

Inferring Video QoE in Real Time

Mukundan Venkataraman and Mainak Chatterjee, University of Central Florida

Abstract

Inferring the subjective perception of a video stream in real time continues to be a stiff problem. This article presents MintMOS: a lightweight, no-reference, loadable kernel module to infer the QoE of a video stream in transit and offer suggestions to improve it. MintMOS revolves around one-time offline construction of a k -dimensional space, which we call the *QoE space*. A QoE space is a known characterization of subjective perception for any k parameters (network dependent/independent) that affect it. We create N partitions of the QoE space by generating N video samples for various values of the k parameters and conducting subjective surveys using them. Every partition then has an expected QoE associated with it. Instantaneous parameters of a real-time video stream are compared to the pre-computed QoE space to both infer and offer suggestions to improve QoE. Inferring QoE is a lightweight algorithm that runs in linear time. We implemented MintMOS by creating an actual QoE space using three parameters and 27 partitions by conducting surveys with 77 human subjects. In a second set of surveys using 13 video clips, MintMOS's predictions were compared to 49 human responses. Results show that our MOS predictions are in close agreement with subjective perceptions.

Multimedia streaming over IP networks is poised to be the dominant Internet traffic in the coming decade. It is predicted that more than 90 percent of Internet traffic will carry multimedia content by 2012 [1]. As customers spend more and more time watching videos online, they are increasingly becoming unsatisfied by low-bit-rate videos available on YouTube and are embracing high-definition (HD) streaming services. Providing high-quality streaming services over a best effort and shared infrastructure such as the Internet is non-trivial. This is increasingly placing loads on network elements to deliver streaming content with high *perceptual* quality, including content delivery networks (CDNs), overlay networks, video on demand (VoD), and Internet television (IPTV) infrastructures. Network service providers need to infer, predict, and improve perceptual experience to ensure long-term success.

Network service providers are hence realizing the need to characterize a video stream in terms of *quality of experience* (QoE) [2–4] rather than quality of service (QoS). QoS has long been used to ensure service guarantees for parameters like bandwidth, delay, jitter, and loss. However, QoS lacks an important element in characterizing video streams: that of human perception. Subjective perception of video is best characterized in terms of QoE, which attempts to directly measure human perception of a video sequence when it is played out. This form of subjective quality assessment is in contrast to traditionally used *objective* assessment techniques. The most popular objective quality inference technique is the peak signal-to-noise ratio (PSNR), which is a pixel-to-pixel comparison of frames to detect degradation. Other popular techniques include the mean squared error (MSE), moving picture quality metric (MPQM) [5] and normalized video fidelity metric (NVFM) [6]. Research has shown that objective assessment does not necessarily correlate with human perception.

A common way of subjectively characterizing video contin-

ues to be the mean opinion score (MOS). MOS is representative of the average human response (say, on a scale of 1 to 5) to a given video flow; more generally, it is a mapping of human inference of distortion on a predefined quality scale. MOS as a metric has been known to have its share of drawbacks [7]:

- Subjects tend to avoid the extreme scores.
- *Forgiveness effect*, where users tend to give a higher rating when a playout is long and smooth, is not uncommon.
- Quality in itself is not a very well defined notion and has many dimensions to it.

Present-day MOS calculations tend to be computationally intensive, cumbersome, and not repeatable. Although prior research has produced a variety of mechanisms to predict MOS, most of the proposed techniques are *full reference* in nature. A full reference framework compares a processed video frame to the original frame at the source to infer degradation and predict perceptual quality. However, the availability of the original frame at intermediate hops and routers all over the Internet is nearly impossible. Availability of the reference frames apart, MOS decisions ought to be based on an arbitrary number of factors that affect perceptual quality, which include network-dependent and -independent parameters. Hence, as a first step in improving perceptual quality, there is a need for a lightweight, nonintrusive framework that can accurately measure MOS inside the network core without using the original frame for reference.

We present MintMOS: a loadable kernel module that is an accurate, lightweight, no-reference framework for capturing video QoE inside the network core. MintMOS can accommodate an arbitrary number of parameters on which to base quality inference decisions, and encompasses both network-dependent and -independent parameters. MintMOS internally consists of an inference engine to infer QoE, a network sniffer to snoop traffic, and a QoE space. A QoE space is a known characterization of perceptual quality for various parameters

that affect it. A QoE space is a k -dimensional space that records the perceptual quality for various values of the k parameters, creating a codebook of QoE values. To create a codebook of N values, we generate N video samples using various values of the k parameters. These samples are shown to a diverse group of human subjects that assign quality ratings to them. Instantaneous QoE is inferred calculating the least distortion between the measured values and the reference points in QoE space. QoE spaces are lightweight and portable from one deployment to another.

We validate the accuracy of MintMOS projections by instrumenting an actual QoE space with 27 partitions using three parameters: loss, delay, and encoding bit rate. We generated 27 video samples from an original sequence, and requested 77 human subjects to assign a perceptual quality score to the samples to create a QoE space. We then put 13 video samples to test by asking 49 human subjects to assign a perceptual rating to them. MintMOS's projections are compared to human perception, and we demonstrate a high degree of correlation between our predictions and human perception. MintMOS's projections also outperform PSNR and visual quality metrics (VQM) [8] predictions.

The rest of the article is organized as follows. The next section investigates QoE as a concept. We then provide an overview of related research. We present a high-level overview of the internal components that make up MintMOS and a measure of MintMOS's complexity. Using the concepts introduced, we instrument an actual QoE space around three choice parameters. We then validate MintMOS. We discuss extensions of the MintMOS as a plug-in for various network elements. The final section concludes the article.

What Is QoE?

QoE describes how well a service performs in meeting user expectations. It is a rating of performance from the user's perspective. For Internet-based streaming to compete with existing cable-based infrastructure, QoE delivered by streaming services has to match or outperform QoE from cable-based streaming. Preliminary investigations of Internet-based QoE, however, show a gap between expected and delivered quality. Network service providers lack tools to infer, predict, or improve QoE in the network. This section looks at what QoE is, why it is important, and the areas of QoE we address in this article.

Why QoE?

Traditionally, Internet QoS was aimed at enabling streaming services. The Internet Engineering Task Force (IETF) standardized integrated services (IntServ) and differentiated services (DiffServ) router mechanisms to improve the quality of streaming content, which required changes to every router in the Internet. Given the scale of the Internet, as well as the diversity of various autonomous systems (ASs) that comprise it, these changes could not be completely coordinated. As a result, even after years of slow adoption there have been no significant performance enhancements in terms of video quality delivery, as the Internet continues to operate on a *best effort* delivery model. QoS mechanisms operate with a notion of providing service guarantees to enhance application performance. However, service guarantees alone are not sufficient to raise perceptual quality. QoS-based quality assessments have often been found to be grossly inaccurate in predicting user experience, and as such are not applicable in evaluating video quality [8, 9].

To understand why objective QoS mechanisms often misrepresent quality, we consider an example scenario. Figures 1a

and 1c show two instances of a video sequence at the source before network transmission took place. In other words, these depict the expected playout at the destination with perfect network transmissions. However, the actual playout at the destination of these two instances was observed as in Figs. 1b and 1d. If we ask any group of human observers to rate the quality of playout in Figs. 1b and 1d, the subjects would unanimously rate the latter playout as far better than the former. Surprisingly, PSNR, the most common objective quality function, rates *both these sequences at par* at about 21 dB.

Clearly, there is a strong need to diverge from objective QoS-based quality evaluation approaches toward QoE-based quality evaluations. Monitoring and improving QoE seems to be the only way by which service providers can prevent churn and raise revenue. Service providers apart, QoE as a concept has been the driving factor for evaluating customer satisfaction in a wide variety of domains: from retail, airlines, and food service to customer support. Over the past few decades, QoE has been the most significant measure of human satisfaction in these domains. Understanding and improving QoE has had a great impact on the long-term success of vendors in all these domains.

What Affects QoE?

QoE is the combined and cumulative result of a wide variety of factors. We discuss below some *measurable* parameters that are known to affect QoE, and motivate our contribution presented in the article.

Media Quality — The most important and obvious metric is an evaluation of how good a video sequence plays out to the user. Ideally, users would look for distortion-free playouts. This implies that frames at a destination are available in order (due to a playout buffer) with the same interspacing with which they were sent. Users also look for high-quality audio and, more important, synchronization of audio and video. Common network ailments include delay, packet loss, and delay jitter. Each of these degradations taken in isolation and combination lead to blockiness, blurriness, or even blackouts (Fig. 1b). Arguably, these areas are of primary importance to the networking research community, which works toward minimizing and working around the network related outages that result in these degradations.

Availability — Users would demand services around the clock, and especially when tuning into an event of importance (conference call, telesurgery, live sports, etc.). Users want services that are stable and reliable. Most service providers target the *five nines* in reliability, aiming to deliver services that are available 99.999 percent of the time. This implies that service providers need supporting middleware and services to *continuously* monitor degradations and proactively fix problems before they arise.

Pricing — The pricing set by service providers will greatly dictate customer retention and service adoption. Customers typically want to pay little more than they already do for acquiring online streaming services. Hence, there is a desire to optimize performance at reduced costs.

Network Loss — Loss contributes to missing information, and as such effects media playout. Network loss can occur due to a variety of reasons, and loss manifests as missing packets. The exact manifestation of loss, however, depends on the type of frame impacted by loss. For example, a loss rate of 0.01 percent can either manifest as a minor glitch that can go unnoticed or severely garble playout [10].

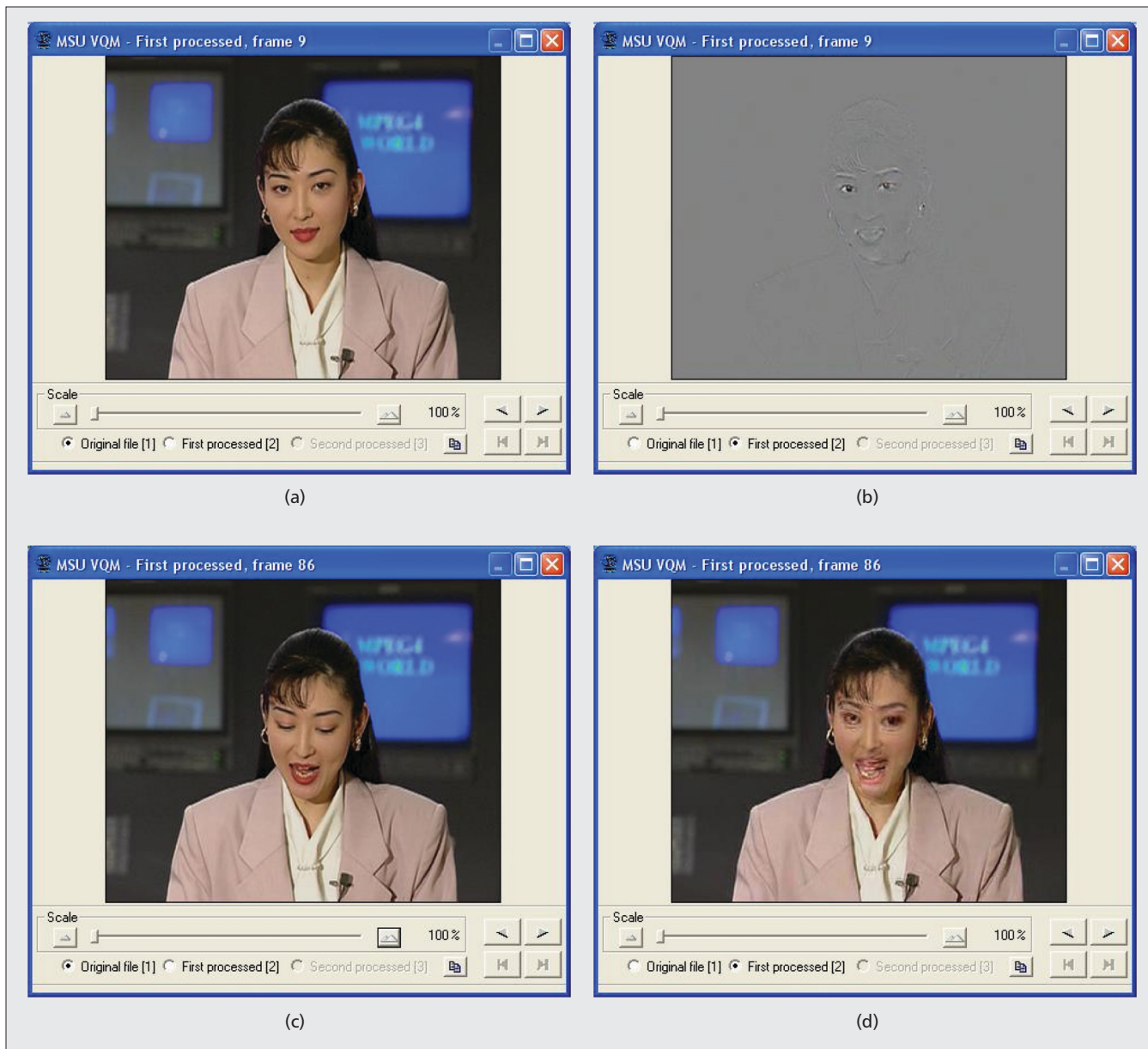


Figure 1. An example of a network transmission with induced degradations: a) *playout at the source of frame 9*; b) *playout of frame 9 at the destination*; c) *playout at the source of frame 86*; d) *playout of frame 86 at the destination*.

Network Jitter — Jitter is the variance in packet arrival times at the destination. Jitter can cause packets to arrive out of order, and most receivers implement a *playout buffer* to nullify the effects of jitter. Although the tolerance to network jitter is high, there is a cap to the amount of jitter a playout can tolerate. Jitter beyond a certain threshold is similar to network loss: if packets arrive out of sequence later than the buffering time, they will be discarded at the receiver. The size of this playout buffer, however, can impact the interactivity presented by the system each time a user flips a channel. This is because channel change includes rebuffering delays: the larger the buffer, the greater the tolerance to jitter, and the more time spent waiting for channel changes to take effect.

Interactivity — This reflects the ability of a system to quickly and correctly change channels (zapping or flipping) as per user request. Acceptable channel change delay is established to be around 1 s end-to-end delay. Channel zapping times of 100–200 ms are considered instantaneous. Much of the delay is incurred in transporting requests from a set-top box (STB)

to a server and back, which entails command processing, queuing delays, and video rebuffer delays. However, much of the functionality is implemented in hardware, making performance evaluation predictable and repeatable. Also, multicasting from edge servers allows switching streams to be more amenable. Channel zapping is fairly well understood and optimized by present-day service providers.

Related Work

Depending on the amount of information needed to perform video quality assessment, architectures can be broadly classified as no reference (NR), full reference (FR), or reduced reference (RR). FR schemes require an entire copy of the original (key frame to infer quality, while RR schemes require partial (key parameters extracted from a video) information. The only feasible schemes that can reside as a standalone module in the network core are the NR types.

Video quality assessment can be either objective or subjective. Objective video quality assessment usually involves a

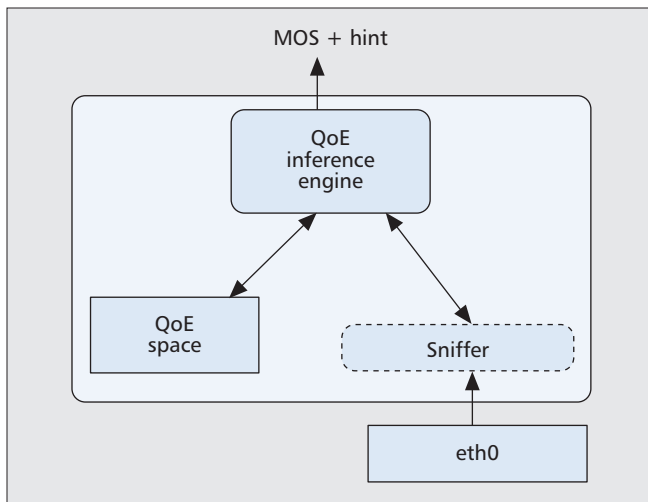


Figure 2. High-level architectural overview of MintMOS.

characterizing function of measurable parameters (networking or transcoding) that evaluates the quality of a reconstructed frame. For example, the media delivery index (MDI) [11] is an NR framework that can reside in network terminals to report loss and delay rates. While it can provide a comprehensive log of these statistics, it does not map them to subjective perception. Other objective quality assessment techniques like PSNR that are FR in nature are both infeasible inside the network core and grossly inaccurate at inferring perceptual quality.

There has been a recent surge of interest in subjectively assessing quality of a video stream. Even at a time when there is little or no consensus on the exact nature of such a model, the basic idea has been to provide a subjective interpretation of various events on a video stream, such as loss, delay, jitter and so on. The video quality metric (VQM) [8] provides a reasonably good indication of subjective perception. However, VQM requires the original frame for reference and is likewise infeasible inside the network core. Apart from these, there are other proprietary video quality metrics exclusive to organizations that own them (e.g., Agilent, Symmetricom). These metrics are not open to the public for free inspection or usage.

Alternates to using MOS have also been suggested. Work by Tasaka *et al.* [12] express QoE in terms of interval scale or psychological scale, where the proposed SCS strategy switches from error concealment to frame skipping based on a threshold of error concealment. Some other studies ask the user to provide certain specific feedbacks. For example, users could be asked to report visibility of certain artifacts on screen, or to suggest changes to artifacts to make the sequence more appealing to them. Another dimension of work tries to capture user dissatisfaction that manifests due to the combined effect of various factors as a single *click* instead of a MOS score [7]. However, MOS continues to be a convenient way of quantifying scores because it provides a uniform interpretation of quality for various purposes (billing, peak hour usage, capacity planning, etc.). The quest for an easily configurable, lightweight, and standalone framework for inferring QoE in real time at arbitrary Internet nodes continues to be an open problem.

Architectural Overview

MintMOS consists of the following components: the sniffer, the QoE inference engine, and the QoE space. Figure 2 provides a high-level overview of the architecture of MintMOS. The sniffer sniffs packets from the networking interface, and generates statistics about the parameters in question (e.g.,

loss, delay, jitter and so on). The sniffer feeds these parameters as input to the QoE inference engine. The QoE inference engine operates on the QoE space to predict the current QoE. Modularity allows these components to evolve independent of each other. We begin with the QoE space and take a closer look at each of these components.

QoE Space

The first step in putting together the MintMOS framework is the creation of a QoE space. A QoE space is a known characterization of expected QoE for various values of the k parameters that affect it. In general, if we assume k parameters that affect the quality of video, those k parameters can be used to express the QoE space in the form of a k -dimensional vector, Γ , as follows:

$$\Gamma = [\gamma_1, \gamma_2, \dots, \gamma_i, \dots, \gamma_k],$$

where γ_i , $1 < i < k$, represents the instantaneous value of the i th parameter. Thus, the vector Γ provides the instantaneous state of the video stream in transit. Due to the dynamism of the network, parameter γ_i constantly changes and so does the vector Γ . The ever changing vector Γ can be interpreted as the motion of a point in a k -dimensional QoE space. We argue that associated with every point in this space is a *QoE index*, which represents the quality of experience offered by the video network to the user.

Since the QoE space is a record of parameter values and measured QoE, a variety of parameters can be used to generate video sequences for conducting surveys. For example, one could construct a seven-dimensional QoE space that consists of parameters such as content type (sports, news, action), mobile hop (Y/N), delay (0–100 s), jitter (0–40), loss (0–100 percent), encoding bit rate, and pricing.

QoE Inference Engine

The QoE inference engine is a component that takes instantaneous values of the k parameters (the vector Γ) as input, and produces the expected QoE as the output. The inference engine searches the QoE space to find the closest match to the partition to which Γ belongs, and returns the QoE associated with that partition.

When a new pair of values are provided to the inference engine, the expected QoE is derived by selecting the least Euclidean distance between the given point and the centroids in the QoE space. It is intuitive that the greater the number of precomputed QoE indices (i.e., M), the less the distortion and the more accurate the QoE prediction.

Implementation Overview

Algorithm 1 shows the internals of the individual components in brief pseudocode. For brevity, we reproduce internals of multiple components in the same algorithm. Lines 5–12 describe the sniffer, which captures packets from the live wire (*dev*), and uses these to compute and update statistics. The inference engine itself is described in lines 15–27, which assume a QoE space stored as a multidimensional array. The algorithm computes the least distortion to return the appropriate index into the QoE space (*handle*).

Complexity of the Framework

For MintMOS to be an effective tool, it must scale well in servicing a large number of flows in real time at various points in the network. To measure the processing overhead of our framework on hardware, we implemented MintMOS in C++

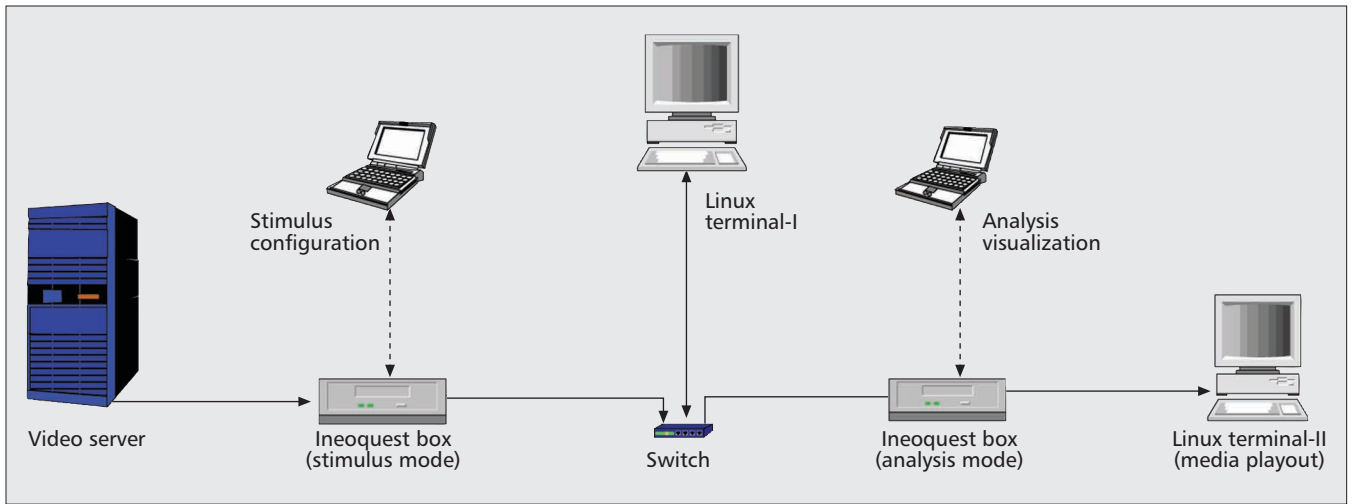


Figure 3. Network testbed used to perform experiments.

with a precomputed QoE space. The module takes as input values of parameters in a specific range, and returns the anticipated MOS upon completion. We report our experience with running MintMOS on standard off-the-shelf Linux terminals (running a Pentium 4 with 1 Gbyte RAM). We measure the time it takes to perform MOS calculations with increasing number of partitions of the QoE space (N) as well as the number of parameters to estimate QoE (k). Our results create a benchmark for using this framework on routers and end-user systems with comparable hardware.

Since it is practically infeasible to create an actual sample space with more than tens of thousands of partitions by conducting subjective surveys, we modified our program to accommodate a varying level of samples by populating the sample space with spurious MOS inputs. This was done in conjunction with preserving the actual number of iterations and comparisons of calculating MOS. We report the *worst case* running time, where we assume that the desired value is found in the last comparison. In other words, if there were 1000 reference points and 20 parameters, the desired MOS is found after having compared the 1000 distortions to all 20 QoE indices.

Experimental Setup

The topology used to measure complexity is shown in Fig. 3. Our setup consists of a video server that initiates a push-based stream onto the network. The server can encode a video stream at various bit rates and frame rates. The server is connected an IneoQuest (IQ) Sigulus G1-T [13] box configured in the *stimulus* mode. In this mode the IQ can generate impairments to the flow by inducing loss, delay, or jitter. The flow is streamed to another IneoQuest G1-T configured in the *analysis and playback* mode. The stream is then passed on to a destination (Terminal-II) for visualization.

The boxes are connected through a switch, and we attach a Linux terminal (Terminal-I) to this switch. Terminal-I runs MintMOS by passively snooping network traffic and inferring QoE. Terminal-II, which is used for playback, also implements MintMOS where it measures perceived quality at the destination. We report the measured complexity of MintMOS at Terminal-I.

The Effect of Number of Samples (N)

The accuracy of MOS projections gets richer with more reference points in the QoE space. An increase in the number of reference points means an increase in the number of comparisons required to infer MOS. While a crude projection of a MOS can be inferred with a handful of reference points (say 5 or 10 samples), it is possible to create a very fine grained and exhaustive set of samples to detect or isolate degradations to a very fine degree.

The effect of increasing the number of QoE space partitions (log scale) to the output rate is shown in Fig. 4a. The output rate is the number of MOS calculations that can be performed *per second* for the given number of QoE indices to be compared. For a small number of QoE indices with less

```

1: Program INPUT: Sampled values of the  $k$ -parameters (i.e.,  $p_1, p_2, p_3, \dots, p_k$ ).
2: Program OUTPUT: MOS projection for the parameters. Return one of  $MOS[N]$ 
3: ...
4: ...
5: char* dev;
6: char errbuf [PCAP_ERRBUF_SIZE] ;
7: dev = pcap_lookup (errbuf) ;
8: pcap_t record;
9: if dev != NULL then
10: record = pcap_open_live (dev, BUFSIZE, 0, -1, errbuf) ;
11: compute_and_update_stats (record) ;
12: end if
13: ...
14: ...
15:  $MOS[N][i][j] \dots [k] \leftarrow$  Precomputed QoE Space
16: handle  $\leftarrow -1$ 
17: min  $\leftarrow 9999$ 
18: for  $i = 1$  to  $N$  do
19:   for  $j = 1$  to  $k$  do
20:     distortion $_j = (p_j - Param_j[i])^2$ 
21:   end for
22: distortion[i] =  $\sum_{j=1}^k$  distortion $_j$ 
23: if min < distortion[i] then
24:   min  $\leftarrow$  distortion[i]
25:   handle  $\leftarrow i$ 
26: end if
27: end for
28: ...
29: return struct{MOS[handle]}

```

Algorithm 1. Internal structure of MintMOS.

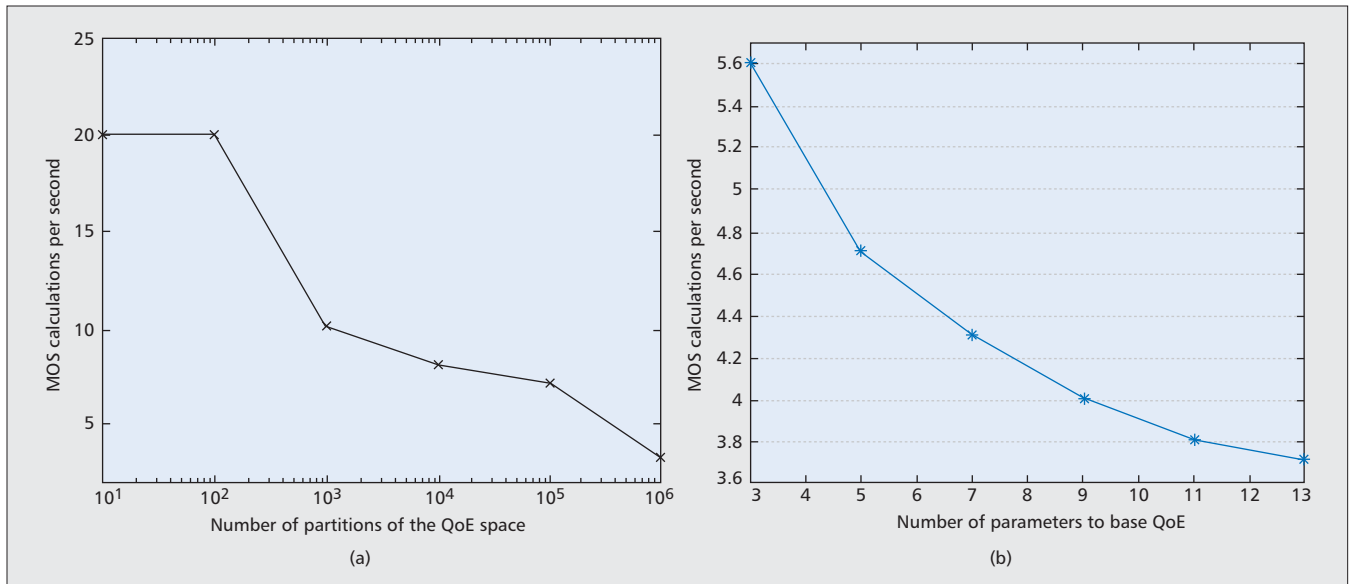


Figure 4. Measuring processing and memory overhead for MintMOS: a) effect of increasing number of partitions for $k = 11$; b) effect of increasing number of parameters for $N = 10^6$.

than 100 partitions, increasing the number of partitions has no real effect on the output rate of the program. The effect only begins to show when the number of partitions approaches 1000 or more. Shown in the plot is the effect of increasing the number of partitions to 1 million samples. Even at such a large number, we could perform 4 MOS calculations/s. Since our time measurement takes into account both the user level interaction with the kernel as well as the actual CPU time the program consumes, this reflects on our ability to perform MOS calculations in real time on a router with comparable hardware. In other words, our module can compute 1200 MOS calculations/min for a small number of partitions (< 100) and 240 computations/min even for 1 million reference points to compare. For a reasonable number of reference points, MintMOS can service more than 1000 flows/min on a router with a similar configuration.

The Effect of Number of Parameters (k)

The cost of adding more parameters, or axes, to the QoE space should not be high. In fact, the greater the number of parameters used in inferring MOS, the more accurate the projections. We examine the effect of increasing the number of parameters in calculating MOS for a QoE space with 1 million partitions by varying the number of parameters from 3 to 13.

Figure 4b shows the effect of increasing the number of parameters. With rising k , the only difference to the module is to compute k distances for each reference point. Even with a large number of reference points in the sample space, increasing the value of k from 3 to 13 has little effect on the output rate, which drops from around 5.5 MOS calculations/s to around 3.75 calculations/s. Again, this can be inferred as servicing 210 flows/min for 3 parameters, and up to 23 flows/min for 13 parameters for 1 million reference points to compare.

Instrumenting MintMOS

We now create a working MintMOS framework with an actual QoE space populated offline by subjective surveys. A QoE space is constructed by obtaining various video sequences with various values of the k parameters. These video sequences are then shown to a large number of human subjects, who individually score the video samples. We create a three-dimensional QoE space that involves three parameters: loss, delay, and encoding bit rate. While loss and delay are network-depen-

dent parameters, encoding bit rate is a source parameter. With a QoE space in place, we use SE to discover relationships within it to create a standalone framework.

Creating a QoE Space: Survey with Human Subjects

For our three choice parameters, we created a total of 27 distorted video sequences using the *foreman* clip for various values of loss, delay, and bit rate. These samples were then put to an open survey, where we requested 80 human subjects to rate these video sequences on a scale of 1 to 5. The survey was double stimulus: subjects were asked to rate a distorted video sequence with respect to the original video sequence. Both the original and distorted video sequences were shown to subjects at the same time on two different (adjacent) screens. The viewer distance was maintained at approximately $8 \times D$, where D is the height of the playback.

Subjects were chosen with sufficient diversity in age, gender, and expertise in subject matter. Outliers were identified by interspersing a shown video sequence multiple times and recording the ratings each time. We identified a total of 3 outliers: subjects who rated the same distorted video sequence differently on different occasions. This effectively accounts for 77 subjective ratings. Table 1 shows the values of our three choice parameters, along with the average MOS and variance obtained from the survey.

A few aspects are noteworthy about human response to various stimuli. Subjects consistently reacted strongly to high levels of loss. For example, samples 3, 8, 11, and 16 all experience a higher fraction of packet loss, and subjects unanimously rated the experience close to *very poor*. Among the other parameters, subjects preferred higher bit rates with delayed frames to be as good as a low-bit-rate playout without delay. For example, subjects thought that sample 7 was comparable to sample 5. Subjects tended to avoid the extreme score of 5 for any video, even if it had very little or no distortion compared to the original video (e.g., sample 1). However, subjects did not hesitate to rate a video extremely poor (score of 1) if the video was highly impaired and if artifacts on screen were barely visible.

Validating the Framework

With the QoE space thus characterized, we finally turn to validating the effectiveness of our framework by running a final set of surveys. We generated 13 video samples using three dif-

Seq. no.	Bit rate (kb/s)	Delay (s)	Loss (frac.)	PSNR (dB)	Avg. MOS	Var
1	1600	0.016	0	42.68	4.16	0.267
2	1600	0.08	0.05	33.64	2.71	0.253
3	1600	0.094	0.85	7.97	1.32	0.212
4	1600	5.809	0	27.13	3.31	0.318
5	1600	2.632	0	36.36	3.97	0.077
6	1100	2.85	0	26.91	3.37	0.29
7	1100	0.019	0	40.75	3.98	0.29
8	1100	0.09	0.76	14.53	1.02	0.026
9	1100	3.981	0.53	22.94	1.96	0.245
10	1100	0.092	0.24	19.35	1.67	0.245
11	600	0.016	0	37.33	1.12	0.139
12	600	3.677	0.43	23.05	2.63	0.257
13	600	0.712	0.73	15.83	1.94	0.202
14	600	3.205	0.11	26.35	3.33	0.249
15	100	0.018	0	30.63	2.66	0.535
16	100	0.084	0.66	15.72	1.09	0.134
17	100	0.087	0.44	20.21	2.38	0.289
18	100	2.819	0	23.47	1.97	0.207
19	2886	2.76	0.71	17.33	1.25	0.103
20	2886	2.62	0.43	20.93	1.5	0.217
21	2886	0.03	0.12	27.24	2.21	0.234
22	2866	1.09	0	29.12	4.87	0.119
23	2886	2	0.18	23.75	1.75	0.101
24	100	0.001	0	28.01	3.375	0.285
25	100	1.65	0.01	24.21	2.215	0.312
26	600	1.65	0.01	22.1	2	0.293
27	2886	0.001	0.01	27.87	4.62	0.165

Table 1. 27 video samples generated by varying three parameters (loss, delay, and encoding bit rate). These samples were put to an open survey to create 27 partitions of the QoE space.

ferent clips at levels of distortion in terms of loss, delay, and encoding bit rate. We used the values of these three parameters as input to our framework, and noted the predicted QoE for the 13 video samples. We then gathered the opinions of the 49 human subjects used in the previous round of survey to score these samples. Once again, this survey was double stim-

ulus (i.e., subjects were presented both the original and distorted video sequences). We use the average scores assigned by subjects as the benchmark to evaluate accuracy of various scoring schemes. We seek to understand how accurately MintMOS's projections match with those from the survey. We also compare our results with more established metrics like PSNR and VQM.

Table 2 tabulates our findings. Shown in the table are the input parameters used to create the sequence of 13 video samples, the type of clip used, the PSNR readings for the samples, the VQM ratings (as obtained from the MSU toolkit [14]), the mean MOS assigned by 49 subjects, the MOS assigned by our framework, as well as the error percentage of our predictions.

Samples that experience a significant percentage of losses are unanimously rated as poor quality clips. All of PSNR, VQM, our projections, as well as user perception are in line with this thought. For example, samples 3 and 10 are rated as poor by MintMOS, PSNR (which records the lowest values), and VQM (which records the highest values).

PSNR, which directly compares a processed sequence with the original, often fails to correctly correlate perceptual quality to degradation. For example, PSNR rates samples 4 and 6 with a similar reading (≈ 20 db). Subjects, however, rate sample 4 poor while they rate sample 6 relatively high. Also, of all the clips used in validation, subjects found clips 13, 11, and 6 to be among the best while PSNR rates them *average* in a range of 17–21 db.

VQM projections work in an inverse way: higher the value indicated, lower shall be the perceived quality. VQM projections provide a better indication of perceptual quality than PSNR. While VQM does identify clips with lower perceptual quality to some degree, the results are not too accurate. For example, VQM rates sample-4 relatively high while subjects rate this clip between *very poor* to *poor*. Again, subjects rate samples 1 and 11 almost at par, while VQM predicts sample 1 to be far better than sample 11. In general, VQM's ability to loosely classify a clip as very poor or very good is accurate, but it often fails to successfully distinguish clips that share similar levels of degradation. This is evident in a direct comparison of VQM scores and subjective perception for samples 1 and 5, and samples 1, 11, and 13. Note that VQM is a full reference scheme, which requires the original frame at all times to provide quality evaluation.

MintMOS's prediction was able to obtain the highest correlation with subjective perception. Noteworthy is the degree of correlation, with a maximum error rate of 4.0 percent for sample 4 and a minimum of 0.4 percent for sample 1. Our MOS projections are dependent on the accuracy of the QoE space. Since the QoE space is in turn

populated by human subjects, the degree of correlation to a past event with a similar newer event is what makes our projections accurate. The mean distortion in using MintMOS with 27 reference points was determined to be 0.53. For smaller sample spaces ($N = 9$), we measured average distortion to 1.02.

Seq. no.	Clip name	Bit rate (kb/s)	Delay (s)	Loss (rate)	PSNR (dB)	VQM	MOS from survey	Projected MOS	Error (%)
1	foreman	1600	0.017	0	25.32	3.47	4.18	4.16	0.4
2	foreman	100	0.0009	0	17.05	7.69	2.71	2.66	1
3	foreman	1600	0.0931	0.88	4.52	33.44	1.29	1.32	0.6
4	foreman	1100	0.577	0.22	20	4.888	1.87	1.67	4
5	foreman	1600	0	0	23.68	3.69	3.29	3.37	1.6
6	news	2886	0.005	0	20.38	6.7	4.6	4.62	0.4
7	news	2886	0.01	0.33	11.04	15.24	1.61	1.5	2.2
8	news	1100	0.06	0	20.38	6.7	4.1	3.98	2.4
9	news	100	0.01	0.55	11.09	14.76	1.2	1.09	2.2
10	coastguard	1600	0.085	0.78	6.73	31.23	1.22	1.32	2
11	coastguard	1100	0.01	0.05	17.14	10.64	4.125	3.98	2.8
12	coastguard	100	0.09	0.33	12.88	14.71	3.22	3.37	3
13	coastguard	1600	0.05	0	17.258	10.42	4.33	4.16	3.4

Table 2. Results from a survey conducted to measure the accuracy of MintMOS projections.

Improving QoE: MintMOS also outputs a hint to improve QoE for every sample. For example, it suggests sample 2 to increase bit rate with a hint of {1}. This is because sample 2's parameters suggest it is operating close to entry 15 in the QoE space. Increasing the bit rate is due to a (15,1) relationship (Table 2). Other suggestions may likewise be investigated by determining the entry point in QoE space to which a sample belongs, and its corresponding *From-To* relationships.

Discussions

Users enjoy online video more than television because they have greater control and customization of what and when to watch. The rise in popularity of user generated content, as well as the potential of moving existing cable-based video content online, would tremendously foster the growth of Internet video. Content providers hence need to both understand and improve QoE for long-term success.

The underlying networking technologies that support streaming services need an in-network tool to infer and improve QoE, including video on demand (VoD), IPTV, content delivery networks (CDNs), overlay networks, and peer-to-peer (P2P) systems. Each of these systems need to make routing and redirection decisions in short timescales, monitor peak hour quality, decide on the number of participating nodes, and make decisions on procuring new hardware. Many of these decisions are currently based on extensive network measurements that usually measure loss, delay, or jitter. However, these numbers provide very little insight on how an end user will perceive quality. A framework such as MintMOS could ideally fit the bill for all of these services by inferring, predicting, and helping to improve QoE.

The fact that our framework is modular, lightweight, and real-time, and scales up to a large number of flows allows us to implement various solutions using it. We show ways to adapt our framework to a wide variety of streaming services: from collecting user-perceived quality from every customer to

overlay streaming networks to VoD and IPTV service providers. We show how we can use the components of the MintMOS framework as a solution to these domains.

Collecting End-User QoE

Most network service providers are at a complete loss when it comes to keeping a log of QoE perceived by *all* of their clients. Our framework can easily be adapted to run on virtually any client laptop or desktop to monitor user perceived quality for a small set of flows the user is viewing. These statistics can be bundled together and sent back to the source, providing continuous feedback and creating a comprehensive log of QoE fluctuations. Such a log can be invaluable to service providers, helping them plan capacity, monitor peak hour QoE, and bill their clients.

A simple adaptation of MintMOS to do this is shown in Fig. 5a. The output from the framework makes its way back to a module that creates a feedback packet and inserts the source's address as the destination, and simply sends it back to the source. The feedback to source module could be further fine-tuned to specify the periodicity of reporting such statistics back to the source.

Overlay Networks

Peer-to-peer and overlay networks are increasingly becoming a popular way of disseminating streaming content. An overlay network is a group of nodes interconnected to each other by virtual links that can potentially exploit paths BGP does not advertise. For example, work on resilient overlay networks (RONs) [15] has shown that Internet outages can sometimes last several seconds to minutes and are quite frequent. Routing using overlay networks often chooses paths based on metrics such as hop count or latency, altogether ignoring perceptual QoE.

We implement a plug-in of our module for RONs (Fig. 5b), which now allows RON nodes to monitor and estimate QoE for various paths. RON nodes continuously exchange probes

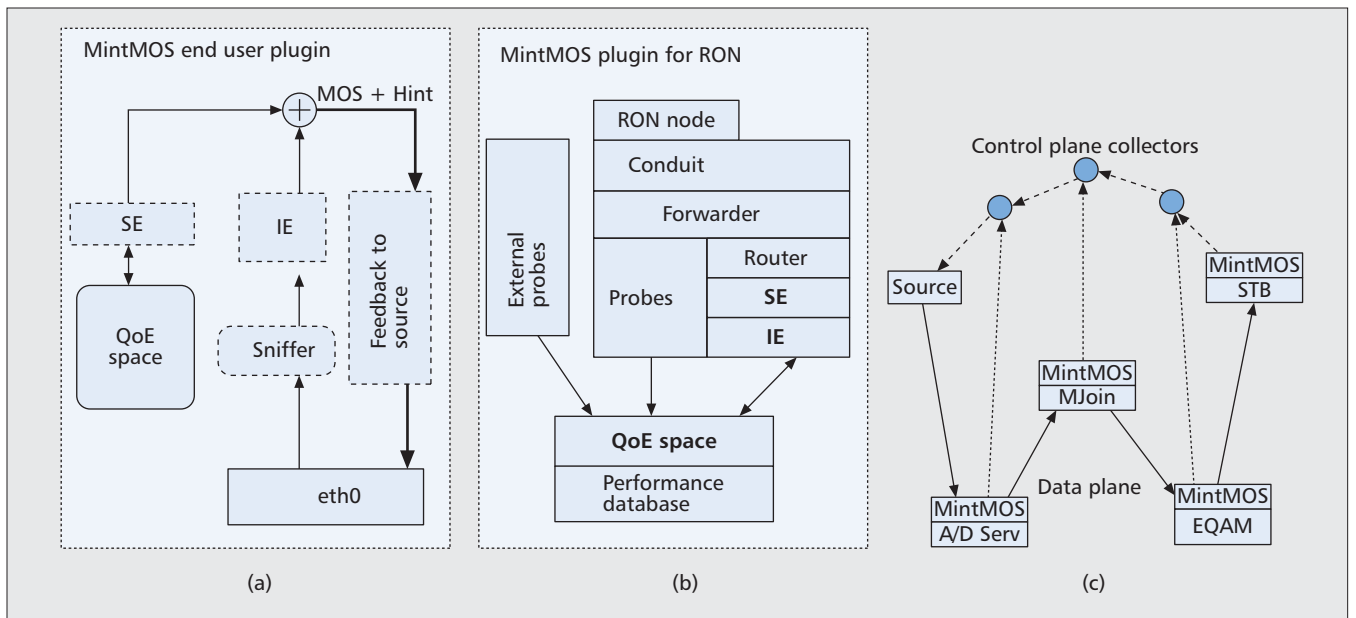


Figure 5. MintMOS Plugins for: a) end-user performance monitoring; b) for overlay networks; and c) for VoD and IPTV service providers.

at regular intervals of time. These probes provide loss and delay statistics for a given virtual link between two RON nodes. Using these probe packets and the statistics from them, our framework running on top of a RON can continuously monitor *expected QoE* as an EWMA between any two nodes in the overlay. For example, if one knows the EWMA of loss and delay for a virtual link as (l_i, d_i) , one could use various values of bit rate to create a series of anticipated QoE for every virtual link. A QoE characterization of virtual links helps RON nodes make routing decisions, choose an appropriate bit rate, and help select the right alternative path when many are available.

VoD and IPTV

Service providers of VoD and IPTV need to continuously monitor QoE at five critical points in their network infrastructure: the source or server, the analog-to-digital (A/D) servers, the MJoin, the EQAM, and the end-user set-top boxes. Nodes before EQAM typically service a large number of flows. Much of the existing infrastructure allows service providers to monitor these critical points to determine usage statistics and capture users inputs (zapping) on a control plane that resides above the data plane.

Our framework can seamlessly be integrated at these critical points and provide perceptual QoE at these points to be streamed to the control plane, even for a large number of flows (Fig. 5c). This would allow service providers to monitor QoE and its respective degradation at all five points. This has direct implications in fault isolation, billing clients, capacity planning and purchasing new hardware to improve QoE.

Content Delivery Networks

CDNs move streaming content *closer* to clients by caching copies on multiple servers worldwide. CDNs perform extensive network and server measurements, and use these to redirect clients to different servers over short time scales (e.g., Akamai [16], the worlds largest and most popular CDN).

A study by Su *et al.* [17] from 140 vantage points in the Internet reveals that Akamai redirections overwhelmingly correlate with network *latencies*. This implies that server redirections are based on delay more than any other parameter. Servers in a CDN can utilize their existing network measurements to estimate loss and delay rates and utilize a QoE space

to perform server redirections based on estimated QoE rather than network latencies. Feedback from MintMOS can further be used to decide encoding bitrates.

Conclusions

Internet-based streaming services are poised to be the next big thing: users have greater control over what and when to watch, user-generated content already accounts for 30 percent of streaming content viewed everyday, and the Internet provides an ideal platform for interactivity with customers where advertisements are one click away from browsing and buying. To successfully tap into this market, content providers need to infer, predict, and improve the quality of experience of their customers. Inferring QoE in real time at various network terminals continues to be hard, computationally intensive, and often requires expensive machinery, which cannot be universally deployed, or reference frames to draw a conclusion.

We present MintMOS: a lightweight, scalable, no reference framework for inferring QoE of a video stream and offering suggestions to improve it. MintMOS is flexible enough to accommodate any number of parameters that can affect video quality, from network-dependent to network-independent. MintMOS revolves around a QoE space, which is a k -dimensional space for k parameters used to measure quality. The QoE space creates a mapping between parameter values and their associated perceptual quality. Inferring QoE or offering suggestions to improve it can be done using the QoE space to base their decisions. We instrumented a QoE Space around 3 parameters and 27 partitions, and demonstrated its effectiveness in projecting MOS. In the end, we believe MintMOS is accurate because it replays a past known association with conditions similar to those observed.

MintMOS proves to be a viable tool for inferring QoE at arbitrary nodes in a network; an implementation of MintMOS on standard off-the-shelf Linux terminals demonstrated this. We could measure the QoE for 1200 video flows per minute for small samples, and for up to 200 flows per minute for large sample spaces. Modularity allows MintMOS to be adapted to a wide variety of network elements.

Lastly, quality is an abstract concept. Even a numerical reduction of quality only helps to indicate it and can never accurately score it. However, QoE is the most *significant* mea-

sure of human satisfaction, and has long been used to characterize customer experience with vendors in a wide variety of domains. Understanding and improving QoE has had a profound impact on the long-term success of many organizations.

References

- [1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2008–2013," White Paper, July 2009; <http://www.cisco.com>.
- [2] P. Brooks and B. Hestnes, "User Measures of Quality of Experience: Why Being Objective and Quantitative is Important," *IEEE Network*, vol. 24, no. 2, 2010.
- [3] K. T. Chen *et al.*, "Quadrant of Euphoria: A Crowdsourcing Platform for QoE Assessment," *IEEE Network*, vol. 24, no. 2, 2010.
- [4] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A Generic Quantitative Relationship between Quality of Experience and Quality of Service," *IEEE Network*, vol. 24, no. 2, 2010.
- [5] C. J. van den Branden Lambrecht and O. Verscheure, "Perceptual Quality Measure using a Spatio Temporal Model of the Human Visual System," *Proc. IST/SPIE Conf., Video and Compression: Algorithms and Technologies*, vol. 2668, Feb. 1996.
- [6] H. R. Wu, T. Ferguson, and B. Qiu, "Digital Video Quality Evaluation using Quantitative Quality Models," *Proc. 4th Int'l. Conf. Sig. Process.*, Beijing, China, Oct. 1998.
- [7] K. T. Chen, C. C. Tu, and W. C. Xiao, "OneClick: A Framework for Measuring Network Quality of Experience," *Proc. IEEE INFOCOM*, 2009.
- [8] M. H. Pinson and S. Wolf, "A New Standardized Method for Objectively Measuring Video Quality," *IEEE Trans. Broadcasting*, vol. 50, no. 3, Sept 2003.
- [9] M. Venkataraman, M. Chatterjee, and S. Chattopadhyay, "Evaluating Quality of Experience for Streaming Video in Real Time," *IEEE GLOBECOM*, Hawaii, Dec. 2009.
- [10] J. Greengrass, J. Evans, and A. C. Begen, "Not All Packets Are Equal, Part II: The Impact of Network Packet Loss on Video Quality," *IEEE Internet Comp.*, vol. 13, no. 2, Mar. 2009.
- [11] J. Welch and J. Clark, "A Proposed Media Delivery Index (MDI)," *IETF RFC 4445*, Apr. 2006.
- [12] S. Tasaka *et al.*, "The Effectiveness of a QoE-Based Video Output Scheme for Audio-Video IP Transmission," *ACM Int'l. Conf. Multimedia*, Vancouver, Canada, Oct. 2008.
- [13] Ineoquest, "Singulus G1-T Equipment" <http://www.ineoquest.com/singulus-family>.
- [14] Video Evaluation Toolkit (Moscow State University, Moscow, Russia), Aug. 2008; <http://graphics.cs.msu.ru>.
- [15] D. G. Andersen *et al.*, "Resilient Overlay Networks," *Proc. 18th ACM SOSP*, Oct. 2001.
- [16] Akamai Inc.; www.akamai.com.
- [17] A. Su *et al.*, "Drafting Behind Akamai," *ACM SIGCOMM*, Pisa, Italy, Sept. 2006.

Additional Reading

- [7] A. Gersho and R. M. Gary, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1991.

Biographies

MUKUNDAN VENKATARAMAN (mukundan@eecs.ucf.edu) is a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at the University of Central Florida, Orlando, where he has been the recipient of a Presidential Doctoral Fellowship. He received a B.E. degree from Visveswararajah Technological University, India. He was a research staff member at the Pervasive Computing team at SET-Labs, Infosys Technologies Ltd., Bangalore. His current research interests include multimedia streaming protocols and architectures to support superior streaming over the Internet. He won the Best Student Paper Award in Asia-Pacific (R10) from the IEEE in 2003 at Penang, Malaysia; and the Best Student Paper awards consecutively in 2002 and 2003 from IEEE Bangalore (B. R. V. Varadan Paper Award).

MAINAK CHATTERJEE (mainak@eecs.ucf.edu) is an associate professor in the Department of Electrical Engineering and Computer Science, University of Central Florida. He received a B.Sc. degree in physics (Hons.) from the University of Calcutta, an M.E. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, and a Ph.D. degree from the Department of Computer Science and Engineering, University of Texas at Arlington. His research interests include economic issues in wireless networks, applied game theory, cognitive radio networks, and mobile video delivery. He has published over 100 conferences and journal papers. He is a recipient of the AFOSR sponsored Young Investigator Program (YIP) award. He serves on the executive and technical program committees of several international conferences.