ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20180212150241
Original scientific paper

An Analysis of Energy Efficient Data Transfer between Mobile Device and Dedicated Server

Predrag SIBINOVIĆ, Mladen NIKOLIĆ, Nemanja ILIĆ, Marko LAZIĆ

Abstract: This paper discusses research results with regard to energy-efficient transmission of serialised data between servers and mobile devices. A test environment was created in which the research authors primarily measured electricity consumption during communication between a mobile device and server. Numerical results were used to determine how well data serialisation was performed on a dedicated server and its effects on the power consumption of a mobile device. The time spent in data serialisation and the size of the serialised file were found to significantly influence energy consumption. Based on that fact, results have been used to create a mathematical model which was later introduced with functional forms. The main variables in those functional forms were time of serialisation and size of a serialised file. The data collected through this research has been used for an experimental API-CB Saver, which based on mathematical models chooses the most favourable manner of serialisation and compression in real time. The results collected during the tests show that the CBSaver-Api approach performs with greater energy efficiency than current techniques. Furthermore, with optimal selection of data serialisation type and compression level in real time the considered system shows better performance in power saving. According to the results, the API-CBSaver tests indicate the direction which one should take for the purposes of improving energy efficiency.

Keywords: data serialisation; energy efficiency; mobile devices; RESTful; mobile application development; data transfer; data compression

1 INTRODUCTION

Nowadays, one can witness that mobile devices have become an inseparable part of everyday life, primarily owing to social networks and faster information flow. According to a study, there are currently eight billion devices connected to the global network, and that number is bound to increase in the upcoming years [1]. The possibilities of these devices were improved on a daily basis, as well as the data transfer rate. Moreover, these devices possess increasing processor power and memory space. Some of the limiting factors were the autonomy and energy independence of devices.

Most mobile devices in everyday use can last one to two working days on a single battery charge. This period is significantly shorter if one is in motion and uses mobile Internet instead of WiFi.

Most applications on modern mobile devices were devised to establish communication with assigned servers. Such communication is reflected in the exchange of data with centralised relational databases. The data transfer process was mostly realised by means of RESTful [2] communication, while data which is most frequently object-oriented is transported serialised. Through the serialisation of list, series of objects from servers become converted into interfiles, and upon their reception on client devices, object lists were reconstructed from the obtained interfile. This form of transfer of object-oriented data is independent of the platform, and interfiles can be of either textual or binary kind [3]. Due to its simplicity of implementation, this method of data exchange has become widely used and dominant [4], [5].

This paper aims to analyse the impact of serialisation on energy efficiency of a mobile device. It aims to analyse performance of various serialisation systems through the process of REST communication and determine their relationship to the electricity consumption in a mobile device modem (3G and 4G). By means of experimental measurements and numerical methods a mathematical model was developed which, on the basis of the current connection parameters and the type of requested data, can compare energy efficiency of different serialisation types.

Since the battery of a mobile device is the main limiting factor to its lengthy use, software optimization methods are recommended for the purposes of increasing the autonomy of mobile devices. By means of this analysis, the authors suggest various methods of designing modern mobile applications, whereby with the help of a high API level, significant improvements in reducing power consumption can be achieved [3].

The paper is divided into six chapters. The second chapter considers previous research studies dealing with serialisation and energy efficiency measurement. The third chapter details the analysis of the energy consumption profile of the REST requests. The fourth chapter analyses energy efficiency performance of various serialisation types and determines mathematical prediction models by means of an experimental measurement. The fifth chapter comprises the summary of results obtained in the previous two chapters through a comprehensive mathematical model which is used as a basis for a simulator of energy efficiency. The sixth chapter represents an experimental energy efficient solution to the transfer of serialised data from the server to a mobile device, based on the previously obtained mathematical model.

2 THEORETIAL BASIS

Serialisation as a process has been widely described within scientific literature in which speed of algorithms and the file size were compared [6]-[9]. The authors of these studies provided comparations on Windows and Linux operating systems and in various programming languages. However, they did not analyse the direct impact of serialisation types on the battery. Bruno (2011) has not directly measured a smartphone battery consumption in his study [10], but has used an approximation programme Power Tutor, and the very influence was approximated. Likewise, the previous references studies have not studied analysis of data influence on the performance. Furthermore, within aforementioned studies, the authors

failed to resort to mathematical and numerical methods to describe the impact of serialisation in the current constellation of the connection between a mobile device and a web server. This paper has taken a special approach to a measurement. The obtained mathematical models could be a starting point for further optimisation in terms of the energy efficiency of mobile devices. Mathematical models can be used to model different scenarios in the use of applications, thus analysing energy efficiency in the early stages of software development.

In order to analyse the impact of serialisation on energy efficiency of mobile devices toward feasibility, the starting assumption should be that serialisation, as a process, can influence energy consumption during the communication. Likewise, the purposefulness of this research can be anticipated in the papers that study the direction and development of connection between mobile applications and the Internet. In most mobile applications resorting to RESTful communication (for example, HTTP Get, Post), a certain set of data from the assigned server is required [11]. As a result of the AVG Technologies study that quest the plan and development of mobile systems for the year 2016, using anonymous sampling of three million users, the authors managed to recognize applications that have the greatest impact on battery consumption [7-11]. By analysing this research it can be easily concluded that applications which mostly use communication with the server have the greatest impact on battery consumption. Likewise, by analysing the API and technologies through which applications from this list were realised one can discern the use of RESTful communication and JSON data serialisation type. If, apart from this assertion, different views and earlier studies are taken into account, it becomes apparent that mobile Internet communication is in the second place in the list of energy consumption, immediately after the system is displayed [12]-[17]. Considering the aforementioned facts, this study began with observing the energy profile of a modem in the course of a typical REST request.

3 MODELLING OF AN ENERGY PROFILE OF A REST REQUEST

Based on analysis of available studies, hardware units of mobile devices as well as operating systems and programme instructions, one encounters two issues which require resolution as a prerequisite for a valid analysis of the battery consumption [15]. The first request is to monitor energy consumption during each REST request instruction. The second deals with mapping the consumption of each programme instruction in order to analyse energy efficiency of the process. Aside from the necessity of obtaining answers to these two issues, it is necessary to devise a very precise manner of measuring energy consumption. Important factors that could influence battery consumption during REST communication are as follows: the size of transferred data, the frequency of transfer and the server performance. Battery consumption could be measured in two ways, by means of a sensor built into the device or through an external sensor. Battery consumption could be influenced by other groups of processes and conditions. These processes could not be compensated or isolated, it follows that the use of an external sensor is more precise and relevant. Likewise, one should bear in mind that battery consumption is a non-linear process [18]. Fig. 1 presents a schematic diagram of the equipment for measuring the intensity of a 3G modem power intensity.

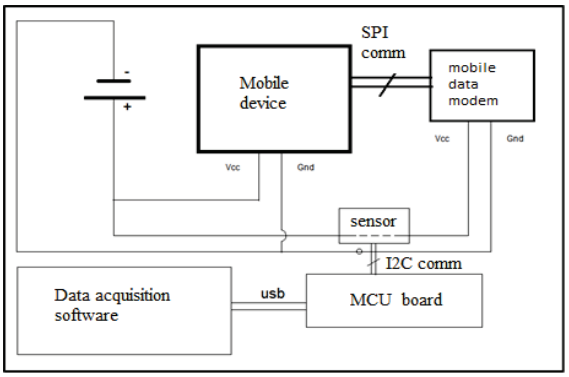

Figure 1 A schematic diagram of the measuring equipment

Aside from already emphasised problems regarding physical measurement and suggestions for potential solutions presented in Fig. 1, a software test environment has been designed. A test programme implies the entirety comprising the set of instructions on the web server, test applications on a mobile device, a suitable data set, as well as the execution of data transfer from the server to mobile devices and vice versa. The role of a test programme on a mobile device was the possibility of setting certain combinations of REST communication parameters. Thereafter, a test programme on a mobile device initiated a REST communication with the defined server. Along with sending REST requests to the server, the previously assigned data were also transferred, and those data were related to a desirable size of data which were retrieved from the server, as well as the server processing time. The server processing time was a server request delay simulation from the moment when the server accepts REST request to the moment of response. Namely, server shall delay the response for the given time by stopping the programme thread in which it accepted the mentioned connection. Test programme is also present on the web server. Based on received request from the test programme of a mobile device, a certain number of rows was selected from the database and returned as a response. Also, based on parameters, regarding the simulation of processing time, it simulates an imaginary processing of predefined duration. The test programme on a mobile device has a task to write in the local CSV (comma separated value) file the time from the beginning of the test, the code of the current operation which it performs and the value read from the microcontroller which, by means of a sensor, measures the current intensity in the power lines of a 3G modem. The acquisition takes place every 5 ms.

The segments of the lifespan of a single REST communication are as follows: the time of connection to the server (t_c), the time of sending the request (t_p), the time of waiting for the reply (t_l) and the time of data reception (t_d). A schematic diagram of segments in the consumption profile is shown in Fig. 2.

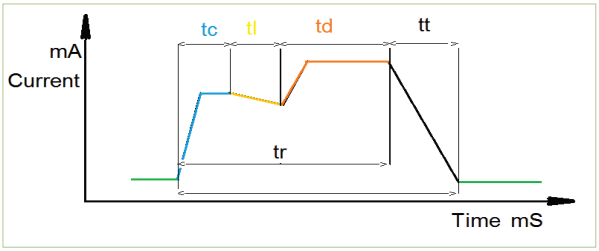

Figure 2 Segmented presentation of an energy profile of a REST request

The total energy consumption during a REST request can be presented by the following mathematical model:

$$E = (a_c \cdot t_c + a_p \cdot t_p + a_l \cdot t_l + a_d \cdot t_d) \cdot U + E_t \quad \text{(Ws)} \qquad (1)$$

Whereby: $t_d = \frac{D_s}{V_d}$, $t_p = \frac{H_s}{V_u}$, a_p - the average current intensity in the modem power supply during sending requires a server; a_c - average current intensity in modem power lines during connection; t_c - the time spent during the connection; H_s - the size of the POST header in bytes; V_u - the rate of sending data in bytes per second; a_l - average current intensity in the modem power lines while waiting for server response; t_l - the time spent on waiting for the reply from the server; a_d - average current intensity in modem supply lines during data acquisition; D_s - the size of data in bytes received from the server; U - supply voltage; V_d - the rate of receiving data in bytes per second; E_t - the energy consumed by a modem after communication break.

Fig. 3 illustrates a recorded energy profile of two identical REST requests submitted in different moments but aligned in terms of temporal axis. Furthermore, based on aforementioned measurement methodology, the graph (Fig. 3) clearly presents the segmentation of phases significant for the lifespan of a REST request.

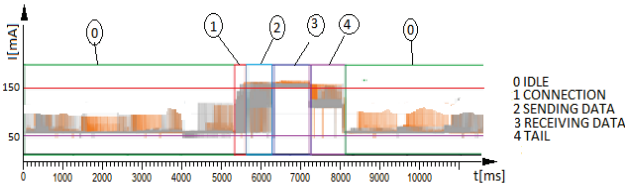

Figure 3 The results of measuring two REST profiles with marked segments

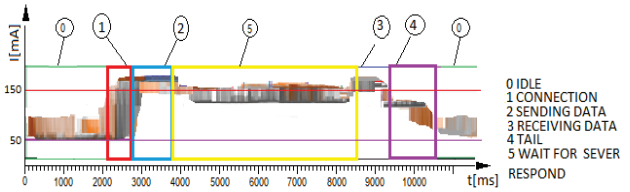

Figure 4 The results of measuring two REST profiles with increased server delay times

Fig. 4 represents a recorded profile of a test during which the time necessary for the server to process the request was drastically increased to 5 s in order to be able to analyse and select more precisely the energy feature in certain segment (5). Normally, this interval should be less than 500 ms (Fig. 6).

Based on measurements and average values in selected segments, the Eq. (1) follows:

$$E = \left(0.123\text{A} \cdot t_c + 0.163\text{A} \cdot (t_p + t_d) + 0.14\text{mA} \cdot t_l\right) \cdot \cdot 3.7\text{V} + 2.875\text{Ws} \quad \text{(Ws)} \qquad (2)$$

4 ANALYSIS OF THE SERIALISATION PROCESS

Considering the previous analysis of a REST communication, it has been concluded that server processing time and data reception time (t_l, t_d) do influence the energy consumption of a device, whereby they were correlated with the process of serialisation. The process of serialisation can influence both of these data. This is due to the fact that raw data, which is serialized, obtains additional size due to special characters, whose serialization process is used to transform raw data into a serial file. Likewise, there are compression techniques which can reduce the volume of data that have been transferred. All these procedures require the time that is necessary for the server to pack the data and send them to the client. It is the time of server delay which is analysed in the measurements of the energy profile of a REST request and it has been proved that it influences the consumption (t_l). When all this is taken into consideration one reaches the conclusion that it is necessary to ascertain two dimensions of performances of a serialisation process and by means of numerical methods reach mathematical equations which would describe them.

During these tests we focused on the combination of XML, JSON, PHP serialisation, YAML and MsgPack, as well as on a gzcompress compression package. The role of the test was to ascertain the efficiency of each command on the tested server. The aim of these tests was to establish mutual dependence between the size of data and performances of different programme packages.

Fig. 5 shows the results of a comparative examination of properties of five different serialisation schemes. The figure shows the size ratios of serialised files, as well as the time necessary for their serialisation. The graphic shows that JSON and MsgPACK had the best performance among all tested schemes. According to this XML, JSON, PHP serialisation and YAML were rejected as a bad candidate for energy efficient data transfer. The testing was continued by adding a *gzip* compression to JSON and MsgPACK, and then performances of each of the nine compression levels were examined through the size of data and time of execution.

Fig. 6 presents graphics that show the impact of a *gzip* compression on the files created through the JSON and MsgPACK serialisation of performances: data size and time of processing. The graphic shows that the compression has a bigger impact on JSON, which is textual, than on MsgPACK, which is binary.

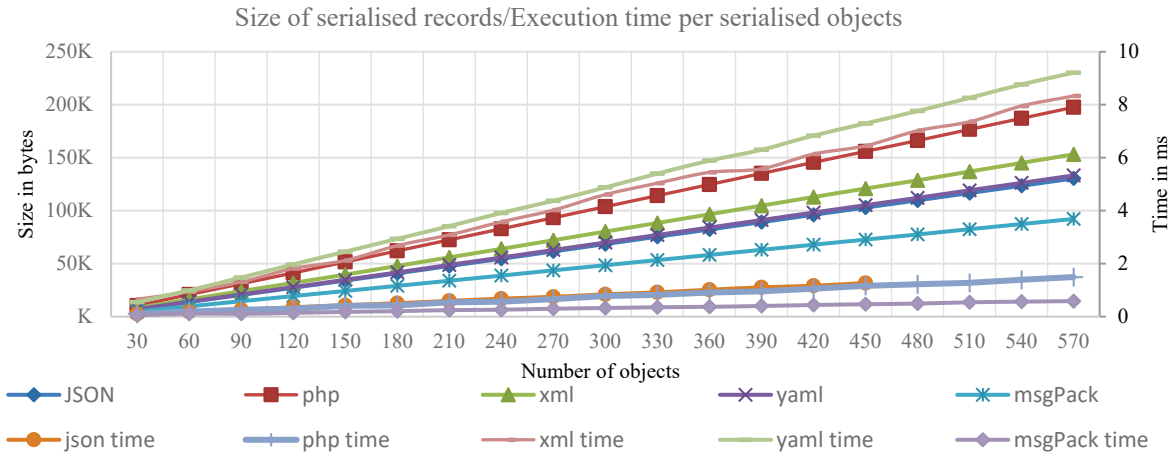

Figure 5 Comparative trend diagrams of analiysed serialisation schemes

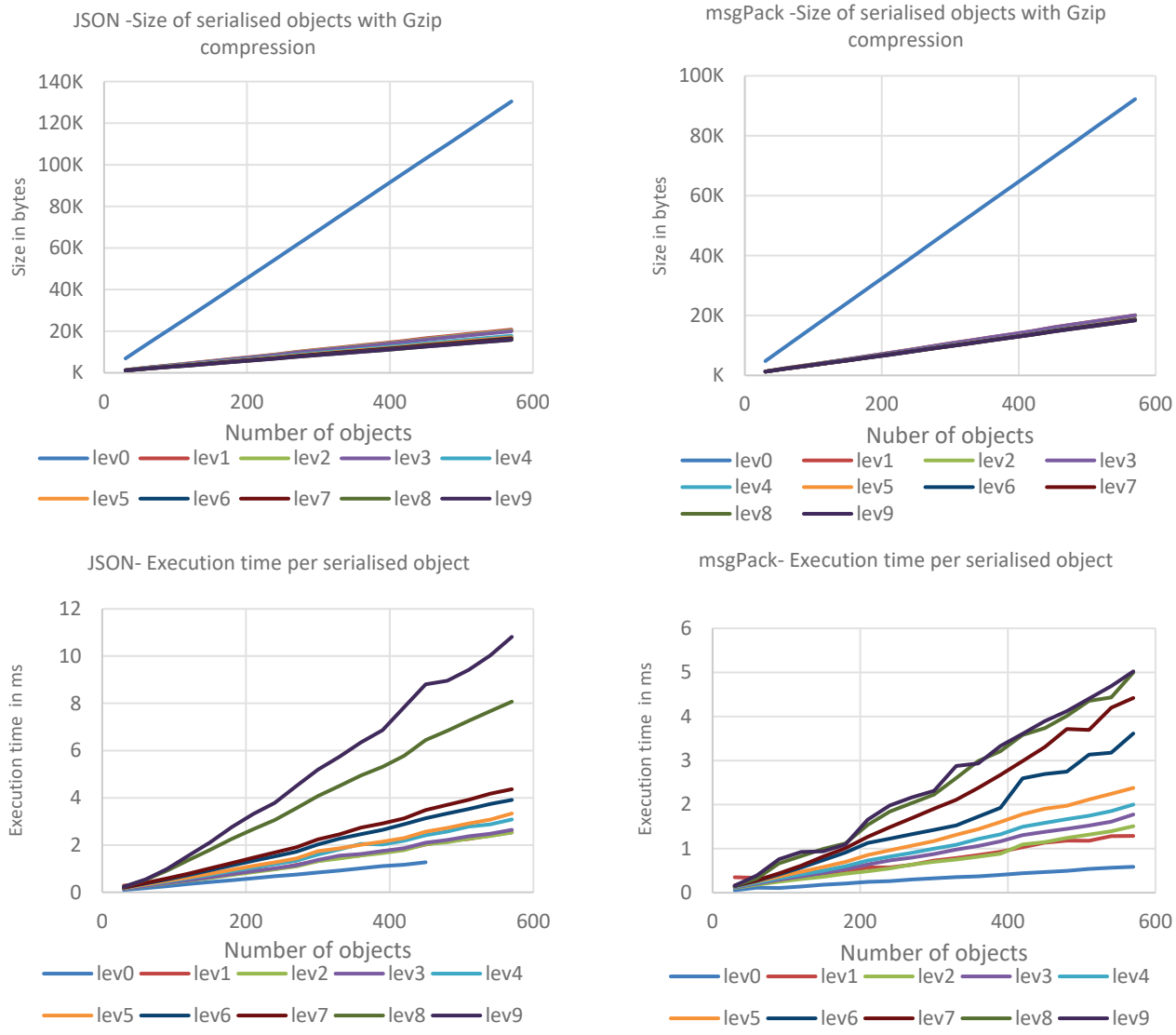

Figure 6 The comparison of JSON and MsgPACK schemes with an added *gzip* compression per levels

5 A COMPREHENSIVE MATHEMATICAL MODEL

By means of serialisation analysis the authors obtained a mathematical model which was embedded within the previous consumption model, so much so that if we take Eq. (2) from the previous chapter and use simple substitution to introduce the obtained models in a general form, we acquire the following:

$$E = \left[0.123\text{A}\cdot t_{\text{c}} + 0.163\text{A}\cdot\left(t_{\text{p}} + F_{\text{d}}(D_{\text{s}}, V_{\text{s}})\right) + 0.140\text{A}\cdot F_{\text{l}}(D_{\text{s}})\right]\cdot \\ \cdot 3.7\text{V} + 2.875\text{Ws} \quad (\text{Ws}) \tag{3}$$

Whereby: $F_d(D_s, V_s)$ the time necessary for the required amount of data to reach the client; $F_l(D_s)$ the time necessary for the server to prepare the data which are to be sent to the client:

$$F_l(D_s) = F_T^{gzi}\left[G_z, F_S^{gzi}\left(G_z, F_S^{ser}(S_t, D_s)\right)\right] + \\ + F_T^{ser}(S_t, D_s) \tag{4}$$

$$F_d(D_s, V_s) = \frac{F_S^{gzi}\left(G_z, F_S^{ser}(S_t, D_s)\right)}{V_s} \tag{5}$$

Whereby: F_T^{gzi} - the function which calculates the time necessary to compress the data depending on the type of compression and data size; F_T^{ser} - the function which calculates the time necessary to serialise the data depending on the data size and type of serialisation; F_S^{gzi} - the function which calculates the data size after the compression depending on compression type and data size; F_S^{ser} - the function which calculates the data size after the serialisation depending on data size and type of serialization; G_z - the compression level; S_t - the serialisation type

$$F_T^{gzi}(G_z, D_s) = \begin{cases} D_s \cdot 8\times10^{-9} - 1\times10^{-6}, & G_z = 1 \\ D_s \cdot 1\times10^{-8} - 3\times10^{-5}, & G_z = 1 \\ D_s \cdot 1.02\times10^{-8} - 2.7\times10^{-5}, & G_z = 3 \\ D_s \cdot 1.08\times10^{-8} - 2.3\times10^{-5}, & G_z = 4 \\ D_s \cdot 1.12\times10^{-8} - 2.9\times10^{-5}, & G_z = 5 \\ D_s \cdot 2.1\times10^{-8} - 2.3\times10^{-5}, & G_z = 6 \\ D_s \cdot 3.1\times10^{-8} - 5.3\times10^{-3}, & G_z = 7 \\ D_s \cdot 5.7\times10^{-8} - 4.3\times10^{-3}, & G_z = 8 \\ D_s \cdot 6.98\times10^{-8} - 5.1\times10^{-3}, & G_z = 9 \end{cases} \tag{6}$$

$$F_S^{gzi}(G_z, D_s) = \begin{cases} D_s \cdot 0.1564 + 374.7, & G_z = 1 \\ D_s \cdot 0.1524 + 349.2, & G_z = 1 \\ D_s \cdot 0.1499 + 380.3, & G_z = 3 \\ D_s \cdot 0.1332 + 422.2, & G_z = 4 \\ D_s \cdot 0.1276 + 409.7, & G_z = 5 \\ D_s \cdot 0.1244 + 353.2, & G_z = 6 \\ D_s \cdot 0.1221 + 359.4, & G_z = 7 \\ D_s \cdot 0.1184 + 369.3, & G_z = 8 \\ D_s \cdot 0.1183 + 349.2, & G_z = 9 \end{cases} \tag{7}$$

$$F_S^{ser}(S_t, D_s) = \begin{cases} D_s \cdot 229 - 179, & S_t = 1 \text{ json} \\ D_s \cdot 234.11 - 183.2, & S_t = 2 \text{ php} \\ D_s \cdot 234 + 179.2, & S_t = 3 \text{ xml} \\ D_s \cdot 0.1499 + 349.2, & S_t = 4 \text{ yaml} \\ D_s \cdot 176.1 + 184.2, & S_t = 5 \text{ msgPack} \end{cases} \tag{8}$$

$$F_T^{ser}(S_t, D_s) = \begin{cases} D_s \cdot 3\times10^{-6} + 4\times10^{-6}, & S_t = 1 \text{ json} \\ D_s \cdot 4\times10^{-6} + 3\times10^{-3}, & S_t = 2 \text{ php} \\ D_s \cdot 2\times10^{-5} + 4\times10^{-6}, & S_t = 3 \text{ xml} \\ D_s \cdot 3\times10^{-5} + 2\times10^{-2}, & S_t = 4 \text{ yaml} \\ D_s \cdot 7\times10^{-7} + 4\times10^{-6}, & S_t = 5 \text{ msgPack} \end{cases} \tag{9}$$

In the case of mathematical models obtained in this fashion it was possible to perform simulations and comparisons of energy efficiency of the systems analysed for serialisation and compression. The mathematical model developed in this paper can be used in future as a guideline for mobile application design. For special purpose data transfer it can help in choosing the serialisation scheme and compression level.

5.1 Simulator

By virtue of the obtained models it was possible to analyse the impact of a serialisation scheme and compression level on the energy efficiency of a client´s device in the very early stages of application design. By using a MATLAB programme package simulator we have simulated comparative scenarios for all of the aforementioned serialisation types and *gzip* compressions. Accordingly, Fig. 7 presents comparative consumption diagrams depending on the data flow rate and the amount of objects requested from the server. The used data volume during the simulation varies from 10 to 400 objects per request, while the range from 0.1 to 8 Mbps was simulated for the data flow rate.

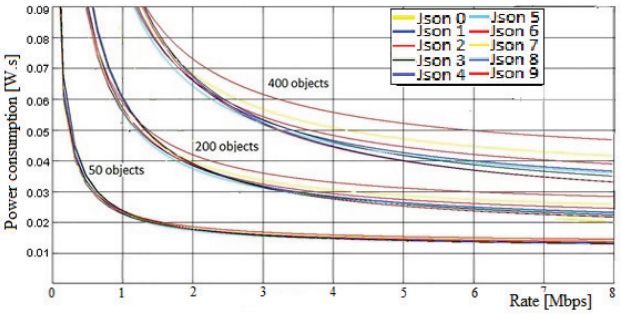

Figure 7 Comparative presentation of energy consumption depending on the speed of flow and amount of data

These simulations pointed that certain combinations of serialisations and *gzip* compressions were more efficient in particular zones than other solutions. Zones were defined by data transfer rate and number of requested objects. During data transfer, the modem continuously operates in a state of high energy consumption, the time that was spent in this state was related to the ratio of the data size and the rate of data transfer. An increased data transfer rate reduces the amount of time within which the modem operates at a high energy state. This was correlated with power consumption.

Fig. 8 shows the relationship between a JSON serialisation and MsgPack with all *gzip* levels of compression, but without the compression itself.

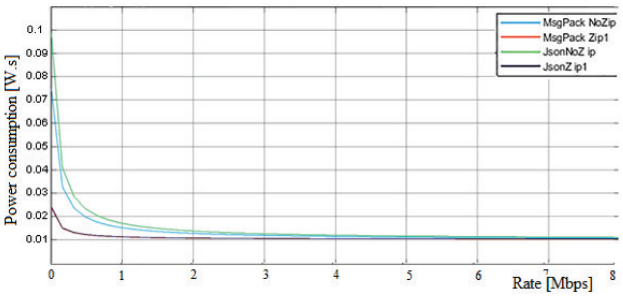

Figure 8 The relationship between the consumption of JSON and MsgPack schemes at different levels of compression

Based on an observation that there was no linearity and a super dominant system, the following question imposes itself: can one reach an optimal energy efficient solution for the current conditions? The answer to this question was provided by a new simulation which aims to present currently the best solution offered by JSON and MsgPack serialisation. The results are presented in Fig. 9 where one can observe that seemingly optimal solutions offered by JSON (with compression) and MsgPack (with compression) cannot generally be considered optimal. A concrete optimal solution offered by JSON was only optimal in the zones of lower speed, while that offered by MsgPack in the zones of higher speed. One should also add that the increase in the number of requested objects moves this point of intersection to the left.

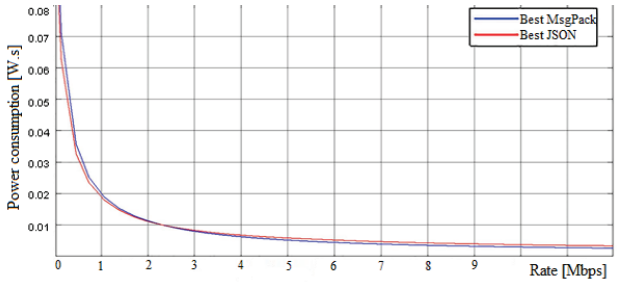

Figure 9 The best JSON and MsgPack solution

5.2 The Data Impact

During the research for the purposes of testing we have resorted to a database comprising varying data type, i.e., numerical and textual fields within the object were equally presented. It has been noted that mathematical models vary depending on the construction of objects, so that the application of the aforementioned simulator can be run only on approximately organised data sets. It has also been noted that a MsgPack scheme corresponds to complex types whose fields are other complex objects, unlike JSON scheme. Furthermore, some lose their advantages if the data which are being serialised are exclusively of textual nature (e.g. MsgPack). A *gzip* compression was applicable and provides positive effects mostly in textual data. In MsgPack only the first level of compression was applicable, while higher levels of compression were counterproductive.

6 CBSaver API

A system for an optimal choice of communication in real time based on a CBSaver algorithm, primarily applied to applications which require data regarding the geographical location of a device.

Based on the aforementioned research studies, the results of which were presented in this paper, the authors came to an idea regarding an energy efficient transference system of object-oriented data between a server and mobile device. The concept was based on mathematical models obtained in the previous research. The operating system uses the current parameters in the course of communication on the basis of the most energy efficient manner depending on the data flow rate and the volume of data which is being transferred.

The characteristic of the mentioned system was a changeable data volume required by the server, i.e., it is not permanent, bearing in mind that these systems were used in motion. For example, in a centre of large city there are ten times more objects of interest than in a small rural town. Likewise, the internet connection and speed vary depending on the location, number of users at the base station and weather conditions. These variable conditions are key factors with respect to the optimal energy efficiency of a device. One should take into consideration that among application users (if it is an application intended for a wider audience) there are a certain number of users who possess devices which can perform at higher internet speeds, as well as those whose devices are up to two generations older than of those aforementioned. Based on predictions by the CISSCO Company (CISSCO, 2017) the share of each technology segment can be seen in users. This means that at the same geographical location and at the same base station there is the same percentage of users who possess 2G, 3G and 4G devices. This implies that even under the same conditions their applications cannot be equally efficient with respect to energy.

All these previous considerations and observations, for the purposes of creating an experimental energy efficient application, lead us to a conclusion regarding the conditions for which one should create an energy efficient algorithm:

- Variable data volume depending on a geographical location;
- Variable data transfer rate depending on a geographical location;
- Unequal maximum data transfer rate depending on the type of a user's device.

From the previous research studies into energy efficiency, published tests and developed mathematical models, one can clearly discern that one cannot create a data transfer system which at a given moment is guaranteed to operate at an optimal energy level, regardless of the degree of compression or selected type of serialisation. When one finally takes into consideration the diversity of devices, a single solution imposes itself: to design a hybrid system which, on the basis of data volume being sent to users and flow rate of said data, chooses an energetically optimal data transfer mode.

To put it simply, if one possesses mathematical models of servers, the volume of data which is sent at a particular moment, as well as the speed of communication between a client and the server, one can determine the energy optimal data transfer mode. This applies to each individual request sent to the server.

Aside from mathematical models which are specific for the executing server, this sort of API possesses its own executive part which has an algorithm for evaluation and choice of work mode. The API itself can be considered a software package for the existing methods already in use. On the basis of the evaluating algorithm, the API determines the levels of data compression and the type of serialisation.

The evaluation algorithm comprises three steps:

- Measuring the size of data which is to be sent;
- Calculating the time of data transfer towards the client by means of mathematical models for each scenario, using the speed of transfer obtained from the client and the size of data which is to be sent to the client;
- Selection and execution of the scenario which provides the least amount of the previously calculated transfer time.

6.1 The Client Side of API

Although the majority of tasks in request processing are at the expense of the server, it cannot perform successfully without co-operating with the client. It has already been emphasised that clients play a significant role in informing the API on the server regarding the flow rate. It does not possess mathematical models nor does it make decisions. In other words, it performs a typical REST request, whereby it provides the server with necessary information in headers, and upon receiving and sending the request, it creates a local base with the latest calculated transfer speeds.

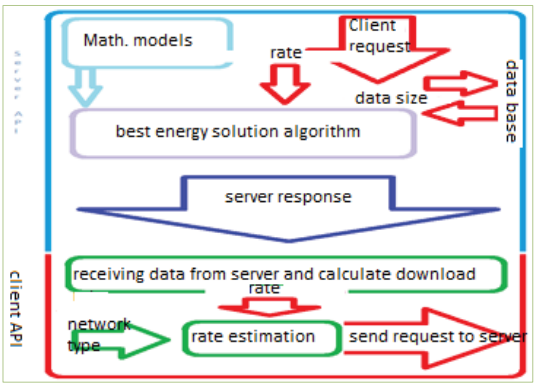

Figure 10 A map of the suggested API CBSaver

6.2 The Entirety of an API Model

Fig. 10 shows a diagram within which clients, on the basis of an estimated speed at their disposal, send the necessary data to the server during the request. The server receives the request and API data (i.e. client's speed) and measures the size of data which needs to be transported back. On the basis of mathematical models specific to the server where the query is executed, and by means of these two parameters, the best data transfer scenario is being calculated and sent back. On the other hand, upon receiving data the client determines the data transport speed and keeps it in the local base. That speed shall be used for some of the upcoming requests. This manner of determining the data flow rate has been selected because it does not require additional resources for test messages to ascertain the speed of communication, but estimates it on the basis of elementary communication with the server with a one message delay.

6.3 Testing

In order to test the new communication system in a real world environment, an application has been designed for the purposes of this research and a server has been formed. The authors have formed a test system which does not differ from a real commercial application which would resort to the aforementioned API from the standpoint of data transfer communication and user exploitation. The differences which are obvious with respect to a real commercial application are data credibility, graphic interface and other options important to the user in order for an application to be user-friendly.

The entire test has been designed so that two test software packages were formed for identical purposes and functionality, the difference being that one solution resorts to a CBSaver API, and the other to a classical REST system with JSON serialisation, which is the case with the majority of modern applications. On the basis of its geographical position obtained from its GPS receiver, the role of the application is to request location data from the server. The application requires data on the basis of moving a mobile device and the time interval. In other words, if a mobile device is in motion, new data is required from the server based on the new position for every 50 metres of motion. If a mobile device is stationary, the data was required in 90 second intervals. This is the basis for applications used as navigation systems, auxiliary systems in space search, warning systems, and even communication systems. Furthermore, this manner of communication is the backbone of future autonomous systems for public transport, as well as in specialised industrial systems.

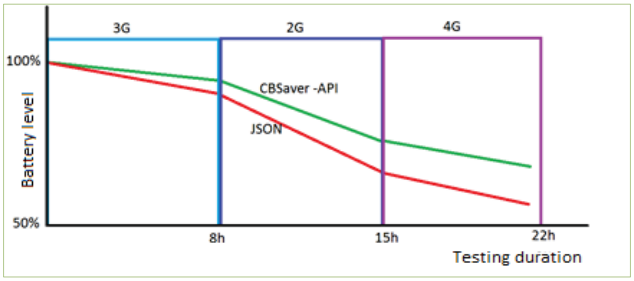

Figure 11 The results of comparative measuring of a CBSaver API and a classical JSON manner of transfer

The test provides as realistic use of the application in everyday life as possible. Therefore, the application was used every day during a seven day period on several different devices. The devices were specially prepared for the testing. All unnecessary applications using the Internet connection had been removed and they operated as background services. On each day of testing, before the beginning of the test, the devices were charged to their maximum battery capacity. Upon finishing the daily testing the value of battery capacity was noted down. Likewise, the device battery management kept the values of the amount of battery capacity spent during the day. After seven days of testing, the completely same manner of testing was performed on devices with an application which does not possess a CBSaver API.

Table 1 A comparative overview of a battery level in percentages on tested devices with and without CBSaver API

Day	Dev A CBSaver	Dev A JSON	Dev B CBSaver	Dev B JSON	Dev C CBSaver	Dev C JSON
1	65	52	60	55	40	10
2	70	62	78	72	52	15
3	45	20	40	36	63	35
4	80	87	70	68	76	43
5	66	70	57	52	49	44
6	19	2	12	0	12	8
7	20	18	10	1	3	0

Aside from the above mentioned tests which indicate that a CBSaver plays a major role in the energy efficiency of a client´s device, a strictly controlled test was performed on a device which tested both systems, so that tests would be performed under the same conditions. Two days in a row, at a designated time of day, an application was initiated so that it performed for 8 hours in a 3G mode, the next seven hours the devices operated only in a 2G mode, and the last seven hours they performed in a 4G mode.

7 RESLUTS

Fig. 11 illustrates the results of the controlled test. It can be seen that the CB Saver API application has much better power efficiency than the JSON format application. In the 2G transfer rate segment, this difference was seen to be the greatest. From there, as the transfer rate gets higher, the difference observed got smaller. This was in line with the results from the simulation shown in Fig. 8.

Generally the power consumption was negatively correlated with the data transfer rate. At low transfer rates the power consumption was larger and at high transfer rates it was smaller. This could explain why the difference between CB Saver API application and the JSON format application was bigger in the low transfer rate segment.

In the main test (the daily test) three classes of devices were tested:

- Class A, devices that have 4G capabilities;
- Class B, devices that have 3G capabilities; and
- Class C, devices that are limited to 2G data transfer rates.

The biggest difference in the energy consumption was found in the Class C category devices, followed by Class B category devices (Tab. 1). This finding was expected and it is in-line with the previous conclusion form the controlled test.

In the Class A devices, day four and five of the test, JSON format application was seen to have better power efficiency than the CS Saver API application. This could be explained through errors in the data flow rate approximation in the CB Saver API application. Class A devices have bigger data flow rate range than Class B and C devices. So errors in the data flow rate prediction could be more frequent than in Class B and C devices. Also physical moving of the device and the condition of the network could affect the frequency of the occurrence of this error.

Following on for another five days of testing, the CB Saver API application achieved better power efficiency than the JSON format application.

This research was based on the fact that data exchange with a dedicated server and mobile device mostly spend energy in the cellular modem. During the research, both test applications (CBSaver and JSON) were analysed additionally by Power Tutor application. During 30 minutes test of each application was recorded the cumulative time of data transfer, approximated energy consumed by CUP and cellular modem. Results showed that CBSaver app spent 30.7 s in communication with a server, total energy consumed by CPU was 4 J and total energy consumed by cellular modem was 28.6 J. The JSON format application spent 37.8 s in server communication, total energy of CPU was 2.7 J and 35 J was consumption of cellular modem. This confirmed that most of the energy was spent during data transfer and the duration of transfer affects power consumption more than deserialization and decompression performed on mobile device.

8 CONCLUSION

The research conducted in this report aimed to determine the existence of a correlation between the electricity consumption of mobile devices, and the type of data serialisation the applications running on these devices receive from the server. Upon successful measurement, mathematical models were developed using which it is possible to more thoroughly analyse the correlation. Using these mathematical models, simulations were run for the purposes of optimising the energy consumption properties of the communications between servers and mobile devices. On the basis of those simulations a CB Saver algorithm was developed recommended for implementation to reduce the power consumption. Tests were conducted in which the CB Saver API had been used as a method for communication in an application which required specific information of interest for that region on the basis of position. These tests showed that, under certain circumstances, this API is more energy efficient in comparison to traditional methods currently in use. Furthermore, based on this research, more solutions could be predicted in the field of software engineering which offer greater energy efficiency than the classical ones.

Acknowledgment

We would like to express our gratitude to Professor Miroslav L. Dukić for the valuable comments and suggestions that added to the quality and comprehensiveness of this article.

8 REFERENCES

[1] CISSCO. (2017). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper*. Evolving toward Smarter Mobile Retrieved from https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html#EvolvingtowardSmarterMobile

[2] Pautasso, C., Zimmermann, O., & Leymann, F. (2008). "Restful web services vs. 'big' web services: Making the right architectural decision". *Proceedings of the 17th International Conference on World Wide Web.* https://doi.org/10.1145/1367497.1367606

[3] Jiri Soukup, P. M. (2014). *Serialization and Persistent Objects*. Berlin: Springer. https://doi.org/10.1007/978-3-642-39323-5
[4] Kuinam J. Kim, C. G. (2016). A continuous playing scheme on RESTful web service. *Cluster Computing*, 379-387. https://doi.org/10.1007/s10586-015-0520-2
[5] Tomé, C. R. (2011). Mobile Application Webservice Performance Analysis: Restful Services with JSON and XML. *Communications in Computer and Information Science*, vol 220. (162-169). Berlin: Springer. https://doi.org/10.1007/978-3-642-24355-4_17
[6] Queirós, R. (2014). JSON on Mobile: is there an Efficient Parser? *OASIcs. SLATE*. 2014 (93-101). Bragança, Portugal: OASIcs.
[7] Sebastian Bittl, A. A. (2015). Performance Comparison of Data Serialization Schemes for ETSI ITS Car-to-X Communication Systems. *International Journal on Advances in Telecommunications*, 48-58.
[8] Maeda, K. (2011). Comparative Survey of Object Serialization. *International Journal of Computer and Information Engineering*, 1488-1493.
[9] Audie Sumaray, S. K. (2012). A comparison of data serialization formats for optimal efficiency on a mobile platform. *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication* (Article No. 48). Kuala Lumpur: ACM New York. https://doi.org/10.1145/2184751.2184810
[10] Bruno Gil, P. T. (2011). Impacts of data interchange formats on energy consumption and performance in smartphones. *OSDOC '11* (1-6). Lisboa: ACM New York. https://doi.org/10.1145/2016716.2016718
[11] Martin Garriga, C. M. (2016). RESTful service composition at a glance: A survey. *Journal of Network and Computer Applications*, 32-53. https://doi.org/10.1016/j.jnca.2015.11.020
[12] P. R. A. (2016). *AVG Technologies Android App Performance Report*. Emeryville: AVG PR.
[13] Abhinav Pathak, Y. C. (2012). Where is the energy spent inside my app? Fine grained energy accounting on smartphones with eprof. *7th ACM euroean conference on Computer Systems* (29-42). Bern: ACM. https://doi.org/10.1145/2168836.2168841
[14] Abhinav Pathak, Y. C.-M. (2011). Fine-grained power modeling for smartphones using system call tracing. *Proceedings of the sixth conference on Computer systems* (153-168). Salzburg: ACM. https://doi.org/10.1145/1966445.1966460
[15] Niranjan Balasubramanian, A. B. (2009). Energy consumption in mobile phones: a measurement study and implications for network applications. *IMC '09* (280-293). Chicago: ACM. https://doi.org/10.1145/1644893.1644927
[16] Rafael Pérez-Torres, C. T.-H.-Z. (2016). Power management techniques in smartphone-based mobility sensing systems: A survey. *Pervasive and Mobile Computing*, 1-21. https://doi.org/10.1016/j.pmcj.2016.01.010
[17] Elhadj Benkhelifa, T. W. (2016). Energy Optimisation for Mobile Device Power Consumption: A Survey and a Unified View of Modelling for a Comprehensive Network Simulation. *Mobile Networks and Applications*, 575-588. https://doi.org/10.1007/s11036-016-0756-y
[18] Jean Araujo, R. M. (2017). Impact of capacity and discharging rate on battery life time: A stochastic model to support mobile device autonomy planning. *Pervasive and Mobile Computing*, 180-194. https://doi.org/10.1016/j.pmcj.2016.10.002

Contact information:

Predrag SIBINOVIĆ, MSc Assistant Professor
Coresponding author
College of Technical and Technological Sciences in Kruševac,
Kosančićeva 36, 37000 Kruševac, Serbia
E-mail: psibinovic@gmail.com

Mladen NIKOLIĆ, PhD, Professor
College of Technical and Technological Sciences in Kruševac
Kosančićeva 36, 37000 Kruševac, Serbia
E-mail: mladennikolic2603@yahoo.com

Nemanja ILIĆ, PhD, Professor
College of Technical and Technological Sciences in Kruševac
Kosančićeva 36, 37000 Kruševac, Serbia
E-mail: 1vodenicar@gmail.com

Marko LAZIĆ, Dipl. Ing.
Lucky Feature
Patrijarha Čarnojevića 20, 37000 Kruševac, Serbia
E-mail: marko.lazic@panrobotics.com