WiFiMon: A Tool for Wi-Fi Performance Monitoring and Verification

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

Contact Person:

Prof. Christos Bouras

Computer Technology Institute & Press "Diophantus", N. Kazantzaki, GR-26504 Patras, Greece

and

Computer Engineering and Informatics Dept., Univ. of Patras, GR-26504 Patras, Greece

Tel: +30-2610-996951

Fax: +30-2610-969016

e-mail: bouras@cti.gr

WiFiMon: A Tool for Wi-Fi Performance Monitoring and Verification

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

***Abstract*—Measuring network quality of a wireless network as experienced by end-users is quite difficult, as there isn't a single tool available that can record measurements on all sides of the system. To fill this gap, there exists a need for better understanding the behavior of network performance issues as well as requirements for Quality of Service (QoS) from the users perspective on wireless networks, especially on sites that serve a large number of users, such as university campuses or conference venues. The approach presented in this research work is based on the end-user feedback, giving the opportunity of visualization of network performance in real time. We initially present an overview of the developed tool, which we call WiFiMon, and which has the ability to capture, record measurements and export statistics on the quality of Wi-Fi network as perceived by the end-users. In the case of WiFiMon, the measurements are initiated by the end-users - without their intervention – after they visit a web page or use mobile applications. WiFiMon aims to give a clear understanding of the Wi-Fi network conditions in which the end-users are connected by measuring specific parameters of the network through JavaScript technology, such as download/upload throughput and round trip time, and correlate these measurements with raw data from various log files to obtain additional information regarding the performance of specific Access Points of the network. The results reveal the functionality of the proposed tool and its scalability in sites that include a large number of wireless access points.**

Keywords: crowdsourced network performance; performance monitoring; performance verification; wireless network; mobile application;

1 INTRODUCTION

In recent years there has been a significant increase in the number of devices that users spend daily working, surfing on the internet, using for social networks or even using to play mobile applications, which normally need internet access. The last technology step of smart devices like smartphones and tablets are becoming an integral part of human life as effective and convenient communication tools.

Therefore, it is imperative to know the requirements of end-users that are connected to a particular Wi-Fi network in order to satisfy their demands for seamless networking. The ability to measure and verify the quality of the Wireless Network (Wi-Fi) as experienced by end-users is quite difficult, since there isn't a single tool that captures and performs network monitoring and verification from the user's perspective [1]. From a statistical point of view, the large amount of measured data (by end-user activities) allow us to analyze the performance of a wireless network, even in crowded sites such as university campuses or conference venues. In these extreme cases, users usually face a number of issues with the performance of the wireless network. The WiFiMon tool, which is being implemented by GEANT[1], comes as a solution to this problem and tries to fill in the gap of network performance as experienced by end-users.

WiFiMon attempts to push the wireless network monitoring and verification cycle at a higher layer. The main focus is placed on improved measurement verification, user friendly GUI development, mobile device app deployment, and WiFiMon as a Service (service automation). It is a tool mainly targeted for wireless network administrators, since the ability to visualize the collected data and monitoring the network status in real time can lead to improved decision making and efficient access points installation.

The process that WiFiMon utilizes can be described as a process of collecting, storing, analyzing, visualizing and consuming data, which aims to correlate the end-users measurements with the information regarding the Access Point (AP) from where the measurement was initiated. This process is not intended to replace the traditional measurements from hardware probes, which are installed at a fixed location. Instead, our intention is to supplement the hardware-probe measurements with "non-invasive" performance measurements captured by the end-users (through WiFiMon) so as to extract a hybrid solution that combines static (hardware-probe based) objective measurements with opportunistic measurements for end-users devices.

The remainder of the manuscripts has the following structure: Section 2 presents the related work in the wireless network performance monitoring. In Section 3 we describe the architectural view of our approach, by presenting the building blocks of WiFiMon as well as their functionality for both the web based and the mobile-app concepts. We also explain the procedure that enables the correlation of the measurements with the information provided in various log files to obtain access point related information, such as Radius logs accounting. Section 4 presents the performance and scalability testing results. Finally, in Section 5 and 6 we conclude the manuscript and provide an overview of the planned future steps respectively.

2 Related work

In the scope of monitoring there are several projects and initiatives designed to allow the remote monitoring of Wi-Fi APs. Today most of these approaches have the objective to monitor and verify the performance using controllers or to monitor stand-alone base stations using ping or, in advanced cases, base station scripts. In a controller-based Wireless Local Area Network (WLAN), base station status is generally observed by monitoring and processing Simple Network Management Protocol (SNMP) Trap messages. In addition to monitoring, controllers improve network quality through centralized channel partitioning and transmit power control. Traditionally, the service quality experienced by the user or the user's terminal has received little attention, even though air interface scanning has been performed to a certain extent. Cisco Wireless Control System (WCS), was mainly developed for the control and monitoring of Cisco WLAN controllers, but also includes features for collecting and visualizing data from the air interface [2]. On the other hand, AirWave's Wireless Management Suite includes several modules for WLAN monitoring and control. For the monitoring of the air interface AirWave has developed the VisualRF module, which can be utilized to check signal strength and channel partitioning [3]. Both Cisco WCS and AirWave are projects developed to perform network monitoring based on data from base stations and the fixed network without taking into account the end-user perspective.

There are of course some products that focus on the end-user side. Motorola AirDefense Mobile, can be installed on laptops running Windows XP or 2000 with an Atheros based IEEE 802.11 a/b/g network card, such as Netgear (WAG511) or Cisco (CB21AG) allowing monitoring of traffic on the WLAN and focusing heavily on information security [4]. Wireshark, is an open source software for Windows and Linux computers for the analysis of network protocols [5]. Wireshark as a service allows the straightforward analysis of traffic on the computer in question, but is not so well suited to a comprehensive analysis of WLANs. Analysis of control and beacon frames requires use of the monitoring mode, which is not supported by all operating systems, adapters and drivers and mobile devices. These approaches however require an installation from the end-user, while the results of the performance are not communicated to the network administrators.

There are of course some online tools/pages, such as Ookla Speedtest [6], where users can manually initiate measurements and get information about the performance of the WiFi network. Performance metrics usually include information about the download and upload throughput and Round Trip Time (RTT). These online tools however require the users to initiate the measurements, but most importantly they only inform the user that initiated the test for the measurement results and not the network administrator who is responsible for setting up the wireless network and optimizing its performance. In other words, these online tools are efficient for checking the quality of the WiFi network in users' premises; however, for sites with a large number of APs a different, more efficient approach should be followed.

WiFiMon differentiates from the solutions presented above since: (i) it allows for "non-invasive" performance measurements without users' intervention, i.e. the tests are initiated automatically, (ii) its results present the performance wireless network as experienced by end-users, (iii) the performance results are associated the APs, i.e. through correlation of results with information retrieved from log files, and (iv) the correlated measurements are posted to the same database so that the network administrator has a complete overview of the wireless network, irrespectively of its size.

[1] The research leading to the results presented in this manuscript has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2)

3 WiFiMon Architecture

The main goal of WiFiMon is to provide a wireless performance and verification schema based on end-user behavior on eduroam enabled Wi-Fi campuses. This indicates that all mobile devices can be considered as raw data suppliers (crowd-sourced approach). The architecture, as depicted in Fig. 1, includes four building blocks, containing the data source block, the relational database, the analytics engine, the query and report generator and finally the visualization block. Each block supports the process of collecting, storing raw data, analyzing, consuming and visualizing of reference data respectively.

The overall operation can be described as follows (see Fig. 1):

- The data source block includes the performance data generated from websites with embedded JavaScript code, and additional raw data retrieved from log files such as server syslogs, Radius accounting logs and DHCP logs. These raw data are used for the correlation of the measurements with the AP-identifiers.
- From the data source, raw date will be fed automatically into the Relational Database (RDB), which consists of a PostgreSQL database for storing the data [7], and an InfluxDB database, for time-series support [8].
- The Analytic Engine (AE) sorts the raw data, analyzes it, and provides visualizations using tools that offer the greatest insight on the wireless network performance. Currently, WiFiMon uses Grafana to create the measurement and monitoring dashboards [9].
- The Query and Report Generation (QaRG) allows a customized search and post them to Web UI.
- The Web UI (Java based) gives real-time options for visualization in form of snapshots (aggregation of correlated data). Collected data for a specific time period, or from a specific AP are some of the options provided through the Web UI.

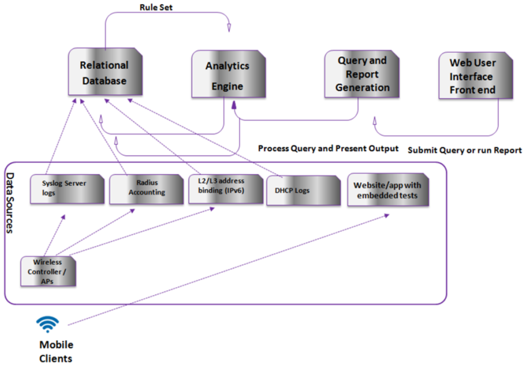

Rule Set
Relational Database
Analytics Engine
Query and Report Generation
Web User Interface Front end
Process Query and Present Output
Submit Query or run Report
Data Sources
Syslog Server logs
Radius Accounting
L2/L3 address binding (IPv6)
DHCP Logs
Website/app with embedded tests
Wireless Controller / APs
Mobile Clients

Fig. 1. WiFiMon architecture

3.1 *Measurement Procedure*

In this section we describe the procedure that enables WiFiMon measurements (see Fig. 2). Initially, the end-user has to connect to an eduroam enabled network [10]. Eduroam is the secure, world-wide roaming access service developed for the international research and education community. To start a new eduroam session, the user has to authenticate through IEEE802.1x (EAP/TTLs) with its Identity Provider (IdP) and get authorized to campus local policy from the Service Provider (SP), receiving a unique IP address by DHCP server. The authentication process creates new entries to DHCP and Radius accounting logs, and also new records are stored at the RDB.

Once connected to the wireless network, the end-user is required to visit a WiFiMon-enabled web page or use WiFiMon-enabled mobile applications for Android or iOS platform, i.e. a web page or a mobile application that has the WiFiMon JavaScript Library embedded. The WiFiMon JavaScript Library automatically triggers performance tests, assuming that the user's IP is included in the registered subnets (see paragraph 3.2). In detail, the library uses open source JavaScript-based tools, such as NetTest [11] or boomerang [12], in order to extract network-performance information. Such information includes download and upload throughput and RTT as experienced by the end-user. Once the tests are completed, the performance results are transferred by JSON post to the RDB, where they are correlated with the DHCP and Radius accounting raw data[2] (see Fig. 3). The correlated data-sets will be also fed into the InfluxDB to allow time series visualizations. Snapshots configured on the Grafana UI enable the visualization of the correlated data-sets [13].

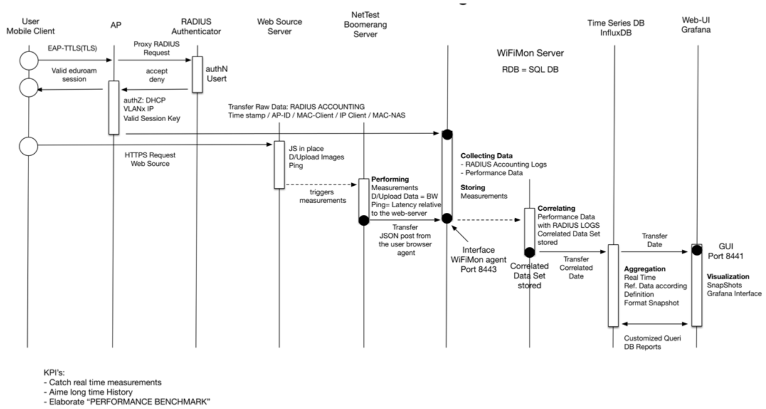

User Mobile Client
AP
RADIUS Authenticator
Web Source Server
NetTest Boomerang Server
Time Series DB InfluxDB
Web-UI Grafana
WiFiMon Server
RDB = SQL DB
EAP-TTLS(TLS)
Proxy RADIUS Request
Valid eduroam session
accept deny
authN Usert
authZ: DHCP VLANx IP Valid Session Key
Transfer Raw Data: RADIUS ACCOUNTING
Time stamp / AP-ID / MAC-Client / IP Client / MAC-NAS
HTTPS Request Web Source
JS in place D/Upload Images Ping
triggers measurements
Performing Measurements D/Upload Data = BW Ping= Latency relative to the web-server
Transfer JSON post from the user browser agent
Collecting Data - RADIUS Accounting Logs - Performance Data
Storing Measurements
Interface WiFiMon agent Port 8443
Correlating Performance Data with RADIUS LOGS Correlated Data Set stored
Correlated Data Set stored
Transfer Correlated Date
Transfer Date
GUI Port 8441
Aggregation Real Time Ref. Data according Definition Format Snapshot
Visualization SnapShots Grafana Interface
Customized Queri DB Reports
KPI's:
- Catch real time measurements
- Aime long time History
- Elaborate "PERFORMANCE BENCHMARK"

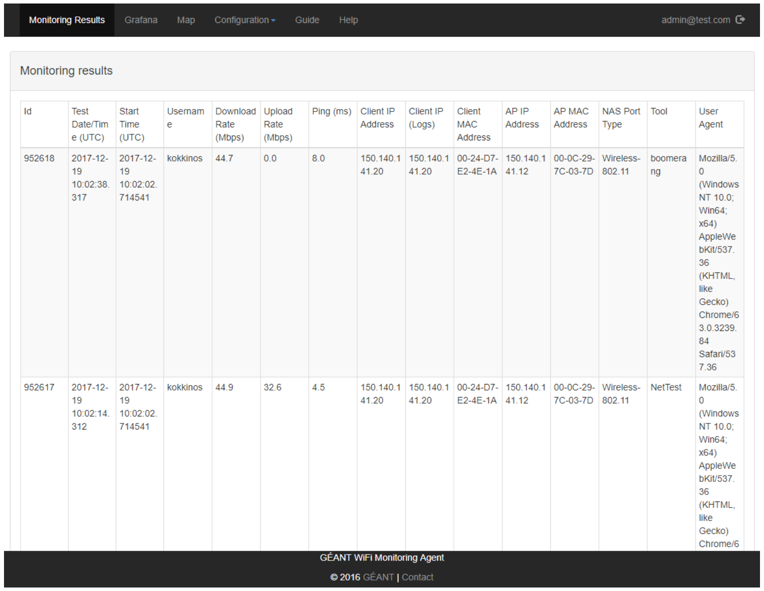
Fig. 2. Architecture flow diagram

[2] If Radius accounting logs is not available, WiFiMon can still record the measurements but the correlation with AP information is not fisible

```
Pseudo code for correlating measurements with Radius logs

CHECK user_IP, timestamp               //from measurements
CHECK client_IP, auth_timestamp     // from Radius logs
WHILE auth_timestamp < timestamp     // in descending order to
                                                  // select the most recent entry
    IF user_IP == client_IP
        INNER JOIN measurement and Radius_entry ON IP
        BREAK
    ENDIF
ENDWHILE
```

Fig. 3. Measurements and Radius logs correlation



Monitoring results

Id	Test Date/Time (UTC)	Start Time (UTC)	Username	Download Rate (Mbps)	Upload Rate (Mbps)	Ping (ms)	Client IP Address	Client IP (Logs)	Client MAC Address	AP IP Address	AP MAC Address	NAS Port Type	Tool	User Agent
952618	2017-12-19 10:02:38.317	2017-12-19 10:02:02.714541	kokkinos	44.7	0.0	8.0	150.140.141.20	150.140.141.20	00-24-D7-E2-4E-1A	150.140.141.12	00-0C-29-7C-03-7D	Wireless-802.11	boomerang	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36
952617	2017-12-19 10:02:14.312	2017-12-19 10:02:02.714541	kokkinos	44.9	32.6	4.5	150.140.141.20	150.140.141.20	00-24-D7-E2-4E-1A	150.140.141.12	00-0C-29-7C-03-7D	Wireless-802.11	NetTest	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/6

GÉANT WiFi Monitoring Agent

© 2016 GÉANT | Contact

Fig. 4. WiFiMon recent measurements

The measurements from all users are stored centrally in the same RDB. The network administrator can access the WiFiMon Web-UI to visualize the results and get an overview of the wireless network overall performance. The Web-UI consists of a Spring Boot web application [14] that is using Hibernate for data query [15]. It includes the functionalities that are necessary to enable the measurements (i.e. add/remove registered subnets), project the correlated data and allow real-time visualization options, such as measurements for a specific time period, from a specific AP, from specific IPs or users as well as results for selected wireless technologies (e.g. IEEE 802.11b, etc.). The table with the recent measurements is also available through the Web-UI (Fig. 4).

3.2 Measures to Avoid Unecessary/Repeated Measurements

Assuming that the JavaScript code is embedded in the official website (or mobile application) of the University that has installed WiFiMon to check the performance of its wireless network, it is obvious that the performance tests should not be triggered by users that visit the website but are located outside the University campus. The results of these tests would be relative to their location (and wireless network performance) and therefore not useful for the WiFi network under study. In addition, NetTest and boomerang download fixed size images from a server and measure the time it took to download them in order to calculate the bandwidth and RTT. This means that these tests increase the total size of the hosting websites. Users browsing experience will not change since the scripts that initiate measurements will run only after the page has loaded, but it will have a negative impact especially if a user is uses a metered internet connection. Again, preventing tests from IPs outside the registered subnets is important.

Equally important is to prevent repeated measurements for users inside the registered subnets in order to avoid network overloading. To this direction, once the measurement is initiated, we set a cookie that prevents re-initiation of the measurements for a specific time period (configurable by the network administrator). Based on the above, the pseudo code to initiate a measurement is presented in Fig. 5

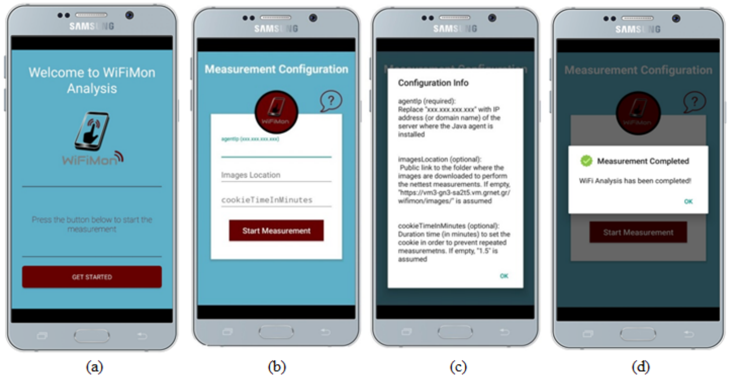
```
Pseudo code for performing/storing measurements

SET registered subnets   //allow measurements only from WiFi subnet
CHECK if cookie is set for the user //avoid repeated measurements and
                                                    //network overloading
IF user_IP inside registered_subnets
    IF cookie is not set
        GET timestamp
        CALCULATE download_throughput, upload_throughput, RTT
        GET user_IP, user_agent
        GET user_location            // with Google API loader
        POST timestamp, download_throughput, upload_throughput,
            RTT, user_IP, user_agent, user_location to Postgres and
            InfuxDB databases
    SET cookie
    ENDIF
ENDIF
```

Fig. 5. Measurement process

3.3 *Mobile App*

Over the last years, end-users satisfy their communication needs through the use of smart phones. Moreover, most of the communication is not carried out using browser-based functions, but rather using mobile applications. Towards this direction, our goal was the creation of a WiFiMon mobile app that on the one hand would work as a standalone application (where the user could take measurements and see the results) and on the other hand it could be integrated in external mobile applications (such as University or conference apps). Fig. 6 shows an overview of the standalone WiFiMon app working on Android devices (available for download at [16]).


SAMSUNG
Welcome to WiFiMon Analysis
WiFiMon
Press the button below to start the measurement
GET STARTED
Measurement Configuration
agentIp (xxx.xxx.xxx.xxx)
Images Location
cookieTimeInMinutes
Start Measurement
OK

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

Measurement Completed

WiFi Analysis has been completed!

(a) (b) (c) (d)

Fig. 6. WiFiMon android app UI

As we can see the WiFiMon disposes a friendly UI ready to be used from any user, giving the opportunity to test his Wi-Fi connection in one click. In the figure above we present the initial page of the smartphone application in which the user is being notified about the application and its purpose Fig. 6(a). In the next step Fig. 6(b), user can optionally set the parameters of the measurements, such as the WiFiMon server IP address (to post the results to), the location of the images for the performance test, and the cookieTimeInMinutes to prevent repeated measurements. If the user leaves the above fields empty the mobile application uses the default configurations to reach the server and generate the measurements. With the above parameters being set up, user can push Start button and then the measurement starts sending data to the server. When measurement has finished, a notification message appears as shown in Fig. 6(d).

4 Performance & Testing Results

WiFiMon gives Wi-Fi network providers the opportunity to have a clear picture of the network performance behavior as experienced by the end-users. Its functionality and performance was tested in two pilot sites: at the Dublin City University (DCU) and at the TNC2015 conference which took place in Porto, Portugal. The results of the pilot operation at DCU are presented in this section.

Apart from its functionality, WiFiMon was also tested in terms of scalability (when it receives a large amount of measurements) and in terms of Web UI performance as the measurements in the two databases (PostgreSQL and InfluxDB) increases. The second part of this section presents and analyzes the stress testing results.

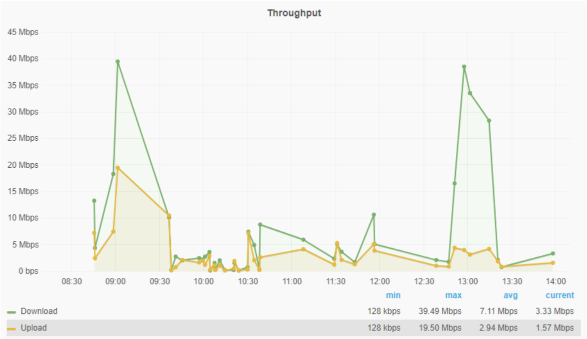
4.1 *DCU Test Case*

The WiFi infrastructure at DCU consists of a pair of Motorola RFS7000 wireless controllers, with a range of Motorola-dependent APs. DCU uses FreeRADIUS [17] with an ISC DHCP server on Linux. DCU campus includes more than 800 wireless APs. The solution is implemented to use non-tunneled bridge mode, where client devices are placed on a local switch VLAN after authentication based on a Radius attribute, which distinguishes categories of users. The authentication to the WiFi network is through the eduroam configured service, for local users and visitors. An AP identifier is included in the Radius logs that enabled the associations of the mobile devices with the APs, allowing location based monitoring.

The procedure for the DCU test case was as follows: On May 26 2017 from 08:30 until 14:00, a number of end-users initiated performance test from different locations while roaming across the campus. As they were roaming, the end-users made handovers to APs that supported different 802.11 technologies. The performance tests were initiated by visiting a WiFiMon-enabled website, which automatically triggered NetTest measurements, i.e. fixed size images were downloaded from /uploaded to a server in order to calculate the bandwidth and RTT. The server was located in Athens, Greece which means that the download/upload rates and RTT were relative to this server’s location. Once the measurements were completed, the results were automatically posted to the RDB and correlated with the Radius logs information (see Fig. 2), allowing the association of performance results with the individual APs. In the figures that follow, we can see the test case results and all the dashboards that are available through the WiFiMon UI.

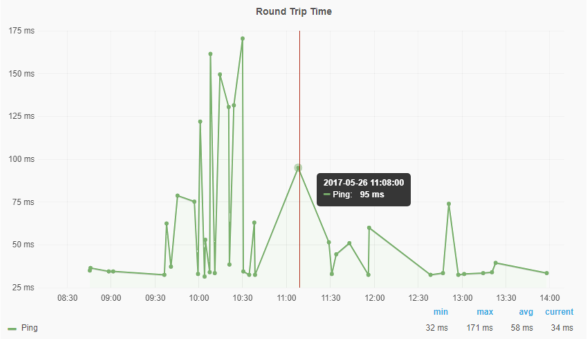
In Fig. 7 we can see the first dashboard that presents the time series of download and upload throughput, as measured by the end-users. The dashboards also provide general information, such as the minimum, maximum and average throughputs achieved. In detail, the maximum value of throughput in the case of download is 39.49Mbps, while the maximum upload throughput achieved is 19.50Mbps. The average throughput is 7.11Mbps for the download and 2.94Mbps for the upload; while the minimum values are 128kbps for both cases. From Fig. 7 it is clear that the download and upload rates show great variation. Yes, these variations may have occurred for example due to temporary problems in the network during the measurement process, or because of temporary poor WiFi connectivity of end-users (e.g. if they were located far from the serving AP).

Apart from the download/upload rates, we can see that there are great variations in the RTT time series (Fig. 8). The minimum RTT achieved is 32ms and the maximum 171ms; however, these values are reasonable since the RTT is measured from Dublin, Ireland to Athens, Greece. In addition, we can see that during the period 09:30 until 10:30 there are many measurements with high RTT values. During this period, the download and upload throughputs achieved were low, indicating poor network performance or temporary network problems, or even connection of end-users to APs that support technologies with lower maximum speeds (e.g. 300Mbps IEEE 802.11an, 130Mbps IEEE 802.11bgn).



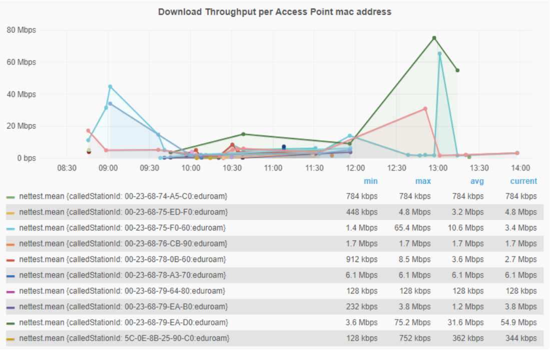
	min	max	avg	current
🟩 Download	128 kbps	39.49 Mbps	7.11 Mbps	3.33 Mbps
🟨 Upload	128 kbps	19.50 Mbps	2.94 Mbps	1.57 Mbps

Fig. 7. Download and upload throughput



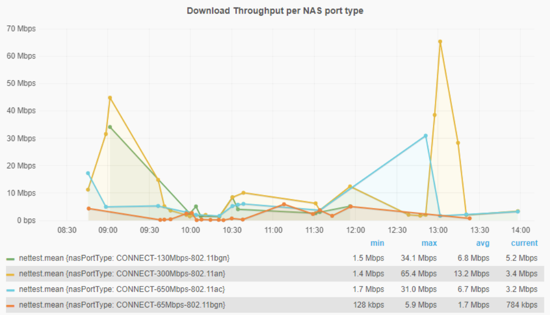
	min	max	avg	current
🟩 Ping	32 ms	171 ms	58 ms	34 ms

Fig. 8. Round trip time



	min	max	avg	current
nettest.mean {calledStationId: 00-23-68-74-A5-C0:eduroam}	784 kbps	784 kbps	784 kbps	784 kbps
nettest.mean {calledStationId: 00-23-68-75-ED-F0:eduroam}	448 kbps	4.8 Mbps	3.2 Mbps	4.8 Mbps
nettest.mean {calledStationId: 00-23-68-75-F0-60:eduroam}	1.4 Mbps	65.4 Mbps	10.6 Mbps	3.4 Mbps
nettest.mean {calledStationId: 00-23-68-76-CB-90:eduroam}	1.7 Mbps	1.7 Mbps	1.7 Mbps	1.7 Mbps
nettest.mean {calledStationId: 00-23-68-78-0B-60:eduroam}	912 kbps	8.5 Mbps	3.6 Mbps	2.7 Mbps
nettest.mean {calledStationId: 00-23-68-78-A3-70:eduroam}	6.1 Mbps	6.1 Mbps	6.1 Mbps	6.1 Mbps
nettest.mean {calledStationId: 00-23-68-79-64-80:eduroam}	128 kbps	128 kbps	128 kbps	128 kbps
nettest.mean {calledStationId: 00-23-68-79-EA-B0:eduroam}	232 kbps	3.8 Mbps	1.2 Mbps	3.8 Mbps
nettest.mean {calledStationId: 00-23-68-79-EA-D0:eduroam}	3.6 Mbps	75.2 Mbps	31.6 Mbps	54.9 Mbps
nettest.mean {calledStationId: 5C-0E-8B-25-90-C0:eduroam}	128 kbps	752 kbps	362 kbps	344 kbps

Fig. 9. Download throughput per AP MAC

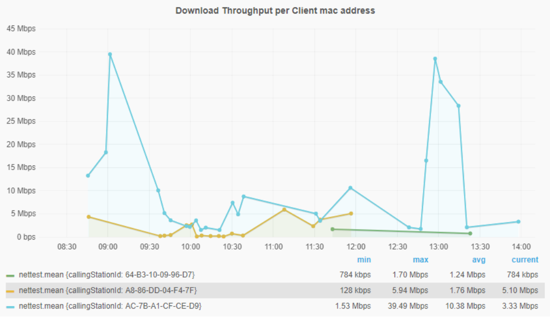


	min	max	avg	current
nettest.mean {nasPortType: CONNECT-130Mbps-802.11bgn}	1.5 Mbps	34.1 Mbps	6.8 Mbps	5.2 Mbps
nettest.mean {nasPortType: CONNECT-300Mbps-802.11an}	1.4 Mbps	65.4 Mbps	13.2 Mbps	3.4 Mbps
nettest.mean {nasPortType: CONNECT-650Mbps-802.11ac}	1.7 Mbps	31.0 Mbps	6.7 Mbps	3.2 Mbps
nettest.mean {nasPortType: CONNECT-65Mbps-802.11bgn}	128 kbps	5.9 Mbps	1.7 Mbps	784 kbps

Fig. 10. Download throughput per NAS port type

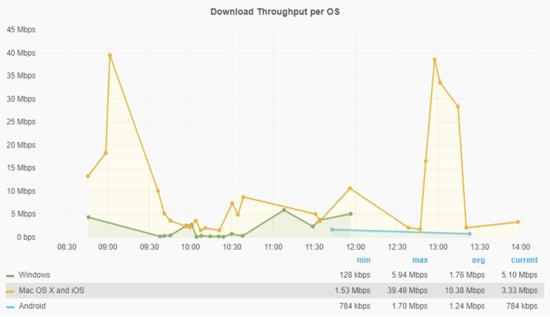
The correlation of the measurement results with the Radius logs offers additional insights. Fig. 9 and Fig. 10 present the download throughput results per AP MAC address and technology respectively. From these figures we can see that WiFiMon provides important information to network administrators regarding the performance of individual APs. Maximum, minimum and average speeds for each AP are calculated and together with the information of each AP technology allows the network administrator to evaluate its performance. Specifically for the DCU case, from Fig. 10 we can see that download throughput results from APs that support 802.11an standard are higher compared to the ones initiated from APs that support 802.11bgn standard.

In addition, the Web UI includes dashboards that evaluate the performance of individual users' devices (MAC address), operating system (OS) and browser (Fig. 11, Fig. 12 and Fig. 13 respectively). The information about the client's device MAC address in included in the Radius logs while the OS and browser are extracted from the User-Agent string in the HTTP header. Such dashboards are extremely useful in order to check whether possible poor results are due to AP's or individual devices/OS/browser's poor performance. The former shows that and improved and more efficient APs installation should take place; while the latter is something out of the administrator's score.



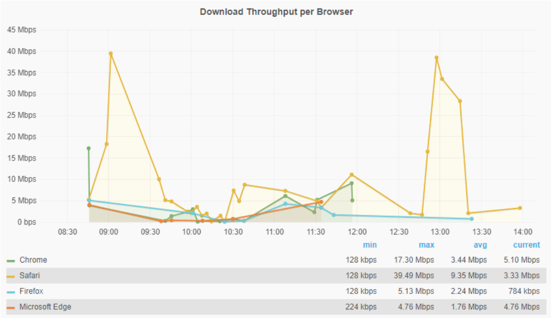
	min	max	avg	current
nettest.mean {callingStationId: 64-B3-10-09-96-D7}	784 kbps	1.70 Mbps	1.24 Mbps	784 kbps
nettest.mean {callingStationId: A8-86-DD-04-F4-7F}	128 kbps	5.94 Mbps	1.76 Mbps	5.10 Mbps
nettest.mean {callingStationId: AC-7B-A1-CF-CE-D9}	1.53 Mbps	39.49 Mbps	10.38 Mbps	3.33 Mbps

Fig. 11. Download throughput per Client MAC



	min	max	avg	current
Windows	128 kbps	5.94 Mbps	1.76 Mbps	5.10 Mbps
Mac OS X and iOS	1.53 Mbps	39.49 Mbps	10.38 Mbps	3.33 Mbps
Android	784 kbps	1.70 Mbps	1.24 Mbps	784 kbps

Fig. 12. Download throughput per OS



	min	max	avg	current
Chrome	128 kbps	17.30 Mbps	3.44 Mbps	5.10 Mbps
Safari	128 kbps	39.49 Mbps	9.35 Mbps	3.33 Mbps
Firefox	128 kbps	5.13 Mbps	2.24 Mbps	784 kbps
Microsoft Edge	224 kbps	4.76 Mbps	1.76 Mbps	4.76 Mbps

Fig. 13. Download throughput per browser

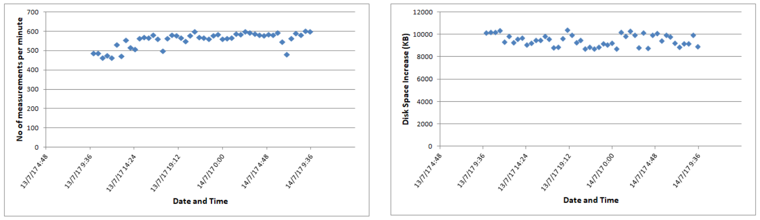
4.2 Stress Testing Results

On the 13th and 14th of July 2017, we initiated a series of tests in order to (i) examine the scalability of the WiFiMon agent when it receives a large amount of measurements, and (ii) examine the performance of WiFiMon GUI as the measurements in the two databases used in WiFiMon increases. To this direction, we generated a real-time stream of JSON data using the JSON Data Generator [18]. The specifications and the usage of the virtual machines (VMs) that we used for the stress testing can be found in Table 1.

Table 1. Simulation Specification

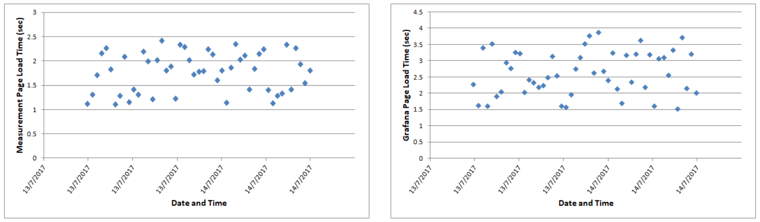
Virtual Machine	Usage	RAM	CPU	Hard Disk	Operating System
vm1	WiFiMon instance	2GB	1	9.8 GB HD	debian-squeeze
vm2	JSON Data Generator	2GB	1	9.8 GB HD	Ubuntu (12.04)
vm3	JSON Data Generator	2GB	1	9.8 GB HD	Ubuntu-trusty (14.04)
vm4	JSON Data Generator	2GB	1	9.8 GB HD	debian-squeeze

The JSON Data Generator in each VM was configured to perform a sequence of 10 measurements (about 1 per 250 ms), then paused for about 1 sec before repeating a new sequence of measurements. These settings generated about 150-200 JSON requests/measurements per minute per VM. The measurements from all VMs were posted in the same WiFiMon instance (vm1), i.e. the specific WiFiMon instance was receiving about 450-600 measurements per minute. The measurements were also correlated with Radius logs so as to increase the complexity. Manual Radius logs were inserted to the PostgreSQL to this direction.



(a) (b)

Fig. 14. (a) Received measurements per minute and (b) Disk space increase (recorded every 30 minutes)



(a) (b)

Fig. 15. WiFiMon Web UI: Load time for (a) measurement and (b) Grafana pages (recorded every 30 minutes)

The simulation lasted for 24 hours. Every 30 minutes, we captured the count of measurements and the used disk space of vm1 that hosted the WiFiMon instance and databases. Fig. 14(a) presents the number of average received measurements per minute and Fig. 14(b) the disk space increase in vm1 as recorded from our team every 30 minutes. In addition, we measured the time required to load two pages of the WiFiMon Web UI, the measurement page and the Grafana page that includes the time series dashboards (Fig. 15(a) and Fig. 15(b) respectively).

Overall, we simulated more than 800000 correlated measurements during the 24 hours period. Even for this large (and dense) amount of measurements, the smooth and robust performance of WiFiMon was not affected as shown in the two figures above. In normal environments (such as university campuses that include a large number of APs), a realistic amount of measurements would be around 500 measurements per hour. Therefore, under normal conditions and according to the stress testing results, the WiFiMon tool can provide a robust and efficient solution to measure (and visualize) the quality of a wireless network as experienced by the end-users.

5 CONCLUSIONS

In this manuscript we proposed and presented WiFiMon, a tool that may measure the quality of a wireless network as experienced by end-users. The tool mainly targets at network administrator in order to provide them a means to visualize and evaluate the performance of their wireless network and detect anomalies in real time. WiFiMon is based on end-users feedback, since the measurements are initiated automatically by the end-users (without their intervention) after they visit a WiFiMon-enabled web page or use a WiFiMon-enabled mobile application. To this direction we deployed and presented a standalone Android mobile application that enables and stores performance measurements. This mobile app concept may also be adapted as a “generic” library to external mobile applications. We also discussed additional measures that WiFiMon is using to avoid unnecessary network overload, such as limiting the performance measurements to defined IP subnets, and preventing the duplication of measurements with the aid of cookies.

In order ascertain the feasibility of WiFiMon, we tested its operation at the Dublin City University, Ireland. The expertise gained from this test showed that it is feasible to measure and record performance data on an eduroam enabled campus and to correlate these measurements with log files in order to retrieve information regarding the performance of individual APs, users’ devices, browsers and operation systems. Finally, we conducted stress testing in order to examine how WiFiMon behaves under extreme circumstances. The results of the stress testing revealed a robust and uninterrupted operation even in cases where WiFiMon receives a large and dense amount of measurements.

6 FUTURE WORK

Having examined the feasibility and performance of WiFiMon, our future work targets are four additional directions: (i) privacy issues, (ii) updates to increase usability and visualization options, (iii) validation of JavaScript measurements with hardware probes measurements, and (iv) prediction of performance degradation.

Privacy issues: Even though the results from the performance test (i.e. download/upload throughput and ping results) are reasonably considered as non-sensitive information, the fact that the performance test are correlated with log files that may contain sensitive data (such as Radius logs) should be handled with the utmost care. Everything should be in accordance with the campus policies and legal obligations. Our intention is to update WiFiMon to this direction and allow additional options such as showing/hiding or even storing/ignoring user-related data.

Updates: The current version of WiFiMon utilizes influxDB and Grafana as a means to obtain time series for the visualization and monitoring of the wireless network. Our team aims to examine similar tools, such as the Elasticsearch and Kibana stack, that could serve as alternatives to influxDB and Grafana. Our intention is to select the most efficient tool in terms of usability and visualization possibilities that of each tool provides.

Measurements validation: To calculate download and upload throughput and RTT as experienced by the end-users, WiFiMon uses open source JavaScript-based tools such as NetTest and boomerang. Since the measurements may depend on the users' position and device status, it is necessary to verify whether the measurements obtained through JavaScripts are accurate enough. To verify their accuracy, our goal is to set up a hybrid environment, consisting of WiFiMon and hardware probes in different rooms of a university campus or a conference venue. This way we could compare the static (hardware-probe based) objective measurements with the opportunistic measurements for end-users devices and examine whether they follow the same trends.

Performance prediction: Prediction of performance behaviour (mainly of performance degradation) will be achieved through recording of long-term histories and making performance benchmarking. Existing work in performance predictions shall be evaluated, and a machine learning approach for mobile-client and APs behaviour will be deployed. The visualization of such predictions will further facilitate the decision-making process of the network administrators during network installation.

References

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

[2] Cisco, "Cisco Wireless Control System (WCS)". Available from: http://www.cisco.com/en/US/prod/collateral/wireless/ps5755/ps6301/ps6305/product_data_sheet0900aecd802570d0.pdf

[3] AirWave, "AirWave's Wireless Management". Available from: http://www.arubanetworks.com/products/networking/management/airwave/

[4] Motorola, "Motorola AirDefense Mobile". Available from: https://www.barcodesinc.com/motorola/airdefense.htm

[5] Wireshark, "Wireshark". Available from: http://www.wireshark.org

[6] Ookla, "Speedtest by Ookla". Available from: http://www.speedtest.net/

[7] PostgreSQL, "PostgreSQL: The world's most advanced open source database". Available from: https://www.postgresql.org/

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

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

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

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

[12] Boomerang, "SOASTA/boomerang". Available from: https://github.com/SOASTA/boomerang

[13] V. Kokkinos, K. Stamos, N. Kanakis, K. Baumann, A. Wilson, and J. Healy, "Wireless Crowdsourced Performance Monitoring and Verification - WiFi Performance Measurement Using End-User Mobile Device Feedback". In Proceedings of the 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT '16), Lisbon, Portugal, pp. 432-437

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

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

[16] GEANT, "WiFiMon mobile app". Available from: http://ru6.cti.gr/ru6/system/files/tools_and_mechanisms/WiFiMon_Mobile_App.zip

[17] D. Nelson, and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", December 2007. Available from: https://tools.ietf.org/html/rfc5080

[18] JSON Data Generator. Available from: https://github.com/acesinc/json-data-generator