WiFiMon App Measuring Wi-Fi Performance as Experienced by End-Users

Kurt Baumann[1], Christos Bouras[2,3], Vasileios Kokkinos[2,4], Nikolaos Papachristos[3], Kostas Stamos[2,4]
[1]Network SWITCH, Zurich, Switzerland
[2]Computer Technology Institute & Press “Diophantus”, Patras, Greece
[3]Computer Engineering and Informatics Dept., Univ. of Patras, Greece
[4]Greek Research and Technology Network, Greece
kurt.baumann@switch.ch, bouras@cti.gr, kokkinos@cti.gr, papachristosn@upatras.gr, stamos@cti.gr

***Abstract*—The measurement of quality and efficiency of a wireless Wi-Fi network is particularly difficult, as there is not a single tool that can record measurements from all sides of the system, i.e. from both the access point and the end-user. Existing tools are able to monitor the overall quality of the wireless network; although they cannot determine how end-users experience the quality of Wi-Fi in a particular part of the network at a given time. In this paper we present a novel tool, named WiFiMon, which enables measuring, recording and exporting statistics regarding the quality of a Wi-Fi network as experienced by the end-users. The measurements are triggered by the end-users when they visit WiFiMon-enabled websites and/or run WiFiMon-enabled mobile applications and are recorded without users’ intervention. Main goal of WiFiMon is to give network administrators a better overview on how the end-users experience the conditions of the Wi-Fi network.**

Keywords—wi-fi; wireless network; network monitoring; mobile application; quality of network;

I. Introduction

Nowadays, there is a great demand for network resources in universities campus areas where users such as researchers, staff and students access Internet through their laptops and/or smartphones. In addition, the wireless network should be designed in such a way so as to satisfy the client requests, irrespectively of their number or location. Measuring and verifying the performance of a Wi-Fi network is extremely challenging, especially if these measurements should take into account the conditions of the end-users [1].

At present, information for Wi-Fi performance measurement and verification can be reported in three ways: (a) mobile end-user device, (b) Wireless Access Points (WAP)/Wi-Fi-Controller and (c) Network Management Systems (NMS). The last two approaches allow basic troubleshooting on clients and indicate potential problem areas; although they do not show the performance of actual applications or the performance of mobile devices without a specific vendor software extension or solution, i.e. they do not tell the whole story from the mobile client device perspective.

Having the above in mind, we focus on mobile end-users’ devices and we investigate how to provide network-relevant data (bandwidth, latency, etc.) based on end-users’ behaviour on the campus wireless network. Previous works, like [2], have tried to understand patterns of activity in the network [3] and improve the capacity of Wi-Fi infrastructures by increasing the AP deployment density [4]; however currently there is not a single tool that can trigger and record measurements from the end-user devices, without user actually intervening to initiate the performance tests.

The tool presented in this paper (named WiFiMon) tries to fill this gap, by integrating a Server Side Service implemented in Java and a service supported by a native mobile application. The service offers an end-to-end tool for measuring the quality of a Wi-Fi network at a given time. It allows users to perform and record real-time Wi-Fi measurements without their intervention, after running WiFiMon-enabled mobile applications. The service is developed and offered by GÉANT [5] and it is based on open source tools (such as Nettest [6]) that adhere to the requirements of the service.

Initially, the WiFiMon architecture and functionality was presented in [7]. The version presented in [7], enabled triggering of performance tests through web browsers when users visited WiFiMon-enabled websites. In this paper we extend the abovementioned work, by developing a mobile application (currently available only for Android devices) that enables measurements through applications running on handheld devices like smartphones or tablets. In detail, through WiFiMon app, users download a number of sample images, and report their download/upload throughput and Round Trip Time (RTT) to a central server. To meet the requirement of clients using different types of mobile devices, WiFiMon will soon be available for iOS platforms. The app is available at [8].

Apart from a standalone application where users may initiate performance test, WiFiMon may also be adopted as an extended library to external mobile applications (such as universities’ applications) as a proof of concept for the performance monitoring.

The remainder of the paper is structured as follows: The overall architecture of WiFiMon is presented in Section II. In Section III, we analyze the WiFiMon functionality and the backend service which handles user requests and records

The research leading to these results has received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

measurements. In Section IV we have an overview of the WiFiMon app procedure. Section V includes screenshots of WiFiMon app running in a smartphone and presents the performance results. Finally, Section VI concludes the paper and presents some future steps that could follow this work.

II. Architecture

To implement WiFiMon on mobile devices, we adopt GEANT libraries and JavaScript files which provide the ability to read, import, export and share the Wi-Fi measurements results [7]. Simultaneously with these libraries, we use a series of open source libraries including Java library, JavaScript library, Android and iOS SDK [9], [10].

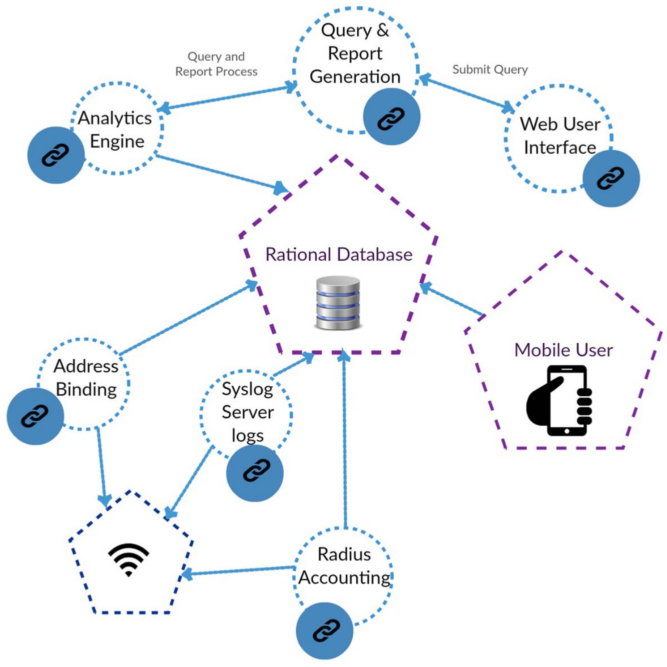


Fig. 1. Components of WiFiMon Architecture

The diagram presented in Fig. 1 shows the main components and mechanisms of WiFiMon architecture. This approach follows the process steps for collecting data on the Wi-Fi interface of eduroam-enabled infrastructures and defined collectors, storing temporary persisted data in (non) SQL databases formats, analysing raw data in real time, and consuming this data by visualizing in real-time the reference data. The architecture can be divided into five blocks, which we present in detail in the following paragraphs.

A. Data Source

The data source layer (Syslog Server logs, Radius Accounting and Address Binding) generates information through the WiFiMon app (or websites with embedded test procedures), namely JavaScript code embedded in the web source that enables users to run process tests without intervention, and is exporting raw data from data source collectors.

Open Source tools, such as NetTest [6] or Boomerang [11] are embedded into websites in order to extract network-related information within the WiFiMon app or a Web browser. Such information includes performance data, such as throughput on downloads and uploads of images with various sizes, and RTT, as experienced by the end-user.

B. Relational Database

The raw data, of the Data Source block are automatically collected in the Relational Database (RDB). The RDB uses SQL as the language for querying and maintenance. The database is based on Open Source technologies, such as PostgreSQL for the relational database, and InfluxDB [12], for a time-series database used for visualization.

C. Analytic Engine

The Analytic Engine (AE) is the architecture block used to examine/analyze the RDB's raw data and preparation of reference data. Thus, the main functional of the AE is to sort the raw data collected, analyze it, and provide visualizations using tools that offer the greatest insight on the wireless network performance. In the current approach Grafana is used to create the measurement and monitoring dashboards [13].

D. Query and Report Generation

The Query and Report Generator (QaRG) is used to obtain specific information from the RDB and the AE. Its main purpose is to search for usable information from these two architecture blocks and to post this information in the form of reports or visualization options to the Web-user Interface (Web-UI), the front end.

E. Web User Interface Front End

The available data from the RDB and the AE is accessible through a network admin Web-UI, which allows data querying. Web-UI allows investigation of collected performance reference data, and in turn, status checks of the wireless network. The Web-UI consists of a Spring Boot [14] web application that is using Hibernate [15] for data query. This block of the architecture shows collected data and allows real-time visualization options, such as collected data from a specific time period, collected data from a specific WAP, min-max-mean values of download/upload/latency measurements, etc.

III. WiFiMon Functionality

The mobile client connects to the eduroam enabled WAP using her/his smartphone [16]. It authenticates by IEEE802.1x (EAP/TTLs), receives an IP address from the DHCP server, and entries are created in the RADIUS accounting log and the DHCP log along with coordinated timestamps, for the successfully authenticated mobile device. All the above data are entered into the RDB.

In the next step, the user is required to open the WiFiMon app (or an external mobile application with WiFiMon embedded as a library) or visit a WiFiMon enabled webpage (this webpage may be a frequently visited page such as a university portal or a conference website). JavaScript running on the WiFiMon app and/or WiFiMon enabled webpage automatically triggers a series of network performance measurements concerning bandwidth (upload and download speeds) and latency (by RTT). The measurement results are populated in the RDB where they are parsed along with the data from the RADIUS and DHCP logs (and perhaps syslog), to correlate the access point identifier (location) with the mobile device and its performance (test results) on the wireless network.

To implement WiFiMon Mobile application we chose the appropriate Application Programming Interface (APIs) that are compatible with the initial browser-based approach [7]. As we discussed earlier, WiFiMon app will adopted by university applications as an external library for Wi-Fi measurement analysis in University Campus. However using Android and iOS SDK makes the application relying on platform-specific APIs, which means that developing Android and iOS projects are independent processes. For this reason we initially focus on the Android application.

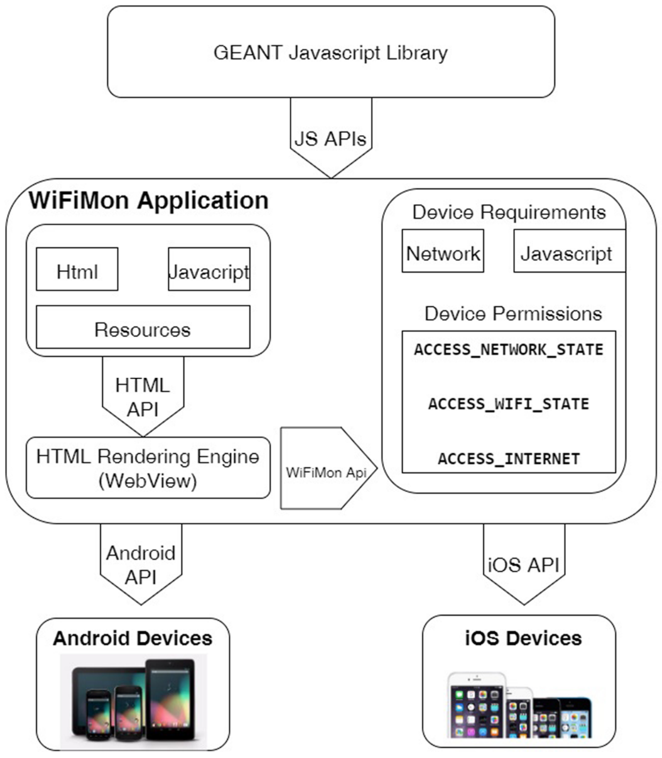


Fig. 2. The Architecture of WiFiMon App

The WiFiMon app architecture, as presented in Fig. 2 consists of three parts:

- the GEANT JavaScript library and Java-based tools, making use of Spring Boot Framework for enabling measurements and monitor analysis,
- the WiFiMon Application that runs on smartphones and sends the measurements to the server, and
- the Mobile OS using the appropriate APIs to build the mobile application.

All JavaScript plugins use HTML and JSON technologies and interact with the Mobile OS via Android and iOS API respectively.

IV. WIFIMON SOFTWARE AND APP PROCEDURE

Considering that our user is already connected to the WAP and receives an IP address, she/he has to follow the following steps to start performance measurements:

- Step 1: Download the WiFiMon App and install it using the steps described in the Readme file [8].
- Step 2: Open WiFiMon App. Through a friendly User Interface - UI (Fig. 3), user has to configure measurements options, e.g. before the process begins for first time (Fig. 4). The configuration includes setting: (1) the IP address (or domain name) of the server where the Java agent and the RDB are installed, (2) the public link to the folder where the images are downloaded to perform the measurements, and (3) the expiration time (in minutes) of the cookie that prevents repeated measurements.
- Step 3: Click on Start Measurement button and wait for the response, until monitor performance has been completed. During this period, the application initiates a series of services through html (such as download and upload file requests to server) and stores the results on the RDB. In each case user will receive a notification message of the condition of the measurements (Fig. 5).

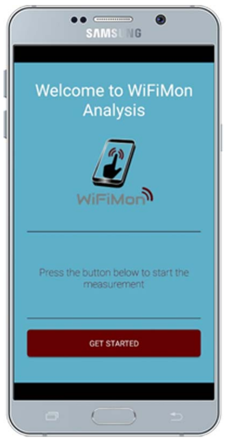
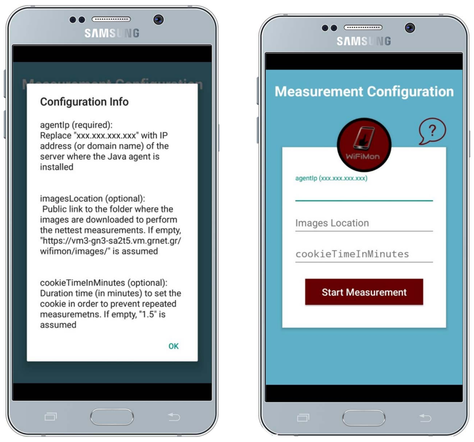

SAMSUNG
Welcome to WiFiMon Analysis
WiFiMon
Press the button below to start the measurement
GET STARTED

Fig. 3. WiFiMon Mobile App Initial Page

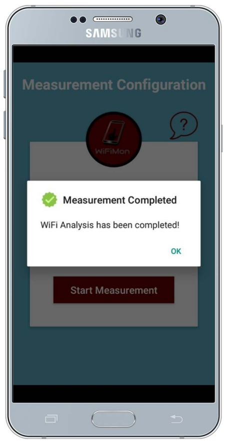

Configuration Info

agentIp (required):
Replace "xxx.xxx.xxx.xxx" with IP address (or domain name) of the server where the Java agent is installed

imagesLocation (optional):
Public link to the folder where the images are downloaded to perform the nettest measurements. If empty, "https://vm3-gn3-sa2t5.vm.grnet.gr/wifimon/images/" is assumed

cookieTimeInMinutes (optional):
Duration time (in minutes) to set the cookie in order to prevent repeated measuremetns. If empty, "1.5" is assumed

Fig. 4. WiFiMon App Configuration Page



Measurement Completed

WiFi Analysis has been completed!

OK

Fig. 5. WiFiMon App Response Message

Storing the results to the RDB is only the half of the process. In universities campuses the Wi-Fi network may consist of a large number of WAPs and the administrators should know from which WAP the measurement was initiated from. Therefore, the second half of the process includes the correlation of each measurement with the unique identifier of the WAP. The information regarding the unique identifier of the WAP can be extracted by the RADIUS and DHCP logs.

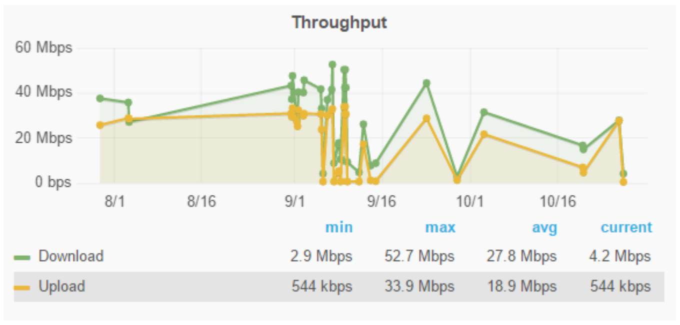
At this point we have to mention that the three steps to start performance measurements are required only when using the standalone WiFiMon app. The WiFiMon app can also be adopted as a library to external mobile applications. In this case the external application developers will have already configured the three parameters of Step 2, and the user will automatically trigger measurements while using the external application, therefore user intervention will not be necessary.

V. WIFIMON EXPERIMENTS AND RESULTS

While roaming, a number of clients visited WiFiMon enabled webpage or initiated measurements through the WiFiMon app. The clients were located in Zurich, CH and in Patras, GR, while the server that hosts the Java agent and the RDB was located in Athens, GR. This Section on the one hand presents the results of the measurements and on the other hand highlights the visualization options provided by the Web-UI.

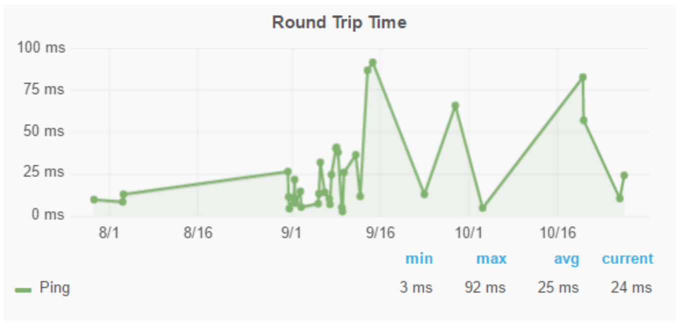
Fig. 6 presents the download and upload throughput as measured from the mobile clients. As we can see from Fig. 6, the average upload throughput is 18.9 Mbps, while its maximum value is 33.9 Mbps. The average download throughput of the mobile clients is 27.8 Mbps, with the max throughput reaching 52.7 Mbps. These variation in the throughput values may be due to the wireless technology used, the user's distance from the WAP during the measurement and the load of the wireless network during the measurement.

In Fig. 7 we have included the results of relative latency, the RTT. The figure shows that the RTT ranges between 3 ms and 92 ms, with an average value of 25 ms. It is worth mentioning that the high RTT values, i.e. RTT > 25 ms, were mainly performed from the clients that initiated measurements in Zurich. Since the measurements are relative to the Java agent location, it is expected that the RTT from Zurich to Athens will be higher that the RTT from Patras to Athens. In addition, a temporary problem on the path could also significantly affect the RTT measurements and justifies the maximum RTT measured.



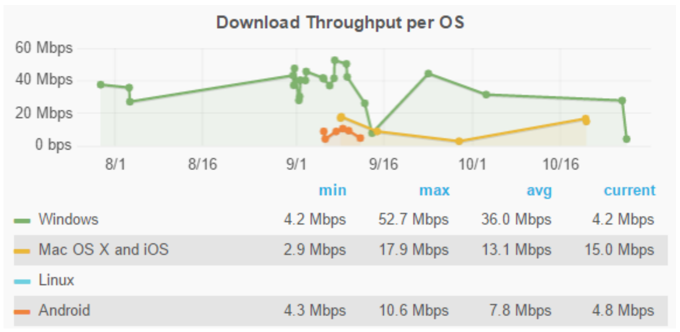
	min	max	avg	current
Download	2.9 Mbps	52.7 Mbps	27.8 Mbps	4.2 Mbps
Upload	544 kbps	33.9 Mbps	18.9 Mbps	544 kbps

Fig. 6. Download and Upload Throughput



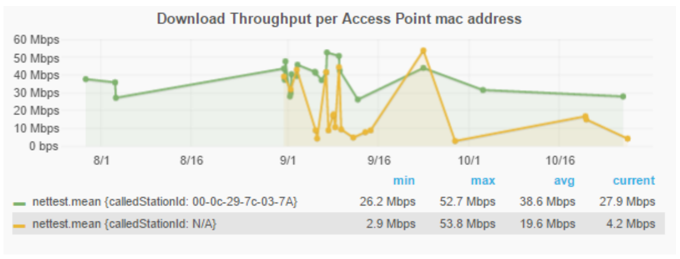
	min	max	avg	current
Ping	3 ms	92 ms	25 ms	24 ms

Fig. 7. Round Trip Time (Ping)



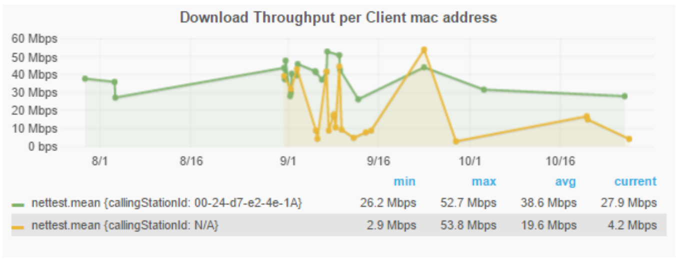
	min	max	avg	current
Windows	4.2 Mbps	52.7 Mbps	36.0 Mbps	4.2 Mbps
Mac OS X and iOS	2.9 Mbps	17.9 Mbps	13.1 Mbps	15.0 Mbps
Linux				
Android	4.3 Mbps	10.6 Mbps	7.8 Mbps	4.8 Mbps

Fig. 8. Download Throughput per OS



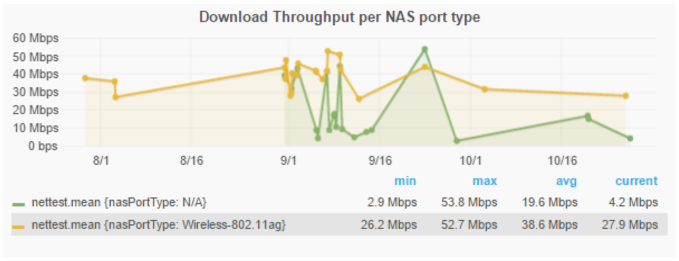
	min	max	avg	current
nettest.mean {calledStationId: 00-0c-29-7c-03-7A}	26.2 Mbps	52.7 Mbps	38.6 Mbps	27.9 Mbps
nettest.mean {calledStationId: N/A}	2.9 Mbps	53.8 Mbps	19.6 Mbps	4.2 Mbps

Fig. 9. Download Throughput per Access Point MAC address



	min	max	avg	current
— nettest.mean {callingStationId: 00-24-d7-e2-4e-1A}	26.2 Mbps	52.7 Mbps	38.6 Mbps	27.9 Mbps
— nettest.mean {callingStationId: N/A}	2.9 Mbps	53.8 Mbps	19.6 Mbps	4.2 Mbps

Fig. 10. Download Throughput per Client MAC address



	min	max	avg	current
— nettest.mean {nasPortType: N/A}	2.9 Mbps	53.8 Mbps	19.6 Mbps	4.2 Mbps
— nettest.mean {nasPortType: Wireless-802.11ag}	26.2 Mbps	52.7 Mbps	38.6 Mbps	27.9 Mbps

Fig. 11. Download Throughput per wireless technology

Fig. 8, Fig. 9 and Fig. 10 are sample figures to highlight the visualization options provided by the Web-UI. The figures present the download throughput per Operating System, AP MAC address and Client MAC address respectively. Fig. 9 is especially interesting for network administrators since it allows them to see from which AP the measurements were initiated and therefore to have an overview of the APs that are performing poorly.

Finally, Fig. 11 shows the monitoring results per AP technology. As Fig. 11 shows, most measurements were taken from APs that use 802.11ag technology, while the technology was not available for the remaining measurements. The average download throughput from 802.11ag APs is 38.6 Mbps, while its maximum value is 52.7 Mbps. Additional visualization options include:

- Measurements per IP
- Measurements per browser
- Ability to view measurements from a specific AP, browser, client, etc.

VI. Conclusions and Future Work

In this paper we developed and presented a standalone mobile application that enables and stores performance measurements. This app can also be adopted as a library to external mobile applications to enable measurements through these applications without user intervention. The measurements from WiFiMon app together with those initiated by WiFiMon enabled webpages can give a better overview of the Wi-Fi conditions as experienced by the end users and could allow network administrators to monitor which AP are performing poorly.

Next step in our work is the deployment of WiFiMon app for iOS devices and the integration of the app as external library to external applications for development and further testing. Moreover our goal is the extension of the standalone WiFiMon app that will allow end-users to display and organize the measurements results in a more useful and friendly way. Finally, our target is extending the visualization possibilities by mapping and displaying the collected data in Google Maps using pins and performance quantity metrics.

References

[1] Nielsen, "SO MANY APPS, SO MUCH MORE TIME FOR ENTERTAINMENT". Available at: http://www.nielsen.com/us/en/insights/news/2015/so-many-apps-somuch-more-time-for-entertainment.html

[2] L. DiCioccio, R. Teixeira, C. Rosenberg, "Measuring Home Networks with HomeNet Profiler", International Conference on Passive and Active Network Measurement (PAM 2013), pp. 176-186, 2013

[3] D. Kotz, K. Essien, "Analysis of a campus-wid wireless network", Wireless Networks, vol. 11, issue 1, pp. 115-133, 2005

[4] K. Sui et al., "Understanding the Impact of AP Density on WiFi Performance Through Real-World Deployment", 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN 2016), pp. 1-6, 2016

[5] GÉANT: https://www.geant.org/

[6] NetTest, "Google Code Archive". Available at: https://code.google.com/archive/p/nettest/

[7] V. Kokkinos, K. Stamos, N. Kanakis, K. Baumann, A. Wilson, J. Healy, "Wireless Crowdsourced Performance Monitoring and Verification - WiFi Performance Measurement Using End-User Mobile Device Feedback", 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT 2016), pp. 432-437, 2016

[8] WiFiMon app. Available for download at: http://ru6.cti.gr/ru6/system/files/tools_and_mechanisms/WiFiMon_Mobile_App.zip

[9] Android Documentation, http://developer.android.com/reference/classes.html

[10] Ios Documentation https://developer.apple.com/library/content/navigation/

[11] Boomerang, "SOASTA/boomerang". Available at: https://github.com/SOASTA/boomerang

[12] Influxdata, "influxDB". Available at: https://influxdata.com/timeseries-platform/influxdb/

[13] Grafana, grafana.org, "Beautiful metric & analytic dashboards". Available form: http://grafana.org/

[14] Spring, "Building an Application with Spring Boot". Available at: https://spring.io/guides/gs/spring-boot/

[15] HIBERNATE, "Hibernate. Everything data". Available at: http://hibernate.org/

[16] Eduroam, eduoram.org, "How to deploy eduroam on-site or on campus". Available at: https://wiki.geant.org/display/H2eduroam/How+to+deploy+eduroam+on-site+or+on+campus