5G Performance Testing of Mobile Chatbot Applications

Vaios Koumaras
INFOLYSiS P.C.
Athens, Greece
vkoumaras@infolysis.gr

Andreas Foteas
INFOLYSiS P.C.
Athens, Greece
afoteas@infolysis.gr

Angeliki Papaioannou
INFOLYSiS P.C.
Athens, Greece
apapaioannou@infolysis.gr

Marianna Kapari
INFOLYSiS P.C.
Athens, Greece
mkapari@infolysis.gr

Christos Sakkas
INFOLYSiS P.C.
Athens, Greece
chsakkas@infolysis.gr

Harilaos Koumaras
NCSR Demokritos
Agia Paraskevi, Greece
koumaras@iit.demokritos.gr

***Abstract* — A Chatbot is an application that is designed to provide automated contextual communication. Today most chatbots are implemented on top of or as a gateway to popular messaging services, such as Facebook Messenger, Skype and Viber. Chatbots can be classified into many categories regarding their usage, such as conversational commerce, customer support, education, marketing and others. Due to their agile deployment ability on top of virtualized and serverless environments, chatbots are expected to play a pivotal role in the forthcoming 5G networks, which support virtualization capabilities at the edge of the network, making feasible the provision of diversified chatbot services customized to each user needs and requests. However, chatbot QoS might be affected under congested network conditions or in areas with poor signal reception quality. Currently, the performance of the chatbot has not been researched, while the users are experiencing only the results of the potential QoS degradation, such as loss or re-ordering of messages. This paper provides an experimental study of the chatbot apps performance/QoS under different network and reception conditions. The experiment was conducted using the 5G mobile network emulation testbed created and provided by the EU-funded TRIANGLE project.**

Keywords—Chatbot, QoS, QoE, 5G, Benchmarkıng

I. INTRODUCTION

A recent trend in the provision of interactive services is the chatbots apps. Chatbot is a computer application, usually powered by machine learning techniques, on top of popular messaging applications, such as Viber, Messenger, and Skype. Chatbots interact with users using natural language in a similar way to imitate a human agent/representative [1]. A successful implementation of a chatbot system can analyze user preferences and predict collective intelligence and in most cases, it can provide better user-centric recommendations. Hence, the chatbot is becoming an integral part of the future consumer services, representing a new way for brands, businesses and publishers to interact with users without requiring them to download a another specialized application, with which they should become familiar or further configure and update regularly, since the already installed messaging application is used.

From the 5G perspective [2], chatbots is one of the next massively deployed applications in the mobile market, which follow the Mobile Edge Computing (MEC) paradigm [3], which means that a distributed serverless computing environment for chatbot application/service hosting is required in close proximity to cellular subscribers, for more agile deployment and faster response time. Especially in chatbots, the achieved QoE [4] and response time is of highly importance in order the user to perceive an appropriate and realistic user experience, depending also on the accuracy the specific chatbot application requires, (e.g. different requirements has an IoT-based informative chatbot app from a remote controlling one).

The motivation for this paper was on the one hand the lack of bibliography in the performance assessment of chatbot applications in relation to the network/reception conditions and on the other hand the increasing popularity of chatbot apps. Although, QoS degradation incidents [5] have been observed in commercial chatbot applications (e.g. messages were delivered in a random order or lost or even delivered partially, usually without the keyboard message), under congested network conditions or in areas with poor signal reception quality, however a sufficiently documented quantitative study has not been conducted. Neither there is any official documentation available by Viber, Facebook or Skype, which describes the network condition requirements and operational values of the chatbot apps in order to reassure a smooth and acceptable QoE and QoS level.

Therefore, this paper presents the performance benchmarking of different chatbot applications on top of the popular messaing platform Viber. The chatbots under test have been assessed in different emulated network conditions, using the TRIANGLE emulated testbed, which provides an experimental facility for testing applications and devices in realistic 5G network conditions. The experiment results are expected to add both direct and indirect value by improving the performance quality of the chatbot apps and providing better quality of service to the end users

The rest of the paper is organized with section II highlighting the background of chatbot apps and section III describes the core components of the experiment including the testbed architecture and the Chatbot applications. Section IV presents the experimental process followed describing the Scenarios which were tested, and the metrics used for the QoS and QoE estimation. Finally, in the Section V are presented the results of the experimental process as well as the evaluation of the results.

II. The Chatbot Apps

Chatbots [6] are stateless web apps that are usually developed with Javascript, such as Node.js, utilizing the API of a popular messaging service. The chatbot functionality is based on events emitters. A Chatbot solution, depending of course on the open API of each platform, may support a diversity of messages [7], where in the case of Viber the following type messages are supported: Text Message, URL Message, Contact Message, Picture Message, Video Message, Location Message, Sticker Message, File Message and Rich Media Message.

Fig. 1. A Viber-based chatbot App with auxiliary keyboard buttons

Bundle with these type of messages, usually it is sent also a Keyboard Message appropriate JSON file, which appears to the users auxiliary keyboard buttons in order to facilitate their navigation within the chatbot app. Such an auxiliary keyboard message is depicted in Fig. 1.

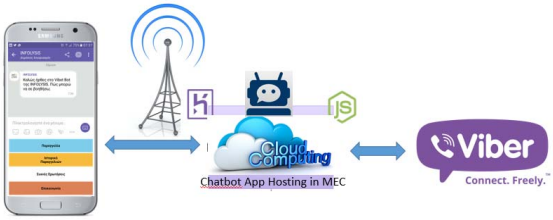

Fig. 2. Viber Chatbot App Ecosystem

The chatbot app in order to communicate with the messaging platform (in our case Viber as Fig. 2 depicts), it is necessary to use a server that will receive, process and send back messages. The server will make use of the Viber Bot Developer API, since this is the primary way to get data in and out of Viber's platform. The server must have an endpoint URL (Webhook) that is accessible from Viber's servers and all new subscriptions to the service have to use a secure HTTPS callback URL. Currently, there are many cloud-based services which are suitable for the development of the chatbot sector, such as IBM Watson, Microsoft bot, AWS Lambda and Heroku [8]. For the deployment needs of the chatbot apps of this paper, Heroku hosting service for managing stateless Node.js web application was selected, which has many similarities to serverless environments related to the processes of managing and maintaining the hosting servers.

III. Components Of The Experiment

In this section are described the core components of the experiment setup for the performance assessment of the chatbot app under different reception conditions.

A. Testbed Architecture

From a high-level perspective the experimental testbed [9] used in this paper is depicted in Fig. 3.

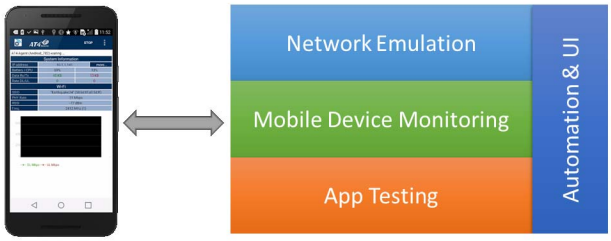

Fig. 3. Testbed high-level overview

As it is shown, the testbed is composed of five main elements:

- **Device cloud**: the testbed provides a set of mobile phones ready to be used.
- **Network Emulation**: from the physical channel to the network core. The system is a 4G/4.5G one, and will undergo upgrades following the 5G standardization.
- **Mobile Device Monitoring**: a set of capabilities which range from software to evaluate the QoE, to hardware probes that measure power consumption
- **App Testing**: automates the user flows in the testing app to be able to repeat the same test under different conditions
- **Automation & UI**: the software that coordinates the experiment, providing a user interface with the appropriate level of details for each user profile

B. Chatbot Applications

Three different types of chatbots over Viber platform have been used for the execution of the experiment, as Table I shows, each one having a different degree of complexity and requirements.

TABLE I. Features of Chatbots under Test

Chatbot No.	Features		
	Static Messages	*Database Communication*	*API-based Communication*
Chatbot #1	✔		
Chatbot #2	✔	✔	
Chatbot #3	✔	✔	✔

Each chatbot was selected because it provides, as Table I shows, a different level of complexity in terms of features and therefore connectivity requirements. A brief description of each chatbot app that was used for the experimental needs of this paper follows.

*1) **Informative Chatbot:*** This chatbot is a testing version of a commercial chatbot created for a popular e-shop. The main functionality is to provide information about the shop and for special offers. The user can browse different categories and get informed on the various offers per category/items. In this chatbot, all the messages are static and locally stored at the cloud computing platform. All messages are coded individually and therefore are prone to shuffling under impaired network conditions.

*2) **Contest Participation Chatbot:*** This chatbot created for a contest event. The main purpose of this chatbot is to assist the participants of the event to learn more about the event, the speakers and finally participate in an online contest. This chatbot involves messages that are static and locally stored at the cloud computing platform, but also messages that ask from the user to enter data, which will be stored to a JawsDB database. The messages are coded in a structured manner and therefore we are expecting the bot to be robust against message shuffling under impaired network conditions. The behavior of the prompt messages for data entry may be affected by the network conditions.

*3) **Order Placement Chatbot:*** This is a testing version of a commercial chatbot created for online food order placement. The main functionality is to provide information about the products and place an online order. This chatbot combines messages that are static and locally stored at the cloud computing platform, but also message that are dynamically generated from an API communication with the ERP of the business owner. The same also applies for the keyboards, which are dynamically generated from the API communication. The messages are coded both in a structured and unstructured manner and therefore we are expecting the performance of the bot to be affected by the network conditions. Moreover, the communication with the API and therefore the appearance of the buttons or the messages may be also affected.

IV. Experimental Process

For the execution of the experiment it was mandatory to use the commercial application of Viber, via which the user would have access to the chatbot service. For the emulation of cellular network, the testbed [9] is using the UXM Wireless Test Platform device by Keysight. This device is capable of modifying a number of parameters of the wireless physical layer and thus emulate various network conditions. The parameters that modified in this experiment are:

- The number of Downlink and Uplink Physical Resource Blocks (PRBS),
- The number of Downlink and Uplink Subframes,
- The Multipath Fading Propagation Conditions (EPA, EVA, ETU),
- The antenna output power,
- The mode, the type (AWGN) and the power of the environmental noise,
- The max Doppler shift

Each mobile device was directly connected to the Keysight Source Management Unit instead of the battery and supplied with 5V DV voltage. This set-up was offering a flexible configuration to meet the power sourcing and analysis requirements.

For the orchestration of the experimental process as well as the configuration of the UXM and SMU devices the experimental testbed [9] used the Keysight KS8400A Test Automation Platform (TAP). This software was enabling a powerful, flexible and extensible test sequence and test plan creation.

The mobile devices were controlled via the Quomation WebDriver, a test automation framework for use with native, hybrid and mobile web apps. Furthermore, the Quamotion Frontend provided a device monitoring and controlling interface. The mobile devices used in this experiment were the Samsung Galaxy S7 and Samsung Galaxy S4. Figure 4 depicts the overall experimental architecture, clarifying that the access to the testbed [9] was achieved via a remote desktop connection, which had installed and properly configured the Quomation Tool and Test Automation Platform.

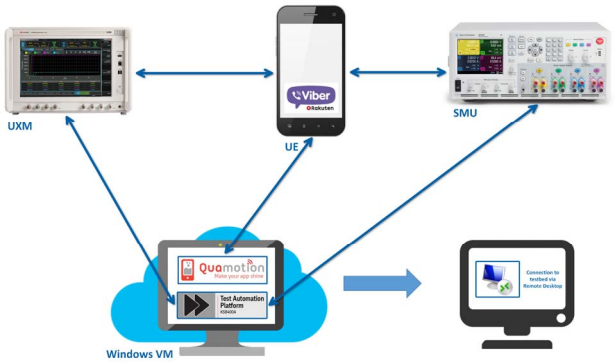

Fig. 4. Technical Architecture & TestBed

Due to the manual testing nature of the experiment it was mandatory to capture the experimental process in video form for later processing, and information extraction of each step. As step is defined each message sent to the chatbot and vice versa.

A. Scenarios Under Test

TABLE II. Triangle Testbed Network Scenarios

Category	Network Scenario
Suburban	Festival (FE)
	Shopping Mall Busy Hours (SM-B)
	Shopping Mall off Peak (SM-OP)
	Stadium (ST)
Urban	Traffic Jam (TJ)
	Internet-Café busy hours (IC-B)
	Internet-Café off peak (IC-OP)
	Office (OF)
	Pedetrian (PD)

The Chatbots were tested in a series of network emulated scenarios that the testbed platform [9] had predefined. These network scenarios are emulating realistic network conditions and they are classified into two major categories:

- Urban,
- Suburban,

Table III lists the network scenaria for each category type. For each of the network scenario, a predefined set of steps was executed for every chatbot application. The outcome of these executions has been used used for the estimatioin and calculation of the metrics and KPIs of the chatbot application performance.

Each network scenario consists of two to four sub-scenarios with different parameters interchanging every 30 seconds. This method enables the simulation of the variations of the signal quality due to changes in the network load or movement of the user.

TABLE III. EXTREME (EX) NETWORK SCENARIO

Physical Parameters	**Subscenarios**		
	ss-1	***ss-2***	***ss-3***
DL PRBS (%)	10	20	5
#/10 DL subframes	1	2	1
UL PRBS (%)	10	20	1
#/10 UL subframes	1	2	1
Output Power (dBm)	-100	-95	-95
AWGN Power (dBm)	5	5	20
Channel Model	ETUC	ETUC	ETUC
Doppler Shift (Hz)	200	20	5
Correlation	MED	MED	MED

The Test Automation Platform was enabling the generation of custom network scenarios. In order to further stress-test the performance of each chatbot, we created, as Table III depicts, a custom network scenario named EX(treme), which stresses the performance of the chatbot app to extreme conditions.

B. Metrics for Performance Benchmarking and QoS Assessment

Benchmarking is the process of comparing a system's performance against a baseline that has been created to determine whether performance is improving or declining and to find deviations across different operating conditions. A critical aspect of a baseline is that all characteristics and configuration options except for those specifically being varied for comparison must remain unchanged.

In the experiments of this paper, the chatbots under test are located at HEROKU/AWS servers and are utilized only for testing purposes in order to reassure that the system is not affected by external factors. Thus, taking into account that Viber architecture is designed in a way that overcomes performance and capacity challenges, the only parameters responsible for the degradation in the chatbot performance are those that are introduced and can be emulated by the testbed platform [9] and are related to the reception quality. The KPIs/metrics used in order to measure the quality of the chatbot service in every network condition are:

1) ***Maximum Step Load Time:*** The max step load time is defined as the maximum time needed for the chatbot to reply to the user's input. For each Chatbot test, the load time that is measured is at the step with the most delayed response.

2) ***Average Completion Time:*** As average Completion time is defined the average time needed for the same user to complete 10 successful executions of the three chatbot types.

3) ***Completion Rate:*** This metric depicts the percentage of successful completions in the set of experiments for each of the chatbots and the network scenarios

4) ***Service Stability:*** The forth Key Performance Indicator in the experiment is the service stability. The service stability metric is directly dependent to the following observed issues:

- Image Loss,
- Message Repetition,
- Message Reorder and
- Keyboard Loss.

Each of these observed issues has a different bias in the calculation formula of service stability, which is provided as eq. 1. In the following equation the N represents the total number of steps in each scenario and m represents the number of successfully executed steps. The variable I_i is equal to 1 for each step that is observed the corresponding issue and equal to 0 elsewhere.

$$S = \frac{1}{N}\sum_{i=1}^{m}\{(1 - 0.5I_i^{IL} - 0.3I_i^{MRP} - 0.2I_i^{MRD})(1 - I_i^{KL})\} \quad (1)$$

Each test score is normalized to the maximum score that can be achieved for each chatbot, so the best value for the service stability is 1 and the worst is 0.

V. EXPERIMENTAL RESULTS

The Chatbot experiment results confirmed the high degree of performance tolerance that chatbot applications have under different network conditions.

A. Network Scenarios Comparison

For the calculation of the max step load time comparison, the steps with the most delayed response are the ones that include an image or thumbnail loading.

From Fig. 5, it is derived that in realistic network conditions the chatbot response time is slightly affected at levels of 1-2 secs, but in the extreme network scenario there is more than 100% increase in the chatbot response time, which reaches 6 to 7 secs.

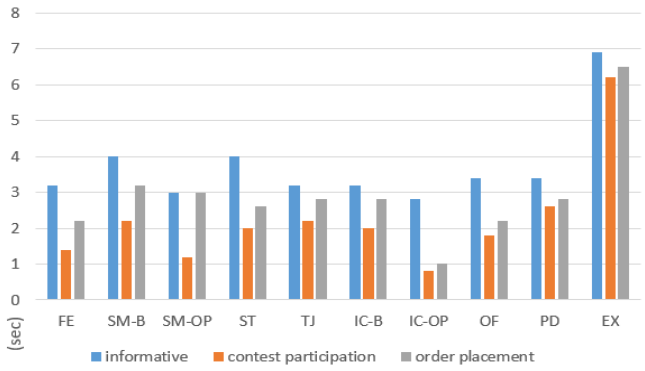

Fig. 5. Max Step Load Time

For the measurement of the average completion time the Samsung Galaxy S4 device was used. As it can easily derived from Fig. 6, the average completion time of the informative chatbot testing scenario was approximately 20 sec, whilst for the contest participation chatbot was 30 sec and for the order placement chatbot was 60 sec. Although it is observed a significant increase in the Max Step Load Time (Fig. 5), the overall completion time is not increased respectively at the extreme network scenario (Fig. 6). This is normal since the step execution time is significantly affected only when they are transmitted images or thumbnails.

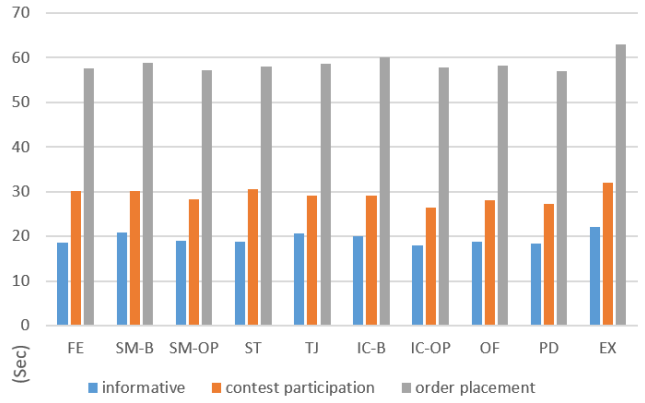

Fig. 6. Average Completion Time

The following metric in our experimental process was the completion rate. Generally, all the chatbots had excellent performance for this metric even in the extreme scenario.

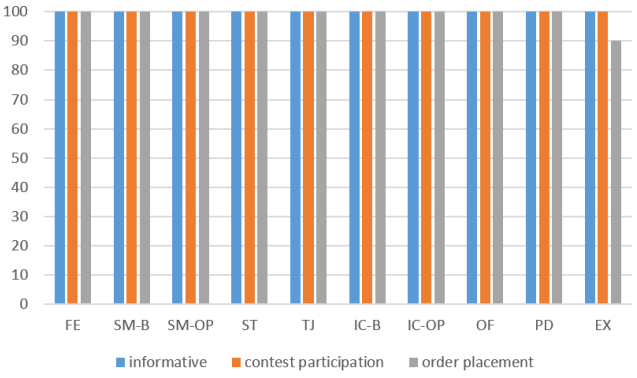

Fig. 7. Completion Rate

The only case in which there was not a successful result was after a keyboard loss in the order placement chatbot under the extreme case scenario. In this occasion the user had to restart the ordering process from the very beginning and every previous step/activity/selection had to be performed again, since all the previous activity had been lost.

Finally, the service stability metric from eq.1 was assessed and Fig. 8 depicts the results in realistic network conditions, where it is evident that high performance/stability values can be achieved with relatively small variation between each scenario. Even in the extreme network scenario the value of the service stability metric is high and only degraded about 10% in comparison to the rest cases.

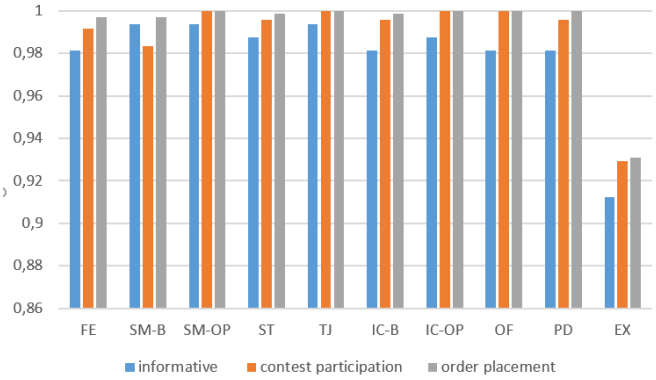

Fig. 8. Service Stability

B. *Mobile Devices Comparison*

Viber application is considered lightweight in terms of resource requirements and thus it is expected to perform satisfactorily in a relevant manner for all the mid-range smartphones that are LTE enabled. The only parameter that could affect the performance of the Viber application and consequently the Chatbot performance is the quality and the Radio Access Technology frond-end that each device is equipped with. The main device used for the experiment was the Samsung Galaxy S7 (SM-G930F). Part of the experimental process repeated with the Samsung Galaxy S4 for comparison purposes.

In the realistic network scenarios of the Table II the performance of the two devices was very similar. However, this was not the case for the extreme network scenario (EX), where performance variation was observed between the two mobile devices. Fig.9, Fig.10 and Fig.11 depict the performance comparison of the two mobile devices in the EX scenario for the three chatbot apps under test. In this network scenario the Samsung Galaxy S4 device had better performance in terms of the response time metric, but on the other hand the Samsung Galaxy S7 was more stable, with less network disconnects and fatal errors such as keyboard loss.

This is clearly depicted in Fig. 9, where the Max step response time is more than 50% increased in each chatbot application when the Samsung Galaxy S7 device is used.

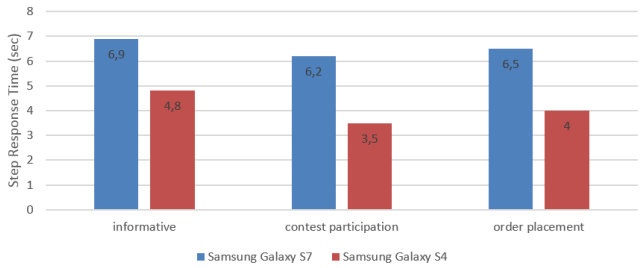

Fig. 9. Max Step Response Time

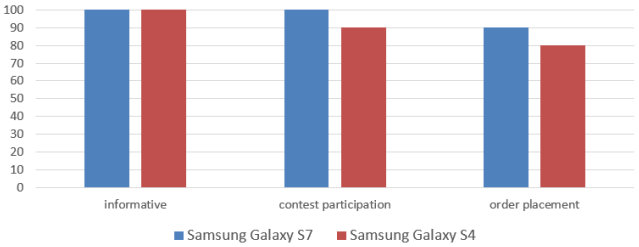

Fig. 10. Average Completion Rate

According to Fig. 10, the S7 device was disconnecting from the eNB more regularly in comparison to the S4. On the other hand the Samsung Galaxy S4 device also was more robust in image and thumbnail loss errors in comparison to S7 device.

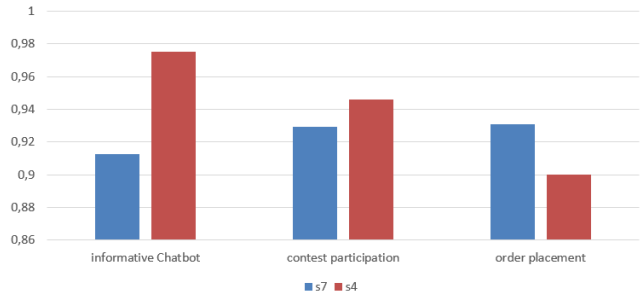

Fig. 11. Average Service Stability

According to Fig. 11, the Samsung Galaxy S4 seems to be more stable at the Informative Chatbot Application, whilst the Samsung Galaxy S7 is more stable at the order placement chatbot. This is explained from the fact that in the Informative and Contest Participation Chatbots the Samsung Galaxy S7 was suffering of more frequent image loss issues, but in the order placement chatbot the performance of Samsung Galaxy S4 was degraded due to more frequent keyboard loss incidents. So, it is evident that the hardware characteristics of each device play a major role in the achieved performance of the chatbot app.

VI. Conclusions

The paper focused on performing a set of experiments in the controlled environment that the experimental testbed of [9] offers, in order to test the various parameters that may affect the performance of a Viber chatbot app. In combination with the provided equipment, it was feasible during this experiment to assess the performance stability of Chatbot applications in various environments, under different conditions and through the use of versatile case scenarios. The results of the experiments assisted to gain additional knowledge on the chatbot apps and the quality of service that the end users experience. Overall, the paper shows that chatbot apps are robust even under extreme network conditions, where the service stability metric, that this paper introduces, reaches high scores (i.e. above 0.9). Finally, the use of different devices affects the performance of the chatbot apps, without however significant QoS degradation to be observed.

Acknowledgment

Part of this work has been conducted in the framework of the open call for experimenters of the H2020/GA no.688712 TRIANGLE project (5G-CHATBOT experiment).

References

[1] A. Argal, S. Gupta, A. Modi, P. Pandey, S. Shim and C. Choo, "Intelligent travel chatbot for predictive recommendation in echo platform," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2018, pp. 176-183.

[2] M. Shafi et al., "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice," in IEEE Journal on Selected Areas in Communications, vol. 35, no. 6, pp. 1201-1221, June 2017.

[3] B. Yang, W. K. Chai, G. Pavlou and K. V. Katsaros, "Seamless Support of Low Latency Mobile Applications with NFV-Enabled Mobile Edge-Cloud," 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, 2016, pp. 136-141.

[4] D. Tsolkas, E. Liotou, N. Passas, and L. Merakos, "A Survey on Parametric QoE Estimation for Popular Services", Elsevier Journal of Network and Computer Applications (JNCA), vol. 77, January 2017..

[5] Michail-Alexandros Kourtis; Harilaos Koumaras; George Xilouris; Fidel Liberal, "An NFV-based Video Quality Assessment Method over 5G Small Cell Networks", IEEE MultiMedia, Year: 2017, Volume: PP, Issue: 99, Pages: 1 – 1.

[6] A. Shah, B. Jain, B. Agrawal, S. Jain and S. Shim, "Problem solving chatbot for data structures," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2018, pp. 184-189.

[7] B. R. Ranoliya, N. Raghuwanshi and S. Singh, "Chatbot for university related FAQs," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, 2017, pp. 1525-1530.

[8] A. M. Rahman, A. A. Mamun and A. Islam, "Programming challenges of chatbot: Current and future prospective," 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, 2017, pp. 75-78..

[9] A. F. Cattoni et al., "An end-to-end testing ecosystem for 5G," 2016 European Conference on Networks and Communications (EuCNC), Athens, 2016, pp. 307-312.