An Extensible Distributed Measurement Platform for Analyzing Quality-of-Experience (QoE) of Multimedia Applications over Wireless Networks

Y. W. Wong, W. C. Lau, K. M. Chan, Y. Yang, C. Y. Tang, F. Lam, K. M. Lo
Department of Information Engineering
Chinese University of Hong Kong
Hong Kong, China

Abstract — **With rapid advancement of 3G wireless network technology, mobile Internet access is becoming more and more popular in recent years, and web access by mobile phones even surpassed that by other means. Users' level of satisfaction of services based on such rapidly developing technology immediately becomes an important issue. The distributed QoS (Quality of Service) / QoE (Quality of Experience) measurement platform proposed here solves this problem by offering network engineers capability to monitor network service quality and the likely QoE as experienced by phone users while watching streaming videos, for purposes like benchmarking and tracking down problematic cells. There are three major components of the proposed system: (1) client program; (2) backend server; and (3) web interface. The client program runs on Android smartphones. Using the various equipped hardware sensors, radio information, location information and network statistics, etc., are collected. By exploiting the mobility of smartphones, data can be collected wherever the phone is located. Those data are uploaded to the backend server for data aggregation and processing. In particular, a QoE computation algorithm recommended by ITU is used to convert raw data into QoE scores. To facilitate easy access to the general network service quality, a versatile web interface is also implemented to give detailed information of individual data collection sessions.**

Keywords - QoE; smartphone; distributed; wireless

I. INTRODUCTION

To provide a benchmark to customers for reference, as well as to network engineers for tracking down problematic cells, a widely acceptable measure of the service quality is always needed in the telecommunication industry.

Previously, voice communication quality was the only major concern for mobile network operators. However, the recent boost of Internet browsing with mobile smart devices suggested a revision to the QoS measurement strategy. Moreover, with the deployment of newer generation mobile network infra-structure and availability of more powerful smartphones, phone users' demand for multimedia content, like high-definition music and movies, is increasing dramatically. An effective way to evaluate user experience for movies watched on those mobile devices is urgently needed.

Towards that aim, two measurement metrics are defined: *Quality of Service* (QoS) and *Quality of Experience* (QoE). Both metrics can be used to quantify the media quality degradation after network transmission. While the former one is objectively taken (e.g. RSSI, uplink / downlink bandwidth, round-trip-time (RTT), packet loss rate), the latter one relies on human's subjective reporting of their impression. The two have a generally positive correlation, yet, as shown in a later section, mismatch between the objective QoS and the subjective QoE occurs from time to time. With increasing demand for multimedia contents over mobile network, it is a must for mobile operators to have an efficient way to measure such QoE scores.

As suggested by ITU ([1-4]), the rigorous evaluation of QoE requires recruitment of a massive pool of subjects to score manually the presented movies, which is just not affordable for the current industry. Obviously, an automated approach is preferred to the traditional human scoring method. Several proposals along such direction have been suggested to ITU for consideration [5], like British Telecom, Yonsei University, and Chiba University. They all aimed at deriving image processing algorithms which approximate the subjective QoE scores that would be yielded if the same set of movies is presented to human subjects. Essentially, media data received at the end-user client side, possibly filled with data loss and transmission delay after network transmission, are compared against the source media, yielding an objective measure of error. This objective measure can then be matched with the subjective judgment by human subjects to establish a correlation relation. Once it is done, a reliable prediction of QoE can be made in an automated manner, no longer with the assistance from human subjects.

In this paper, our major goal is to apply these leading edge algorithms on mobile networks to implement an automatic mobile QoE evaluation system. Among the proposals in [5], as a proof of concept, we implemented the Yonsei University's approach [6]. We decided to implement the measurement system with a client-server model, in which smartphone serves as the client for data collection and backend server for data processing. The system was designed so to exploit mobility of smartphones on one hand, yet at the same time avoid processing power bottleneck in mobile devices by lifting CPU-intensive video processing tasks to the backend.

The current paper contributes to solve the problem of QoE calculation by implementing a distributed QoE data

This research was supported by a grant from the Hong Kong Innovation and Technology Fund under the project# ITS/045/09.

978-1-4673-1881-5/12/$31.00 ©2012 IEEE

collection-processing system. The whole path of operations is automatic. With high accuracy QoE prediction by Yonsei's algorithm, our automatic approach can totally replace the laborious human effort originally needed. Here is the organization of this paper: The software system architecture will be introduced in Section II. The current implementation on hardware, as a proof of concept, will be shown in Section III. Client- and server-side (backend and web-UI) components and algorithms will also be detailed. Section IV will describe preliminary observations from collected data. Finally, Section V will conclude the whole paper.

II. System Architecture

Two major components, *client* and *server*, make up our system. Smartphones play the role of client that collects QoS data as well as raw data for QoE calculation. Data are stored temporarily in internal storage until being uploaded to the backend through encrypted channels according to a pre-defined schedule. The backend server is responsible for processing the collected data. Aggregation is done to combine data collected from multiple sensors (e.g. GPS, RSSI, cell-ID). Besides that, a QoE calculation algorithm is invoked to convert the raw packet capture data into standardized QoE units (e.g. difference mean opinion score (DMOS), edge PSNR (EPSNR)). Summary of the processed data is sent back to the client for display. There are also various upload / download connections between the mobile phone and the backend for downloading throughput testing files, RTSP video stream, etc., during QoS and QoE measurement, as shown in Fig. 1.

A. Client component

QoEApp (the client component) consists of three layers: User interface, measurement platform, and data collection modules, as shown in Fig. 2.

As primarily a data collection program, the core of QoEApp is a *measurement platform*, which is responsible for: (1) initiating QoS / QoE collection sessions and serialize the collected data; (2) collecting auxiliary data like system loading, GPS location, radio parameters, surrounding WIFI MAC's; (3) feeding data to user interface for status display; (4) communicating with backend to receive application configuration file; and (5) uploading data to the backend,

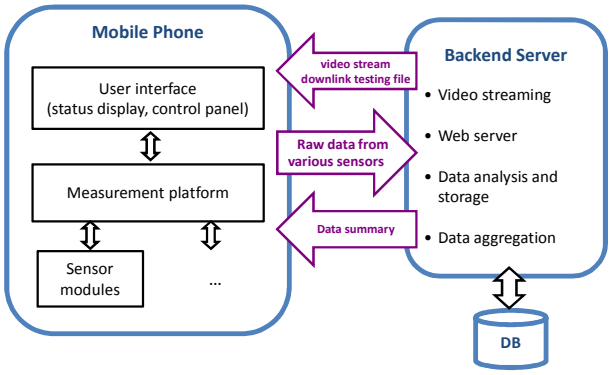

Figure 1. Overall architecture of distributed QoE measurement system.

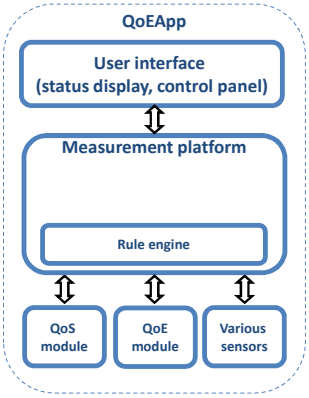

Figure 2. Components of the client application, QoEApp.

through encrypted channels to secure personal privacy over wireless transmission.

The measurement platform invokes several modules for data collection. The QoS and QoE modules mainly collect data through network socket operations. In addition to that, several auxiliary sensors are also present to collect data relevant to follow-up data analyses. Each of these sensors is encapsulated to expose a similar calling interface to the measurement platform. With this design, there is a flexibility to append further sensors to the measurement platform. The sampling frequency of each sensor can be fine-tuned on a user-by-user basis to accommodate different data collection schedules. A *rule engine* is involved to schedule the data collection processes at either a predefined schedule (*proactive* measurement) or in response to 'interesting events' (*reactive* measurement).

The *QoS module* measures *round trip time* (RTT) and *network throughput*.

The *QoE module* is responsible for collecting video frames which are used for frame-by-frame QoE computation algorithms at the backend.

Video streaming protocols can be built on top of either connection-oriented or connectionless protocols. We focus on the latter one here. There is no in-built mechanism to verify if video packets sent out by the streaming server have reached the client side, and in a correct order. Packet drops and out-of-order arrival often occur as a result. RTP protocol used in RTSP video streaming is one such example.

The data processing flow is shown in Fig. 3, divided into two parts. The video packet reception, which is relatively lightweight, is done on the QoE module in QoEApp. A *thin video client* stores temporarily individual video packets as well as the corresponding arrival timestamps, and uploads them to the backend for processing. To save network bandwidth, the packet arrival summary is compressed before uploading. On the backend, a *thin video server*, implementing the same streaming protocol as the original video streaming server, streams the decompressed video packets to a *generic video client*. With this thin video server, each video packet is streamed according to the recorded inter-packet delay. The generic video client can be any one of the widely used clients. However, the pre-requisite is that, the video client must be

able to dump individual video frames. Such dumped video frames forms the input to QoE computation, to be discussed in a later section.

In addition to QoS and QoE modules for network condition data, several *hardware sensors* are also present for collecting relevant data: *GPS sensor* detects the current phone location; *Radio sensor* collects information like the current serving cell ID and RSSI; *System sensor* monitors the current smartphone system statistics like battery and CPU load; and *WIFI sensor* scans surrounding WIFI devices, recording their MAC addresses, which will later on assist determining smartphone location.

B. *Server component*

Fig. 4 shows breakdown of the server component. The QoE computation module is responsible for converting the raw video packet traces into meaningful QoE metrics. The whole procedure consists of (1) *frame reconstruction* and (2) *QoE score calculation*.

To avoid system overloading, only the arrival patterns of video packets are saved on smartphone handsets, leaving the video decoding to the backend. Step 1 completes the whole video playback loop with the unhandled video decoding process. Theoretically, this frame reconstruction process yields a video frame sequence that is identical to the case as if the video player was run on the smartphone, given the same packet arrival pattern.

Step 2 is the most complicated part, since it involves implementation of a particular QoE prediction algorithm. Because QoE computation algorithms work on a frame-by-frame basis, the image sequence generated from Step 1 has to be compared with another image sequence generated from a perfect playback for frame alignment. After the frame alignment process, the video frames are ready for QoE computation, with exact calculation steps depending on the actual system implementation.

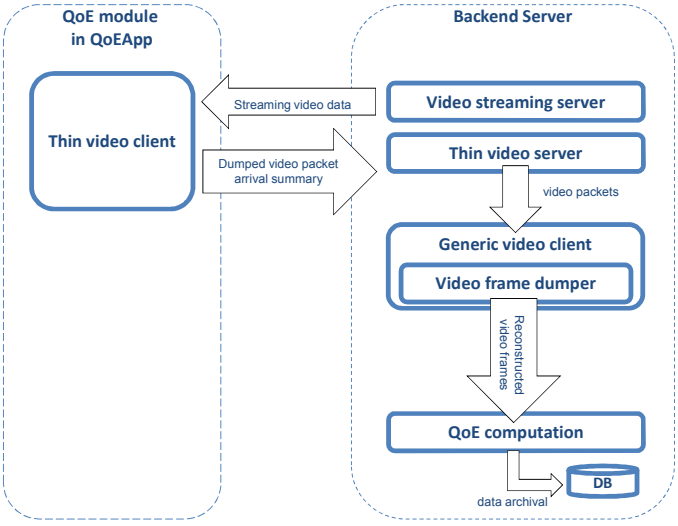

Figure 3. Video playback loop for QoE measurement.

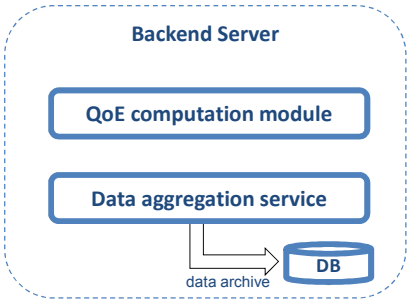

Figure 4. Backend server component.

Besides, the backend aggregates data streams from various sensors with timestamp as the key. The aggregation results in a multi-dimensional time-series (e.g. GPS, radio information, QoS information, etc.), facilitating study of inter-correlation between individual measures.

III. System Implementation

This section describes our current implementation of the architecture of the distributed QoE measurement system detailed in the previous section. Details about the particular software / hardware used will be shown. Besides the smartphone client and backend server components, a web interface has been implemented to enable easy monitoring of the overall network condition by network engineers. Graphical interfaces within provide quick access to relevant information.

A. *Smartphone client*

Our client data collection software, QoEApp, is currently implemented to run on Android smartphones (version 2.1 or above) as a background service, coded entirely in Java.

The measurement platform (1) *collects* data from the various sensors, (2) *uploads* them to the backend, and (3) *provides* data summary periodically to the user interface. All the data collected are uploaded to the backend through encrypted HTTPS connections.

The array of sensors (e.g. QoS sensor, QoE sensor, GPS sensor, radio sensor) are encapsulated as Java classes, feeding data to the measurement platform. All the internal complications, for instance, interfacing with hardware and invoking socket calls, are hidden from the measurement platform. This design offers the flexibility for future feature expansion through implementation of new software plugins. Each sensor works independently in its own thread.

The QoS module currently measures RTT and downlink / uplink throughput. A terminal 'ping' command is used to send the packets and retrieve the RTT statistics from the terminal output. This seemingly indirect way of obtaining RTT is required since a non-rooted Android system does not expose APIs necessary in constructing and sending network-layer ICMP packets. To measure downlink / uplink throughput, an HTTP GET / POST request is sent to the backend server to retrieve / send a specified large-size file. The total file size is then divided by the total time required for such file download / upload to obtain the network throughput (in KB/s).

For performance consideration, QoE measurement is implemented with a distributed design as aforementioned.

Video packets are collected at the handset, leaving the eventual CPU-intensive QoE computation to the backend. We currently have implemented the solution for RTSP streaming video. A further improvement has been incorporated in the QoE sensor: Instead of capturing whole RTP packets, the QoE module only records the packet index and the arrival time to minimize the performance impact (due to file dumping) on other mobile phone applications during measurement. The collected packet indices, together with the timestamps, are uploaded to the backend for frame reconstruction and QoE computation.

B. *Backend server*

The backend server currently runs on a Linux system (Fedora Core 12) on an Intel Core 2 Quad CPU. In addition to the two basic services discussed in Section IIB, there are two additional components, *streaming server* and *web server*.

Since our current implementation works on a connectionless protocol (RTP protocol on UDP), architecture like that in Fig. 3 is used to generate video frame sequences. After then, a frame realignment process prepares the video frames for QoE computation. The QoE computation module makes use of Yonsei's algorithm, which involves EPSNR calculation, to compute QoE score (i.e. DMOS). To boost performance, all the codes for the computation are in C.

The final stage of QoE computation involves the mapping from EPSNR obtained to DMOS, which is an impressionistic judgment of the difference between reference and distorted videos. We use logistic-regression to map EPSNR to DMOS, with regression coefficients previously derived through the public UT LIVE Wireless Video Quality Assessment Database.

The data aggregator collects data from multiple sensors and aligns them with the key field *timestamp*, before injecting them into the database, where MySQL is used currently. The aggregation routines are coded in Java, running periodically to identify and process any candidate data file newly uploaded to the backend.

Darwin Streaming Server (DSS/5.5.5, build 489.16) is responsible for streaming videos to the mobile clients through RTSP protocol. As no data exchange between it and other components is needed, the server has not been modified in any way, and works in a standalone manner to serve incoming video requests.

The backend relies on Tomcat 6.0.26 to provide web service. Java based technology (e.g. Java servlet and JavaServer Pages) are enabled on the web server to provide dynamic web content pages for monitoring QoS and QoE statistics.

C. *Web interface*

To provide a tool for monitoring the collected QoE data, a web interface has been set up. A sample showing the DMOS data collected on 27th August, 2011 is shown in Fig. 5. The access to this interface is restricted currently by a username-password authentication scheme.

The major function of the web interface is for locating problematic network cells. Any one of the QoS (RTT, packet loss, uplink / downlink throughput, RSSI) and QoE (DMOS, EPSNR) data types can be selected for a map plot. Charting function is also provided for users to get a statistical illustration of the collected data. Time-series, histogram and cumulative-frequency-distribution (CDF) plots are available.

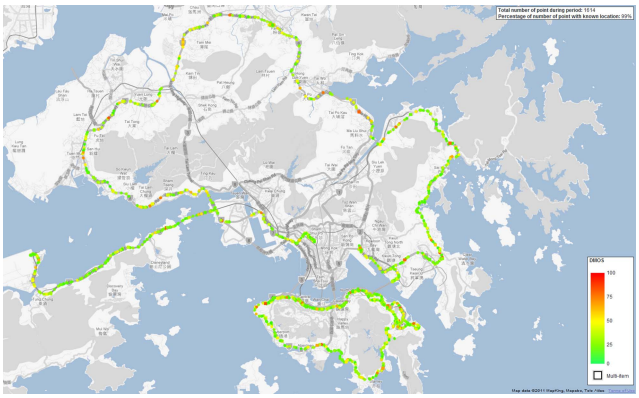

Figure 5. Web interface for browsing collected data.

IV. Data Collection and Analysis

We continuously collected data to verify the scalability of our QoE measurement system. Up to the moment, 47,205 QoE data, 84,406 QoS data, 190,766 packet loss, and 7,964,241 radio data have been collected. Items differ in terms of sample pool size due to the different sampling frequencies used.

In the following paragraphs, we make use of some subsets of such database to describe various initial operational insights we obtained during data analyses.

A. *Independence of QoE of QoS*

Fig. 6 highlights the significant contribution of our system to measure QoE. The top two streamgraphs are from two streaming sessions of the same source video, both with the same zero packet loss. The DMOS is much better (the lower the more satisfactory) on the left (9.64) than the right (42.89). Packet arrival jittering on the right seemingly lowers the QoE. As a further example, the two bottom streamgraphs have both similar video packet arrival jittering and packet loss rate. However, the left one (DMOS = 35.6) scores much better than

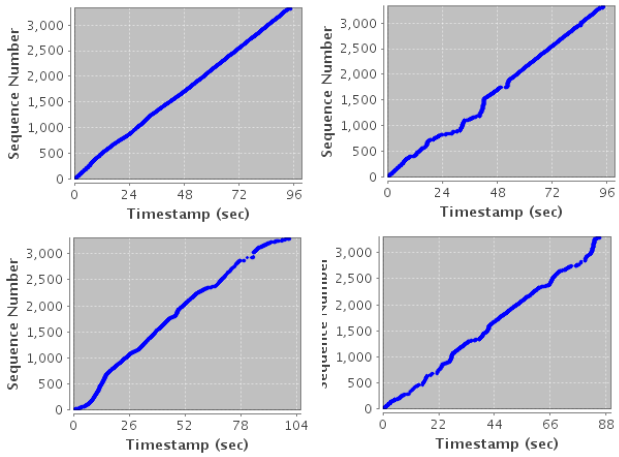

Figure 6. Failure of objective QoS as a subjective QoE-reflecting VQM.

the right (DMOS = 76.68). These two examples shows that QoE is a combination of many factors, and that QoS measures alone cannot easily account fully for the subjective impression as experienced by human viewers.

B. *Effect of network reception area*

While comparing QoE performance metrics obtained, we found that data points obtained in crowded areas generally performed better than those obtained while traveling on highways. Fig. 7 is an illustration of this contrast. There are altogether 294 samples for highways and 270 for crowded areas. The histogram of normalized number of records shows better performance while the QoEApp operates in crowded areas. This is probably due to the fact that video sessions on highways usually involve more cell handovers, resulting in a higher probability of network transmission loss.

C. *Effect of testing video length*

To indirectly adjust density of QoE data points, we used testing videos of different lengths. A shorter video takes a shorter period for complete streaming, and thus shorter lapse before the next sampling. Fig. 8 shows the performance comparison between QoE measures resulting from testing videos of different lengths. An immediate observation is that a shorter video length leads to a better QoE measure. This difference can be attributed by a similar reasoning as above, that downloading a shorter testing video likely experiences a smaller number of network cell handovers.

D. *Handset differences*

While collecting data from various handsets, quite considerable performance variability was noticed across brands and even handsets by the same manufacturer. This is expected due to the heterogeneities existing in the handset price range and production technologies. GPS data, for example, were of vastly different accuracy across brands. Besides, some handsets offer RSSI reading at much coarser intervals than others. Fig. 9 shows one such example contrasting Nexus S and Nexus One. The data were collected on 2nd Aug., 2011 afternoon, and totaled 280 for Nexus S and 284 for Nexus One. From the normalized QoE record counts, we may find that Nexus S yielded a better (smaller) overall DMOS, probably due to its more powerful CPU for handling incoming video packets on time.

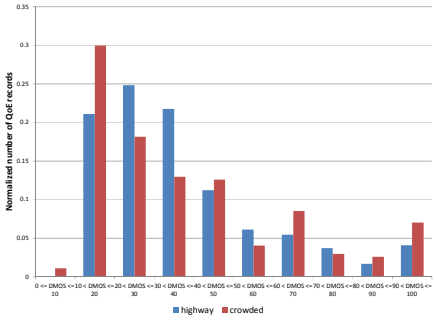

Figure 7. Effect of network reception area.

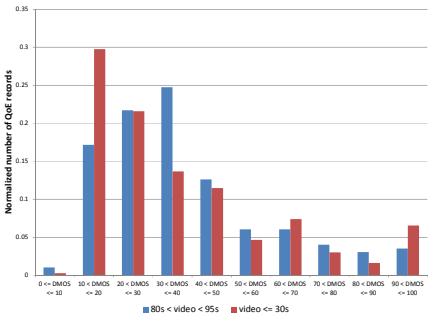

Figure 8. Effect of testing video length.

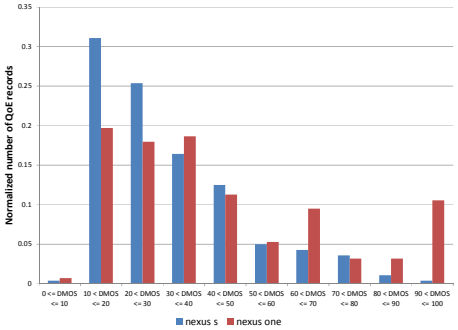

Figure 9. Effect of handset model on DMOS.

V. CONCLUSION

To rapidly estimate the perceptual quality as experienced by mobile network users while watching streaming videos, we developed a distributed QoE measurement system. It consists of three major components: (1) a client program running on Android platform to collect various network statistics; (2) a backend server for CPU-intensive QoE computation tasks based on the Yonsei algorithm; and (3) a versatile web interface to give easy access to the collected information.

The current system provides an automated QoE evaluation solution for RTSP streaming video. To further modify our system to accommodate connection-oriented video streaming standards, like HTTP progressive video, some refinement is required.

REFERENCES

[1] ITU.T J.144 - Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference.

[2] ITU.T J.246 - perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference.

[3] ITU.T J.247 - Objective perceptual multimedia video quality measurement in the presence of a full reference.

[4] ITU.T P.910 - Subjective video quality assessment methods for multimedia applications.

[5] Final report from the video quality experts group on the validation of objective models of video quality assessment, Phase II ©2003 VQEG.

[6] Chulhee Lee, Sungdeuk Cho, Jihwan Choe, Taeuk Jeong, Wonseok Ahn and Eunjae Lee, Objective video quality assessment, Optical Engineering 45, 017004 (Jan 24, 2006).