Assessing QoE of On-Demand TCP Video Streams in Real Time

Amy Csizmar Dalal, Andy K. Bouchard, Sara Cantor, Yiran Guo, Anya Johnson
Department of Computer Science, Carleton College, Northfield, MN, USA
Email: {adalal, bouchara, cantors, guoy, johnsona}@carleton.edu

***Abstract*—Real-time stream quality assessment can assist network operators, content providers, streaming servers, and ISPs in evaluating their customers' *quality of experience (QoE)*, and can lead to better design of streaming protocols, computer networks, and content delivery systems. In previous work, we demonstrated that objective data collected from an instrumented media player application can be used to *predict* video QoE of UDP streams with a high degree of accuracy and without the inherent scalability and interpretation issues of subjective ratings. In this paper, we extend our previous work to consider real time QoE assessment of TCP video streams, using as little as 15 seconds worth of data. By examining two video-on-demand scenarios and a general video scenario in which content changes rapidly, we find that we can accurately assign user quality ratings between 75 and 87% of the time using objective application-layer measurements from TCP video streams, such as bandwidth and frame rate, and that various combinations of stream state measurements produce accurate ratings in this range. We discuss the implications of these findings on the design of a real time video QoE assessment system.** [1]

***Index Terms*—Video streaming, quality of experience (QoE), quality of service (QoS), multimedia applications, measurement, subjective quality**

I. Introduction

It is difficult to assess the subjective, user-perceived quality of a media stream in a scalable and quantifiable way. As with all Internet-based applications, there is a complex interplay between network congestion conditions and the effect these congestion conditions have on application performance. Knowing how end users perceive the quality of audio and video streamed on-demand over computer networks, and the relationship between stream quality and network congestion, can lead to better design of streaming protocols, computer networks, and content delivery systems. Real-time stream quality assessment is important for network operators, content providers, streaming servers, and ISPs to help determine if their customers' *quality of experience (QoE)* is acceptable, but presents its own set of challenges: determining what to measure and how often, interpreting the findings correctly so that the right changes can be made to the system, and so on.

QoE assessment for Internet and/or multimedia applications, without using subjective ratings such as the MOS (mean opinion score) [13] with their inherent scalability and interpretation issues, is a topic of great interest for these reasons; [14] provides an overview of this area. Knowledge of the source stream is required for QoE assessment in [1], while others [3], [4], use the Emodel [8], an objective mechanism for assessing audio quality using transmission parameters. An alternate approach is to utilize application-layer objective metrics, taken at the client's machine through an instrumented media player application [3], [12]. These approaches allow one to take measurements as close to the user as possible, in some cases without requiring the user's participation, providing a more accurate assessment of the state of the application at any given time. [15] computes the QoE of YouTube video by estimating the remaining content in the play-out buffer, using a combination of packet sniffing and browser plugin. While that study and our study share similar goals and approaches, we utilize a single plugin and player-state measurements alone, as well as machine learning techniques, to determine degraded QoE.

In previous work [5], [6], we demonstrate that objective data collected from an instrumented media player application can be used to *predict* video QoE with a high degree of accuracy (typically 70-90%). We use machine learning techniques to discern the connection between this objective data and video QoE, using a nearest-neighbor heuristic and dynamic time warping (DTW) as its distance measure. Our approach is similar to [7] in that we use machine learning techniques (they use neural networks) and our algorithm is no-reference, meaning we do not require information about the source stream, although they are more concerned with assessing encoder performance. Our previous studies utilized UDP streams. Because most streams are transmitted over TCP, we are interested in the key differences between our UDP results and our TCP results. In particular, TCP hides network congestion from the application, meaning that lost and retransmitted packets (the key indicator of video QoE from our previous studies) will not be explicitly indicated in the application-layer measurements. We wish to determine, in the absence of this information, if it is still possible to discern video QoE solely from application layer video measurements.

In [6], we demonstrated that our algorithm accurately assesses video QoE with as little as 10-15 seconds of stream state data. A drawback of this study, besides the use of UDP streams, was that our source streams had only a single rating assigned at the end of the stream, which we retroactively assigned to each 10-15 second portion of the stream. In reality, subjective stream quality is not fixed, but rather fluctuates over

[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.

the lifetime of the stream depending on the underlying network conditions. We wish to discern whether or not we can assess video QoE for TCP video streams over short time scales. The ultimate goal of our work is to discern degraded video QoE in real time, so that we can mitigate the conditions that cause the degraded quality (by switching the stream to another server, playing content out of a local cache, etc.) before the end user notices any video quality degradation.

The key contributions of this study are as follows. Leveraging our existing work in stream quality assessment for UDP-based streams, we modify our stream quality assessment algorithm for TCP-based streams, which lack some of the key indicators, namely lost and retransmitted packets, of UDP-based streams. Based on lessons learned from our previous experiments, and with an eye towards real-time stream quality assessment, we modify our data collection mechanism by logging stream quality ratings continuously in time using a sliding scale, and by assessing smaller portions of each stream to allow us to monitor stream quality changes throughout the stream. We describe these modifications in Section II. Using the data from a new set of user experiments, described in III, we determine the accuracy with which we can assign user stream quality ratings using the aforementioned algorithm, on 15 second portions of streams and using different combinations of stream state measurements as input to the algorithm. By examining two video-on-demand scenarios and a more generalized video scenario in which content changes rapidly, we find that our system can accurately assign user quality ratings between 75 and 87% of the time, and that various combinations of stream state measurements produce accurate ratings in this range (Section IV). This indicates a good amount of flexibility in designing an actual stream quality assessment system to operate in real time, which we discuss in Section V.

II. System Architecture

Our framework for collecting and analyzing data from video streams to discern video QoE is shown in Figure 1. The framework is the same basic framework we've developed in previous work, with some key changes in how we collect, process, and analyze data to allow for evaluation of TCP-based video streams with an eye towards real-time video QoE assessment. In this context, a *stream* refers to a single video watched and rated by a single person.

The basic idea behind our framework is as follows: use historical data about how stream state measurements, such as bandwidth and frame rate, influence video quality ratings, and use this information to predict the video quality rating for an unrated stream. We use machine learning techniques, specifically a K-nearest-neighbor algorithm using dynamic time warping [9] as its distance metric, to assign video quality ratings to unrated streams. Dynamic time warping is used because it allows us to match up portions of a stream that are shifted in time, recognizing that similar streams may have similar patterns that do not occur at exactly the same time intervals, as Figure 2 shows. We first collect historical

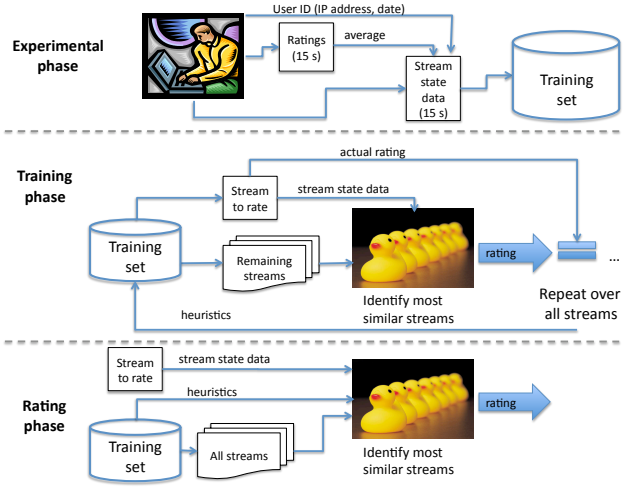

Fig. 1. Video QoE assessment framework.

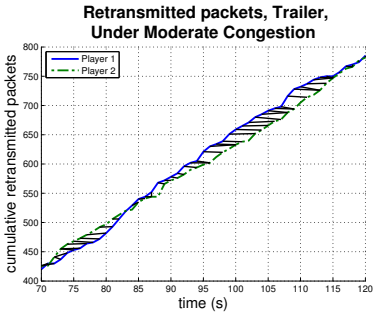

Fig. 2. The cumulative number of retransmitted packets for a portion of a UDP video stream, measured at two different clients under moderate congestion. The solid black lines connecting the two plots show the point-to-point matching performed by DTW, illustrating that the closest matches between points do not always line up exactly in time.

data about video streams, consisting of stream state data and quality ratings. This historical data forms the *training set* for our machine learning algorithm. During the training phase, we determine two *heuristics* for the algorithm: how many neighbors (closest to the stream to rate) to use when assigning a quality rating (K), and how far forwards and backwards in time we'll shift when measuring the dynamic time warping distance between two streams. These heuristics are determined using leave-one-out cross-validation. These two phases occur off-line. In the ratings phase, which will eventually occur online in real time, we use the historical data to assign a rating to an unrated stream, by identifying the K streams with the closest dynamic time warping distance to the unrated stream and taking the median of their ratings—this is the rating we assign to the unrated stream.

The historical data is tagged with identifying information (user's IP address and datestamp), used solely to normalize the ratings, which we discuss below. In previous experiments, we input all of the data associated with a single stream at

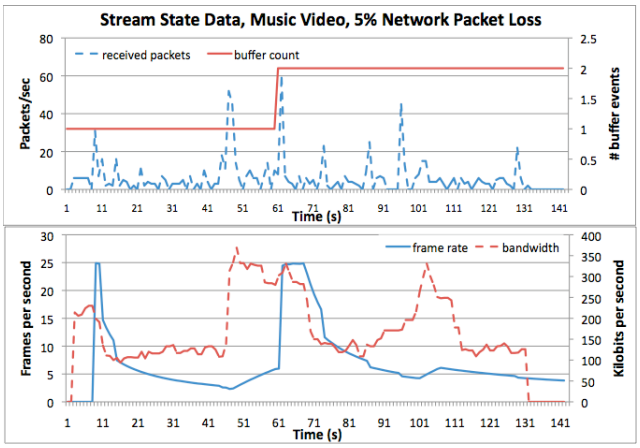

Fig. 3. Stream state data collected by our instrumented media player for a music video experiencing 5% network packet loss and 100ms network delays.

once, and the algorithm assigned a rating to the stream as a whole. For real time QoE assessment, we instead need to assign ratings more frequently. The tradeoff is that we need enough data to reliably assign a rating to the stream, without missing crucial stream quality degradation events. Based on our results in [6], we divide our data into 15-second subsets, or *substreams*, each with its own quality rating, which is the average of the ratings collected each second over this interval. Each substream looks like an individual stream to the algorithm. While it may be useful to the algorithm to retain some historical knowledge of the stream, our goal here is to determine the feasibility of assessing stream quality on shorter time scales.

Figure 3 is an example of the data collected by our system for a TCP stream several minutes in duration that experiences a moderate level of network congestion (5% packet loss and 100 ms delays). Because TCP hides network congestion from the application, we see the loss of key stream quality indicators from our UDP experiments, namely lost and retransmitted packets. The plot shows how the remaining stream state measurements respond to network congestion. For instance, we see spikes in the number of received packets when the server attempts to recover from previous periods of few to no packets received (overcompensation). Note that while bandwidth and frame rate rise and fall at similar times, bandwidth tends to rise first, indicating that bandwidth may be a *leading indicator* of stream quality degradation. Buffer count, which responds later than the other measurements, records the number of times the player entered *buffer starvation* mode during stream playback, including the initial startup buffering period.

In our original experiments, participants assigned a single rating to a stream at its conclusion, using a numerical 7-point modified MOS. To get a finer grained idea of how stream quality perception changes over the course of a stream in response to network congestion, we changed the ratings system so that our participants entered a rating at any time during stream playback, using a continuous sliding scale. A sliding scale requires less active cognitive thought and allows for faster and more frequent judgements [2], [10], making this type of scale better suited for rating streams in real time. Our slider bar was labeled at the beginning and end with 0 and 100%, respectively, and at points in the middle at approximately 33 and 66%. These labels gave our users visual cues as to the length of the scale and allowed them to calibrate their scores to these markers. The location of the slider bar was sampled and logged every second, along with the stream state data. Because subjective ratings are inherently biased, influenced by the person's prior experience with video streams online, preference for content, mood, etc, the system normalizes the ratings using the z-score, $z_s = \frac{r_s - \bar{r}}{\sigma_r}$, where r_s is the user's quality rating for substream s, $\bar{r}$ is the average of the user's quality ratings on all substreams, and σ_r is the standard deviation of the user's quality ratings. Both the raw and the normalized ratings are stored in the database as part of the 15 second streams' information.

We consider three scenarios in this paper, mimicking the design of three different types of video systems. The first two scenarios consider *video on demand* (VOD) systems, where the owner has complete control over the video content and thus has the opportunity to develop a set of historical stream data that exactly matches the content served out to users. The *specialized VOD scenario* considers the case where the streams in the historical data represent exactly the streams to be rated. For instance, for rating a clip of the Disney movie *The Lion King*, the historical data would only include clips from *The Lion King*. The *generalized VOD scenario* considers the case where the historical data consists of data from all streams currently in the system. Using our example above, in this case to rate *The Lion King*, the historical data would include clips from all Disney animated movies. We use this scenario to determine whether having more streams in the training set improves hit rates. 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 [16]. In this scenario, we cannot guarantee that a stream to be rated is at all represented in the training set. In the *general video scenario*, there is no overlap between the streams in the historical data and the streams to be rated.

III. Experimental Setup

Our testbed network consists of 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 [11] 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 variety in

TABLE I
DESCRIPTION OF TEST VIDEOS

Name	Time (mm:ss)	Description	Action level
cow	1:57	Dialog	Shifting scene with little action
okgo	3:06	Music video	Stable scene with heavy action
up	4:40	Animated short	Frequent scene shifting with heavy action

their duration, style, and amount of action. We introduced four levels of congestion onto the network: 0%, 5%, 10%, and 15% average network packet loss, with a delay of 100ms. For space reasons, we present the results for 10% or less network packet loss.

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 of the video 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 at 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 an instrumented version of Windows Media Player [5], as shown in the Experiment phase in Figure 1.

We trained the system on the full set of experimental data, as indicated in the Training phase in Figure 1. To test the system's accuracy, we removed one substream at a time from the training set, removed its normalized and raw ratings, and had the system assign a normalized rating to the substream. We then compared the assigned rating to the actual rating. If the two scores were within 0.65 of each other, we counted this as a successful rating (referred to as the *hit rate* in the Results section). The tolerance was chosen by experimentation, and represents the amount of allowable perceptible differences between quality levels. We measured the *hit rate* of the system by calculating the percentage of time it assigned a correct rating to a substream.

We tested the system using various combinations of stream state data as input, from single entities such as bandwidth to all available stream state information. The idea behind this analysis is to determine what pieces of stream state information are the strongest and weakest indicators of video QoE. Because additional pieces of stream state information add dimensions to the dynamic time warping calculation, increasing its time and complexity, reducing the amount of input to the system reduces the time required for dynamic time warping. For readability, we indicate the stream state data using the abbreviations listed in Table II.

IV. RESULTS

We first examine how our participants utilized the continuous rating scale. Figure 4 shows the probability density functions of the raw slider ratings, separated out by the amount of congestion on the underlying network and by stream. Our participants heavily favored the ends of the ratings scale, with a slight preference for the other labels (33 and 66%) as well. The slider started out at 100% for each stream, and participants tended to leave it there at least through the initial buffering period. Significant stream degradations often caused our participants to move the slider all the way to 0% and leave it there until conditions improved. Our participants were able to distinguish between the network congestion levels even without knowing to which congestion level the stream was exposed. Contrary to our expectations, the videos exposed to no network congestion do not have perfect or near-perfect subjective ratings. This demonstrates the challenge in soliciting subjective measurement input, and for the need to normalize user ratings to remove some of these user biases.

TABLE II
ABBREVIATIONS USED FOR THE STREAM STATE DATA

Abbreviation	Data
TP	Received packets (pkts/s)
BW	Bandwidth (kbps)
FR	Frame rate (fps)
BC	Buffer count (# times buffered)

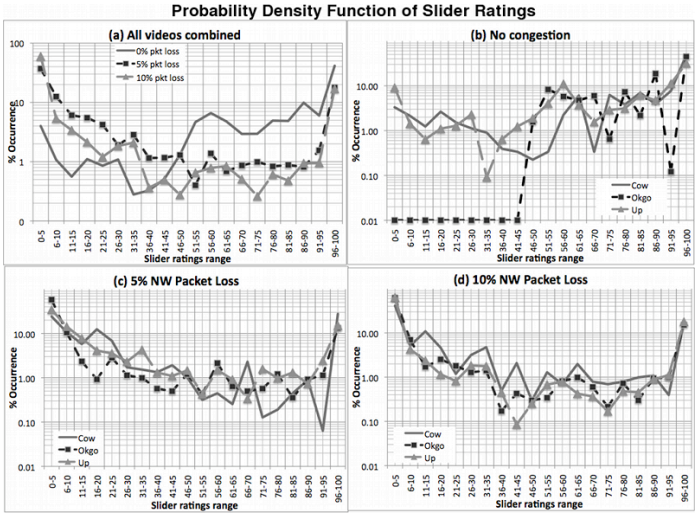

Fig. 4. Probability density function of raw slider ratings for all videos, separated out by the amount of packet loss on the underlying network and by stream. Note that the y-axis is logarithmic.

We next present results obtained in assigning normalized stream quality ratings to the substreams rated by our participants. We first examine the specialized VOD scenario, shown in the top plot in Figure 5. Buffer count is clearly a poor indicator of stream quality; the system fails to assign a correct rating over half the time. For the rest of the stream state measurements, the system is successful in assigning stream quality ratings more than 75% of the time. If we know in advance which video we are rating, we can select the best pieces of stream state information to use as input. The combination of received packets and bandwidth (with or without buffer count) yields the most accurate ratings

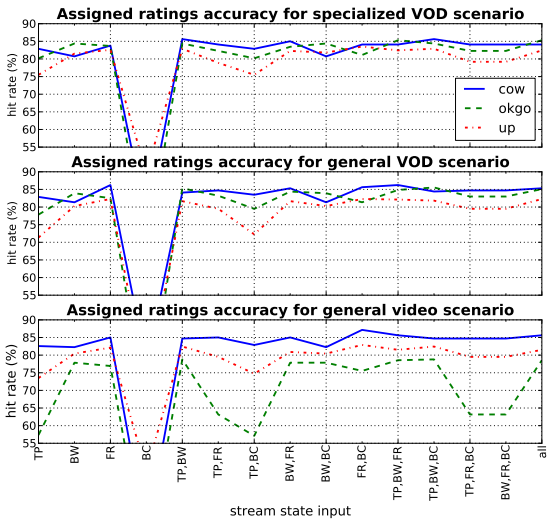

Fig. 5. System accuracy when assigning normalized subjective video quality ratings for (top) the specialized VOD scenario, (middle) the generalized VOD scenario, and (bottom) the general video scenario.

for cow (85.6% accuracy). For okgo all of the stream state information should be used (85.3% accuracy) and for up, frame rate and buffer count in combination indicate quality ratings correctly 83.5% of the time. In general, any of the combinations mentioned here for individual videos are great indicators (better than 80% accuracy) for the other videos as well.

The middle plot in Figure 5 shows the general VOD scenario, with the larger training set. These results are similar to the results for the specialized VOD scenario, although hit rates actually increase slightly for cow in some cases, and decrease slightly for up. An increased number of substreams in the training set mean an increased number of potential matches for an unrated substream. In the case of cow, these additional matches are slightly better, in the aggregate, than when only other cow substreams could be considered matches. The closest substreams to cow are not necessarily always other cow substreams. However, for up, these matches actually contribute "noise" to the system—they are closer in distance, but not better matches than if we limit the training set to other up substreams. As with the specialized VOD scenario, if we know in advance which video we are rating, we can select the best pieces of information to use as input. For cow and up, frame rate alone yields the best results (86.2 and 82.3% accuracy, respectively); for okgo, the combination of received packets, bandwidth and buffer count will give us accurate ratings 85.5% of the time. Similar to the specialized VOD scenario, any of these combinations are strong QoE indicators (better than 80% accuracy) for the other videos as well.

We next examine the general video scenario, shown in the bottom plot of Figure 5. For up and especially for okgo, received packets alone is not a good indicator of stream quality, although it improves in combination with bandwidth. When assigning quality ratings to either cow or up, the combination of frame rate and buffer count assures us of the most accurate ratings in these experiments (87.2 and 82.9%, respectively). This same combination also does fairly well when we are assigning a rating to okgo (accurate 75.5% of the time), but to achieve the best results we should use received packets and bandwidth in combination (78.8% accuracy).

The ratings for the substreams corresponding to cow and up are about as accurate under the general video scenario, in most cases, as in the two VOD scenarios. The ratings for the substreams corresponding to okgo are notably less accurate. Okgo is unlike the other two videos in that it is filmed against a constant backdrop. Thus, the nature of the encoded frames may be different enough between okgo and the other two videos to make finding matches slightly more challenging. Note, however, that there are still a number of stream state measurements that yield ratings with between 75 and 80% accuracy for okgo.

V. Discussion

In examining the results in aggregate, several trends are apparent. First, there is little discernible difference in QoE assessment accuracy between the specialized VOD and generalized VOD scenarios. This is not surprising, as we expect the closest streams to a given stream to be those from the same source video. This result indicates that a highly focused training set, which reduces the time necessary to train the system, would be an acceptable design decision in a stream quality assessment system. It is also promising that the system can assign quality ratings correctly over 75% of the time under less than ideal conditions, i.e. when we attempt to rate a substream corresponding to a video not in the training set. This allows for some flexibility in system design: we do not necessarily have to know in advance which videos the system will rate, although the results improve when we do.

With the exception of when buffer count alone is used, for the VOD scenarios our assigned ratings are accurate more than 70% of the time regardless of what combinations of stream state information we input. Even better, there are a number of combinations for which our assigned ratings are accurate over 80% of the time. This is a promising result, because it means we have a lot of flexibility in the design of a real-world stream quality assessment system in terms of selecting stream state data as input. We can select a set of inputs that works well for all possible streams we will see without being optimal for one or even any of them. This cuts down on the time necessary for the training phase as well, since we don't have to spend time analyzing each set of inputs separately.

For all three scenarios, received packets does not work as well as an indicator of stream quality as bandwidth and frame rate. At first this result surprised us, since the packet-based measurements were such strong indicators of stream quality in our UDP experiments. We surmise that for received packets to work as a stream quality indicator, we need information about the *expected* pattern of received packets, so that we can compare the number of packets received so far with

the number of packets we should have received. Doing so, however, means we would have to abandon the no-reference characteristic of our stream quality assessment system, which would add complexity to the system and limit its flexibility to assess previously unseen video streams.

Our system is able to assign accurate user quality ratings over 75% of the time with just 15 seconds of data. This bodes well for future real-time stream quality assessment for two reasons. First, we are able to quickly detect when stream conditions change and affect user QoE. Ideally, this means we can find ways to use this information to identify the source of the problem and, with enough lead time, mitigate the problem before QoE degrades. Second, it is not necessary to have our system consider all available information about the stream in order to assign an accurate rating—we can use any 15 seconds worth of a video stream. While it is possible that our ratings assignment may improve by using as much information as possible (and indeed is a question we are actively studying), 75% accuracy with 15 seconds of data is very promising already.

Finally, it is important to recognize that these results indicate that assessing video QoE can be done *without the need to poll the users directly about their perceptions of video quality*. While in these experiments we do collect user quality ratings, we do so only to demonstrate that our system is able to assign the same ratings, within a tolerance, that our user population would. In an actual stream quality assessment system, we would only collect stream state information, and not require users to continually rate the stream as we did in our experiments. This mitigates the aforementioned scalability issues inherent in subjective quality assessment systems.

VI. Conclusions

In this paper, we explore changes to the design of a proof-of-concept system whose aim is to assess video QoE solely from objective, application-layer measurements. The original system assessed video QoE offline using UDP streams. In this study, we modify the previous stream quality assessment algorithm, a machine learning algorithm, to collect and assess data for TCP-based streams, and log stream quality ratings continuously in time. We test the modified algorithm's ability to assign stream quality ratings in real time by having the algorithm assign ratings to 15 second portions of a stream, and examining these ratings to see how closely they match up with the actual ratings assigned by our participants. Our results indicate that for video on demand systems and for general video systems, there are several combinations of stream state data for which our algorithm can assign quality ratings with 75 to 87% accuracy. These results indicate that it is possible to accurately assess video QoE using several combinations of stream state data without the need to poll users about their perceptions of video quality. With some additional knowledge of the source video, we can design a training set that improves accuracy further. These results show promise for the design of future real time stream quality assessment systems, because they indicate that assessment can be done accurately on short time scales without direct user participation.

We are working to validate these results with a larger user population and a larger set of videos. Doing so allows us to expand the historical data in our training set, to determine the variability in normalized ratings as the user population increases, and to determine what effects, if any, the characteristics of the videos in the training set have on our results. We are also exploring the differences between TCP and HTTP video streams, to determine which stream measurements are the best indicators of HTTP video QoE.

References

[1] W. Ashmawi, R. Guerin, S. Wolf, and M. H. Pinson. On the impact of policing and rate guarantees in Diff-Serv networks: A video streaming application perspective. In *Proc SIGCOMM*, San Diego, CA, August 2001.

[2] A. Bouch, M. A. Sasse, and H. DeMeer. Of packets and people: A user-centered approach to quality of service. In *Proc. IWQoS2000*, pages 189–197, 2000.

[3] P. Calyam, M. Sridharan, W. Mandrawa, and P. Schopis. Performance measurement and analysis of H.323 traffic. In *Proc. PAM 2004*, Antibes Juan-les-Pins, France, April 2004.

[4] R. G. Cole and J. H. Rosenbluth. Voice over IP performance monitoring. *SIGCOMM Comput. Commun. Rev.*, 31(2):9–24, 2001.

[5] A. Csizmar Dalal. User-perceived quality assessment of streaming media using reduced feature sets. *ACM Transactions on Internet Technology*, 11(2), December 2011.

[6] A. Csizmar Dalal, E. Kawaler, and S. Tucker. Towards real-time stream quality prediction: Predicting video stream quality from partial stream information. In *Proc QShine*, Las Palmas de Gran Canaria, Spain, November 2009.

[7] Dubravko Ćulibrk, Dragan Kukolj, Petar Vasiljević, Maja Pokrić, and Vladimir Zlokolica. Feature selection for neural-network based no-reference video quality assessment. In *ICANN '09: Proceedings of the 19th International Conference on Artificial Neural Networks*, pages 633–642, Berlin, Heidelberg, 2009. Springer-Verlag.

[8] ITU-T Recommendation G.107. The Emodel, a computational model for use in transmission planning. Recommendations of the ITU, Telecommunications Sector, 1998.

[9] E. Keogh and C. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, March 2005.

[10] J. A. Krosnick and L. R. Fabrigar. *Survey Measurement and Process Quality*, chapter Designing rating scales for effective measurement in surveys, pages 141–165. Wiley-Interscience, 1997.

[11] The Linux Foundation. netem. http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[12] J. Nichols, M. Claypool, R. Kinicki, and M. Li. Measurement of the congestion responsiveness of Windows streaming media. In *Proc NOSSDAV*, Kinsdale, Ireland, June 2004.

[13] ITU-T Recommendation P.910. Subjective video quality assessment methods for multimedia applications. Recommendations of the ITU, Telecommunications Sector.

[14] 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 Evgeny Osipov, Andreas Kassler, Thomas Bohnert, and Xavier Masip-Bruin, editors, *Wired/Wireless Internet Communications*, volume 6074 of *Lecture Notes in Computer Science*, pages 252–263. Springer Berlin / Heidelberg, 2010.

[15] 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.

[16] YouTube, LLC. YouTube. http://www.youtube.com.