_{mean} < 15$ Mbps or $\text{BUSY}_{mean} > 60\%$. We diagnose events where $\text{TxPHY}_{mean} < 15$ Mbps as having *low PHY rate*, and events where $\text{BUSY}_{mean} > 60\%$ as having *high medium occupation*. On poor QoE events with high medium occupation, we get further insight on the root cause by comparing WiFi_{mean} and nonWiFi_{mean}.

Table VII summarizes this analysis. Low PHY rate is the most common cause for long poor QoE events. We find that 79.2% of poor QoE events show low PHY rate, of which 49.9% and 19.7% are consistent and intermittent poor QoE events with only low PHY rate. On 9.6% of poor QoE events we observe both low PHY rate and high medium occupation. The presence of both kinds of Wi-Fi impairments makes it more likely to occur consistent (8.3%) than intermittent (1.3%) poor QoE events. On 16.5% of poor QoE events we observe high medium occupation and not low PHY rate, with a balance between consistent (8.9%) and intermittent (7.6%) poor QoE events. We further identify the root cause of the high

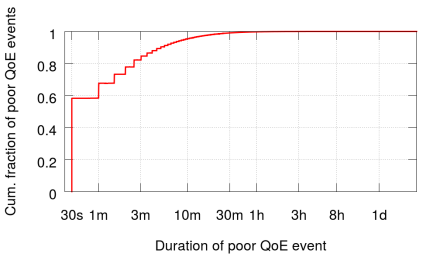


Fig. 17: Duration of poor QoE events for $T = 2\ min$.

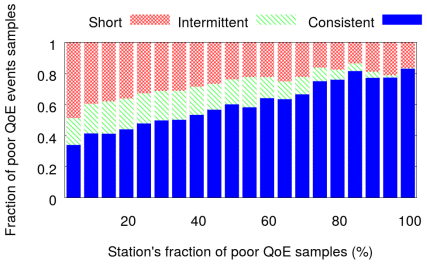


Fig. 18: Classification of poor QoE samples per group of station.

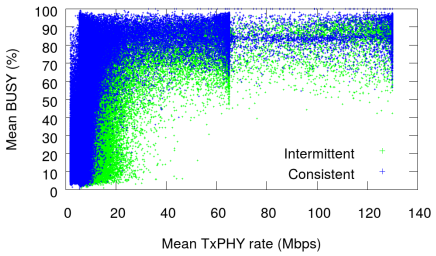


Fig. 19: Average Wi-Fi metrics of poor QoE events.

medium occupation as Wi-Fi interference when $\text{WiFi}_{mean} > \text{nonWiFi}_{mean}$, and non Wi-Fi interfernce otherwise. We find a very similar number of poor QoE events caused by Wi-Fi (8.3%) and non Wi-Fi (8.2%) interference. Interestingly, we found consistent poor QoE events more often under Wi-Fi interference and intermittent poor QoE events more often under non Wi-Fi interference. Finally, we observe 4.3% poor QoE events with no clear indication of Wi-Fi problems by looking at QoS metrics alone. These cases occur on the prediction boundary of poor QoE samples and they occur more often for intermittent events.

VI. Related Work

Quality of Experience. Subjective quality assessment uses human subjects to explicitly evaluate user experience [16], [18], [25], [33]. Objective quality assessment proposes quality metrics that correlate with user opinion, but can be calculated without user involvement [33]–[35]. We use objective quality metrics to assess the effect of Wi-Fi quality on QoE. Mapping network QoS to QoE for a number of applications has been a popular research topic [4], [7], [20], [26], but none of these studies consider the mapping of Wi-Fi QoS to QoE. There are only two exceptions. Chakraborty et al. [5] propose ExBox, a system that relies on predicting the impact of an incoming flow on QoE of a set of flows to make flow admission control decisions in Wi-Fi entreprise networks. ExBox assumes that some of the stations in the Wi-Fi network are instrumented to report application QoS metrics for online learning of a binary classifier that decides whether incoming flows can be accepted or not. The only Wi-Fi metric that ExBox considers is SNR. In contrast, our goal is detection and diagnosis in home Wi-Fi, which brings different operational constraints (e.g., no knowledge of applications, no instrumentation of user devices). Although we study a similar set of applications, we offer a more in-depth study of the relationship between multiple Wi-Fi metrics and QoE. Closest to our goal is a workshop paper that studies the effect of Wi-Fi on Web QoE [10]. Our paper improves on their Web QoE model, adds predictors for three other applications, proposes a definition of poor QoE events, and performs a characterization of poor QoE in the wild.

Wi-Fi performance characterization. Pei et al. [23] deploy 47 APs in a university campus and show that the Wi-Fi link significantly contributes to end-to-end latency. Ioannis et al. [22] characterize the Wi-Fi performance with passive measurements from 167 homes, over a period of four months. Biswas et al. [2] characterize Wi-Fi network usage and performance on over 20 thousand industrial networks. These studies improve our understanding of Wi-Fi quality in different settings, but none of them addresses the issue of how Wi-Fi quality affects QoE as we do.

VII. Conclusion

This paper was the first to shed light on the effects of home Wi-Fi quality on QoE of different applications. Our first contribution was to develop a method to detect instances of poor QoE from Wi-Fi quality metrics. We showed that we can predict application QoE from Wi-Fi metrics available on commodity APs with low errors. We built on these predictors to identify poor QoE samples. ISPs can use this method to detect when Wi-Fi quality is likely to degrade customers' QoE to proactively fix Wi-Fi problems. In fact, we have interest from ISPs to incorporate these predictors with the Wi-Fi monitoring system, which is already deployed in field trials. Our second contribution was to characterize the effect of Wi-Fi on QoE in the wild. We apply our predictors on Wi-Fi metrics collected from 3,479 APs over one month. Our results were reassuring in that in the vast majority of samples Wi-Fi quality can sustain good QoE; still we found 21% of stations have a fraction of poor QoE samples above 25%. ISPs can use the fraction of poor QoE samples to identify stations with frequent poor QoE due to Wi-Fi. In particular, intermittent events are challenging to troubleshoot and require a long-term monitoring approach, as we propose here.

References

[1] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a predictive model of quality of experience for internet video. In *ACM SIGCOMM CCR*, 2013.

[2] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-e, A. Bhartia, and D. Aguayo. Large-scale measurements of wireless network behavior. In *Proc. ACM SIGCOMM*, 2015.

[3] E. Bocchi, L. De Cicco, and D. Rossi. Measuring the quality of experience of web users. In *Proceedings of the 2016 workshop on QoE-based Analysis and Management of Data Communication Networks*. ACM, 2016.

[4] P. Casas, M. Seufert, and R. Schatz. YOUQMON: A system for online monitoring of YouTube QoE in operational 3G networks. *ACM SIGMETRICS Performance Evaluation Review*, 2013.

[5] A. Chakraborty, S. Sanadhya, S. R. Das, D. Kim, and K.-H. Kim. Exbox: Experience management middlebox for wireless networks. In *CoNEXT*, 2016.

[6] L. DiCioccio, R. Teixeira, J. Kurose, and M. May. Pinpointing home and access network delays and losses using wifi neighbors. Technical report, UPMC, 2012.

[7] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki. Measuring video QoE from encrypted traffic. In *Proc. IMC*. ACM, 2016.

[8] W. K. Edwards and R. E. Grinter. At home with ubiquitous computing: Seven challenges. In *International Conference on Ubiquitous Computing*. Springer Berlin Heidelberg, 2001.

[9] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler. Waiting times in quality of experience for web based services. In *QoMEX*. IEEE, 2012.

[10] D. N. d. Hora, R. Teixeira, K. Van Doorselaer, and K. Van Oost. Predicting the effect of home Wi-Fi quality on Web QoE. In *Proc. SIGCOMM Internet QoE workshop*. ACM, 2016.

[11] E. Ibarrola, F. Liberal, I. Taboada, and R. Ortega. Web QoE evaluation in multi-agent networks: validation of ITU-T G. 1030. In *Proc. Int. Conf. on Autonomic and Autonomous Systems*. IEEE, 2009.

[12] ITU-T. Methods for subjective determination of transmission quality, Aug 1996.

[13] ITU-T. Recommendation P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, 2001.

[14] ITU-T. Recommendation P.862 annex a: Reference implementations and conformance testing for ITU-T Recs P.862, P.862.1 and P.862.2, 2005.

[15] ITU-T Recommendation G.1030. Estimating end-to-end performance in ip networks for data applications, 2005.

[16] D. Joumblatt, J. Chandrashekar, B. Kveton, N. Taft, and R. Teixeira. Predicting user dissatisfaction with internet application performance at end-hosts. In *Proceedings of IEEE INFOCOM*, 2013.

[17] P. Kanuparthy, C. Dovrolis, K. Papagiannaki, S. Seshan, and P. Steenkiste. Can user-level probing detect and diagnose common home-wlan pathologies. *ACM SIGCOMM CCR*, 2012.

[18] M. Katsarakis, R. Teixeira, M. Papadopouli, and V. Christophides. Towards a Causal Analysis of Video QoE from Network and Application QoS. In *Proceedings of the 2016 workshop on QoE-based Analysis and Management of Data Communication Networks*. ACM, 2016.

[19] K. Lakshminarayanan, S. Seshan, and P. Steenkiste. Understanding 802.11 performance in heterogeneous environments. In *Proc. SIGCOMM workshop on Home networks*. ACM, 2011.

[20] M. Papadopouli, P. Charonyktakis, M. Plakia, and I. Tsamardinos. On user-centric modular QoE prediction for VoIP based on machine-learning algorithms. *IEEE TMC*, 2015.

[21] A. Patro, S. Govindan, and S. Banerjee. Observing home wireless experience through wifi aps. In *Proc. international conference on Mobile computing & networking*. ACM, 2013.

[22] I. Pefkianakis, H. Lundgren, A. Soule, J. Chandrashekar, P. Le Guyadec, C. Diot, M. May, K. Van Doorselaer, and K. Van Oost. Characterizing home wireless performance: The gateway view. In *Proc. IEEE INFOCOM*, 2015.

[23] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei. Wifi can be the weakestlink of round trip network latency in the wild. In *Proc. IEEE INFOCOM*, 2016.

[24] S. Rayanchu, A. Patro, and S. Banerjee. Airshark: detecting non-wifi rf devices using commodity wifi hardware. In *Proc. IMC*. ACM, 2011.

[25] R. Schatz, T. Hoßfeld, L. Janowski, and S. . From packets to people: quality of experience as a new measurement challenge. In *Data traffic monitoring and analysis*. Springer, 2013.

[26] T. Spetebroot, S. Afra, N. Aguilera, D. Saucez, and C. Barakat. From network-level measurements to expected quality of experience: the skype use case. In *M & N workshop*, 2015.

[27] D. Strohmeier, S. Egger, A. Raake, T. Hoßfeld, and R. Schatz. Web browsing. In *Quality of Experience*. Springer, 2014.

[28] L. Sun and E. C. Ifeachor. Voice quality prediction models and their application in voip networks. *Transactions on multimedia*, 2006.

[29] S. Sundaresan, N. Feamster, and R. Teixeira. Home network or access link? locating last-mile downstream throughput bottlenecks. In *PAM*. Springer, 2016.

[30] I. Syrigos, S. Keranidis, T. Korakis, and C. Dovrolis. Enabling wireless lan troubleshooting. In *Proc. PAM*. Springer, 2015.

[31] M. Varela, L. Skorin-Kapov, T. Maki, and T. Hosfeld. QoE in the Web: A dance of design and performance. In *QoMEX*. IEEE, 2015.

[32] M. Varvello, J. Blackburn, D. Naylor, and K. Papagiannaki. Eyeorg: A platform for crowdsourcing web quality of experience measurements. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. ACM, 2016.

[33] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hossfeld. Modeling the YouTube stack: From packets to quality of experience. *Computer Networks*, 2016.

[34] T. Wang, A. Pervez, and H. Zou. VQM-based QoS/QoE mapping for streaming video. In *Proc. IC-BNMT*. IEEE, 2010.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), 2004.

[36] Z. Wang, L. Lu, and A. C. Bovik. Video quality assessment based on structural distortion measurement. *Signal processing: Image communication*, 19(2), 2004.

[37] Xiph.org media. http://media.xiph.org/video/derf/.

[38] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hoßfeld. Impact of frame rate and resolution on objective QoE metrics. In *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*. IEEE, 2010.