Cost-Performance Trade-offs in Fog Computing for IoT Data Processing of Social Virtual Reality

Songjie Wang*, Samaikya Valluripally*, Reshmi Mitra*, Sai Shreya Nuguri*, Khaled Salah,† Prasad Calyam*
*University of Missouri, Email: svbqb, snd45@mail.missouri.edu, wangso, mitrare, calyamp@missouri.edu
† Khalifa University of Science, Technology and Research, Email: khaled.salah@kustar.ac.ae

***Abstract*—Virtual Reality (VR)-based Learning Environments (VRLEs) are gaining popularity due to the wide availability of cloud and its edge (a.k.a. fog) technologies and high-speed networks. Thus, there is a need to investigate Internet-of-Things (IoT)-based application design concepts within social VRLEs to offer scalable, cost-efficient services that adapt to dynamic cloud/fog system conditions. In this paper, we investigate the cost-performance trade-offs for an IoT-based application that integrates large-scale sensor data from Social VRLEs and coordinates the real-time data processing and visualization across cloud/fog platforms. To facilitate dynamic performance adaptation of the IoT-based application with increased user scale, we present a set of cost-aware adaptive control rules. The implementation of the rules is based on an analytical queuing model that determines the performance states of the IoT-based application, given the current workload and the allocated cloud/fog resources. Using the IoT-based application in an exemplar VRLE use case, we evaluate the cost-performance trade-offs with three system architectures i.e., *cloud-only*, *edge-only* and *edge-cloud* architectures. Experiment results illustrate the best/worst practices in the cost-performance trade-offs for a range of simulated IoT scenarios involving monitoring user emotional data collected by using brain sensors. Our results also detail the impact of the system architecture selection, and the benefits in enabling feedback about student emotions to instructors during Social VR learning sessions. Lastly, we show the benefits of integrating our model-based feedback control in maximizing IoT-based application performance while keeping the associated costs at a minimum level.**

***Index Terms*—IoT-based Application, Cloud/Fog System Architecture, Model-based Resource Management, IoT Data Processing/Visualization, Social Virtual Reality**

I. Introduction

Cognition refers to processes such as memory, attention, language, problem solving, and planning [1]. A virtual learning environment with a cognition-sensing application to detect the emotions of a person can help the instructors understand what tasks are causing a change in emotions such as engagement, stress, frustration, relaxation and focus. Implementing a cognition-sensing mechanism in a social Virtual Reality Learning Environment (VRLE) paves the path to a more immersive and dynamic learning environment in e.g., surgical training, and first-responders training.

Online social VRLE based learning as shown in Figure 1 normally involves geographically distributed students and instructors who teleport into a common virtual classroom along with Internet of Things (IoT) devices (such as head-mounted sensors) to measure their emotions. In this paper, we describe a novel IoT-based application in a social VRLE system that

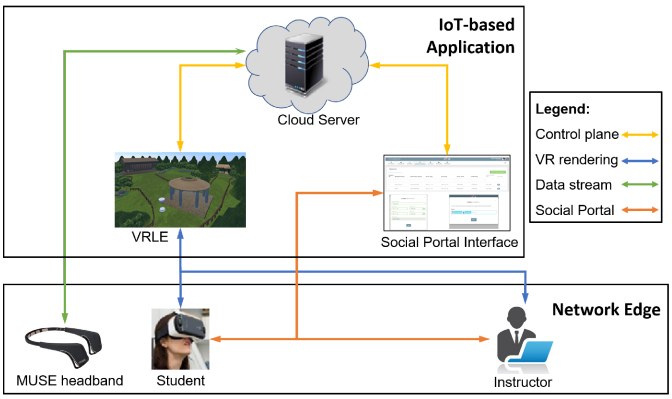


Fig. 1: Illustration of our IoT-based application integrated in a Virtual Reality Learning Environment (VRLE) with student emotion data collection using EEG sensors.

can help instructors to monitor students' learning progress. It collects the relevant emotion data from the users (through e.g., electroencephalogram (EEG) sensors such as Muse headsets [2]), and provides services based on cloud and edge (a.k.a. fog) technologies to store, visualize, and analyze the data. Based on the feedback through coordination of cloud, edge and sensor data sources for real-time data processing/visualization, the instructor can modify the training content delivery dynamically to improve learning effectiveness of the students.

The success of such a distributed IoT-based learning application depends on the architecture of the underlying cloud system resources and its extension into fog and edge resources. Our system could either run on either a *cloud-only*, *edge-only* or an *edge-cloud* architecture configurations. These architectures exemplify the fog computing concept by collecting, processing and visualizing IoT data using both edge and cloud resources, and providing a bridge between these modules. Understanding the pros and cons and evaluating these three underlying system architectures can guide a VRLE administrator to choose a suitable system configuration to satisfy social VRLE user Quality of Experience (QoE) requirements.

The novelty of our work is in the adoption of social VRLE as an exemplar use case to understand the system architecture requirements that motivate the design and evaluation of the three cloud/fog architecture candidates for real-time IoT data processing/visualization. Using an integrated set of modules, we perform a series of realistic experiments in a VRLE

testbed. Experiment results on the cost-performance trade-offs presented in this paper have the potential to pave the way for deployment of scalable VRLEs with seamless delivery of learning content. It also characterizes analytical model-based adaptive control involving processing/visualization of large-scale user emotion data within either *cloud-only*, *edge-only* or *edge-cloud* system architectures.

The main contributions of this paper are:

- To suit the purpose of monitoring user emotion data and providing real-time feedback, we have developed an IoT-based application for VRLEs; the IoT-based application automates data collection, real-time analysis and visualization through coordination of cloud/fog resources.
- Performance validation of the IoT-based application with three different system architectures: (i) *edge only*, (ii) *edge-cloud*, and (iii) *cloud only*, for large-scale IoT data processing, analytics, and visualization requirements.
- An adaptive closed-loop feedback control mechanism that can modify system behavior by adjusting system parameters and cloud resource allocations according to the system performance level, and thus maximizes the users' QoE while keeping the associated costs in the cloud at a minimum level.

The remainder of the paper is organized as follows: Section II discusses prior related works. Section III discusses the VRLE system infrastructure, implementation of the functional modules and their flow interactions. Section IV discusses the modeling and implementation of our rule-based adaptive feedback control scheme. Section V highlights the evaluation of the IoT-based application with testbed details and experiment results to compare the cost-performance trade-off for the three system architectures and benefits of the model-based adaptive feedback control. Section VI concludes the paper.

II. Related Work

A. Cloud and Sensing Integration

A large network of IoT sensor devices could generate massive volume of data whose use requires scalable cloud/fog storage systems and data analytics/visualization applications. Cloud computing offers services that can scale to IoT storage and processing requirements with elasticity and hardware diversity. Authors in [5] present an IoT system configuration and a method of EEG sensor data collection from smart helmets to a cloud-hosted server, which analyzes and visualizes data to predict soldiers state. Similarly, authors in [6] propose a suicide risk scouting prototype. In this system, patients' vital diseases symptoms are collected through wireless body sensors and then analyzed in a cloud platform with patient's historical records of diseases, habits, rehabilitation and genetics. In the work in [7], authors propose a cloud-supported Cyber-Physical localization system using smart phones to acquire voice and EEG signals for patient monitoring.

In the above exemplar works, cloud platforms are used to provide large-scale computation and communication for geographically distant users. However, these works do not provide real-time data analysis/visualization and feedback capabilities to the users as done in our work by considering low-latency data processing offered by fog platforms or a combination of cloud/fog resources.

B. Computing Architectures

Cloud platforms provide highly available computing resources, distributed storage, and offer more flexibility to users with easily customizable and configurable features. However, they are proprietary and require higher resource costs over time for using their services. Depending upon the services used and the usage level, the cost considerations for the "pay-as-you-go" can vary drastically. In this sense, both performance and cost need to be considered when evaluating system architectures that involve public cloud services such as Amazon Web Services (AWS) or local edge resources.

Similar to the focus of this study, the ability of cloud platforms to host scientific applications and the related costs for running such applications were investigated in [9]. Authors in [4] presented an elastic adaptive controller framework that can continuously detect and self-adapt to workload changes in an AWS cloud testbed. However, the tests and the proposed frameworks in these works assume operation with *cloud-only* services, and do not take into consideration of alternative cloud/fog system architectures.

The emergence of edge computing has particularly provided a scope for competent solutions that enable context-aware, real-time and low-latency response services for users. By leveraging the potential of edge computing, the authors in [8] propose an autonomic framework designed to process big data as part of a decision support system. When compared to cloud computing, edge computing could foster faster data analysis, lower costs, lower network traffic and better application speed which ultimately translates to better Quality of Service (QoS). In comparison, our work involves processing of large-scale IoT-based application workloads with low-latency demands by considering the relevant cost-performance trade-offs.

C. Model-based Resource Adaptation

For service-centered cloud applications, QoE metric plays an important role as it conveys a measure for the customer convenience and satisfaction. Since it is difficult to quantify human-subject experience in real-time due to the subjective nature of QoE, many prior studies rely on QoS metrics as an indirect measure for overall user satisfaction level[12], [10], [11]. Although the above works provide exemplar QoE-QoS correlation models, their main consideration for the QoS metrics is at network layer with limited focus on the Quality of Application (QoA) metrics at the application layer. Our work addresses this issue for IoT-based application data processing by proposing a model-based adaptation mechanism to vary QoS and QoA in order to meet the satisfactory QoE levels. In other words, our work leverages the interplay between the QoE and QoS, and applies adaptation rules that improve the VRLE user QoE by using best practices.

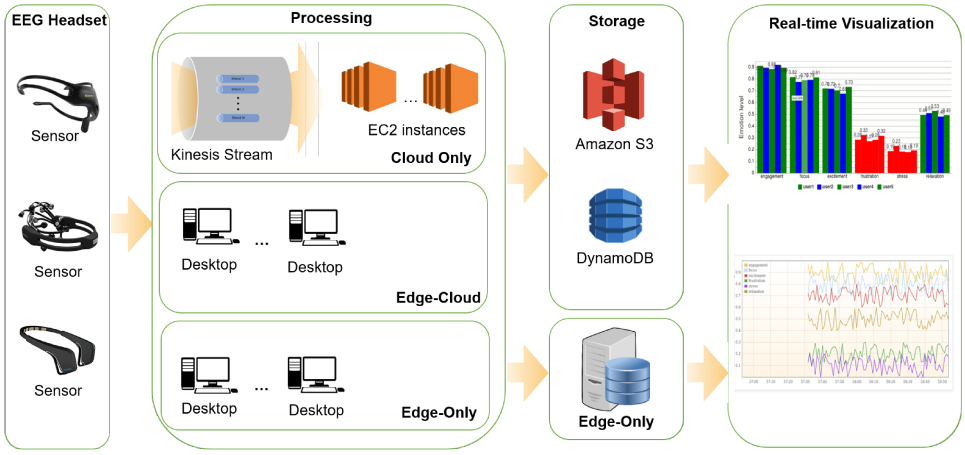


Fig. 2: Infrastructure of the IoT-based application to support the VRLE sensing data collection, processing, storage and visualization features with the three system architectures: *cloud-only*, *edge-only*, and *edge-cloud*.

In addition to QoS metrics at network layer, works such as [13] studied video-audio delivery for mobile users to demonstrate the correlation of streaming parameters at application layer with the user. Our work builds upon the recent work in [14] that addresses the issue of delivering satisfactory user QoE in cloud infrastructure reservation by taking into account all the three indices i.e., QoE, QoS, and QoA. Our analytical queuing model uniquely captures the interplay relationships for social VRLEs and considers factors such as data size, number of users, and delay in data processing, in addition to the network conditions.

III. IoT-based VRLE Application Design

A. Social VRLE Use Case

To motivate the need for a cognition-sensing application, we use system requirements from a social VRLE that was designed to improve social interaction skills. As shown in Figure 1, students log into the VRLE environment and perform specific learning sessions that are coordinated by a remote instructor on a Cloud server. The same cloud server also controls the social portal, which allows the instructor to keep schedule and monitor VR sessions as well as monitoring the emotional states of the students in the VR session. Students wear EEG headsets, which collect raw EEG data during various user learning actions. The raw EEG data are then processed and visualized on the social portal for instructor to track their engagement and other user experience metrics during the learning sessions. Based on the monitoring, the instructor can make decisions to give rewards or strikes, or acknowledge any issues to improve student learning outcomes.

B. System Architectures

As shown in Figure 2, our IoT-based application is suitable to be used with three candidate system architectures, namely the *cloud-only*, *edge-cloud*, and *edge-only*. The *cloud-only* architecture is based on Amazon AWS Kinesis, EC2 computing, and DynamoDB/S3 storage services (DynamoDB for real-time visualization and S3 for longer-term storage), while the *edge-cloud* architecture uses the DynamoDB/S3, and the *edge-only* architecture uses the DynamoDB Local (the local version of AWS DynamoDB).

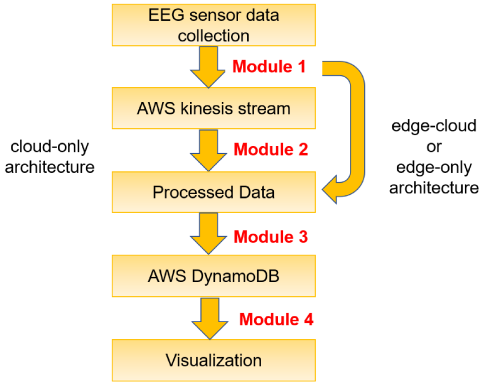


Fig. 3: Implemented modules of the IoT-based data streaming-visualization application.

User EEG data are collected in real-time by the application. Then, depending on the chosen system architecture, data will be either: (i) processed, stored and visualized in the cloud (i.e., *cloud-only* architecture); (ii) processed locally at the edge and stored and visualized in the cloud (i.e., *edge-cloud* architecture); or (iii) processed, stored and visualized locally at the edge (i.e., *edge-only* architecture). Based on the decision, the real-time data visualization will be rendered either in the cloud or at the edge and available for instructors to frequently monitor the students' emotional states during the learning activities.

C. IoT-based Application Implementation

AWS provides numerous services [15] that can be used as middle-ware for engineering solutions based on user requirements. One such sample application demonstrates how to generate statistics from a stream of online click data

and visualize the results [16]. Our IoT-based application for streaming, processing and visualizing users' emotion data is based on the framework of this application, however extended into three system architecture designs defined above. Figure 3 shows the four modules we implemented to evaluate suitable system architecture configurations:

Module 1: includes an EEG sensor application component that detects a headset and collects EEG data from the user. Based on the architecture configuration, this module either sends data to Kinesis stream (in the *cloud-only* architecture), or processes EEG emotion data and delivers the data to Module 3 (in the other two configurations).

Module 2: is only in the *cloud-only* configuration. It includes an AWS Kinesis module that consumes data from the Kinesis stream, processes the data, and then delivers the processed data to Module 3.

Module 3: accepts the data from either Module 1 or Module 2, transforms processed data into our desired data structure, and persists the transformed data into a DynamoDB table.

Module 4: creates an HTTP web server, retrieves data from the DynamoDB table, and renders data into a dynamically updated diagram to visualize the student's emotional states in a real-time manner.

IV. Model-based Adaptive Feedback Control

Cloud-hosted applications and services need to be highly scalable so that they can satisfy QoE requirements with large number of users with the least cost. In this section, we present a QoE-QoS-QoA (3Q) based feedback control mechanism that captures the dynamics of the data processing and adaptively allocates required cloud resources to satisfy users' QoE requirement (i.e., perceived visualization delay) while keeping costs at a minimum level.

A. QoE-QoA-QoS (3Q) Interplay Model

QoE-driven services have become the main focus of many cloud providers due to the vast growing cloud-based platforms and applications. In the social VRLE system, we model the performance of our IoT-based application using the 3Q factors, i.e., QoE, QoA, and QoS.

QoE is a measure of the perceived satisfaction or annoyance of a customers experiences with a service. In this work, we use objective QoE to evaluate users' QoE level. The metrics include: perceived delay in visualization of user emotion data, which indirectly also relates to the perceived system adaptation response time. QoS comprises of requirements on all the aspects of a connection, such as network bandwidth, packet loss, jitters, and delays. To simplify the complexity of our feedback control mechanism implementation, we use network bandwidth, the most deterministic metrics of network quality, to evaluate the QoS level. QoA reflects the key characteristics of an application or service in terms of processing capacity. Our QoA metrics include: the number of users, data rate, data size, exceeded write/read throughput in Kinesis stream, maximum age of data records (IteratorAgeMilliseconds) in Kinesis stream, and throttled write/read requests in DynamoDB.

These 3Q factors and their measurements are inter-related and have successions of impact to each other. The interplay among these factors help us to implement an adaptation control mechanism that is discussed in detail in the following section.

B. Cost-aware Adaptive Feedback Control Scheme

Based on the 3Q metrics, we present our rule-based adaptation control scheme for management of our IoT-based application at high loads in a social VRLE. This scheme promotes intelligent decision-making and on-demand service provisioning to ensure satisfactory user QoE with relevant cost-performance considerations.

As shown in Figure 4, when VRLE learning sessions start to operate, our IoT-based application first runs in a lower cost/performance scheme for real-time visualization. This visualization of emotion data acts as the QoE feedback to the application. Then our adaptive feedback control evaluates the objective QoE metrics to see if the visualization is satisfactory. If not, the feedback control then identifies QoA or QoS issues, such as delays in data streaming, processing or visualization rendering. Based on the identified issues, the feedback control takes appropriate adaptation action to solve the issues. However, taking one adaptation action might not be able to completely solve the issue, as there might be multiple issues related to a system performance degradation. The iterative property of the adaptive feedback control scheme will keep looping these processes until the problem is completely solved and desired QoE level is achieved. In this feedback control scheme, we also keep in mind the related cost in using cloud services. In the figure, we highlighted the adaptation schemes that are free at the edge locations as well as those that result in a higher cost in the cloud platform case. If the cost resulted from the proposed adaptation exceeds user's budget, the feedback control scheme can alternatively use different adaptation schemes when feasible, e.g., reducing the data size (i.e., changing system architecture from *cloud-only* to *edge-cloud*) or reducing data rate at the edge. However, such an adaptation will require interrupting the current running application so that changes can be made at the edge platform setup.

C. Analytical Model for Estimating Response Time

In our feedback control mechanism, there is a need of a model to capture the pattern of application performance, especially with regards to response time, given the workload in the amount of data records and currently allocated cloud resources.

The entire time of data record flow and processing in our IoT-based application can be divided into three parts, each taken at one of the three system layers, i.e., data input layer, data processing layer, and visualization rendering layer. The behavior of data processing represents a queue, we thus model this layer into an *M/M/1/K* finite queuing system. This analytical model is based on the embedded Markov Chain, featured by states, events, transitions, as described in [18].

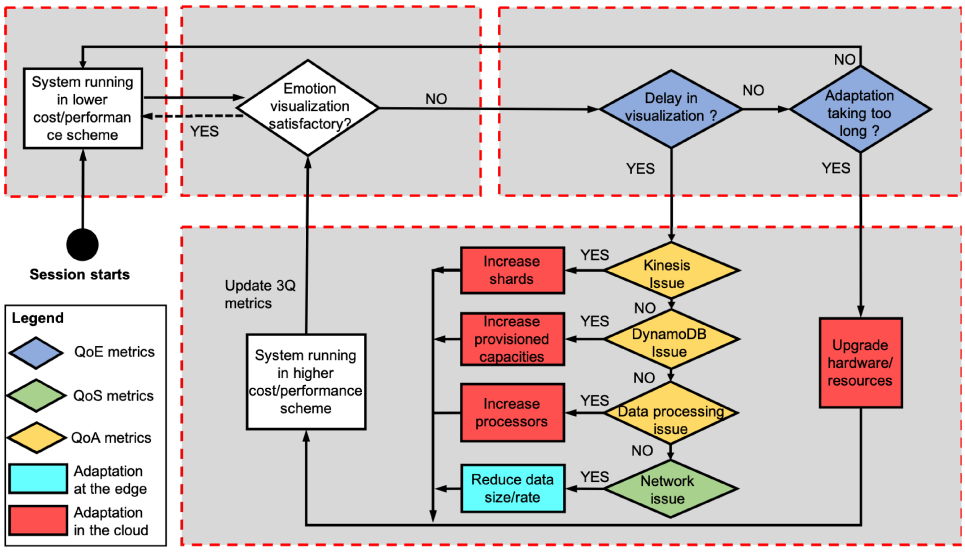


Fig. 4: Work flow of the 3Q model in the IoT-based Application. This work flow monitors the QoE, QoS and QoA metrics of the system and facilitates the feedback control mechanism to identify issues and trigger system adaptation.

The requests that enter in the queue are users' raw EEG data records. Data records are put into the queue and are processed on a First Come First Service (FCFS) basis. As seen in Figure 5, the processing of an incoming request includes three stages: stage 1 (retrieval from queue), stage 2 (record processing), and stage 3 (pushing into buffer). After Stage 3, the processed data record leaves the queue. From Figure 5, we can see that each of these stages has a different average service rate, represented as μ_1, μ_2, and μ_3. Thus, the overall response time of the system in processing one data record can be computed by solving the Markov chain transition model as described in [18]. In this process, the execution of the three stages is mutually exclusive, which means that the second record will not be processed until the previous one is completed. We assume the processing times at each stage is exponentially distributed, and the data records follow a Poisson arrival with an expected rate of λ.

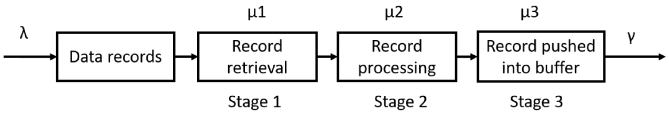


Fig. 5: Processing sequence of an incoming raw EEG data record.

When the queue contains multiple data records, our data processor processes the jobs following the Markov state model in Figure 6 with a state space $S = \{(k,n), 0 \leq k \leq K, 0 \leq n \leq 3\}$. The steady-state probability of each state in the Markov chain model can be derived similar to those presented in [18].

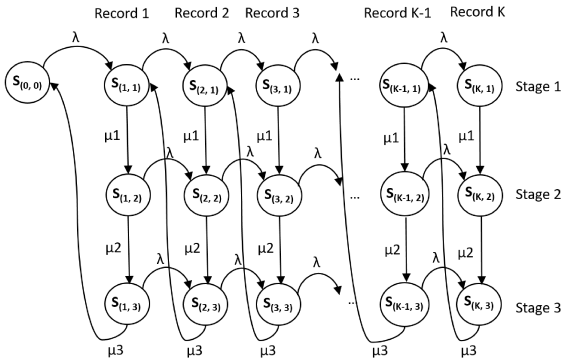


Fig. 6: State transition diagram with the three data processing stages in the Markov chain.

Based on the derived state probability of each stage, the initial state probability $p_{0,0}$ can be expressed using the normalization condition in the following form:

$$p_0 = p_{0,0} = \frac{1}{1 + \sum_{k=1}^{K} \sum_{n=1}^{3} \frac{p_{K,n}}{p_{0,0}}} \tag{1}$$

The $p_{0,0}$ can be used to compute the probabilities of all other states. The metric for the mean system throughput γ, also known as the departure rate, is

$$\gamma = \mu \sum_{k=1}^{K} p_{K,3} \tag{2}$$

The mean system throughput can equivalently be expressed as

$$\gamma = (1 - p_0)/\bar{X} \tag{3}$$

$\bar{X}$ is the sum of the mean service time for all three stages, and can be written as

$$\bar{X} = \sum_{n=1}^{3} 1/\mu_n \tag{4}$$

The departure rate γ can also be expressed as

$$\gamma = (1 - p_0)/\bar{X} = \lambda(1P_{loss}) \tag{5}$$

The loss probability P_{loss} can be expressed as

$$P_{loss} = p_k = 1 - \frac{1 - p_0}{\rho} = \frac{p_0 + \rho - 1}{\rho} \tag{6}$$

where $\rho = \lambda\bar{X}$ is referred to as traffic intensity. We can also express P_{loss} as the probability of being in states (K,1), (K,2) or (K,3), which is

$$P_{loss} = \sum_{n=1}^{3} p_{K,n} \tag{7}$$

The mean number of records in the system is

$$E[K] = \sum_{k=1}^{K}\sum_{n=1}^{3} kp_{K,n} \tag{8}$$

The mean number of records in the queue is

$$E[K_q] = \sum_{k=1}^{K}\sum_{n=1}^{3}(k-1)p_{K,n} = E[K] - (1 - p_0) \tag{9}$$

Using Little's formula, the mean time a record spent in the system, which is also the system response time, is

$$T = \frac{E[K]}{\gamma} = \frac{1}{\gamma}\sum_{k=1}^{K}\sum_{n=1}^{3} kp_{K,n} \tag{10}$$

In our feedback control mechanism, the above equations will be used to determine the system throughput, response time, and thus the cloud resources needed to handle data processing workload that come from all the distributed students in the social VRLE sessions.

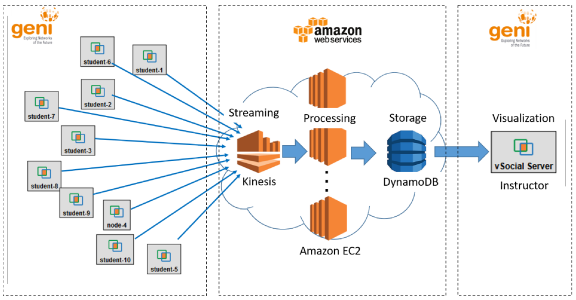
V. Experimental Evaluation

In this section, we first describe the testbed setups used for our evaluations. Next, we test the integration of the IoT-based application. We then discuss the results from two sets of experiments to evaluate the cost-performance trade-offs for *cloud-only*, *edge-cloud* and *edge-only* system architectures. Lastly, we present benefits of our analytical queuing model through experimental results.

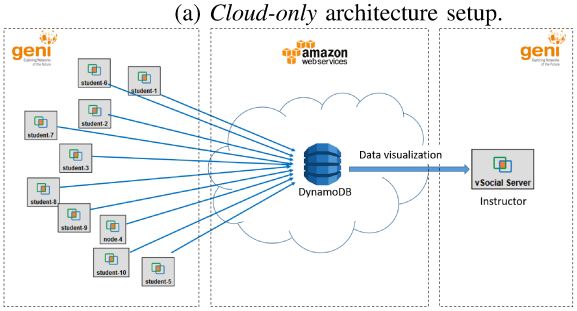
A. Testbed Setup

In the context of the social VRLE system, our testbeds for the three architectures were set up to have an instructor site and multiple student IoT devices, depending on specific test scenario for the three architectures, as illustrated in Figure 7. Both instructor and students are simulated on GENI (Global Environment for Network Innovations) edge nodes. Synthetic raw sensor data were generated from student nodes for system testing purpose. Flow of modules and data are designed according to the description in Sections III-B and III-C.

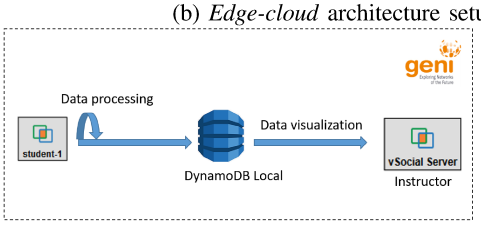
Our parameter settings for various tests are as follows: For the number of users, we tested various settings, ranging from 1 to 20 for the cost-performance trade-off analysis, and between 1-200 users for the scalability tests. We only tested 1 user setting for *edge-only* because this architecture



(a) *Cloud-only* architecture setup.



(b) *Edge-cloud* architecture setup.



(c) *Edge-only* architecture setup.

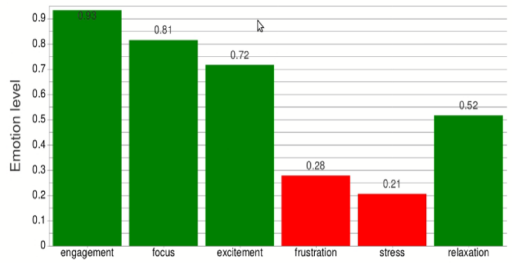
Fig. 7: Testbed setup for the three system architectures.

is not scalable for the intended comparison tests. For data size, we used two different settings, i.e., larger data object (200 KB) representing the raw sensor data collected from students and smaller data object (1 KB) representing the classified emotions. For different data rates, we tested the application with users sending emotion data once per 1 second, 5 seconds, or 10 seconds. For network quality settings, we used the following settings: (a) a higher speed network with a bandwidth of 1 Gbits/sec represented by simulating users on AWS EC2 T2.micro type instances, (b) a medium speed network with a bandwidth of 200-300 Mbits/sec represented by users on GENI edge nodes, and (c) a low speed network with a bandwidth of about 15 Mbits/sec with users set up on a wireless-edge access to a Wide Area Network (WAN) offered by a public Internet Service Provider.

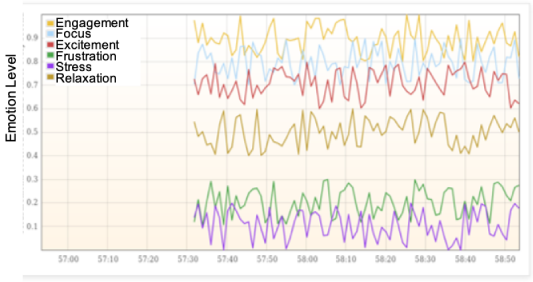
B. End-to-end Integration of IoT-based application and social VRLE

We first conducted a series of tests to ensure that the IoT-based application is able to collect, process and visualize users' sensor data in all the three system architectures. The integration of the four modules were changed based on the system architecture configuration selections. Figures 8a and 8b) show data stream from one student that was sent at 1 record/sec and visualized in all the architectures. The main visualization pages include the upper panel showing the

emotion data of the user in the most recent time stamp (1 second in this case) and the lower panel showing the history of the emotion data for this user in the last 120 seconds.



(a) Overview of the six emotions for the current student at the current time stamp.



(b) History of the emotion data for the current student in the past 120 seconds.

Fig. 8: Visualization of IoT-based application data trends.

C. Best and Worst Practices

Due to the use of AWS Kinesis, EC2, and DynamoDB services for streaming, processing, storage and real-time visualization of users' emotion data, the settings of these cloud services in our IoT-based application need to be in compliance with the limits defined by the service provider. Based on these limits, as well as considering the costs of using these services, we summarize some of the best and worst practices, as shown in Table I. The main settings and metrics, both in the cloud and at the edge, include number of shards (Kinesis stream), write and read throughputs (Kinesis stream), provisioned write and read capacities (DynamoDB), consumed write and read capacities (DynamoDB), throttled write and read requests (DynamoDB), number of users, data sizes, data rates, and network bandwidth. These best and worst practices, based on the selected parameters, provide fundamental test scenarios when evaluating the costs and performance trade-offs of the three candidate system architectures, as will be discussed in the next section. The results from Table I show that:

- Performance of *edge-only* architecture depends on the storage, network connections, and computing power on the server side; due to storage restrictions, it normally requires to have a backup storage in a cloud platform;
- Performance of *edge-cloud* architecture depends on the provisioned write and read capacity, the rate and the size of data traffic, which determines the write and read throughput, into and from DynamoDB tables;
- Performance of *cloud-only* architecture depends on the rate and size of the data, the write and read throughput, into and from both Kinesis stream and DynamoDB tables.

D. Cost-Performance Trade-off Analysis

The results in the previous section discussed some of the best and worst practices when implementing our IoT-based application with the three system architectures. In order to fully understand the effects of these practices on cost and performance, along with identifying the trade-off between these two important decisive factors, we set up experiment scenarios with 'stress-test' type settings, shown in Table II. The bottom 2 rows in the Table show the delay in visualization, which represents the performance of the application for the selected architecture and parameter settings in Table II.

As shown in Figure 9, we analyze the costs for running both *cloud-only* architecture (DynamoDB and Kinesis) and *edge-cloud* architecture (DynamoDB alone), the exceeded write/read throughputs, which represent the delays in Kinesis stream, and the throttled write/read requests, which represent delays in writing/reading data in DynamoDB table. In the best practices settings, the *cloud-only* and *edge-cloud* architectures have no throttled requests (BP1, BP2), indicating no performance degradation, and the costs for both architectures are at the lowest level. While in the four worst practice scenarios, there were either high exceeded throughput or throttled requests (WP1, WP3, WP4). Note that WP2 did not have either exceeded throughput or throttled requests, but in this practice the data size was small but its cost was much higher than BP1, which was also sending small size data. It can also be observed that the cost of Kinesis plus DynamoDB, which is used by the *cloud-only* architecture, is much higher than using just the DynamoDB service, as in the *edge-cloud* architecture. The above results suggest that:

- The *cloud-only* architecture is more prone of delays in rendering the emotion data visualization, due to the large data that exceeds the write/read throughput of Kinesis stream, as well as the provisioned write/read capacity of DynamoDB tables;
- With the *edge-cloud* architecture, there is a less chance of delays in visualization since the data streams are processed at the edge nodes, and hence small size data is sent to the cloud platform for visualization;
- For the *edge-only* architecture, we could not acquire results owing to the limited scale property of this system architecture.

Building upon the above results, we summarized the scalability tests and the typical costs when 10, 100, or 200 users are using various architectures, as shown in Table III. These results, together with the data shown in Table I and Figure 9, suggest that based on the social VRLE user requirements:

- *Edge-only* architecture is the cheapest option, but it is horizontally non-scalable and comes with the need to store backup data on cloud resources;
- *Cloud-only* architecture is the most expensive option but provides the advantages of nearly unlimited scaling to

Test case parameters	*edge-only* architecture (DynamoDB Local)	*edge-cloud* architecture (DynamoDB)	*cloud-only* architecture (Kinesis + DynamoDB)
Best Practices	**Key-Findings:** ∗ Save data backup on cloud due to storage restrictions ∗ Remote DynamoDB connections need high network bandwidth ∗ Maximize computing resources	**Key-Findings:** ∗ Provisioned write capacity = consumed write capacity ∗ Provisioned read capacity = consumed read capacity ∗ Write throughput <= 1KB/write capacity ∗ Read throughput <= 4KB/read capacity	**Key-Findings:** ∗ Max. data blob size <= 1MB ∗ Write throughput < 1 MB/sec/shard or 1,000 records/sec/shard ∗ Read throughput < 5 transactions or 2 MB/sec/shard ∗ No. of shards (integer) = ceil[(no. of users * data rate * data size)/1000] ∗ For DynamoDB: same as *edge-cloud*
Worst Practices	**Key-Findings:** ∗ Not storing backup on cloud due to storage restrictions ∗ Not enough network bandwidth ∗ Not enough computing resources	**Key-Findings:** ∗ Provisioned write capacity > consumed write capacity ∗ Provisioned read capacity > consumed read capacity ∗ Write throughput > 1KB/write capacity ∗ Read throughput > 4KB/read capacity	**Key-Findings:** ∗ Increase Data blob size > 1 MB/record ∗ Data size > 1 MB/sec/shard ∗ Read throughput >5 transactions or 2MB/sec/shard ∗ no. of shards >> no. needed ∗ For DynamoDB: same as *edge-cloud*

TABLE I: Comparison of the best and worst practices for the three system architectures.

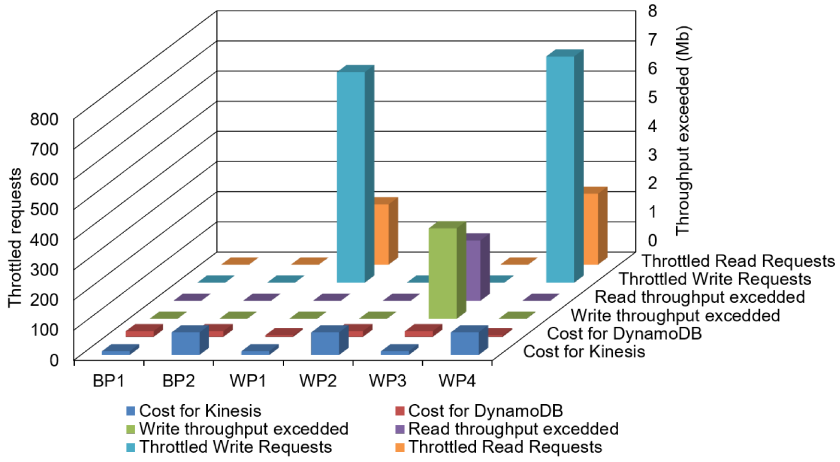


Fig. 9: Comparison of the cost-performance trade-offs in various best and worst practices for using AWS Kinesis and/or DynamoDB cloud services in *edge-cloud* or *cloud-only* architecture.

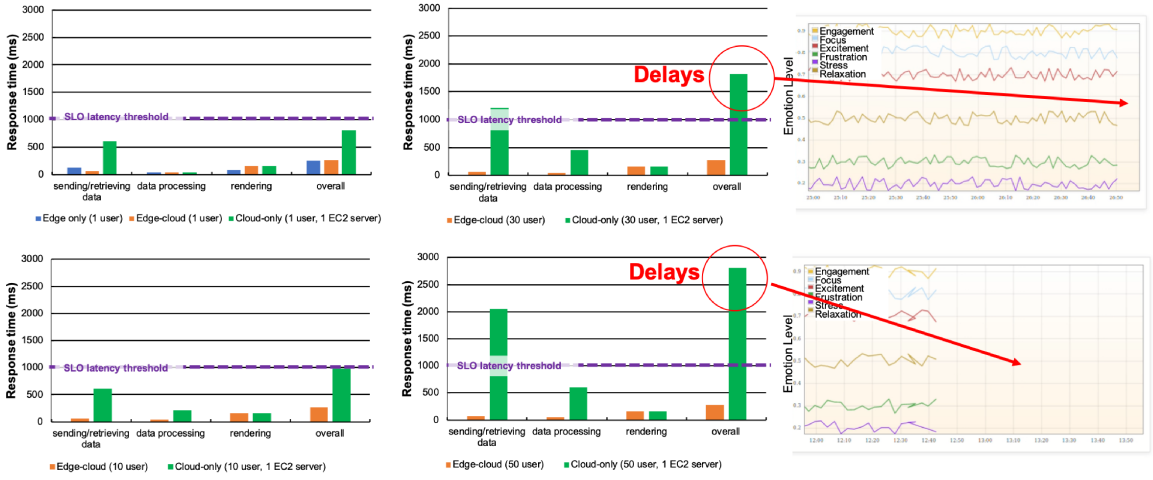


Fig. 10: Comparison of visualization delay with respect to the *edge-only*, *edge-cloud*, and *cloud-only* system architectures. The response time in sending/retrieving data from stream, data processing, visualization rendering and the overall time are compared when 1, 10, 30 and 50 users are serviced in the system. Delays (e.g., when response time is over 1000ms) are referenced to the data history diagram. *Edge-only* architecture has only 1 user data since it is not scalable.

deal with large data inputs and fluctuating workloads; plus, the *cloud-only* architecture is most robust and adap-

Parameter Setting	BP1	BP2	WP1	WP2	WP3	WP4
Max No. of users tested	20	20	20	20	20	20
Data Size (KB/record)	1	200	1	1	200	200
Data Rate (seconds)	1	1	1	1	1	1
Number of Shards	1	4	1	4	1	4
Provisioned W/R Capacity	20	20	10	20	20	10
Visual Delay (*cloud-only*)	/	No	/	/	Yes	Yes
Visual Delay (*edge-cloud*)	No	/	Yes	No	/	/

TABLE II: Best and worst practices tests in cost-performance trade-off analyses for *cloud-only* and *edge-cloud* architectures. **Legend:** BP - best practice; WP - worst practice; W/R - write/read.

Test case parameters	*edge-only*	*edge-cloud*	*cloud-Only*
User Scalability	Cost:$2.31, 100 GB storage/month	Cost:$136, User:200 200 records/sec	Cost:$634, User:200 200 records/sec
		Cost:$65, User:100 100 records/second	Cost:$314, User:100 100 records/second
		Cost:$5, User=10 10 records/second	Cost:$30, User=10 10 records/second
	Comments: Scales up, not out Limited storage and computing resources	**Comments:** Scales up and out, limited to local computing resources	**Comments:** Scales up and out, no limitations

TABLE III: Performance vs. Cost (per month) for 1 KB data input/user/sec and 1 Kinesis stream.

tive to data traffic demands imposed by the VRLE;

- *Edge-cloud* architecture is the less cheaper option (when compared to the *cloud-only* option), and avoids the heavy workload on the cloud resources by data processing at the edge resources. However, reducing the data structure could result in significant information loss as well as burdening edge nodes with heavy computations.

E. Adaptive Feedback Control

Previously we identified performance issues, i.e., delays, with our IoT-based application when the number of users increased. The issues were mostly seen in the *cloud-only* architecture. When testing the application with the number of users at 30 to 50, even with well defined number of shards and provisioned write/read capacities, we still experienced serious visualization delays (Figure 10). This was caused by computationally overloaded EC2 instance when processing large-scale raw sensor data. To reduce the computational stress on the EC2 processor, there was a need to scale up the number of EC2 instance processors to offload the workload.

The distribution of the average time that a single data record goes through the whole application includes: inputting into Kinesis stream and visualization rendering each taking 125ms and 120ms, respectively; retrieval from stream taking 3ms; data processing taking 40ms; pushing into DynamoDB buffer taking 2ms; and writing into DynamoDB table taking 8ms. As specified previously, the queuing stage includes data retrieval from the stream, processing on EC2 processor, and pushing into buffer. Thus our queuing analytical model was used to analyze the time that data records spent in these three steps.

Based on the equations specified in Section IV-C, we predicted the overall system response time when there are between 1 and 150 users, with various numbers of EC2 instance processors, as shown in Figure 11. To meet user's QoE requirement, we needed the system to process and render visualization within 1 sec from data collection. Since the average time for outside the queue added up to about 250ms, we defined 0.7s (700ms) as our tolerable latency threshold for determining the minimum number of processors required. The result from this analytics justified our earlier finding that when the number of users was at 30 or 50, there was a high level of delays in processing the data records using just one EC2 instance processor.

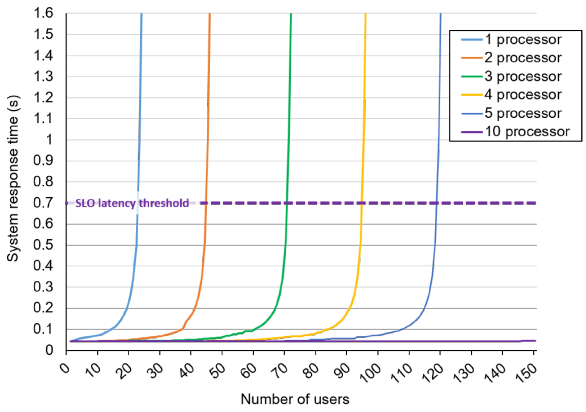


Fig. 11: Impacts of the number of users and EC2 data processors to system response time.

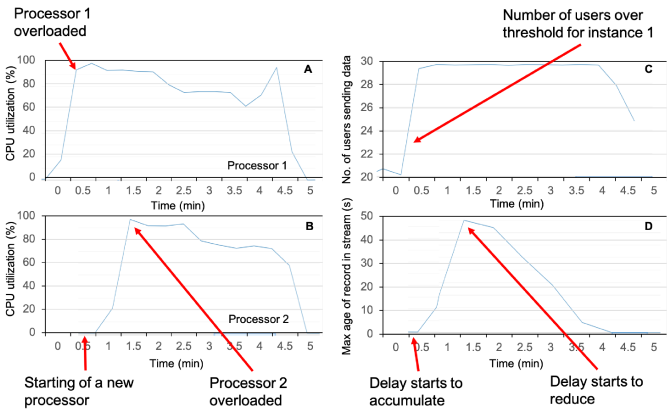


Fig. 12: Adaptive feedback control mechanism scales-up by increasing the number of EC2 data processors.

As part of our evaluation experiments, we tested the control feedback loop with the IoT-based application running on the *cloud-only* architecture. And we focused on the adaptation on the number of data processors. In the experiment, we setup the testbed as shown in Figure 7a.

Figure 12 shows our adaptation process test case. We started the application with 20 user sending data, and one EC2 instance as the data processor. As shown in panel (D), there was no delay at this stage. Then when users increased to 40, the CPU utilization in Processor 1 reached to almost 100%

(panel A), which resulted in the delays in processing data record (panel D). Upon detection of the increase in the delay in processing data, our feedback control mechanism calculated the required number of processors needed at this stage, which was 2, by using the analytical queuing model shown in Figure 11. Then a second EC2 processor was initiated to offload the workload in processing data records (panel B). The CPU utilization on Processor 2 quickly increased to almost 100%. With the two processors working together, the delays started to reduce until there was no significant delay any more. It took about 3 minutes for the two processors to clean up the delayed data in the queue. During this time period, no additional data processor was initiated because the delay was decreasing. At the end of this period, our application recovered and was able to visualize all users emotion data in real-time. After we reduced the number of users back to 20, our algorithm adaptively stopped one of the two processors according to the current workload requirement and delay status. Our design and initial experimental evaluation of the adaptive feedback control scheme for scaling up cloud resource allocation shows that:

- Our analytical queuing model can be used to effectively determine the number of data processors required for scenarios with different number of users in a given IoT-based system; the model also justified our earlier findings of system delays with higher number of users;
- Our feedback control scheme is able to predict system performance levels using selected performance metrics, and use this knowledge to scale out the processor resources according to the analytical model guided rules;
- To our knowledge, the feedback control scheme presented in this paper is the first adaptation framework supporting social VRLE based IoT application. The merits of this adaptation scheme include: minimal resource over-provisioning and thus user costs; fine control by using the maximum age of records metrics to identify both system performance issues and progress in adaptation; and high intelligence enabling the system to wait for adaptation progress before further scaling-out resources.

VI. Conclusion

In this paper, we present an IoT-based application designed to manage visualization of the sensor data from geographically distributed users in a social VRLE. Our work addressed the challenges in handling the cost-performance trade-off analysis for a distributed system with multiple devices generating real-time high volume data. The challenges related to configuring a suitable system architecture amongst options in the fog/cloud computing: *edge-only*, *edge-cloud* and *cloud-only*. Our cost-performance analysis results provide insights on the best practices that need to be followed for obtaining maximum performance for supporting a large number of users, yet at minimum cost.

We also described an analytical model-based dynamic performance adaptation scheme that can trigger rules to deal with high-scale loads to maximize the user experience (i.e., perceived visualization delay) by controlling the application (VRLE) and the system (sensors, network) parameters. Our analytical queuing model with derived formulas is shown to be beneficial to monitor the overall system response time and adapt the edge and cloud resources suitably.

Our future work is to investigate our IoT-based application within VRLE education content that serves different learning curriculum objectives, e.g., public safety best practices training for first responders and incident commanders. In addition, machine learning based algorithms can be developed to correlate sensor data and use context in social VRLE systems with feedback to enhance student-instructor collaboration.

References

[1] H. Okon-Singer, T. Hendler, L. Pessoa and A.J. Shackman. "The neurobiology of emotioncognition interactions: fundamental questions and strategies for future research", *Frontiers in Human Neuroscience*, 2015.

[2] Muse: Meditation made easy, the brain sensing headband. [Online]. Available: http://www.choosemuse.com

[3] C. Zizza, A. Starr, D. Hudson, S. Nuguri, P. Calyam, Z. He. "Towards a Social Virtual Reality Learning Environment in High Fidelity", *Proc. of IEEE Consumer Communications & Networking Conf. (CCNC)*, 2018.

[4] A. Khoshkbarforoushha, A. Khosravian, R. Ranjan. "Elasticity management of streaming data analytics flows on clouds". *Journal of Computer and System Sciences*, Vol. 89, No. 1, pp. 24-40, 2017.

[5] G.H. Jo, S.B. Jeon, H. Chung, Y.J. Song, "Sensor Data Analysis and Visualization of IoT System for Combat Helmet", *Advanced Science Letters*, Vol. 23, No. 10, pp. 10342-10345, 2017.

[6] M.G.R. Alam, E.J. Cho, E.N. Huh, C.S. Hong, "Cloud based mental state monitoring system for suicide risk reconnaissance using wearable bio-sensors", *Proc. of ICUIMC*, Article: 56, 2014.

[7] M. Hossain, "Cloud-Supported CyberPhysical Localization Framework for Patients Monitoring", *IEEE Systems Journal*, Vol. 11, No. 1, pp. 118-127, 2017.

[8] M.P. Hosseini, T.X. Tran, D. Pompili, K. Elisevich, H. Soltanian-Zadeh, "Deep Learning with Edge Computing for Localization of Epileptogenicity Using Multimodal rs-fMRI and EEG Big Data", *Proc. of IEEE ICAC*, 2017.

[9] I. Sadooghi, J.H. Martin, T. Li, K. Brandstatter, K. Maheshwari, T.P.P. de Lacerda Ruivo, G. Garzoglio, S. Timm, Y. Zhao, I. Raicu, "Understanding the performance and potential of cloud computing for scientific applications", *IEEE Transactions on Cloud Computing*, Vol. 5, No. 2, pp. 358-371, 2017.

[10] T. Hossfeld, P. Tran-Gia, M. Fiedler, "Quantification of quality of experience for edge-based applications", *In Managing Traffic Performance in Converged Networks, Springer, Berlin, Heidelberg*, pp. 361-373, 2007.

[11] R. Koshimura, Y. Ito, Y. Nomura, "Evaluation of relationship between QoS and QoE for web services considering the hierarchical structure with principal component analysis and path analysis", *Proc. of IEEE Asia-Pacific Conference on Communications (APCC)*, pp. 266-270, 2014.

[12] M. Li, C. Lee, "A cost-effective and real-time QoE evaluation method for multimedia streaming services", *Telecommunication Systems*, Vol. 59, No. 3, pp.317-327, 2015.

[13] M. Seufert, P. Casas, F. Wamser, N. Wehner, R. Schatz, P. Tran-Gia, "Application-layer monitoring of QoE parameters for mobile YouTube video streaming in the field", *IEEE Intl. Conference on Communications and Electronics (ICCE)*, 2016.

[14] D. Chemodanov, P. Calyam, S. Valluripally, H. Trinh, J. Patman, K. Palaniappan, "On QoE-oriented Cloud Service Orchestration for Application Providers", *IEEE Transactions on Services Computing*, 2018.

[15] J. Varia, S. Matthew,"Overview of Amazon Web Services", 2014.

[16] J. Ganoff, Amazon Kinesis Data Visualization Sample Application. [Online]. Available at: https://github.com/awslabs/amazon-Kinesis-data-visualization-sample, 2014.

[17] M Berman, J S. Chase, L Landweber, A Nakao, M Ott, D Raychaudhuri, R Ricci and I Seskar, "GENI: A federated testbed for innovative network experiments", *Computer Networks*, 61(1):5-23, 2014.

[18] K. Salah, P. Calyam and R. Boutaba. "Analytical model for elastic scaling of cloud-based firewalls", *IEEE Transactions on Network and Service Management*, 14(1):136-146, 2017.