User-Centric Quality of Experience Measurement*

Bachir Chihani[1,4], Khalil ur Rehman Laghari[3], Emmanuel Bertin[1,4], Denis Collange[2], Noël Crespi[4], and Tiago H. Falk[3]

[1] Orange Labs, 42 rue des Coutures, 14066 Caen, France
[2] Orange Labs Sophia Antipolis, 905 rue Albert Einstein, 06560 Valbonne
firstname.lastname@orange.com
[3] Institut National de la Recherche Scientifique (EMT-INRS), Montreal, QC, Canada
[4] Institut Mines-Telecom, Telecom SudParis, CNRS 5157
9 rue Charles Fourier, 91011 Evry, France
firstname.lastname@mines-telecom.fr

Abstract. Quality-of-experience (QoE) produces the blue print of human perception, feelings, needs and intentions, while Quality-of-Service (QoS) is a technology centric metric used to assess the performance of a multimedia services and/or network. .It is quite important for service/content providers to understand user/customer experience requirements in order to improve the service quality or the content recommendation. With advent of 3G and 4G wireless networks, and efficient smart phones, the band-width hungry multimedia applications are becoming common in use on end-user devices. Thus,it is also important for telecom operators to understand the impact of wireless network performances on the user experience in mobile environment. On the fly evaluation of user experience for multimedia services is a challenging problem especially in mobile environments. It implies the collection and the correlation of a mixture of variables on network conditions, on the service, as well as on the user itself. This paper proposes an innovative mobile application that can be used for measuring user quality-of-experience on the fly with a high accuracy and the consideration of multiple parameters about the user, the network and the system. This application takes advantages of current advances in mobile technologies to measure user experience directly on the user device. In addition, it aims to preserve the user privacy by transmitting only estimated quality-of-experience to the service provider.

Keywords: QoE, QoS, context, mobile computing, 3G UMTS, video streaming, machine learning.

1 Introduction

The wide spread deployment of Wi-Fi, 3G and 4G cellular networks has increased the use of smart phones, which has changed the landscape of information and

* A short abstract of this article has been accepted as a work-in-progress report in the IEEE Pervasive Computing magazine, Oct-Dec issue, 2012. [1]

G. Memmi and U. Blanke (Eds.): MobiCASE 2013, LNICST 130, pp. 33–46, 2014.

communications technology. Due to advanced operating capabilities of smart phones, multimedia applications are now being developed massively and made available through Google or Apple stores. These services have stringent Quality-of-Service (QoS) requirements. However, in a mobile environment, the user context (e.g. location) and network QoS change continuously, in turn continually influencing the user's behavior and experience. Thus, it is critical to identify requirements for mobile multimedia applications that are not only related to the wireless network QoS but also to the user context and feedback. These requirements can be derived from user Quality-of-Experience (QoE) demands that can be understood by mapping the user's subjective ratings to the objective QoS and contextual parameters.

We propose in this paper an innovative user-centric, context-aware solution that can be used for measuring QoE on smartphones. The objective is to design an intelligent and user-centric QoE measurement framework for Android-based smartphones. Such framework can be used to analyze and evaluate user experience requirements for multimedia services and applications in a mobile environment. In this paper, we propose a framework which is implemented with a standalone intelligent QoE application installed on smartphone. End-user usesa multimedia service, and s/he gives a QoE score using our framework. These subjective scores are correlated with QoS and context parameters. The resulting dataset is then analyzed locally by our proposed framework in order to generate a personalized QoE model to assess the user perception regarding the studied service. The generated QoE model is updated over time with respect to changes in the QoS or contextual parameters, i.e. network or application performance criteria. This application not only captures QoS, contextual parameters and the user ratings but also analyzes and generates the personalized QoE results for a given user session. Furthermore, QoE is never a fixed value; it keeps updated over the time with respect to changes in QoS or in contextual parameters.

The novelties of our solution are first, the collection of QoS, contextual and user ratings locally on user smartphones; and second, the client-side analysis of the collected data to generate a personalized QoE model locally on smartphones. The data are analyzed as soon as the user finishes interacting with the studied service or after a consequent change in the user perception.

2 Challenges and Motivations

From the telecom perspective [2], the network's performance can be monitored by collecting and investigating key performance indicators such as QoS parameters. These technical indicators are measured at the different levels in a wireless network. The examples of theseindicators collected at network layer are bandwidth, delay, jitter, packet loss rate, etc., and at end-user device level(e.g. noise/interference level, signal strength, connection establishment time, drop rate, etc.).

On the other hand, from the user perspective [2], the network's performance can be monitored by collecting user feedback, i.e. QoE data. In contrast to QoS, QoE provides an assessment of human perceptions, feelings, emotions and intentions with respect to a particular product, service or application [18]. QoE is affected by various technological, business and contextual factors [19] [3].

It is extremely difficult for telecom operators to measure QoE as it depends on various factors [4]: objective ones related to network condition and subjective ones related to user perception. For example, the QoE for a video streaming service depends on network conditions (e.g. bit rate, packet loss rate) and viewing conditions (e.g. type of used device, at home or work, etc.).

Moreover, it is quite challenging to establish an accurate QoS to QoE mapping method for different applications as it is hard to choose the relevant QoS parameters for a given application [5]. It is also challenging to evaluate feedback with respect to QoS and context data as acquiring these different parameters is difficult in a mobile environment. Another challenge [6] was due to the limited computing capabilities of user terminals which make QoE processing on these devices hardly possible. This challenge was valid for traditional featured phones as they were limited in terms of processing powers and not designed for calculation task. This is no longer valid as current smartphones have improved processing capabilities and they are equipped with flexible operating system allowing the development of advanced applications. For example, the Google Nexus runs the Android 4.1 operating system; it has 1 Gb RAM memory and 1.2 GHz CPU processor.

These improvements in mobile device capabilities as well as the fact that the mobile devices are the closest elements to end-user motivate our work for a full client-based QoE measurement framework. This proposed framework aims to collect and process both QoS and QoE data locally on the user device and create a personalized QoE model. Compared to QoS-based approaches, this approach is closer to user and provides better insights about user experience.

3 User-Centric QoE Measurement

Existing QoE frameworks tend to upload the data needed for generating QoE model from multiple users to a central server to process and aggregate them. Our objective is to avoid unnecessary Internet traffic generated by uploading data to a distant server by performing a local management of QoE parameters. This enables the generation of a personalized QoE model and better user privacy by storing and processing user information locally on his device. We propose a user-centric way for measuring QoE parameters, directly on the user device.

3.1 Framework Architecture

Our architecture is composed of an Android application running on the user Smartphone for measuring user QoE; multimedia server (e.g. YouTube) from which the videos will be streamed over a 3G/WiFi connection via Real Time Streaming Protocol (RSTP).

Figure 1 presents main components of the Android application responsible for QoE measurement, interaction with the end-user and with the remote multimedia service provider (MSP).

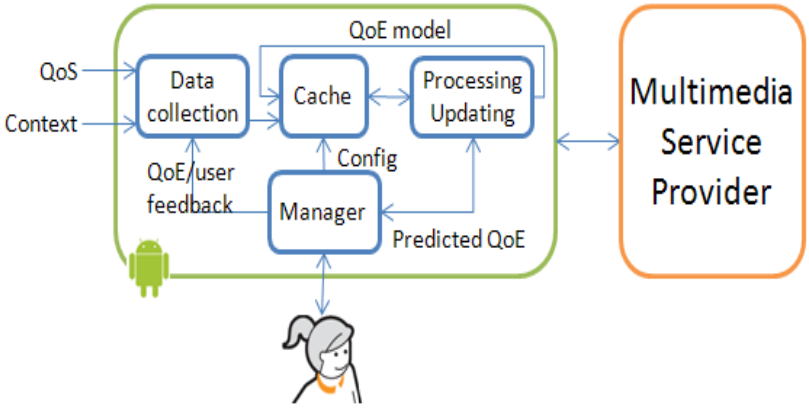


Fig. 1. Architecture of the Android application for QoE measurement

The manager component (MC) is the main component responsible for interacting with the outside world (user and service provider) and managing rest of the system components. The data collection component (DCC) is responsible for acquiring QoS (e.g. jitter, packet loss) and user context (e.g. GPS data) related information. The Cache Component (CC) is responsible of caching temporarily, a set of collected data (QoS, context and QoE) and the generated QoE model. The processing/updating component (PUC) works in two modes: learning and automation modes. In the learning mode, this component uses a supervised learning algorithm (for instance a linear regression) to generate a personalized QoE model and stores it into the cache component. The generated model is updated continuously with the cached data and each time the cached data is consumed, the cache is emptied.

In the automation mode, the component is responsible for predicting QoE parameters (e.g. did the user like the video content?), with the use of the cached QoE model. Thanks to this mode, a multimedia and telecom service provider can use our framework as an integrated component to its multimedia service to evaluate the the user experience regarding the usage of the service. In this case, the predicted QoE values can be for instance sent to the multimedia and telecom service provider in order to personalize the recommended videos.

3.2 Collected Parameters

Smartphones are a rich source of information about user and his/her environment. Table 1 summarizes the data we are collecting on the client-side for generating user QoE model. These collected data belong to the following categories.

User related information: input from user describing his satisfaction through various ratings after viewing a video;

Table 1. Collected parameters for QoE model generation

Parameter	Unit	Value	Sampling
User related information			
Satisfaction	state	[yes, no]	On thumbs up/down
Video Quality	integer	[1, 5]	When user stop watching
Video Content	integer	[1, 5]	When user stop watching
Application related information			
Watched	%	[0, 100]	When user stop watching
Error	%	[0, 100]	On error
Device related information			
CPU	%	[0, 100]	Each second
Memory	%	[0, 100]	Each second
Battery level	%	[0, 100]	Each second
Latitude	double	[0, 180]	On location changes
Longitude	double	[0, 90]	On location changes
Network related information			
Jitter	second	[0, ∞[	On RTSP packets arrival
Loss rate	%	[0, 100]	On RTSP packets reordering
Network Type	state	[WiFi,3G,LTE]	On changes
RSSI	dBm	]-∞, +∞[	On changes

Application related information: Video parameters like time spent watching the video (i.e. if or not the whole video was watched) or the moment when an error had happened (e.g. related to a bug in the application) while the user was watching the video;

Device related information: battery related information like level, its health (e.g. good), its status (e.g. charging); CPU usage (e.g. percentage consumed by our application); memory usage (e.g. amount of memory needed by our application); Location information like the name of the location provider, altitude, longitude, etc.

Network Performance related information like signal strength, QoS parameters like delay and jitter, received packets; network type (e.g. UMTS, LTE, GPRS);

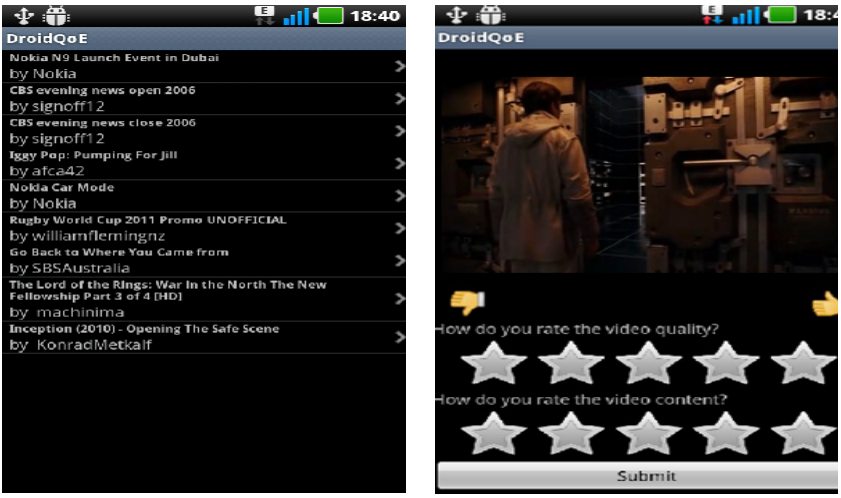
In our implementation, there is no fix sampling rate as the Android platform allows applications to subscribe for specific events (e.g. network type/location changes) to be notified on their occurrence. This way, there is no need for a continuous polling of the event source (e.g. GPS sensor, network manager).

3.3 Implementation Details

The different components in Figure 2 are implemented as Android threads (i.e. AsyncTask) except the cache which is implemented as an Android ContentProvider able to store data locally into the Android SQLite database. The application has two Android activities: the first one displays a list of videos; the second one displays the chosen video. We used the YouTube API (Application Programming Interface) to

stream videos from the multimedia service provider. Some Android APIs are used to get contextual information (e.g. location) and QoS parameters (e.g. jitter).

When the application is started, a list of videos is displayed from which the user can choose one video to watch. Two ways are provided to user to report his/her satisfaction (which is represented by the reported QoE score): While s/he is watching the video thanks to thumbs up (QoE score = 4) or thumbsdown (QoE score = 1) buttons, or at the end of the video by answering the questions (QoE score ranges from 0 to 5). The reported QoE will be stored in the cache to be processed later when there will be enough available data; i.e when the cache becomes full.



(a) (b)

Fig. 2. Screenshots from the QoE measurement application

Figure 2 depicts a screenshot from an implementation of the QoE measurement frame work. The GUI (Graphical User Interface) showed in (a) displays a list of videos. The one in (b) is composed of a top area where the video is displayed. In the middle, the user can use the thumbs up/down buttons to express his current liking and disliking of the displayed video. At the bottom, there is a button for submitting this user survey.

3.4 Components Interaction

Figure 3, illustrates the framework sequence diagram. When the user reports its QoE, the Manager sends this value to the data collection component. In addition to the QoE value, it collects the current QoS and user context information, and stores them into the cache. When the stored examples in the cache reach a certain value (configurable parameter), the processing-updating component is notified to consume them and to generate an updated version of user QoE model.

When the multimedia service provider requests a QoE value for the currently streamed video, the manager component sends back the user reported QoE (if there is) or a predicted value generated by the processing-updating component.

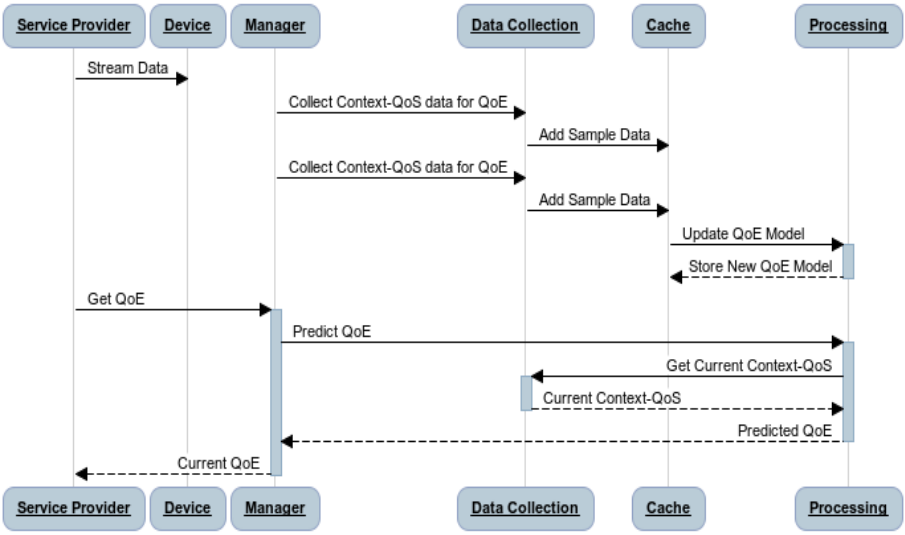


Fig. 3. Sequence diagram for QoE management

4 Learning and Processing

All the data is not available at a single time but it is gathered continuously and progressively over the time. Thus, the iterative nature of linear regression may help in building accurate model which fits our needs. Our learning algorithm, implemented by the processing/updating component, is based on the multivariate linear regression [7] where input parameters are QoS and contextual information and QoE is the output or target variable. For each learning phase, the size of the training set or number of samples is '*m*' which is also the size of the cache. The hypothesis (*h*) represents the model to be learned for predicting future values of QoE ($y_{predicted}$) for a giving sample vector (*X*), i.e. $y_{predicted} = h(X)$. Mathematically, *h* is defined in equation (1), where x_i is an input parameter, *n* is the number of input parameters, and θ_j the weight of the corresponding input parameter. It is the set of weights that represent the parameters to be learned.

$$h(X) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \quad (1)$$

The learning algorithm tries to predict the best values of the hypothesis parameters (vector of θ values) minimizing the difference between the output QoE value and real value ($y_{predicted} - y_{real}$). Equation (2) defines mathematically the cost function '*J*' which is based on a model (vector of θ values) to output the cost of this model by the

summation of distances between predicted values $h_\theta(x)$ and real values y for all samples (rows) of the dataset.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 \quad (2)$$

To predict best values of θ parameters, we use a modified version of Batch Gradient Descent (BGD) [8]. BGD is an iterative optimization algorithm that requires the whole data set to be available and then it does line search to find the best step size, which makes it a slow algorithm. Instead, our modified version (which is also an iterative optimization algorithm) operates on the data stored into the cache when '*m*' (cache size) samples become available, i.e. when the cache becomes fullThe motivations behind M-BGD is that on a mobile environment the samples are streamed (i.e. continuously collected) and thus traditional BGD cannot be applied in a single learning phase as we cannot have a full dataset. In this case, the learning should instead be continuously performed.

M-BGD (Modified Batch Gradient Descent) first normalizes input parameters (P) as shown in equation (3).. This normalization aims to project data into the [-1, 1] interval in order to avoid parameters scaling problem that may influence the resulting model.

$$Normalized\ P = \frac{P - mean(P)}{\max(P) - \min(P)} \quad (3)$$

Second, M-BGD updates θ values continuously until convergence or stagnation at a local minimum given the following algorithm:

```
Initialize θ parameters (e.g., to 0);
Repeat until convergence:
```

$$\theta_j := \theta_j - \alpha * \frac{d}{d\theta_j}J(\theta)\ \text{j=1, n}$$

By replacing J derivative with its value, the last loop becomes:

```
Repeat {
```

$$\theta_j := \theta_j - \alpha * \frac{1}{m} * \sum_{i=1}^{m}(h(X^i) - Y^i) * X_j^i$$

```
}
```

Where X^i is a vector representing the i^{th} sample/input features, Y^i is the QoE value corresponding to the i^{th} row of the training set, and θ_j represents the learned parameters corresponding to the j^{th} feature/column. The latter are initialized the first time to zero. Then, after each training phase, θ_j are stored to be reused the next phase as initialization values. The 'α' regulate the convergence speed of θ_j values.

The cost function '*J*' is a convex function; it has then a unique minimum which is the global minimum at which θ values are best values that gives the minimal distance between predicted and real output values. Convergence of θ_j to best values is guaranteed. But gradient descent is an iterative algorithm and it is known to be too

slow as the all dataset is used many times during each iteration. The 'α' parameter needs to be well chosen to speed up the algorithm convergence.

5 Evaluation

Our first goal is to understand the impact of the modification brought to the original Batch Gradient Descent (BGD) algorithm with respect to the optimization of an objective function. In our case, this optimization aims to calculate the best weights that correspond to the QoS and context variables used in the objective function to measure the QoE score. We implemented the original Batch Gradient Descent (BGD) algorithm and our variant Modified BGD (M-BGD) algorithm to compare their performance in term of evolution of the output cost function (equation 2) after each algorithm step. Figure 4 depicts the graphs related to cost function calculated for each algorithm. To generate these graphs, we used some data collected from a QoE study of a multimedia service (video streaming) that involved 24 subjects (6 women and 18 men) aged between 20 to 35 years. The data is composed of output parameters (QoE values given by users) and input parameters including the video category ('0' for fast videos like football match, and '1' for slow videos like a ship moving in the large sea), and QoS parameters (packet loss, packet reorder, video bit rate).

In case of BGD, the cost function is calculated for the whole dataset each time and this is why its graph is smooth (it can be represented with a linear function) and the cost value is decreasing in a steady way. At the other hand, the cost function of M-BDG is calculated only for the available data in the Cache component which makes the cost value oscillate continuously as the model may fit current data while not perfectly fit the next set. The BGD need more data to output a low cost value, while M-BGD is able to output an acceptable cost (less than 1).

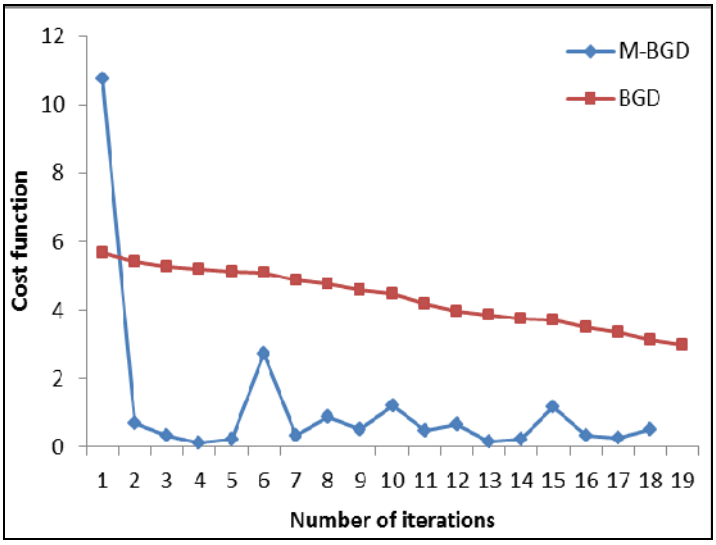


Fig. 4. Cost function graphs of two methods

A second goal is to understand the relation between QoS parameters and QoE scores. For this, we conducted a set of experiments with our framework to collect QoE scores under varied QoS conditions (network related). After aggregating the resulting data, the Figure 5 shows the relation between users QoE and network QoS. It is clear that the obtained QoE scores are inversely related to disturbance of QoS parameters as stated in [9].

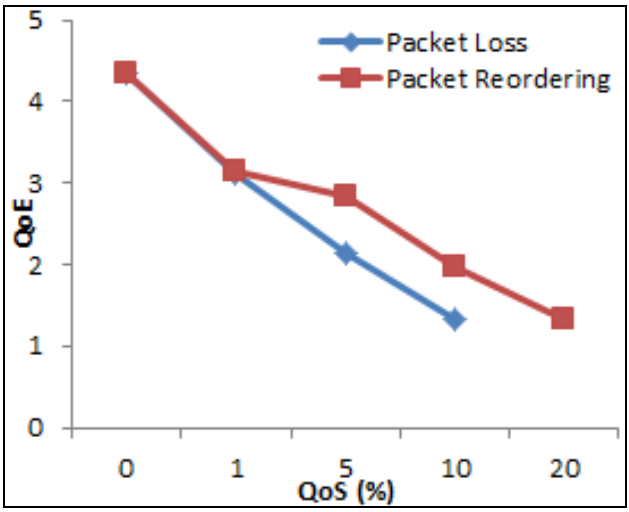


Fig. 5. Relationship between QoE score (y-axis) and QoS values (x-axis)

6 Related Works

Commonly, QoE is evaluated in Living Labs [9] which is a user-centric ecosystem that involves users in testing/assessing new services (e.g. multimedia, games). Another possibility for measuring QoE is to hire a representative panel of real users of the service (e.g. telephony). In both cases, the evaluation is based on questionnaires where users have to answer after a service usage session. After collecting multiple answers from the participant users, the Mean Opinion Score (MOS) [10] method is used to evaluate the overall QoE of the service. These methods are experimental and passive in way they need to: hire a group of users, put them in a controlled environment, experiment the service under study in different conditions, collect data from users and correlate them with experiments setups to finally generate an aggregated QoE model. An example of such approach is presented in [11] where the authors propose a QoE framework for smart phones and use subjective assessment technique for the measurement of QoE. Their framework is based on a client-server model. Once, user data are collected; the server side takes the control of all user data and analyzes it. The purpose of the client-side application is limited to video

streaming and reporting user feedback data to the server side. It is not intelligent enough to make any analysis over data and/or produce personalized QoE results for smart phone users.

Objective QoE assessment methods represent another class of approaches which are more active [12] as they attempts to measure QoE by mapping it to some QoS parameters without end-user involvement. An example of such approaches is presented in [6] where the authors proposed a QoE measurements method for smart phones. The method is based on the collection and the processing of QoS data on the user terminal and reporting QoE based on objective (QoS) assessment. Hence they do not require any user feedback. However our work is based on a subjective assessment scheme and it provides more reliable and accurate user QoE. In fact, the generated QoE model is personal as it relies on user input as well as system and network information acquired directly from the user device. These methods rely heavily on QoS indicators to try to approximate the evaluation of the user perception ignoring user contextual information like location. Also, if a QoS to QoE mapping is accurate for a given class of applications, it may become obsolete for another class as different applications have different QoE/QoS requirements. For example, some application may be sensitive to jitter and delay like online video games while others are more sensitive to packet loss like file transferring.; some applications may need a quiet environment to be used (e.g. telephony), while others may need a suitable lighting arrangement (e.g., texting).

Table 2 summarizes the description of these two main QoE measurement approaches, and illustrates as well a comparison between them.

Table 2. QoE measurement approches comparaision

	Data Collection	**Data Transmission**	**Comparison**
Objective Methods	QoS	Huge data transmission	Generalized QoE (QoS- specific), Saves time. No User feedback, lacks accuracy
Subjective Methods	Surveyed QoS / QoE	No need for data transmission	Personalized QoE (User-Specific), Time consuming Reliable and Accurate QoE Based on user feedback

Most of the existing QoE measurement tools aims to analyze the user web browsing activities, especially video downloading as it represents a major part of the Internet traffic [13]. Some of these tools usually implement a polling interface to ask more or less interactively the users about their satisfaction. For instance, HostView [14] is an end-host tracing tool that implements a combination of objective and subjective QoE measurement methods. It collects network traffic, system performance information, and prompts also the user for feedback on network performance. Another tool combining both QoE assessment approaches is presented in [15]. This tool does not require any installation on the user side; it uses a heuristic approach to collect user feedbacks in an explicit way. It is able to infer the user impatience from collecting and analyzing the last flags of the TCP connections generated by the user activity as

well as the end-to-end network performance. QOM frame work [20] combines both subjective and objective factors, but the most of the QoE processing and management is done at server side.

Typical examples of objective QoE measurement tools include: Netalyzr [16] and a modified version of FasterFox [17]. Netalyzr [16] is client-server application that allows the user to download an applet through which active tests are conducted and collected data are uploaded to some of the predefined Netalyzr servers. In [2] the authors attempted another deployment architecture based on plugins (e.g. browser plugin.); they modified FasterFox [17] which is a Firefox plugin originally developed to speed-up network performances. They used this plugin to collect data from the user browser and to report it to a remote server.

The existing tools relying on QoS data imply the transfer of an important quantity of low level data about network metrics. The aggregation made at the back-end side produce a generalized model about user experience which may lack accuracy. The tools combining objective with subjective measurement approaches provide an enhanced accuracy with a more personalized QoE assessment. Nevertheless, most of these tools do not consider information about user situation which may be important for a more precise user experience assessment. The following table summarizes the description of the presented QoE measurement tools and attempt to compare those tools regarding different implementation and operational characteristics.

Table 3. Summary of existing QoE measurment tools

	Architecture	Measurement technique	Personalization	Metrics
Z. Qia et al. [6]	Client side	Objective QoE Assessment	No	Network parameters
I. Ketykó et al. [11]	Client-Server	Subjective assessment	Partial	Network parameters User feedback
HostView [14]	Client-Server	Combined approache	Yes	Network parameters System performance User feedback
D. Collange et al. [15]	Network centric	Objective QoE	No	Network parameters
Netalyzr [16]	Client-Server	Objective QoE	Partial	Network parameters
J. Shaikh [2]	Client-Server	Objective QoE	No	Application parameters
Laghari et al.[20]	Client-Server	Combined approch	Partial	Network parameters Application params User information
Our proposal	Client side	Combined approache	Yes	Network parameters Device parameters Application params User information

Our proposed QoE framework is a simple, intelligent and self-functioning QoE framework which not only monitors contextual, QoS and user ratings but also makes QoE analysis and decisions on its own at the client side. It does not require any third party servers for data analysis and it produces run time QoE Evaluation. However, the used machine learning technique is rather simple which makes the accuracy of the generated QoE model relatively low. More advanced techniques (e.g. neural networks, Bayesian networks) should be used to enhance the accuracy. To our knowledge there are currently no robust and reliable libraries implementing these techniques on mobile Operating Systems. In further studies, we will investigate the possibility of using such advanced machine learning techniques on mobile platforms, like Android, by porting existing libraries in the Android environment.

7 Conclusion

In this paper, we propose a smartphone-based framework that enables the evaluation of the user experience regarding multimedia streaming services. We present the framework architecture and implementation details. The advantages related to our solution are twofold. First, from the service provider perspective, the framework provides a better user perception assessment as the processed technical and user parameters (QoS, context and user rating data) are collected close to user, directly from the his/her device. Second, from the user viewpoint, he/she has freedom to give his feedback about offered quality at any time through thumbs up/thumbs down icon and/or user rating, with respect to a particular service, and in any situation. Third, from the telecom operator perspective, our framework handles "monitor, analyze and decide" functions on user data on smartphone and it does not require any other server side for these functions, hence there is no need for bulk data transfer. Also, it may give a privacy control to user behavioral requirements. In a future work, we plan to investigate the possibility of using more advanced machine learning techniques (on an Android device) like neural networks to generate a QoE model with better accuracy.

References

1. Chihani, B., Bertin, E., Crespi, N.: Android-based QoE Management Framework. Work in Progress report, IEEE Pervasive Computing, Issue (October/December 2012)
2. Shaikh, J., Fiedler, M., Collange, D.: Quality of Experience from user and network perspectives. Annals of Telecommunications 65(1-2), 47–57 (2010)
3. Chihani, B., Bertin, E., Jeanne, F., Crespi, N.: Context-aware systems: a case study. In: International Conference on Digital Information and Communication Technology and its Applications, France (2011)
4. Hubbe, P., Kerboeuf, S., Leprovost, Y., Mahfoufi, Y.: An Innovative Tool for Measuring Video Streaming QoE. TECHzine Technology and Research E-ZINE (2011)
5. Serral-Gracià, R., Cerqueira, E., Curado, M., Yannuzzi, M., Monteiro, E., Masip-Bruin, X.: An overview of quality of experience measurement challenges for video applications in IP networks. In: International Conference on Wired/Wireless Internet Connections, Sweden (2010)

6. Qiao, Z.: Smarter Phone based Live QoE Measurement. In: 15th International Conference on Intelligence in Next Generation Networks (ICIN 2011), Berlin, Germany (2011)
7. Kaw, A., Kalu, E.: Numerical Methods with Applications: Abridged, 2nd edn. (2011) ISBN: 9780578057651
8. Nabney, I.: NetLab: Algorithms for Pattern Recognition. Springer (2002)
9. Rifai, H., Mohammed, S., Mellouk, A.: A brief synthesis of QoS-QoE methodologies. In: 10th International Symposium on Programming and Systems (ISPS), Algiers, Algeria (2011)
10. International Telecommunication Union, "Methods for Subjective Determination of Tranmission Quality," ITU Recommendation, p.800 (August 1996)
11. Ketykó, I., De Moor, K., De Pessemier, T., Verdejo, A.J., Vanhecke, K., Joseph, W., Martens, L., De Marez, L.: QoE measurement of mobile YouTube video streaming. In: Proceedings of the 3rd Workshop on Mobile Video Delivery (MoViD 2010), Firenze, Italy (2010)
12. Calyam, P., Ekicio, E., Lee, C., Haffner, M., Howes, N.: A gap-model based framework for online VVoIP QoE measurement. Journal of Communications and Networks 9(4), 446–456 (2007)
13. Schatz, R., Egger, S.: Vienna Surfing - Assessing Mobile Broadband Quality in the Field. In: Workshop on Measurements Up the STack (W-MUST), collocated with ACM SIGCOMM, Toronto, Canada (August 2011)
14. Joumblatt, D., Teixeira, R., Chandrashekar, J., Taft, N.: HostView: Annotating End-Host Performance Measurements with User Feedback. In: HotMetrics Workshop, collocated with SIGMETRICS, USA (2010)
15. Collange, D., Hajji, M., Shaikh, J., Fiedler, M., Arlos, P.: User impatience and network performance. In: 8th Euro-NF Conference on Next Generation Internet (NGI), Karlskrona, Sweden (June 2012)
16. Kreibich, C., Weaver, N., Nechaev, B., Paxson, V.: Netalyzr: Illuminating the edge network. In: ACM Internet Measurement Conference (IMC), Melbourne, Australia (2010)
17. FasterFox, `http://fasterfox.mozdev.org/` (accessed on March 2013)
18. Laghari, K.U.R., Molina, B., Palau, C.E.: QoE Aware Service Delivery in Distributed Environment. In: IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA 2011), March 22-25, pp. 837–842 (2011)
19. Laghari, K.U.R., Connelly, K., Crespi, N.: Toward total quality of experience: A QoE model in a communication ecosystem. IEEE Communications Magazine 50(4), 58–65 (2012)
20. Laghari, K.R., Pham, T.T., Nguyen, H., Crespi, N.: QoM: A new quality of experience framework for multimedia services. In: Proceeding of IEEE Symposium on Computers and Communications (ISCC), pp. 851–856 (2012)