Systems Considerations in Real Time Video QoE Assessment

Amy Csizmar Dalal
Department of Computer Science, Carleton College
Northfield, MN 55057
Email: adalal@carleton.edu

***Abstract*—We consider several key questions in the design of a real time video quality assessment system. How frequently can we generate subjective video quality ratings with some degree of accuracy? How often should we sample the data? How do we weigh the need to consolidate data collection (arguing for fewer, less frequent data points) with the need to monitor video quality in real time (arguing for more frequent data points)? What are the timing requirements for such a system, both in training the system and in assigning ratings to videos? Our results demonstrate that we can achieve accurate video quality ratings by using small portions of the video and a frequent sampling rate, and that in the worst case the system can be trained in ten minutes.**[1]

I. Introduction

Measuring and evaluating user quality of experience (QoE) specifically for video applications, without explicitly using the Mean Opinion Score (MOS) [1] with its scalability and interpretation issues, is an area of active research [2]–[5]. Determining video QoE can lead to the development of better protocols and better network structure to better support video applications in the Internet.

In previous work, we have demonstrated that we can evaluate the QoE of on-demand video in real time using objective measurements collected from two instrumented media players, Windows Media Player [6], [7] and Flash Player [8]. Our approach, which uses data mining to process application-layer, stream state measurements (bandwidth, frame rate, and the number of received packets), and compare these to previously-obtained subjective ratings, is able to assess video quality with between 75 and 87% accuracy on just 15 seconds of application-layer measurements [7].

In this paper, we explore the system-level considerations in building a real-time video quality assessment system. Our goal is to keep ratings accuracy high while keeping timing requirements—the time it takes to train the system and the time it takes for the system to assign a quality rating to video data—and resource utilization low. Resource savings take on increased importance as we consider increasing the number of measurement probes in the system. We consider characteristics of the stream state measurements, factors such as how often to sample the data, how much time should elapse between ratings assigned by the system, and which combination of stream state measurements to input into the system. We are interested in finding the combination of factors that maximizes ratings accuracy while keeping the time required to train the system within acceptable bounds. We consider three scenarios: video on demand, with a training set tailored to the video to rate; video on demand, with no attempt to tailor the training set to the video to rate; and general video, where the content may change rapidly. We demonstrate that sampling data every second, using a combination of transmitted packets and bandwidth, and assigning ratings every 20 seconds, yields the most accurate ratings for all three scenarios. We also find that the time it takes to train the system ranges from 10 seconds to 10 minutes. Given that the training phase is completed off-line, these results indicate that modifying the training set occasionally within such a system is feasible, allowing for flexibility in the design of such a system.

In the following section, we review the architecture of the system and describe the design tradeoff categories under consideration here. In Section III, we describe the experiments used to evaluate the tradeoffs. Section IV presents the results, and provides some context for the results for system designers.

II. System architecture

Figure 1 illustrates the proposed architecture for our video QoE assessment system, as described in [9]. The system utilizes information from the network and from stream state measurements from the video players to form a picture of the current health of the network. The system will then feed this information back to the content distributors, network operators, and other interested parties. Each party acts upon the information received about the system health however it sees fit. As updated application and network measurements are fed back into the system, the system becomes self-supporting, reflecting both the current state of the network and the relevant history of the network and application states, similar to the system proposed in [10].

In this paper, we concentrate on the information coming out of the QoE rating algorithm, which is affected by the training algorithm as well. Figure 2 shows the detail of the training and rating portions of the system. The clients collect stream state data (and, during our user experiments, video quality ratings) every second. The collected data is then sent to a database on the assessment server. The assessment server compares the newly-collected data to the training set data to determine

[1]This work was sponsored by grants from the Clare Booth Luce Foundation, Howard Hughes Medical Foundation, and Carleton College. Earlier portions of this work were sponsored by Hewlett-Packard Laboratories.

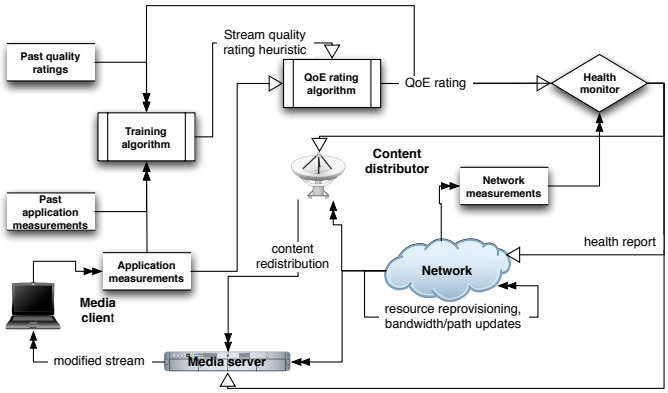

Fig. 1. The proposed video QoE assessment framework.

which stream(s) in the training set are "closest" to the current stream, using k-nearest neighbors with a distance metric of dynamic time warping (DTW). The system takes the median of the closest k streams' ratings to assign a quality rating to the current stream.

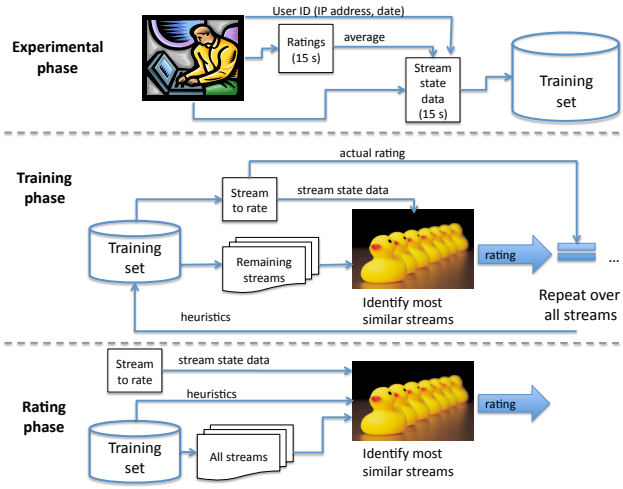

Fig. 2. Detail of the training and rating algorithms in the video QoE assessment framework.

Our system aims to assign quality ratings in real time. From a resource usage perspective, we would like to calculate ratings as infrequently as possible. Gauging the current status of the application and, indirectly, the network, however, requires frequent measurements and ratings assignments, so that we do not miss key events such as buffer starvation or a degradation in frame rate. In the remainder of this section, we discuss some key design tradeoffs to minimize resource usage while still maintaining an accurate picture of the overall health of the application and underlying network. While there are many design tradeoffs we could consider, we limit our discussion to tradeoffs related to stream state measurements. In future work, we will consider other system design tradeoffs such as the number of servers, placement of probes, etc.

Tradeoff 1: Sampling rate. Previous studies assumed that the system collects and analyzes stream state data every second. While this means we will not miss any events such as decreases in bandwidth or frame rate, or buffer starvation events, it is the most taxing on the system in terms of resources. For a video stream that is 60 seconds long, if we sample the data every second, we will have approximately 1400 bytes of data. While this is not a large amount for a single client, it adds up for multiple measurement clients. If our video stream is 2 minutes long, sampling every 30 seconds will give us only 4 data points. On the other hand, if we can achieve accurate quality ratings by collecting data every 5 seconds, this represents a resource (storage) savings. For this paper, we examine sampling rates between 1 and 5 seconds. For a 2 minute stream, sampling every 5 seconds will give us 24 data points.

Tradeoff 2: Interrating time. Our next tradeoff determines how much of a stream's data to examine before assigning a rating. A too-short time scale may not yield enough information to generate an accurate rating, yielding false positives or false negatives. A too-long time scale means we may miss key events until it is too late to self-correct. The key is to determine how often to rate the stream to achieve both accurate and useable ratings. We set an upper limit on the time between ratings at 30 seconds, and a lower limit of 10 seconds [11]. Thirty seconds was chosen as an adequate tradeoff between resource expenditure and frequency, to minimize the odds that we will miss a crucial stream event.

Tradeoff 3: Stream state data combinations. Previous work [7], [11] demonstrated that certain pieces of stream state information are more valuable than others in terms of influencing video QoE. The presence of additional stream state data can introduce "noise" into the ratings. The pieces of data available to us include the number of times buffered, frame rate, bandwidth, and the number of packets received per second. We wish to determine which combinations of stream state data, in conjunction with different sampling rates and durations, achieve the most accurate QoE ratings. We also wish to determine if the choice of stream state data depends on the video to be rated. Frame rate may be an excellent indicator of QoE for one video, but for another the combination of frame rate and received packets may improve ratings accuracy. Increasing the amount of stream state data used to calculate QoE increases the number of dimensions for the dynamic time warping distance calculation, which in turn increases both training time and rating time. Thus, we are interested in optimizing the amount of stream state data from a timing perspective as well, which we discuss below.

Tradeoff 4: Training set composition. There are two main strategies in defining the composition of a training set. One strategy includes all possible items to increase the odds of a match for the stream you are trying to rate, at the expense of a longer training time and more storage space. A second strategy tailors the training set to the stream(s) you anticipate rating. The idea is to reduce noise in the training set by only

including streams likely to be similar to the stream to rate. Reducing the size of the training set reduces the time required to train the system as well as the storage requirements, at the possible expense of ratings accuracy. In this paper, we consider both strategies.

Tradeoff 5: Timing concerns. When assigning quality ratings in real time, it is crucial that our system be able to assess the state of the application quickly, so that action can be taken to mitigate impending poor conditions. Previous work indicates that the system assigns ratings in well under a second, so we do not consider rating time further in this paper. Even though training is not in the critical path of the system, keeping training times reasonably short has several advantages. If training times are short, then swapping out one training set for another as the system is running becomes feasible. As we get more information about the system and how it reacts to network congestion, or as new videos are added to the system, we can add this data to our training set. If the time scale on which to train the system is short, we can retrain the system using this additional data.

III. Experimental Setup

Our testbed network contains 15 client machines on an uncongested subnet of our campus network, running Windows XP version 5.1, with 2.4 GHz Intel Core 2 Duo processors, 2 GB RAM and Windows Media Player version 11.0.5721.5268. A media server, a 2.4 GHz machine with 512 MB of RAM running Windows Server 2003 and Windows Media Server 2003, sits on an isolated subnet behind a router, which has an Intel Pentium 4 3.4Ghz processor, 1GB RAM, and runs Red Hat Linux Enterprise Server 5.5. The router separating the media server from the campus subnet runs netem software [12] to control the level of congestion from the media server to the media clients.

Table I lists the characteristics of the video streams used in this study. The streams were selected to provide some variety in their length, style, and amount of action. We introduced four levels of congestion onto the network: 0%, 5%, 10%, and 15% average network packet loss, each with a delay of 100ms.

TABLE I
Description of test videos

Name	Time (m:s)	Description	Action level
Cow	1:57	Dialog	Moderate: frequent scene shifting
OKGo	3:06	Music video	Moderate: stable scene, heavy action
Up	4:40	Animated movie short	High: frequent scene shifting, heavy action

We ran a blind experiment using 22 participants, all but one between 18-22 years of age. Participants watched each video four times. They first watched a 10-second clip at 0% packet loss to norm their expectations of video quality. The same video clip was then shown at full duration three more times, twice at a randomly-chosen network congestion level and once a repeat of a previous congestion level. Other than the reference clip, the users were not aware of the congestion level of the particular clip they were viewing. These ratings were logged every second, along with the stream state information reported by Windows Media Player, and presented to the training algorithm. (Table II lists the stream state measurements reported by Windows Media Player.) Because user ratings may be biased, depending on the user's prior experience with streamed video, personal preferences, etc, we normalize the user ratings by calculating the z-score, $z_s = \frac{r_s - \bar{r}}{\sigma_r}$, where r_s is the user's quality rating for stream portion s, $\bar{r}$ is the average of the user's quality ratings on all stream portions viewed, and σ_r is the standard deviation of the user's quality ratings on all stream portions viewed. We average the user ratings over an N-second window, where N is the interrating time. We measure the *accuracy* of the system by calculating the percentage of time it assigns the same rating (z-score) to a video stream portion that our participant did, within a tolerance of 0.65, which roughly corresponds to one level of perceptible quality difference.

TABLE II
The stream state data collected by Windows Media Player.

Name	Description
Received packets (TP)	Number of packets received by the player per second.
Bandwidth (BW)	Current bandwidth, in kilobits per second.
Frame rate (FR)	Current frame rate, in frames per second.
Buffer count (BC)	Number of times (cumulative) the player has buffered.

We consider three scenarios in the makeup of the training set. The first two scenarios consider *video on demand* (VOD) systems, where the owner controls the video content and can develop a set of historical stream data that exactly matches the content. The *fine-tuned VOD scenario* considers the case where the streams in the historical data represent exactly the streams to be rated. The *general VOD scenario* considers the case where the historical data consists of data from all streams currently in the system. We also consider the case where the owner does not have complete control over the content on the site, and where perhaps the content changes rapidly, similar to YouTube [13]. In this *general video scenario*, there is no overlap between the streams in the historical data and the streams to be rated.

IV. Results

A. Ratings accuracy

The top third of Table III lists the top three combinations of stream state measurements, sampling rates, and interrating times for each source video individually in the fine-tuned VOD scenario. For all three videos, the combination of bandwidth and received packets has the greatest influence on video quality; our system is able to accurately rate the video 81-82% of the time for cow, 84% for okgo, and 79-80% for up. For both okgo and up, sampling once a second and rating the video every 20 or 30 seconds is a very successful strategy. For cow, we are better off using longer (40-60 second) interrating times and sampling less frequently (2 or 4 seconds). This is somewhat surprising given that cow is our shortest video.

However, cow is the lowest-action video, with scene changes occurring every 30-40 seconds on average, so it is possible that the results reflect the video content.

TABLE III
RATINGS ACCURACY, TOP INPUT COMBINATIONS FOR INDIVIDUAL VIDEOS. "TIME" = "INTERRATING TIME".

Video	Stream state measurements	Sample rate (s)	Time (s)	Accuracy (%)	(K, w)
Fine-tuned VOD					
cow	{TP,BW}	2	60	82.05	5, 6
	(tie) BW/{BW,FR}	2	60	82.05	5, 7
	{TP,BW}	4	40	81.25	4, 2
	(tie) BW	4	40	81.25	4, 1
	(tie) FR	1	50	81.25	4, 12
okgo	{TP,BW}	1	20	84.10	7, 4
	{TP,BW}	2	20	84.01	15, 2
	{TP,BW}	1	30	83.80	11, 10
up	{TP,BW}	1	20	79.85	20, 4
	{TP,BW}	1	40	79.36	16, 12
	{TP,BW}	1	30	79.20	12, 7
General VOD					
cow	FR	5	50	88.16	4, 1
	FR	1	60	85.90	7, 11
	BW/{BW,FR}/{TP,BW}	5	50	85.53	3, 1
okgo	{TP,BW}[a]	1	50	85.64	6, 13
	{TP,BW}	1	30	84.51	9, 6
up	{TP,BW}	1	20	80.60	14, 3
	{TP,BW}	1	30	79.73	9, 6
	FR	2	40	79.12	5, 6
General video					
cow	FR	1	50	83.33	8, 10
	FR	1	40	81.25	4, 11
	FR	2	40	80.95	5, 6
okgo	{TP,BW}	2	20	79.19	7, 6
	{TP,BW}	1	20	78.80	10, 4
	{TP,BW}	1	50	77.35	4, 7
up	{TP,BW}	1	50	75.21	3, 11
	{TP,BW}	1	30	75.20	13, 8
	{TP,BW}	1	20	75.19	11, 6

[a] We get slightly lower accuracy with BW alone (84.53%) and with BW, FR (83.43%) for the same duration, sampling rate, and (K, w) values.

The general VOD scenario results appear in the middle third of Table III. Similar to the fine-tuned VOD scenario, the combination of received packets and bandwidth most strongly influences video quality for okgo and up, and sampling frequently is the best strategy. Sampling less frequently (every 5 seconds) is a better strategy for cow. We see similar trends in interrating times for cow and up as in the fine-tuned VOD scenario, but increasing interrating time for okgo is a better strategy here. Intuitively, we expect that fine-tuning our training set allows the algorithm to focus on streams with similar characteristics. In reality, having more source data in our training set demonstrates that for some videos under some conditions, the best match may be from a different source video. This is a testament to the power and flexibility of DTW: it is able to pick out similarities that may be difficult to detect among streams otherwise.

There is a bit of overlap between the best strategies for okgo and up, but not for cow, for the general video scenario, as seen in the bottom third of Table III. For cow, our best strategy is to use frame rate, with an interrating time of 50 seconds and a sampling rate of 1 second. For okgo and up, received packets and bandwidth with frequent sampling best predict video quality.

For the greatest degree of generalizability and simplicity, we would like to have a single strategy that assesses QoE accurately for all three scenarios. We first start by identifying successful general strategies for each scenario, listed in Table IV. The top strategy is identical for all three scenarios: use transmitted packets and bandwidth for the stream state measurements, sampled every second, with an interrating time of 20 seconds. In each scenario, with these options, the system achieves at least 75% accuracy.

TABLE IV
TOP INPUT COMBINATIONS FOR INDIVIDUAL SCENARIOS. "TIME" = "INTERRATING TIME"

Scenario	Stream state data	Sample rate (s)	Time (s)	Accuracy (%)		
				cow	okgo	up
Fine-tuned VOD	{TP,BW}	1	20	77.83	84.10	79.85
General VOD	{TP,BW}	1	20	84.73	83.61	80.60
General video	{TP,BW}	1	20	78.82	78.80	75.19

To maximize rating accuracy, the training set should contain streams from all of the videos in our working set (the general VOD scenario). Ideally the stream to be assessed will either already be represented in the training set, or will be very similar to a source video already in our training set. In this scenario, our system can accurately assess a stream's quality more than 80% of the time. Tailoring the training set to the video to be rated results in a slight loss of ratings accuracy.

B. Timing

Our final goal is to determine the time required to train the system. We chose frame rate alone and the combination of received packets and bandwidth as stream state data. This allows us to see how adding pieces of stream state information increases the training time. It is possible that different pieces of stream state information may affect training time differently, but for now we assume that all combinations of, say, two pieces of stream state information will yield approximately the same training time. We also varied the interrating time between 10 and 60 seconds while keeping the sampling interval at 1 second. Doing so allows us to see how the amount of information within a training stream affects training time. Finally, we varied the number of stream portions in the training set, which is done naturally via the three scenarios discussed previously.

Figure 3 shows the training times for the three scenarios. The fine-tuned VOD plots demonstrate that training time increases with the total duration of the source video. The plots also show a slight increase in training time when two pieces of stream state data are used over one piece of stream state data. This difference tends to be more pronounced as the number of stream portions in the training set increases. Training times range from 10 seconds (cow with an interrating time of 60 seconds) to just over 2 minutes (up with an interrating time of

10 seconds). We see similar trends in the general video and general VOD scenarios: the more video portions in the training set, the longer the training time. The worst case training time overall occurs for the general VOD scenario (labeled "all" on the graph) when the interrating time is 10 seconds. In this scenario, the system takes 10 minutes to train. The worst case general video scenarios take about 2/3 of the training time as the general VOD scenario, meaning we can train a general video system in under 7 minutes.

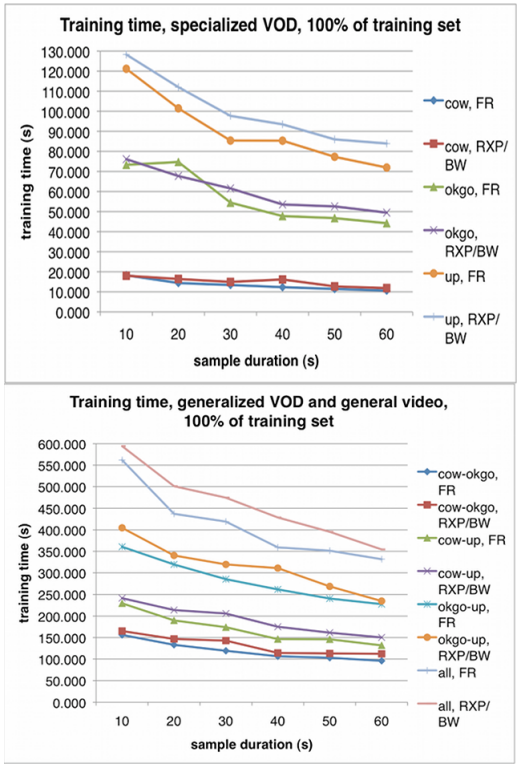

Fig. 3. Training times, in seconds, for the three scenarios.

Our results demonstrate that retraining the system off-line is feasible, which gives system designers flexibility in modifying and updating the contents of the training set. With a VOD system, new videos can be added to the training set, if during testing the content providers or volunteers are willing to assign ratings to the streams (which they may already be doing for quality control purposes). With a general video system, community volunteers may discover and rate newly uploaded videos, similar to Netflix's voluntary ratings. Crowdsourcing can be used to add to the system's knowledge of both videos and video quality; [14] provides an example of this using pairwise video comparisons.

V. Conclusions and Future Work

This paper discusses system design considerations when building a real-time video QoE assessment system. While previous work demonstrated the feasibility of using stream state measurements to discern video QoE, this paper concentrates on how to best use these measurements to design a scalable video QoE assessment system. We consider tradeoffs in both the composition of the individual data sets (which measurements to include, how often to sample the data, and interrating intervals) and the composition of the training set (whether to include all available videos or a representative sample of videos). We also examine the timing requirements to train the system. We find that the system is most accurate when the individual data sets contain data on received packets and bandwidth, sampled every second, with 20 seconds between ratings, and when the training set contains all available videos (the general VOD scenario). The system tradeoff here, with the large size of the training set, is in the training times, but we demonstrate that we can train the system off-line, worst case, in about ten minutes.

References

[1] I.-T. R. P.910, "Subjective video quality assessment methods for multimedia applications," Recommendations of the ITU, Telecommunications Sector.

[2] R. Serral-Gracia, E. Cerqueira, M. Curado, M. Yannuzzi, E. Monteiro, and X. Masip-Bruin, "An overview of quality of experience measurement challenges for video applications in IP networks," in *Wired/Wireless Internet Communications*, ser. Lecture Notes in Computer Science, E. Osipov, A. Kassler, T. Bohnert, and X. Masip-Bruin, Eds. Springer Berlin / Heidelberg, 2010, vol. 6074, pp. 252–263.

[3] J. Nichols, M. Claypool, R. Kinicki, and M. Li, "Measurement of the congestion responsiveness of Windows streaming media," in *Proceedings of NOSSDAV*, Kinsdale, Ireland, June 2004.

[4] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "YoMo: A YouTube application comfort monitoring tools," in *Proceedings of QoEMCS@EuroITV*, Tampere, Finland, 2010.

[5] D. Ćulibrk, D. Kukolj, P. Vasiljević, M. Pokrić, and V. Zlokolica, "Feature selection for neural-network based no-reference video quality assessment," in *Proceedings of the 19th International Conference on Artificial Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 633–642.

[6] A. Csizmar Dalal, "User-perceived quality assessment of streaming media using reduced feature sets," *ACM Transactions on Internet Technology*, vol. 11, no. 2, December 2011.

[7] A. Csizmar Dalal, A. Bouchard, S. Cantor, Y. Guo, and A. Johnson, "Assessing QoE of On-Demand TCP Video Streams in Real Time," in *Proceedings of the IEEE International Conference on Communications*, Ottawa, Ontario, Canada, June 2012.

[8] H. French, J. Lin, T. Phan, and A. Csizmar Dalal, "Real time video QoE analysis of RTMP streams," in *Proceedings of the 30th IEEE International Performance Computing and Communications Conference*, Orlando, FL, November 2011.

[9] A. Csizmar Dalal, "Revisiting a QoE assessment architecture six years later: Lessons learned and remaining challenges," in *Proceedings of the Third International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*, Las Palmas de Gran Canaria, Spain, November 2009.

[10] M. Allman and V. Paxson, "A reactive measurement framework," in *Proceedings of the Passive and Active Measurement Conference*, Cleveland, OH, April 2008.

[11] A. Csizmar Dalal, E. Kawaler, and S. Tucker, "Towards real-time stream quality prediction: Predicting video stream quality from partial stream information," in *Proceedings of the International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Las Palmas de Gran Canaria, Spain, November 2009.

[12] T. Linux Foundation, "netem," http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[13] YouTube, LLC, "YouTube," http://www.youtube.com. [Online]. Available: http://youtube.com

[14] K.-T. Chen, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, "A crowdsourceable QoE evaluation framework for multimedia content," in *Proceedings of the Seventeen ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2009, pp. 491–500.