SMART CITY WEBGIS APPLICATIONS: PROOF OF WORK CONCEPT FOR HIGH-LEVEL QUALITY-OF-SERVICE ASSURANCE

A. Noskov

Institute of Geography, Heidelberg University, Germany - a@n-kov.com

KEY WORDS: Quality of WebGIS Services, Bitcoin's Proof of Work, Spatial Information Theory, User Activity Monitoring, User Requests Balancing, Quantity of Spatial Information

ABSTRACT:

In the frame of smart city initiatives, map-based web applications (WebGIS) are distinguished by the complexity of client-side implementation and high load on web servers. Web-map applications provide non-typical web content usually visualized by JavaScript code. For such applications, higher-level advanced approaches to quality of service (QoS) assessment are required. In this paper, a method based on the estimation of users' interest in a web page and the information quantity provided by a web page is introduced. In order to implement it, a proof of work (PoW) concept is applied for verification of real users. Moreover, a novel metric is introduced to calculate the information quantity provided by web pages. PoW is used for verification of real users and distinguishing them from bots. Additionally, it can be utilized for balancing of web server load. For monitoring of web pages, an image-based approach is introduced. Web pages are rendered into raster images. A number of metrics describing an image are calculated. This approach allows users to compare and track various web pages, including rich web applications providing complex WebGL content. Several web services utilize the developed solutions. Analysis of harvested data proves the effectiveness of the proposed solution.

1. INTRODUCTION

In the frame of smart city initiatives, map-based web services (WebGIS) are distinguished by the complexity of web applications and high load on web servers. Several WebGIS applications are developed in the frame of the WeGovNow (We Government Now) project funded by the EU Horizon-2020 program. WeGovNow (Boella et al., 2018) is a research and innovation action focused on civic participation in local government aiming at using state-of-the-art digital technologies in community engagement platforms to involve citizens in decision-making processes within their local neighborhood. WeGovNow enables a new type of interactivity in the co-production of citizen-centered services and the co-development of strategic approaches to community development. The platform provides several e-Government components and related services. The mentioned components provide the following facilities: citizens' urban activities coordination and collaboration, reporting local issues to a public administration, opinion formation on a given issue, and web mapping tools. All components implement map-centered web applications.

As known, web-map applications provide non-typical web content (mainly, graphical) usually visualized by JavaScript (JS) code. For such applications, higher-level advanced approaches to quality of service (QoS) assessment are required. Quality of Service assurance is usually based on testing a number of parameters (e.g., reliability, compatibility, code maintainability, availability, effectiveness, vulnerability, efficiency). From tens to hundred metrics can be applied for web service evaluation. Many existing approaches are described in the literature considered. One can describe them as low-level solutions. Map-based web applications (a.k.a., WebGIS) require higher-level approaches.

In this work, the author proposes an approach based on the evaluation of users' interest in a web service and the quantity of information provided by a web page. In order to conduct this, proof of work (PoW) verification is applied to distinguish real users from bots (or auto-ware, like spiders or crawlers). Moreover, spatial information theory based metrics are applied to evaluate the quantity of information provided by web pages.

This article proposes PoW-based solutions for QoS assurance, monitoring of user activity and, prospectively, balancing of WebGIS services. For the first part of the approach, we use a JavaScript function to implement PoW verification in client side. Only proved by work users' activity is considered further. This allows monitoring of users' activity and setting up some initial variables employed in the further processes. Next, the author introduces a novel approach based on spatial information theory to assess the quantity of information delivered by WebGIS pages. The proposed users' verification technique and the procedure for evaluation of web pages both construct a framework for QoS assessment.

2. RELATED WORK

(Saleem et al., 2016) mentioned several issues regarding the quality of web services. Testing of web services is a significant problem that should be studied carefully; the testing has to be extensive and comprehensive to all important levels, unit, component and system level.

For assurance of quality of web services, the testing mechanism for reliability, compatibility, code maintainability, availability, complexity measures, effectiveness, vulnerability, efficiency and performance measures were discovered. In (Shafin et al., 2012) a Web Service Regression Testing Model (WSRTM) was presented. Many approaches are applicable for testing the reliability of web services (Eler et al., 2010). For proper functioning of web services, it is important to work orderly, have the capability of efficient integration and operation. Efficiency means the quality or property of being efficient and near to half techniques caters for efficiency quality parameters; all other techniques are not considering the efficiency in their techniques. It is an essential need and the most important aspect that to ensure efficiency while performing and providing services to their user. Besides efficiency, the effectiveness of providing services is also important. Efficiency parameter is assured by many techniques (Masood et al., 2013; El Ioini and Sillitti et al., 2011; Shafin et al., 2012; Yan et al., 2012). Excepting (Yan et al., 2012) all mentioned techniques are suitable for compatibility. (Zheng et al., 2014) proposed an approach to the harvesting of data regarding QoS assessment conducted for thousands of real world websites. The proposed techniques allow low-level evaluation. As mentioned, this paper introduces a higher-level QoS assurance approach aimed mainly at WebGIS services.

In this work, we introduce a novel Proof of Work (PoW) based approach to quality of service assessment. PoW concept was described by (Nakamoto, 2008). He described this process as follows. A PoW system similar to Adam Back's Hashcash (Back, 2002), rather than newspaper or Usenet posts. The PoW involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash. For the timestamp network, the PoW is implemented by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the PoW, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.

The Blockchain's PoW concept is utilized for distinguishing real users and bots. Alternatively, it can be achieved by using session management or analyzing users' behavior (Tran and Nakamura, 2016). The former approach may significantly decrease the number of users, because of required authorization. The latter required the deployment of complex systems for user behavior analysis and modeling. In many cases, it is not applicable.

It is popular to use information theory to estimate the quantity of information delivered by a map (Noskov and Doytsher, 2014). The entropy of maps with different degrees of generalization derived from a one detailed map should be almost identical if extraneous data is removed. A good example of the task was demonstrated by (Clarke and Battersby, 2001). The developed Coordinate Digit Density (CDD) function measures redundancy. The removal of extraneous data in the dataset should prove to increase information content through the reduction of redundancies in the coordinate set. A metric based on CDD is utilized in this work.

3. METHODOLOGY

Various approaches for low-level QoS assessment were discussed in the previous section. These approaches are suitable for any kind of web services. Developers use such methods for low-level monitoring. In this work, we propose a solution for higher-level QoS assurance oriented to a rich web application. The solution consists of two main steps. First, we define pages which will be evaluated. In order to achieve it, we apply an approach for distinguishing real users and bots. Second, pages visited by real users are assessed. We utilize an image-based approach for this task. All processes are carefully logged; that allows us to verify the results of QoS assessment and provide a full-history information to developers and users in the future.

3.1 Client Side PoW: Verification of Real Users

As known, HTML has been designed as a markup language representing text documents. Now, its use case is much wider. It has been enriched by incorporating CSS and JS into HTML pages. As a result, HTML pages can provide rich content. HTML5 offers WebGL functionality which enables to implement advanced graphics applications covering even 3D visualization.

Despite the structured manner of HTML, it is difficult to analyze rich web pages, because JS can turn a document into a very complex entity. WebGIS applications illustrate this. WebGL based map applications do not contribute to the Document Object Model (DOM). They rather provide results of processing in a form of raster images. Attempts to analyze structurally modern web applications remain a complex task. Thus, in this article, we introduced an image-based approach to web page assessment and monitoring.

PoW-based user verification followed by the web page information quantity calculation allows us to assess the quality of WebGIS service. The proposed approach is suitable for web services providing rich visual content. It is not limited only by map-based web applications.

In order to recognize bots from real users, JS-based solutions are applied quite often. For instance, a media file could be accessed by a script function. Bots are usually written in scripting languages (e.g., Perl, Python). Bots access multiple web pages for various reasons: harvesting content, collecting statistics, seeking vulnerabilities, spamming (e.g., referrer spam), etc. Bots can easily manage cookies. As a result, bots cannot be easily filtered out by web service owners.

Normally, bots do not evaluate JS code. To prevent email spamming, many people prefer to publish their email addresses as links generated by a JS function. This approach is quite effective since significant computer resources are required to acquire information generated on-the-fly in web browsers.

The PoW concept. The Proof of Work (Pow) concept utilized by Bitcoin enables to significantly decrease the probability of recognizing bots as real users. On the Blockchain technology, the complexity is based on probability to find a hash integer value lower than a defined level.

In order to implement a proof of work approach, we propose to use the xxHash algorithm (Mashtizadeh, 2017). Developers describe it as "an Extremely fast Hash algorithm, running at RAM speed limit". According to the testing results, xxHash runs 16 times faster than MD5 and 19 times faster than SHA1. This is important because in our case hashes are calculated in a web browser environment. Proof of work processing cannot be very sensitive to a regular user. At the same time, it should be costly for bots. xxHash libraries are implemented for many programming languages, including JS and Tcl. We use the latter for backend development. Two parameters are required to generate an xxHash. The first parameter is any hashing object. The second is a seed integer (usually a random object). We generate a hash according to the following JS listing.

```
XXH.h32(navigator.userAgent+”–”+window.
    location.href+”–”+String((new Date()).
    getTime()),Math.floor(Math.random() *
    999999999999)).toNumber()
```

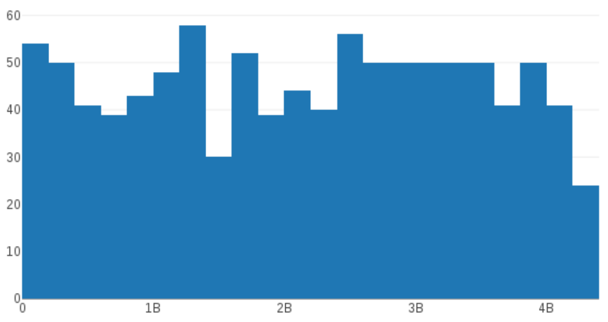

Figure 1. Histograms of hash values for 1000 random strings generated by 2 attempts.

$XXH.h32$ is a function for hash calculating. In the listing, a hash is calculated for a string object consisting of a user agent (a description of a web browser sending in headers to a server), a URL address of a web page and a time value in milliseconds joined by a dash symbol. The function is seeded by a random integer in the interval $0 - 10^{12}$. Developers can use various approaches for calculating hash numbers. They can apply another hashing algorithm, hashing parameters, various ways to combine parameters. The listing demonstrates just one of the infinitely possible solutions.

From the blockchain solution, one can notice that the proof of work concept based on the difficulty to find a hash integer lower than the defined level. Bitcoin defines this level according to the speed of calculation of the next block in a chain. If a next block is found too fast a minimal hash level is decreased to increase the complexity of a next block definition. Otherwise, the level is increased.

We use a simpler approach to define a minimal level. In order to obtain such value, we tested our solution on several devices. A special web page was prepared. The page generates a number of graphs allowing us to define the minimal level empirically.

Several types of graphs were analyzed. The following figures provide typical graphs. In Figure 1, the first type of graphs represents the domain of hash values calculated for random strings with a random seed.

From Figure 1 one can conclude that the domain of hashes in $0 - 10^9$. The distribution is almost equal. Thus we can suppose that the probability of a random hash value from 0 to the half of the domain equals approximately 0.5. For 1000 attempts hash integers are distributed with an average frequency from 30 to 60 for 22 considered intervals. It should be mentioned that the probability of values in an interval of $4.2 \cdot 10^9$ to $4.5 \cdot 10^9$ is relatively lower.

In the next step, various minimal levels are considered. In Figure 2 a histogram of attempts number required to obtain a hash lower than a defined minimum level is demonstrated. One can observe, that in the most cases number of attempts is lower than 3000. Only in a few cases, more than 3000 attempts are required. These attempts can be ignored.

In Figure 3, a histogram of time required to obtain a hash lower than a defined minimum level is presented. In most cases, a hash is obtained in time less than 30 milliseconds. Few attempts required more than 30 milliseconds can be ignored.

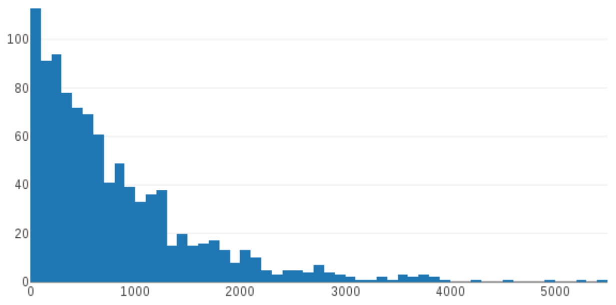

Figure 2. Histogram of attempts number required for the definition of a hash less than $5 \cdot 10^6$ for 1000 random string objects.

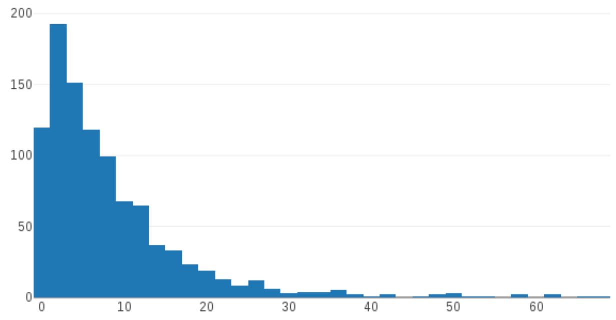

Figure 3. Histogram of time (in milliseconds) required for the definition of a hash value less than $5 \cdot 10^6$ for 1000 random string objects.

Using the discovered properties of hash values probabilities we can construct a framework for the analysis of users' activity. Usually, cookies and/or user registration (session management) are utilized to distinguish bots and real users. Currently, this problem in many cases can be avoided by bots, because bots are capable to manage cookies and even pass some basic registration process. Complex registration processes require additional efforts from the users' side. Most users prefer not to visit websites accessible only to registered users. The registration can be a serious obstacle to attract new users, especially for smaller sites.

Alternatively, some websites use a script-based initial verification of users. A user visiting a web page waits several extra seconds, while a number of JS routines are carried out to verify if a user agent is a real web browser and is not a bot script. Then, the web site returns a cookie with a session id. The process is repeated when the session is expired. This approach requires a users' patience and implementation of a session management in a server side. The session management raises problems with personal data processing because giving a session id allows websites to track the activity of a certain user. In many cases, developers avoid tracking users, because it can lead to unpleasant legal issues, especially in some developed countries. Additionally, in the age of shared VPNs and anonymous networks, like Tor, many users often share IP addresses. Thousands of bots try to automatically acquire information through these IPs. As a result, when someone visits a website, she/he is forced to pass a captcha manually. This makes web surfing for such users very problematic.

The described problems can be resolved by utilization of probability properties of hash values. The solution is based on the definition of the balance between usability of a web service and complexity of a hash-based verification task. It is implemented by the definition of a minimal hash level. In Figures 2 and 3, the

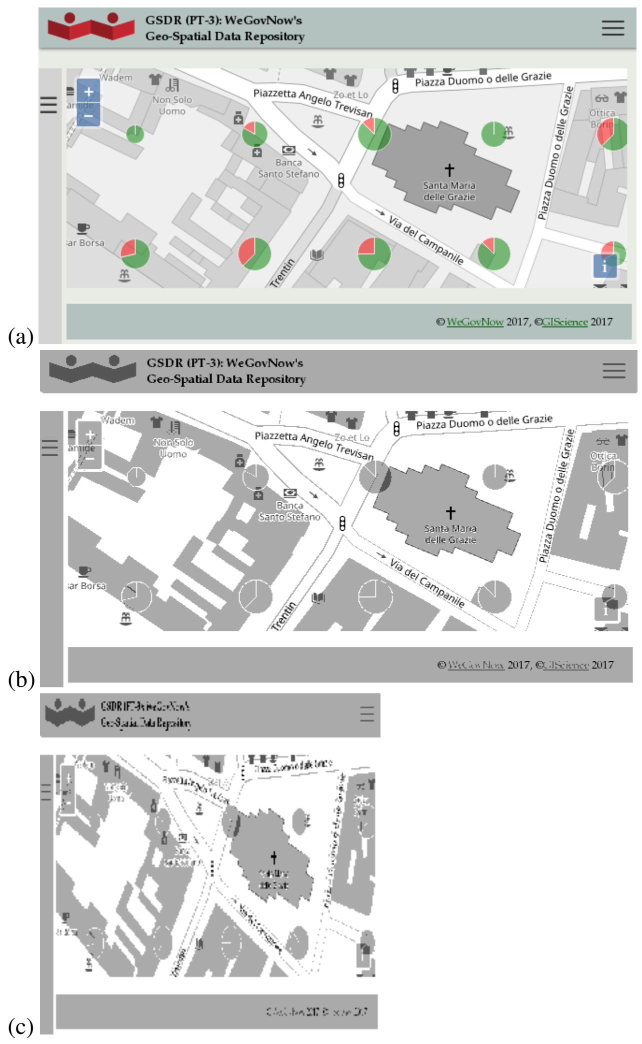
(a)
GSDR (PT-3): WeGovNow's Geo-Spatial Data Repository
Piazzetta Angelo Trevisan
Piazza Duomo o delle Grazie
Santa Maria delle Grazie
Banca Santo Stefano
Via del Campanile
©WeGovNow 2017, ©GISicence 2017
(b)
(c)

Figure 4. Normalization of rendered images. Top-down: original, gray-scaled and resized image.

solution has to be in the interval from 0 to $5 \cdot 10^6 - 1$, i.e., the minimal hash level equals $5 \cdot 10^6 - 1$. This value configures the complexity of a task. Decreasing of a level leads to increasing the complexity and, thus, processing time. Hence, in cases of increasing number of website hits, decreasing a minimal hash level will lead to slowing down a real user activity. It is useful for the users' requests balancing. Additionally, it allows administrators to identify and block bots interrupting the normal functionality of a web service.

The described Proof of Work based methodology allows us to distinguish bots and real users, avoid session management in many cases and balance request through slowing down user agents. The proposed approach is useful for indicating the usability of a web page. Only web pages visited by users passed PoW verification will be considered further.

3.2 Server Side PoW: Image Based Assessment of Web Pages

In 3.1, we described a mechanism for recognizing real users. Using it, we can see web pages visited by real users (users passed PoW verification). This automation shows us essential web pages, excluding documents playing a role of service pages (e.g., temporarily, debug and testing pages) that are not visited by real users. Knowing a list of pages, we need an approach to assess and monitor them. As mentioned above, it is difficult and unproductive to analyze DOM of web pages. In this work, we introduce an image-based approach to web page assessment and monitoring. The approach consists of the two following steps: rendering of a web page and obtaining a number of quantitative parameters from a rendered raster file.

Nowadays, multiple tools are available for automatic rendering of web pages. Most of them are divided into two categories. The fist is designed on top of the QT programming environment. The second utilizes facilities provided by the Node.JS library. Both solutions allow rendering of web pages in the command line mode. That makes them suitable to be used on a server for real-time rendering of web pages.

Rendering of rich web pages is a resource-consuming process. A web browser instance is run in memory and renders web page evaluating rich JS code. Such rendering can be applied to a limited number of web pages. Thus, it is important to consider only real users. If a page is not visited it should not be assessed. Hence, a kind a PoW procedure is applied on the server side because the solution requires significant resources and cannot be applied for very massive assessment of web pages.

A web page is rendered to a regular PNG raster file. Two significant (for assessment and monitoring) basic metrics can be extracted from a PNG file: size and height of a file. A virtual monitor utilized for web page rendering always returns same-width images, thus, it makes no sense to extract a width of a page. A minimal height of images equals the height of a virtual monitor, a maximal height is not limited.

During the rendering of web pages, all required CSS rules are applied and scripts are implemented. Thus, a rendered image file provides the same view of a web page as users see through a regular web browser.

The aim of web page monitoring is to recognize the essential properties of a web page. We do not try to catch every small change of a web page, only a significant dynamic should be registered. For this, the quantity of information provided by a web page is calculated.

Our central monitoring service is designed for multiple websites in a row. Thus, only simple information quantity metrics were considered. It was decided to use three metrics.

The first metric is the size of a PNG image in bytes (an integer value). The size can be easily obtained from an image file; it reflects the quantity of information provided by a page. Web pages rendered into same size images several times are marked as unchanged for a considering term. If the size of an image grows, the quantity of information provided by a web page increases.

The second metric is the height of the image in pixels (an integer value). This an important parameter. For two images with the same height, the bigger size image provides more information. As mentioned, all images have the same width.

Pixel Value Density. The third metric is the Pixel Value Density (PVD); it has been developed by the author. The metric uses ideas behind Coordinate Digit Density described in 2. PVD is an advances information theory based parameter designed to evaluate the amount of essential information provided by a raster image. In order to eliminate non-informative (mainly decorative) elements of web pages (e.g., color gradients, background patterns,

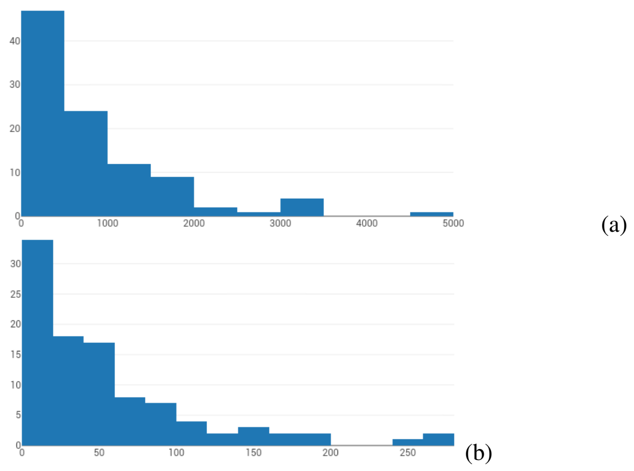

Figure 5. D3: number of attempts (top) and time (bottom) to achieve the minimal level.

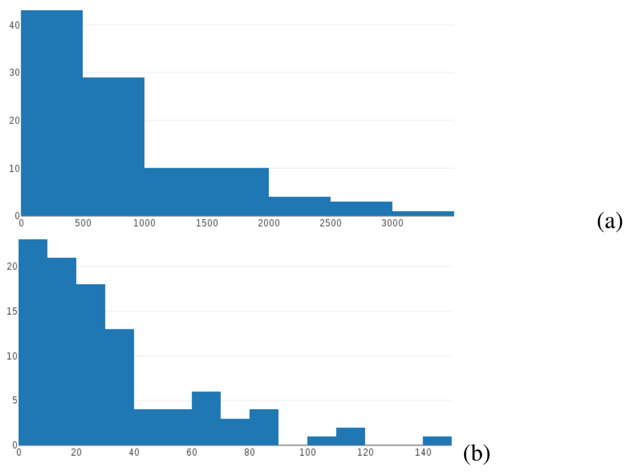

Figure 6. D2: number of attempts (top) and time (bottom) to achieve the minimal level.

etc.) and make possible to compare different-size web pages, rendered images have to be normalized. The normalization consists of converting a color image to grayscale and resizing an image to rectangular canvas. In Figure 4, the normalization process is illustrated.

PVD is calculated for normalized images according to the following equation:

$$PVD = \sum_{i=0}^{maxindex} \mid \frac{n_i}{N} - O \mid \tag{1}$$

In the equation, i is an index of a pixel value. For instance, a black-and-white image contains only two indexes of pixel values, 0 and 1-, 3-byte grayscale image provides 8 indexes for a pixel value. PNG data format defines indexes for each pixel. That requires less computer memory and can be effectively compressed. Colors of pixels are defined by a palette which establishes correspondences between indexes and colors (in RGBA format). n_i is the number of pixels with index i. N is the number of pixels in an image. O is an overall variation of the index. It is calculated as $1/2^{bits}$, where "bits" means a number of bytes required for a pixel. For instance, overall variation of a 3-bit pixel image is calculated as $\frac{1}{8} = \frac{1}{2^3}$.

The proposed metric can be refined during further verification. More web pages and longer time period are required for this.

4. IMPLEMENTATION

Refinement of shared HTML code. An empty scalable vector graphic (SVG) file is used for implementation of the client-side PoW. The SVG data format has been chosen because it is extremely easy to generate it. In comparison to raster images, any information could be easily included in a file. That is useful for debugging. Moreover, a request for an SVG file from another server is not restricted by any browser. An AJAX request requires permissions granted by a website developer. Otherwise, it will be rejected to prevent Cross-Site Scripting (XSS) vulnerabilities. Thus, our central server for user activity monitoring provides and Application Programming Interface (API) responding an SVG file. In normal mode, it responds an empty SVG file

Device Id	Name	CPU	RAM
D1	Fujitsu Desktop	Dual Core 2.4 GHz	16GB
D2	Lenovo X200	2xIntel 2.4GHz	4GB
D3	Samsung Galaxy S3	S4 1.5 GHz	2GB

Table 1. Specifications of the devices utilized in testing.

comprising an XML comment with a word "Accepted" for validated requests and "Rejected" otherwise. In debug mode, API generates SVG images comprising a green rectangle for validated requests and a red rectangle for rejected.

In most cases, modern websites provide common header and footer HTML parts. Thus, adding a script function requesting a media file automatically reflected by all pages of a website. In order to implement this idea, three components should be integrated into a common header or footer of a website. The first component is a link to a script file or inline script itself. The script contains the main function requesting a media file from a central service for monitoring user activity. Additionally, a script file contains required dependencies. Second, an empty DOM image object should be defined. And third, a trigger of a function requesting a media file needs to be specified. Thus, a maximum of three additional lines of HTML code is required.

Several devices were used to test the approach. The Table 1 provides technical characteristics of the devices.

Several websites are involved in monitoring. ***.GSDR.GQ** (gsdr.gq, wgn.gsdr.gq and wgn-pt3.gsdr.gq) is a group of sites described as "Geo-Spatial Data Repository for Grand Quality". The group consists of different instances (or versions) of a web service for, mainly, quality assessment of open geospatial data developed in the frame of the WeGovNow project. **IGIS.TK** is a web frontend of a version control system that manages an open source project. An Integrated Geographic Information System Tool Kit is developed in the frame of the project. **Tiles.CF** is another open source initiative for developing an open source library for manipulating of geo-tiles. ***.n-kov.com** (a.n-kov.com and n-kov.com) is a personal website and a home page of several small open source tools. All the mentioned websites are monitored by the central service. SQLite database is used.

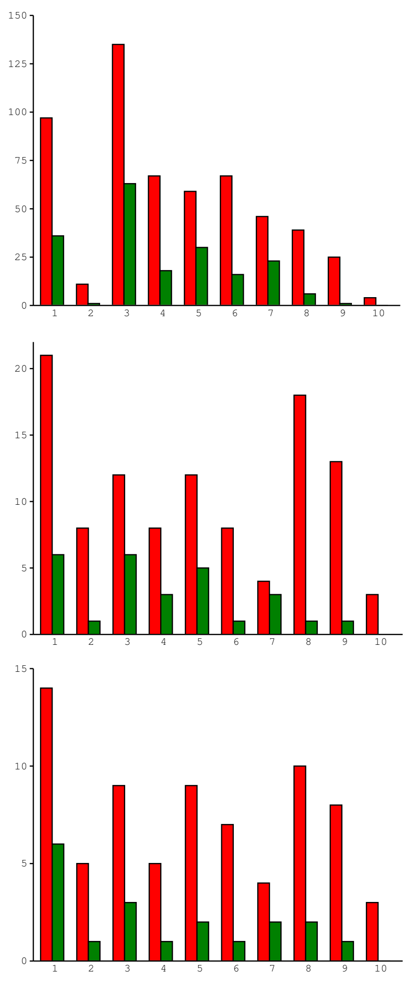

Figure 7. Number of requests (a), number of unique IP addresses (b) and user agents (c). Left (red) bars are logged by the web server data, right (green) bars are verified by proof of work. The Y-axis represents the time interval, one is 10 hours.

4.1 Client-side applications

In order to define an optimal minimal hash level or minimal hash value, a number of tests were carried out. We defined empirically that $5 \cdot 10^6$ can be used. In Figures 2-3, histograms reflecting the required number of attempts and time are provided for device D1. Figures 6 and 5 demonstrates same information for devices D2 and D3 correspondingly. One can conclude that for all devices in the worst case (excluding some extremely rare cases) only 0.2 seconds are required to find a hash with a value lower than $5 \cdot 10^6$. The fastest device is D1 (the desktop computer). It is twice faster than the slowest device D2 (the smart phone).

In web pages, recommendations provided in 3.1 were applied. The procedure is as follows. First, a web application checks the time of a previous hash definition. If the application generates a hash less than 3 seconds ago the procedure is rejected. It is useful for dynamic web applications which send monitoring requests many times, usually when the URL of a web page is modified (e.g., to update zoom level and central point of a map). Then, an iteration limited by 10^6 passes and 0.2 seconds is started. In order to find a hash lower than the predefined minimal hash value, for each iteration, a hash of string concatenating user agent, the URL of the web page and time in milliseconds seeded by a random integer (for each iteration a new random is generated) is calculated. If the generated hash is lower than the minimal level, then a request for evaluation is sent in the following form: https://wgn-pt3.gsdr.gq/wm/api.tcl?hash=1835626&time=1518404673682&seed=30631033887. The multiple iterations take time when verification implemented in a server-side is very fast. Moreover, it is impossible to know in advance which seed is required. Only multiple random tries work. This is the main idea behind the proof of work.

4.2 Server-side applications

The described solution was deployed on a VPN server with 4 dedicated ARM cores, 2 GB of memory and 50GB SSD disk. GNU/Linux Debian Jessie operating system was installed on the server. The Apache HTTP web server was installed from the official repository. The latest Apache Rivet (version 3.0.1) was built from the source code. Apache Rivet uses the Tcl programming language for server-side functionality.

The procedure of request verification is as follows. First, only requests sent from web pages belonging to the predefined list of domains are considered. Second, the time is evaluated. Allowed time deviation is one hour. Then the hash is calculated and checked if it is lower than the predefined minimal level. If so, the request is accepted.

Next, the web page from which request was sent is accessed. First of all, the page is rendered into a PNG file. File size and image height are calculated. Then, PVD is evaluated. All data are committed to the database.

5. RESULTS

The data considered in this work were collected from Wed Feb 7 15:33:59 CET 2018 to Sun Feb 11 14:07:38 CET 2018. Two data types are processed: verified by proof of work requests and a regular web server log covering only pages rendering by a web browser (e.g., *.html, *.php, *.rvt, *.cgi, etc., media files, such as *.js, *.css, *.png and APIs are not considered).

For the considered term, 550 requests were logged by the web server and just 194 requests were verified by proof of work. In Figure 7, logged and verified data are compared. One can notice that most of the requests were made by bots. In the bar charts, the number of requests, unique IP addresses, and user agents are compared. All charts represent a relatively similar ratio between real users and bots.

Figure 8 represents collected data in a histogram view. Charts (b) and (c) were clipped because for Y axis more than 6 the represents Y value 1. The first histogram reflects collected seeds (random values). The following histogram represents metrics of the rendered web pages: size, height, size/height ration and PVD value of the processed images.

Charts (b) and (c) of Figure 8 represent the same distribution of images. Charts (d) and (e) provide more representative and useful distribution for analysis allowing us to access the quality of services. Additionally, it can be mentioned that some web pages were not rendered because rendering took too much time (we limited it by 30 seconds). Such pages can be a source of users' troubles and developers can consider improving their performance. This can be utilized as a Boolean indicator of web page quality.

The distinguished by the histograms rendered images are presented in Figure 8, metrics are provided by Table 3.

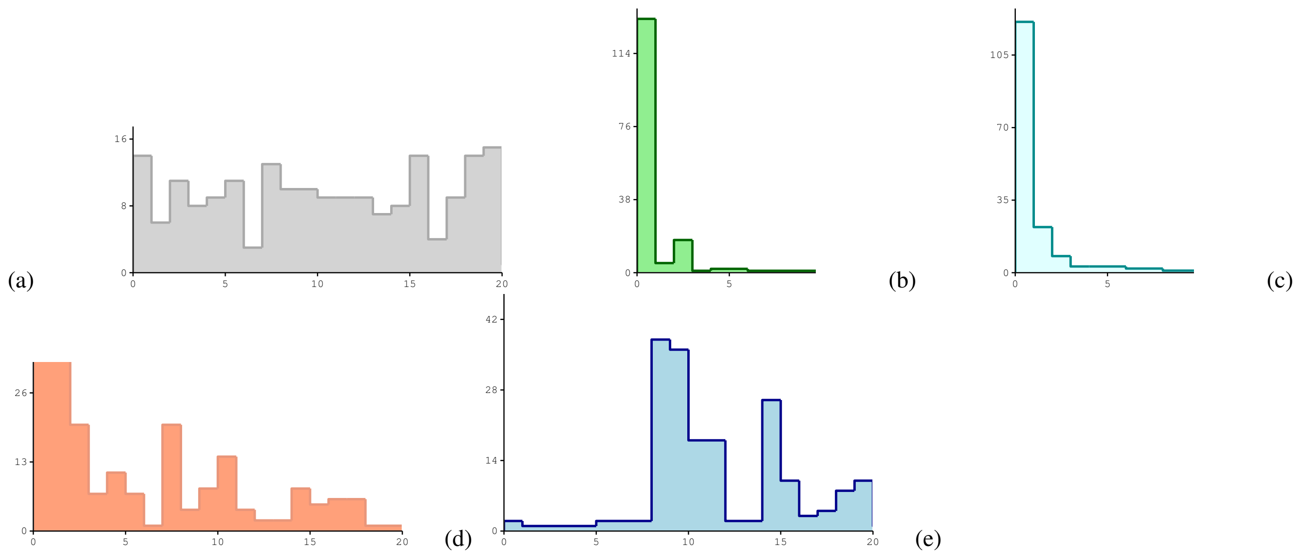

Figure 8. Histogram of seed (a) random values used for the hash generation and size (b), height (c), size/height (d) and PVD (e) values. For the meaning of the intervals of X-axis see Table 2

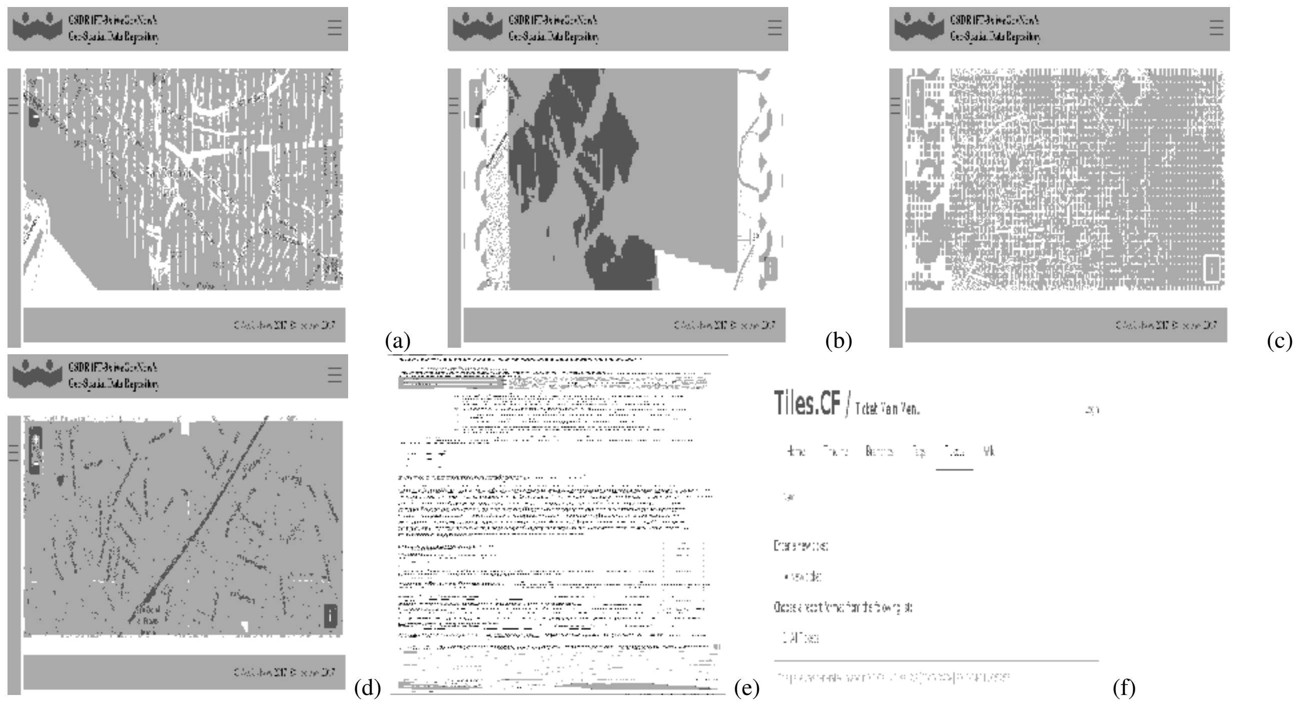

Figure 9. Normalized rendered web pages utilized for PVD calculation.

Histogram, data type	Minimal	Maximal	Interval(Y)
(a), seeds	$1.12 \cdot 10^{10}$	$9.95 \cdot 10^{11}$	$4.9 \cdot 10^{10}$
(b), sizes	17214	4930211	245649
(c), heights	360	26269	1295
(d) size/height ratio	1.0	784.0	39.15
(e) PVD	0.5547	1.4516	0.045

Table 2. Minimum, maximum and interval data for histograms in Figure 8

Image in Figure 9	Size	Height	Size/Height	PVD
(a)	282549	360	784.86	0.95
(b)	116877	360	324.66	0.74
(c)	130829	360	363.41	0.97
(d)	204716	360	568.65	1.09

Table 3. Image data

6. CONCLUSIONS

In this work, a novel approach to QoS assurance, monitoring, and balancing of WebGIS services is introduced. Proof of work concept is widely utilized.

In order to implement the concept, client and server-side applications have been developed. The approach enables distinguishing bots and real users. Configuring minimal hash value allows developers to balance load of web services.

Image-based monitoring of web pages was introduced. Web pages are rendered to raster images; a number of metrics are calculated. The metrics are used to assess the quality of service. They can detect poorly designed web pages and compare various types of web pages.

The presented method is effective and promising. Prospectively, more websites will be covered by the presented implementation. A frontend web page displaying collected real-time data will be developed. More data will be harvested and analyzed.

ACKNOWLEDGEMENTS

This work has been funded by the European Union's Horizon 2020 research and innovation programme under the grant agreement n. 693514 ("WeGovNow"). The article reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

Boella, G., Francis, L., Grassi, E., Kistner, A., Nitsche, A., Noskov, A., Sanasi, L., Savoca, A., Schifanella, C. and Tsampoulatidis, I., 2018. WeGovNow: A Map Based Platform to Engage the Local Civic Society. *In Companion of the The Web Conference 2018 on The Web Conference 2018, International World Wide Web Conferences Steering Committee*, pp 1215-1219 https://doi.org/10.1145/3184558.3191560

Back, A. 2002. Hashcash - a denial of service counter-measure. http://www.hashcash.org/papers/hashcash.pdf (4 July 2018)

Clarke, K. and Battersby S., 2001. The Coordinate Digit Density function and Map Information Content Analysis. *Proceedings of the American Congress on Surveying and Mapping Annual Meeting*, Las Vegas, NV, USA.

Eler, M., Delamaro, M., Maldonado, J. and Masiero, P., 2010. Built-In Structural Testing of Web Services. *Brazilian Symposium on Software Engineering*, Salvador, Bahia https://doi.org/70-79 10.1109/SBES.2010.15

El Ioini, N. and Sillitti, A., 2011. Open Web Services Testing. *IEEE World Congress on Services*, Washington, DC, pp. 130-136 https://doi.org/10.1109/SERVICES.2011.48

Mashtizadeh, A., Garfinkel, T., Terei, D., Mazieres, D. and Rosenblum, M., 2017. Towards practical default-on multi-core record/replay. *Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, pp 693-708 https://doi.org/10.1145/3037697.3037751

Masood, T., Nadeem, A. and Ali, S., 2013. An automated approach to regression testing of web services based on WSDL operation changes. *IEEE 9th International Conference on Emerging Technologies (ICET)*, Islamabad, pp 1-5 https://doi.org/10.1109/ICET.2013.6743536

Nakamoto, S., 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf (4 July 2018)

Noskov, A. and Doytsher, Y., 2014. Preparing Simplified 3D Scenes of Multiple LODs of Buildings in Urban Areas Based on a Raster Approach and Information Theory. *Thematic Cartography for the Society*, Springer International Publishing, pp 221-236 https://doi.org/10.1007/978-3-319-08180-9_17

Nyquist, H., 1924. Certain factors affecting telegraph speed. *Transactions of the American Institute of Electrical Engineers*, 43, pp.412-422 https://doi.org/10.1002/j.1538-7305.1924.tb01361.x

Saleem, G., Azam, F., Younus, M., Ahmed N. and Yong L., 2016. Quality assurance of web services: A systematic literature review. *2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, , pp. 1391-1396 https://doi.org/10.1109/CompComm.2016.7924932

Shafin, S., Zhang, L. and Xu, X., 2012. Automated testing of Web Services system based on OWL-S. *World Congress on Information and Communication Technologies*, Trivandrum, pp 1103-1108 https://doi.org/10.1109/WICT.2012.6409240

Tran M. and Nakamura Y, 2016. Web Access Behaviour Model for Filtering Out HTTP Automated Software Accessed Domain. *10th International Conference on Ubiquitous Information Management and Communication (IMCOM '16)*, ACM, New York, NY, USA, , Article 67 , 4 pages https://doi.org/10.1145/2857546.2857614

Yan, M., Sun, H., Wang, X. and Liu, X., 2012. Building a TaaS Platform for Web Service Load Testing. *IEEE International Conference on Cluster Computing*, Beijing, pp 576-579 https://doi.org/10.1109/CLUSTER.2012.20

Zheng, Z., Zhang, Y. and Lyu, M., 2014. Investigating QoS of Real-World Web Services. *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32-39 https://doi.org/10.1109/TSC.2012.34