Prometheus: Toward Quality-of-Experience Estimation for Mobile Apps from Passive Network Measurements

Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Shobha Venkataraman, He Yan
AT&T Labs - Research
1 AT&T Way, Bedminster, NJ
{vaneet,emir,jeffpang,shvenk,yanhe}@research.att.com

ABSTRACT

Cellular network operators are now expected to maintain a good Quality of Experience (QoE) for many services beyond circuit-switched voice and messaging. However, new smart-phone "app" services, such as Over The Top (OTT) video delivery, are not under an operator's control. Furthermore, complex interactions between network protocol layers make it challenging for operators to understand how network-level parameters (e.g., inactivity timers, handover thresholds, middle boxes) will influence a specific app's QoE. This paper takes a first step to address these challenges by presenting a novel approach to estimate app QoE using passive network measurements. Our approach uses machine learning to obtain a function that relates passive measurements to an app's QoE. In contrast to previous approaches, our approach does not require any control over app services or domain knowledge about how an app's network traffic relates to QoE. We implemented our approach in *Prometheus*, a prototype system in a large U.S. cellular operator. We show with anonymous data that Prometheus can measure the QoE of real video-on-demand and VoIP apps with over 80% accuracy, which is close to or exceeds the accuracy of approaches suggested by domain experts.

1. INTRODUCTION

The proliferation of smartphone apps has dramatically expanded the services that cellular operators support. Subscribers now want wireline-comparable QoE for a vast array of rapidly evolving applications ("apps"), including OTT video delivery, voice over IP (VoIP), real-time video calling, and other apps [23]. Hence, monitoring app *QoE metrics*, which we define as objective measures of specific app services (e.g., number of video stalls for a video-on-demand app), is critical because there are direct relationships between QoE metrics, customer satisfaction and business objectives [1, 4, 13]. Unfortunately, in the industry-to-date, there is little understanding of how network operators can measure QoE metrics of the diverse array of mobile app services or how they can tune their networks to improve a specific QoE metric.

Measuring app QoE metrics is challenging for cellular operators due to three reasons: lack of control, limited vantage points, and protocol complexity. Unlike circuit-switched voice and messaging, the vast majority of app protocols run as OTT services that are not under an operator's control. Thus, operators generally do not have access to the logs of app services or knowledge of packet payload formats, which are required by previous QoE estimation techniques (e.g., [16, 19]). Moreover, the physical environment plays a critical role in performance that wireless users experience. As a consequence, it is impractical to use active probes to measure QoE at scale over the entire range of possible vantage points. Finally, applications running over cellular networks can have complex application-layer protocols and experience complex interactions between a number of different network layers. This complex layering of protocols results in trade-offs in performance characteristics, such as interactive latency vs. capacity [14] and average throughput vs. link disruptions [9]. Thus, the relationships between network parameters and QoE metrics is often poorly understood.

In this paper, we present a new approach for operators to estimate QoE metrics for arbitrary apps. We hypothesize that there is a learn-able relationship between the behavior of an app's network traffic and each relevant QoE metric. Our approach is to find this relationship, i.e., a function F : $\mathsf{app\ traffic} \rightarrow \mathsf{app\ QoE\ metric}$. With such a function, an operator can estimate the app's QoE metrics over all its subscribers simply by monitoring the app's network traffic.

To overcome the challenges presented above, we combine data collected in controlled experiments and anonymized network data passively collected for mobile handsets. Our approach works by first collecting a *training set* where the operator has both the app traffic and the app QoE metric, using instrumentation on a few mobile devices. We then use a machine-learning algorithm to learn the function F, similar in spirit to video service providers' recent efforts to estimate user satisfaction from video QoE metrics [1]. Unlike service providers, network operators have no control over the endpoints to collect the "ground truth" of user satisfaction.

To evaluate the efficacy of our approach in practice, we built *Prometheus*, a prototype implementation in a large U.S. cellular operator. Using Prometheus, we apply our approach to two widely used smartphone app services: a popular video-on-demand app and a VoIP app. We show that *Prometheus* can estimate the relevant QoE metrics of these two apps (e.g., video stalls and Mean Opinion Score) more accurately or very close to approaches suggested by domain experts.

A key feature of our approach is that it *does not require detailed domain knowledge* about app protocols, which often takes too long to acquire and relate to QoE in the rapidly evolving ecosystem of smartphone apps. Previously proposed frameworks for measuring QoE (e.g., [10, 18, 19, 22]), have relied on domain knowledge to relate network metrics to QoE. Moreover, while previous studies have shown that machine learning can be used to model user satisfaction (e.g., [3, 11]), they relied on manual user input to collect training data. Prometheus is the first to show promise in the synthesis of QoE measurement frameworks and automated modeling using machine learning.

We evaluated Prometheus on a modest dataset and our results suggest that learned models can accurately estimate the QoE of apps nearly as well or better than models devised by domain experts. Our models predict QoE of video-on-demand more accurately than a throughput-based model used in practice by 8 percentage points. Our model predicts QoE of a VoIP app within 3 percentage points of well-known domain expert model, even though we do not use any domain knowledge to generate it. Furthermore, our initial results suggest that our models are robust to certain types of training bias. Thus, our initial study suggests that automatically learned QoE models are promising, particularly since they overcome the challenges of lack of control, limited vantage points, and protocol complexity.

2. THE PROMETHEUS APPROACH

This section presents the design and architecture of our approach. At a high level, our approach approximates a function F : app traffic $\rightarrow$ app QoE metric, given a specific app QoE metric.

We first illustrate our approach with two example apps: VideoApp and VoIPApp. VideoApp is a popular video streaming service (anonymized for business confidentiality) that uses progressive HTTP streaming [7]. That is, it uses HTTP to download a short video, which is buffered for a short time before playback starts. If the playback out-paces the download of the video, then the playback stalls until the video is again sufficiently buffered. Buffering Ratio is one app QoE metric that we want to estimate and is defined as the ratio of the time a video spends stalled (i.e., buffering) and the total video playback time. It is well established that buffering ratio is negatively correlated with user satisfaction [1, 13]. Approximating buffering ratio from network traffic is non-trivial because stalls depend on many non-network factors, such as how much the player buffers, the rate-pacing strategy of the video server, the variability in video bit-rate, etc.

VoIPApp is CSipSimple, a popular open source VoIP application that uses SIP protocol. An important QoE metric that we want to estimate for VoIPApp is an objective Mean Opinion Score (MOS), such as PESQ-MOS [15]. This score is defined as a value between -0.5 (worst) and 4.5 (best) that represents the quality of the output audio signal at the receiver compared to the input audio at the sender. As with Buffering Ratio, the network provider can not observe the input or output audio signals, so the challenge is to approximate PESQ-MOS from the network traffic.

Since business objectives and anomaly detection applications typically dictate thresholds for acceptable levels of customer satisfaction, we formulate the QoE metric we wish to predict as a binary variable: does the network traffic satisfy the QoE threshold or not? In other words,
$F(\text{network traffic}) = 1$ if QoE metric $\geq T$, and 0 otherwise, where QoE metric is either Buffering Ratio or PESQ-MOS.

Our goal is to find a function that accurately approximates each ideal function F — that is, a function that matches the empirical output of F with high probability. To find this function, our approach proceeds in five stages, as depicted in Figure 1.

Instrumentation. The first step is to design a benchmark that executes a user interaction with the app from which we can directly measure the app QoE metric, i.e., video stalls. For VideoApp, the benchmark is simply playing a random video from the VideoApp service and monitoring the screen to see whether the video stalls or not. A modest number of phones can be instrumented to perform these benchmarks automatically and repeatedly using GUI testing frameworks (e.g., [2]). This is the only stage that requires human intelligence, as the benchmark to measure the app QoE metric is designed and scripted by a person. However, no domain knowledge of the app internals or network protocols are required — the designer merely needs to understand how to use the app's GUI. In our prototype implementation on Android, we use the `VideoView` API to play random standard quality videos from this popular video service. We limited the duration of video playbacks to 5 minutes by stopping the video player at that time (the average duration was approximately 4 minutes).

For VoIPApp, we implemented a benchmark that received an incoming call on the handset, initiated by an automated call server. The server plays a random 16-bit PCM/8 kHz audio file containing a recording of a selection of Harvard Sentences, a standard data set designed for speech quality testing.

Data Collection. Once a few phones are instrumented, the benchmarks are automatically and repeatedly executed on them in a range of real conditions on the cellular network. For example, an operator can instrument the phones it already sends out with field testers. Because the tests are automated, no user interaction is required while running the tests and a large number of benchmarks can be run in a short period of time. Two sets of data are collected for each benchmark b_i: First, the "ground truth" QoE_metric_i is collected by the benchmark instrumentation on the phone (e.g., 0 or 1 for video stalls). Second, passive network data for each app is collected within the cellular network. This passive network data may include IP flow records, which summarize each TCP/UDP flow as a set of metrics (e.g., volume, throughput, loss rate, etc.),[1] and radio layer statistics (e.g., received signal strength). We summarize the network data into a vector $network\_data_i = (f_{i1}, f_{i2}, \ldots, f_{im})$, where f_{ij} is a feature of the passive network data of benchmark b_i, such as average throughput.

Our implementation uses two network data streams to build the *network_data* feature vectors for each benchmark sample. The first set, called flow record features, are derived from IP flow-level measurements collected by actual monitors in the core of the cellular network (Figure 1). Each

[1]Full packet traces could be captured in principle, but it is not cost effective to capture, save, and process full packet captures for all subscribers of a large cellular network.

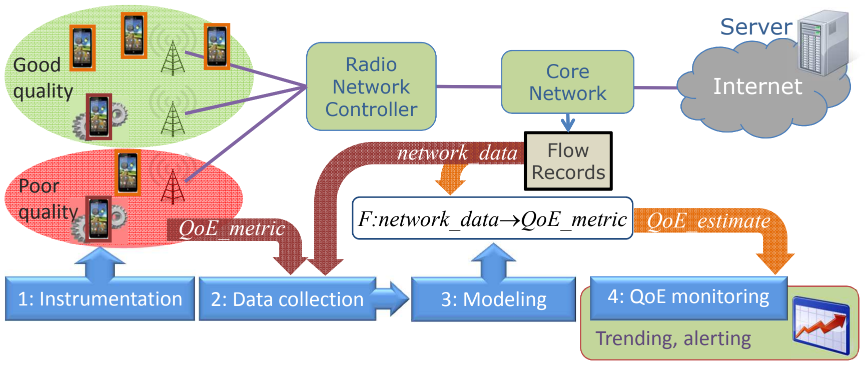

Figure 1: Each stage of our approach.

Table 1: The features in a *network_data* vector.

flow record features (currently collected by probe)
flow duration, app signature, 2G/3G/4G network type, type of service (ToS), ToS count, bytes and packets (throughput, sum, min, max, mean, median per 1-min interval), TCP features (flags, options, retransmissions, out-of-order, 0-win), TCP data (throughput, sum, min, max, mean, median), TCP sequence number range (throughput, sum, min, max, mean, median), HTTP requests, response codes and their counts
timeseries features (for throughput timeseries (TS))
TS stats (min, max, mean, stddev, kurtosis, skewness, 1-norm, frobenius- and infinite-norm), spline interpolation features, TS FFT (power sum, min, max, mean), derivative of TS summary norms, TS periodograms

flow is labeled with a domain and application, so we can determine which flow belongs to which app. One record is generated for each complete TCP or UDP flow and one additional record per minute that the flow is active. To create flow record features for the *network_data* from a benchmark's largest flow, we use all the fields in the flow record and compute basic summary statistics (mean, median, variance, min, max) over the fields in the one minute records. The full list of *network_data* features are described in Table 1.

To evaluate how flow features not yet implemented by the production monitor would improve the model, we use a second set of features called timeseries features. We use `tcpdump` to capture packet traces on the instrumented phones. Although the capture of traces at the phones will not precisely emulate the traces that would have been captured within the network, the summary statistics we compute from flows would be very similar because hybrid ARQ at the radio link layer prevents almost any packet loss on the radio link [12]. To create timeseries features from these traces, we measure a throughput time series, sampled once per second over a flow. From this time series, we extract a number of standard timeseries features, shown in Table 1.

For passive network data for actual subscribers (i.e., in the QoE Monitoring phase below), device and user identifiers are anonymized to protect privacy. In addition, the model output is aggregated (e.g., per region and/or network element), so it does not permit the reversal of anonymization or re-identification of users.

Modeling. The next objective is to approximate the function $F : network\_data \rightarrow QoE\_metric$. We use an offline machine learning algorithm, where we take data from n benchmarks $b_1, b_2, \ldots, b_n$ as the input training set. This data can be viewed as an $n \times m$ input matrix (one row for each $network\_data_i$ vector) and an output vector $(QoE\_metric_1, QoE\_metric_2, \ldots, QoE\_metric_n)$:

$$F : \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1m} \\ f_{21} & f_{22} & \cdots & f_{2m} \\ \vdots & & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nm} \end{bmatrix} \rightarrow \begin{bmatrix} QoE\_metric_1 \\ QoE\_metric_2 \\ \vdots \\ QoE\_metric_n \end{bmatrix}.$$

An important property of the input matrix is that, if viewed as a set of linear equations, it is likely to be severely over-determined. Most of the elements in *network_data* vectors are likely to be unrelated with *QoE_metric*, since we do not have a domain expert to cherry-pick the relevant features f_{ij}. To meet the challenge of this over-determined learning problem, we use LASSO regression [21], which enables us to predict real value as well as categorical *QoE_metric*s. Theoretical analysis indicates that LASSO regression is particularly effective when there are many irrelevant features and only a few training examples, which is why LASSO is a good fit for Prometheus. In addition, LASSO produces a linear equation F'. In F', LASSO associates a coefficient to each feature in the *network_data* feature vectors that indicates the degree to which it influences the *QoE_metric*. By ranking the features by the absolute value of this coefficient, operators can determine the network metrics that most influence the QoE metric. Our implementation uses the `glmnet` implementation of LASSO [8], which uses cyclical coordinate descent to deal efficiently with sparse features.

QoE Monitoring. The output of the previous stage is a function F' that accurately approximates the ideal function F. F' can now be applied to anonymous network data of real subscribers to estimate whether their Buffering Ratio or PESQ-MOS exceeds the desired threshold. We leverage previous work to identify network flows from different apps and to segment them into samples, from which we can extract a *network_data* feature vector [23]. By computing $F(network\_data)$, we obtain estimates of *QoE_metric* for each sample, enabling anomaly detection, trending, and other applications.

Open Questions. A central contribution of our work is to answer two important questions about the efficacy of the approach described above.

Is there sufficient information in the *network_data* to predict *QoE_metric*s for real apps? Our evaluation indicates that our derived models could perform nearly as well or better than models designed by domain experts.

Will biases in the data collection stage cause the learned model to be too inaccurate when applied to anonymous network data from real users? In practice, there are a number of factors that may bias the learned model, e.g. the network conditions we encounter in the data collection stage may differ from those encountered by other users. Our evaluation provides evidence that many of the biases that are introduced do not substantially degrade the accuracy of the learned model.

3. EVALUATION

Our results in this section provide evidence that the answer to the first open question is affirmative and the answer to the second is negative, thereby suggesting the effectiveness of our approach. We first describe some statistics about the data collection we performed using our prototype (§3.1). Second, we show that the accuracy of Prometheus' models on video and VoIP are at least as accurate as models designed by domain experts (§3.2-3.3). Third, we use our data to show that potential biases in data collection do not substantially impact results in most cases (§3.4).

3.1 Experimental Setup

Data Collection. We used the Prometheus prototype to collect data for the VideoApp and VoIPApp QoE metrics described in §2. The data used to train models in this section were collected for Samsung Galaxy S devices carried by four research staff during their daily routine at various times during the fall of 2012. The majority of benchmark samples were collected in five cities in northern New Jersey and the San Francisco Bay Area in California, United States.

Evaluation Criteria. The primary objective of our evaluation is to show that the QoE prediction model generated by Prometheus is accurate. To determine the accuracy of the models, we use a standard method to evaluate machine learning models. We divide the benchmark samples (*network_data*, *QoE_metric* pairs) into two sets: a training set and a test set. We train the LASSO model using the training set and evaluate the accuracy of the model on the test set by comparing the model prediction *QoE_prediction* to the actual *QoE_metric* in the test data. We balance the training set by randomly selecting the same number of positive and negative samples so that we obtain unbiased classifiers — i.e., one with roughly equal true positive rate (TPR) and true negative rate (TNR). Except in §3.4, we partition the set randomly 100 times and report average accuracy over those trials.

3.2 Video QoE Accuracy

Our VideoApp benchmark measured the QoE metric of Buffering Ratio. In total, 1464 benchmarks were conducted. The empirical distribution of Buffering Ratio is shown in Figure 2(a). We observe that 65% of buffering delays are 0 since there are no stalls in those samples.

We compare the accuracy of the Prometheus models with two baseline methods: (1) a naïve baseline (naïve) that predicts the mean value of the training set (or most common value for binary classifications) and (2) a model that uses linear regression on the (total bytes) throughput feature only (throughput-only), i.e., the average throughput of the video flow. The latter model is simple, but used in practice due to the belief that lower quality environments exhibit lower throughput and more stalls.

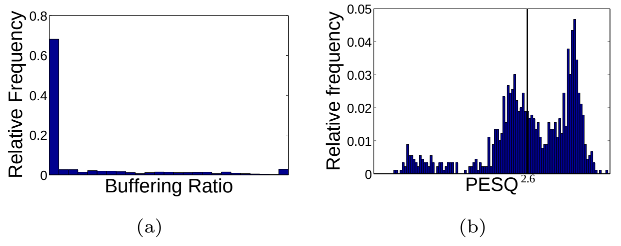

Figure 2: Distribution of (a) Buffering Ratio in the VideoApp, (b) PESQ-MOS in the VoIPApp benchmarks. (These results do not represent a systematic evaluation of this network.)

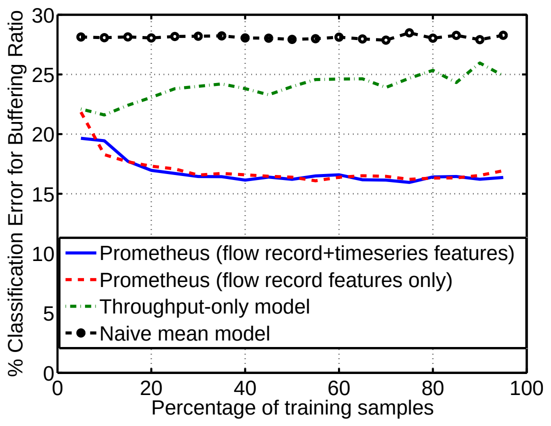

Figure 3: Classification error of thresholded Buffering Ratio.

We set the Buffering Ratio threshold to 0.1, since this is approximately the value when almost all users will stop watching the video [13]. We show classification accuracy of predicting whether Buffering Ratio > 0.1 in Figure 3, as a function of the training set size. The Prometheus models perform better than naïve and throughput-only, achieving classification error of close to only 16%. We found similar accuracy for a model predicting a video's startup delay, another important video QoE metric (results omitted due to space constraints).

3.3 VoIP QoE Accuracy

Our VoIPApp benchmark measured the QoE metric PESQ-MOS. A total of 997 benchmarks were conducted. The empirical distribution of PESQ-MOS values obtained is shown in Figure 2(b). We observe that the distribution has two strong modes.

We again compare the accuracy of the Prometheus models with two baseline methods: (1) a naïve baseline (naïve) that predicts the mean value of the training set and (2) the USI model [4] that was previously proposed as the most accurate

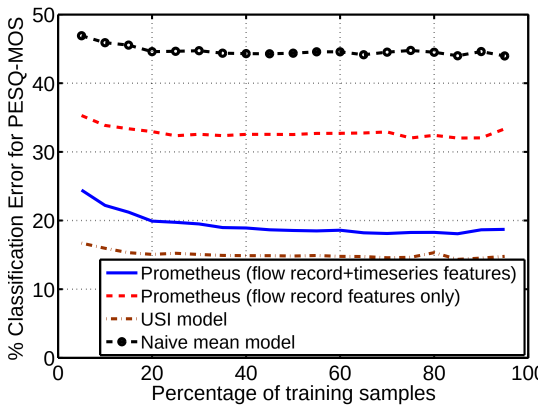

Figure 4: Classification error of thresholded PESQ-MOS.

measure of QoE for Skype voice calls. USI is defined as:

$$\mathsf{USI} = 2.15\log(\mathsf{bitrate}) - 1.55\log(\mathsf{jitter}) - 0.36\mathsf{RTT}, \quad (1)$$

where bitrate is best approximated by mean throughput, and jitter by standard deviation of throughput [4]. Due to its minimal impact, we fix RTT at 0.2. Note that USI requires features from the set of timeseries features to compute. Since the coefficients of USI were originally chosen to predict session time rather than MOS, we use linear regression on the training set to find the best linear predictor of PESQ-MOS using USI.

Figure 4 shows the classification error of each predictor on whether PESQ-MOS > 2.6, the threshold between unacceptable and acceptable audio quality. As expected, the domain expert devised metric, USI, has the lowest error rate. However, the Prometheus model with flow record+timeseries features is only slightly worse (3 percentage points), which is significant because we did not require any domain expertise in developing the model. This model is 27 percentage points more accurate than the naïve baseline.

The flow record features only model, while better than naïve, has lower accuracy than the flow record+timeseries features model, suggesting that the IP flow features currently collected by the network monitor are not sufficient to predict PESQ-MOS accurately. This implication is not surprising, since the flow record features merely consist of volume and duration statistics, whereas jitter/variance are important to the quality of a real-time voice call. Only modest engineering is required to also capture the relevant timeseries features in the network monitor.

3.4 Impact of Data Collection Bias

Thus far, we have evaluated our models by taking a random training and test sets from our collected benchmark data. We balanced the TPR and TNR to avoid biasing the model in one direction. However, it is possible that *network_data* samples taken in times and contexts that are different than those encountered in our data collection phase could result in bias as well, thus reducing the accuracy of our models on real data.

Setup. To evaluate the possible impact of bias in data collection, we first identify three factors that may introduce bias in our training set: the location where the samples are collected, the users carrying the phones doing the data collection, and the device models used in data collection. We then evaluate the potential degradation in accuracy caused by each of these factors by training on a subset of our data that is biased by each factor. For example, to evaluate location bias, we train a model on all samples collected in California, while we test the model on all samples collected in New Jersey. Similarly, to evaluate user bias, we train the model on all samples collected by two staff and evaluate the model on data collected by the other two staff. To evaluate device bias, we train the model on data collected by the Samsung Galaxy S devices and test on data collected by the Galaxy Nexus and Samsung Galaxy S3 devices.

Table 2: Error rate (%) of VideoApp and VoIPApp QoE metrics when training on biased samples.

Metric	Device	Location	User	Random
Buffering Ratio	-	18.8	17.1	17.6
PESQ-MOS	22.3	16.8	23.9	18.9

Results. Table 2 shows the accuracy of the VideoApp and VoIPApp QoE metric predictions when the models are trained on data biased by location and user. We also show the models trained on unbiased data (Random) for comparison. We find that the accuracy with the biased samples for VideoApp is comparable to the random sampling. We only collected VideoApp benchmarks on one device model so we do not include results for device bias.

For VoIPApp, we find that the accuracy degradation due to any of these biases is small (at most 5 percentage points). In all cases, the accuracy is still substantially better than the naïve baseline but slightly worse than the domain expert USI model. We conclude that bias due to device, location, and user factors for this model has limited impact on accuracy.

4. RELATED WORK

There is extensive literature on assessing and modeling QoE. The most direct way of assessing the QoE experienced by users is via subjective evaluation. The MOS is such a subjective metric designed to obtain user-perceived service quality for telephony, obtained by asking human listeners to score a call's quality as they perceived it [17]. Although subjective QoE assessment methods are the most direct way to capture user-perceived service quality, they are inconvenient and expensive to conduct.

Objective assessment uses numerical quality metrics to approximate user-perceived service quality (e.g., [5, 20]) — i.e., what this paper terms a QoE metric. Thus, the assessment process can be done repeatedly and automatically. A number of QoE measurement frameworks have been proposed to incorporate objective assessment methods (e.g., [18, 22]). Although objective assessment methods are more convenient than the subjective methods, they typically require control of the application service to collect the reference input and outputs signals. These are not available to network operators for OTT services.

Estimation of QoE from network traffic is closest in spirit to our work. HostView [11] and OneClick [3] used regression models to predict user satisfaction or dissatisfaction with apps on laptops, but relied on users to express their dissatisfaction by manually clicking a button. Our approach collects ground truth automatically using benchmarks and can handle more features with LASSO. Q-score estimates a single score that represents user-perceived quality in an IPTV service by using coarse-grain network metrics [19]. Similarly the relationship between network metrics and QoE for web browsing [6] and Skype calls [4] have been studied. In contrast to our work, these approaches require significant domain knowledge about the application services and/or access to service logs not available to cellular operators.

5. DISCUSSION AND FUTURE WORK

Our initial experience with Prometheus indicates that automatically learned QoE models are promising, particularly since they overcome the challenges of lack of control, limited vantage points, and protocol complexity. To conclude, we describe some remaining research challenges to fully realizing Prometheus' potential.

Automating benchmarks. In order to collect sufficient training data, Prometheus relies on automated benchmarks to simulate user behavior on a target app. Some apps' benchmarks may not be as straightforward as video streaming and voice calls due to personalized content, variation in user behavior, and other sources of nondeterminism. Thus, a more comprehensive methodology for defining benchmarks would help generalize our approach further.

Developer interaction. One appealing alternative to using benchmarks to collect training data would be to standardize a mechanism for an app to directly report its QoE (e.g., as an upload to a server for users that opt-in). This alternative would leverage the domain expertise of app developers and collect training data from real users, but it requires developer integration and trust.

Applications. QoE estimates are useful for a variety of network applications, such as trending and anomaly detection. The more accurate the QoE estimates, the better these applications will perform, as there will be fewer false alarms. Furthermore, Prometheus can help in diagnosing root-cause of poor QoE by identifying the important network features, many of which can be tuned or improved with more resources (e.g., RSSI may be raised with higher transmit power). An open question is how much these applications benefit from each point in increased accuracy.

Improving accuracy. Finally, more sophisticated learning methods may further improve the accuracy of Prometheus' QoE models. For example, semi-supervised learning approaches may be able to reduce data collection bias and improve accuracy on data from real users.

6. REFERENCES

[1] A. Balachandran, V. Sekar, A. Akella, S. Seshan *et al.* A quest for an Internet video Quality-of-Experience metric. In *ACM HotNets*, 2012.

[2] T.-H. Chang, T. Yeh, and R. C. Miller. GUI testing using computer vision. In *ACM CHI*, 2010.

[3] K. Chen, C. Tu, and W. Xiao. OneClick: A framework for measuring network quality of experience. In *IEEE INFOCOM*, 2009.

[4] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei. Quantifying Skype user satisfaction. In *ACM SIGCOMM*, 2006.

[5] P. Corriveau and A. Webster. VQEG evaluation of objective methods of video quality assessment. *SMPTE journal*, 108(9):645–648, 1999.

[6] H. Cui and E. Biersack. On the relationship between QoS and QoE for Web sessions. Technical report, RR-12-263, EURECOM, http://www. eurecom. fr/cui/techrep/TechRep12263. pdf, 2012.

[7] J. Erman, A. Gerber, K. K. Ramakrishnan, S. Sen, and O. Spatscheck. Over the top video: The gorilla in cellular networks. In *ACM IMC*, 2011.

[8] J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2 2010.

[9] R. Hsieh and A. Seneviratne. A comparison of mechanisms for improving mobile IP handoff latency for end-to-end TCP. In *ACM MobiCom*, 2003.

[10] Y. Jin, N. Duffield, A. Gerber, P. Haffner *et al.* NEVERMIND, the problem is already fixed: Proactively detecting and troubleshooting customer DSL problems. In *Co-NEXT*, 2010.

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

[12] H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen, and V. Niemi. *UMTS Networks: Architecture, Mobility and Services.* Wiley, New York, NY, 2001.

[13] S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. In *ACM IMC*, 2012.

[14] F. Qian, Z. Wang, A. Gerber, Z. M. Mao *et al.* TOP: Tail optimization protocol for cellular radio resource allocation. In *ICNP*, 2010.

[15] A. Rix, J. Beerends, M. Hollier, and A. Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *IEEE ICASSP*, 2001.

[16] R. Schatz, T. Hossfeld, and P. Casas. Passive YouTube QoE monitoring for ISPs. In *IMIS*, 2012.

[17] I. T. U. T. S. Sector. *Methods for subjective determination of transmission quality.* International Telecommunication Union, 1996.

[18] D. Soldani. Means and methods for collecting and analyzing QoE measurements in wireless networks. In *WoWMoM*, 2006.

[19] H. Song, Z. Ge, A. Mahimkar, J. Wang *et al.* Q-score: Proactive service quality assessment in a large IPTV system. In *ACM IMC*, 2011.

[20] O. Sugimoto, S. Naito, S. Sakazawa, and A. Koike. Objective perceptual video quality measurement method based on hybrid no reference framework. In *IEEE ICIP*, 2009.

[21] R. Tibshirani. Optimal reinsertion: Regression shrinkage and selection via the LASSO. *J.R.Statist. Soc. B*, 58(1):267–288, 1996.

[22] M. Volk, J. Sterle, U. Sedlar, and A. Kos. An approach to modeling and control of QoE in next generation networks [Next Generation Telco IT Architectures]. *IEEE Communications Magazine*, 48(8), 2010.

[23] Q. Xu, J. Erman, A. Gerber, Z. Mao *et al.* Identifying diverse usage behaviors of smartphone apps. In *ACM IMC*, 2011.