Quota: Wi-Fi Data Managing and Costing System

Version 1.0 – for Windows platforms

JALP Jayakody
Department of Computer Science and Engineering,
University of Moratuwa, Sri Lanka.
lahiiru.13@cse.mrt.ac.lk

***Abstract*— Project Quota is about wireless local area networks and managing its data usage. There is no simple way to calculate and restrict individual data usage in a small scale wireless local area network. Quota enables that ability by forcing network users to use a third party software when connecting to the network. That software syncs users' usage statistics to a central node. Owner of the wireless router or access point can remotely manage data usage of clients after signing in to a web application hosted in central node. This document describes the design and implementation of the Quota 1.0 which includes a client application and a central server web application. Since windows platform is targeted in this version of the product, client application consists of a windows Graphical User Interface application and a windows service.**

Keywords—wireless LAN;data usage monitoring; data usage tracking; Wi-Fi data usage;

I. INTRODUCTION

Today, most of the developing countries do not provided completely free internet. Hence people have to purchase and use metered networks. Widely using and flexible way of sharing internet connectivity today is wireless local area networks over Wi-Fi. These connections are shared among several people in most of the cases, due to various reasons. Single person may not be able to afford the monthly bill or the package sizes offered by internet service provider may not fit in to their needs.

In case of such a situation, owner of the internet connection has to monitor and restrict others' network utilization. There is no existing direct simple way to achieve that necessity in small scale Wi-Fi networks. Alternatively, the Wi-Fi router can be flashed with a firmware that is capable of measuring individual usages of clients. However It voids the router warranty and gives limited features. Fortunately, Quota solution can integrate with existing Wi-Fi network as a completely separate system. It is a complete solution for sharing a Wi-Fi internet connection with multiple users in a proper way.

The software solution consists of a web application and a desktop application. Owner of the Wi-Fi router registers as a Quota admin using the web application hosted in [1]. Thereafter he or she can manage shareholders who willing to use Quota admin's internet package. Those shareholders have to install Quota desktop application. That software handles the data usage of the device and sends usage statistics to Quota web application.

At present people are moving towards Wi-Fi and number of small scale Wi-Fi networks are growing rapidly. In such a situation applications of Quota system will be many more. It will be used in houses with home broadband, in boarding places with fixed 4G connections. Further, it can be distributed free of charge as a product as the income would be generated by displaying advertisements in desktop application.

This document describes and explains about the design and implementation of the Quota solution in following sections. Section II reviews relevant systems and technologies. Section III is about an overview of the system. System models, System design, implementation and testing are discussed thereafter. Final section of the paper includes conclusion and future works.

II. LITURATURE REVIEW

When comparing and contrasting similar systems, an important observation is that there are no known similar products or alternatives for Quota. However, there are products and procedures that can be used together to fulfil some requirements from the complete requirements specification covered by Quota. Those products, procedures and capabilities are explained hereafter in this section.

Major activities in a Wi-Fi data managing and costing system are as follows,

T1 – Measuring Wi-Fi network usage.
T2 – Persisting usage statistics.
T3 – Ensuring the controllability of the connection.
T4 – Connecting to given access point with a passkey.

In order to meet the requirements of the system, T1, T2 and T3 are compulsory. Depending on the approach T4 can be optional. There is no known system which can achieve all the considered requirements. But following methods can be compared with Quota.

Method A: Configuring the router to monitor bandwidth.

Method B: Installing a bandwidth monitoring software in each client.

In method A, we have to update firmware of the router. That process needs technical knowledge and can be critical. These firmware such as DD-WRT [2] and Gargoyle [3] may not support for home routers. Although supporting firmware exists, it voids service provider warranty and has limited capabilities. Therefore, method A is not a feasible option for general users. The requirement is a single point and platform independent solution which accomplished T1, T2 and T3. Quota system does not modify router and doesn't need technical knowledge. Therefore Quota is a feasible option for general users and it is platform dependent. Platform dependency is a disadvantage in Quota compared to method A.

In method B, usage statistics are manually read from the device operating system tools or pre-installed third party tool which accomplishes T1 and T2. Method B is simple to do than method A but less reliable, as a client user can bypass or modify those tools. Further, most tools don't count network usage separately for each access point. These methods are described in "Howtogeek" by Hoffman, C [4].

In Quota system, special application and Windows service is used to monitor the usage and achieve T3 [5]. Therefore, unreliability of method B is avoided and all manual steps are automated. Task T4, which is not covered by either method A or B, is accomplishes Quota by means of Native Wi-Fi API [6]. Temporal profiles are used when connecting [7]. Then passkey is not stored anywhere and T4 becomes more secure.

Hence in Quota, accuracy and performance is low compared to method A. In general usage, Quota implementation is desirable and acceptable rather than the two discussed methods. Further, Quota is the only product available, which achieves four tasks listed above to provide complete Wi-Fi data managing and costing system.

III. SYSTEM MODELS

A. System requirments

Main functional requirement of the system is measuring usage statistics and syncing with the central server. Users should be able to connect to and disconnect from the network using the client app. Client app should display the usage summary of a user for the current billing month. Additionally, the user can send requests to admin. Main request types are register request, package change request and message request. Register requests are used to send newly registering users' information. Package change request can be sent when user needs to change his or her current package. Message request carries a custom message to the Quota admin.

Quota admin should be capable of registering new users and monitoring their usage remotely using the system. It is achieved by the central web server application. Additionally, Quota admin can adjust client packages, respond to user requests and add user payments. Further, he or she can download latest version of the client app from admin dashboard anytime. Quota system has to maintain a web server which costs monthly. These costs are to be covered by displaying advertisements in client applications. Therefore, the client application should be capable to display dynamic advertisements.

As a usability requirement, one click installation for client app is preferred [8]. It is nice to have the ability to connect to access point without asking passkey at the very first time too, then users doesn't need another internet connectivity for registration as Quota user. By providing Facebook and Google logins for system users, their sign up is simplified [9].

As supportability requirements, Quota admin users are guided at first time as they can easily get started. Client app should update automatically. It is convenient since it helps to release bug fixes and security patches immediately. All contributors for the project should follow naming conventions and edits should be well commented with proper reasoning.

B. Use case diagram

.

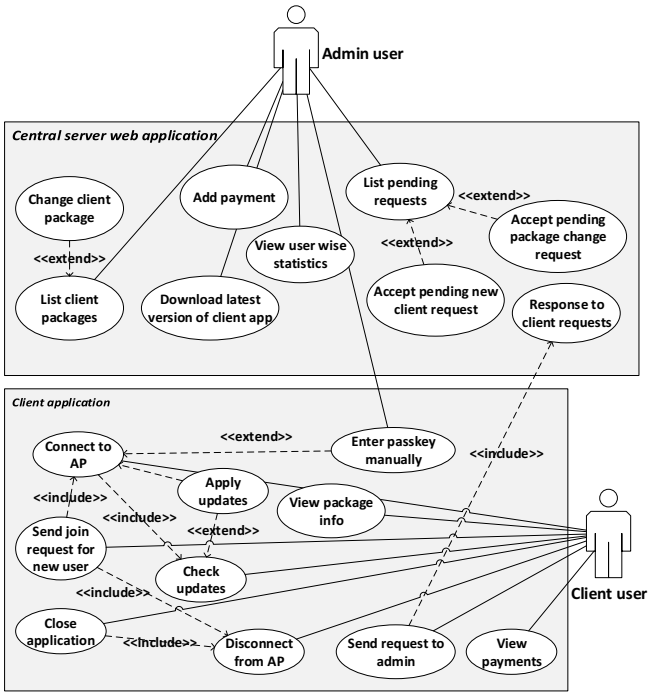

Fig. 1. Abstract use case model for Quota system

Figure 1, the use case view describes the set of scenarios, or use cases, that represent some significant functionality that have a substantial architectural coverage. Use cases related to the core functionalities are included and optional use cases are omitted to simplify the diagram. The overview of the system can be represented using the case diagram.

Quota system consists of two main sub system as represented in the diagram. In the central server web application, all use cases except "Response to client requests" are followed by a signing in process. The "Response to client requests" use case is triggered when client app sends a request to central server and acknowledged by a JavaScript Object Notation (JSON) response.

"Enter passkey manually" use case is triggered when "Connect to AP" is unable to find correct passkey. Then system asks the client user to enter passkey manually. Then system admin comes and enter the passkey.

IV. System Design

A. Logical view

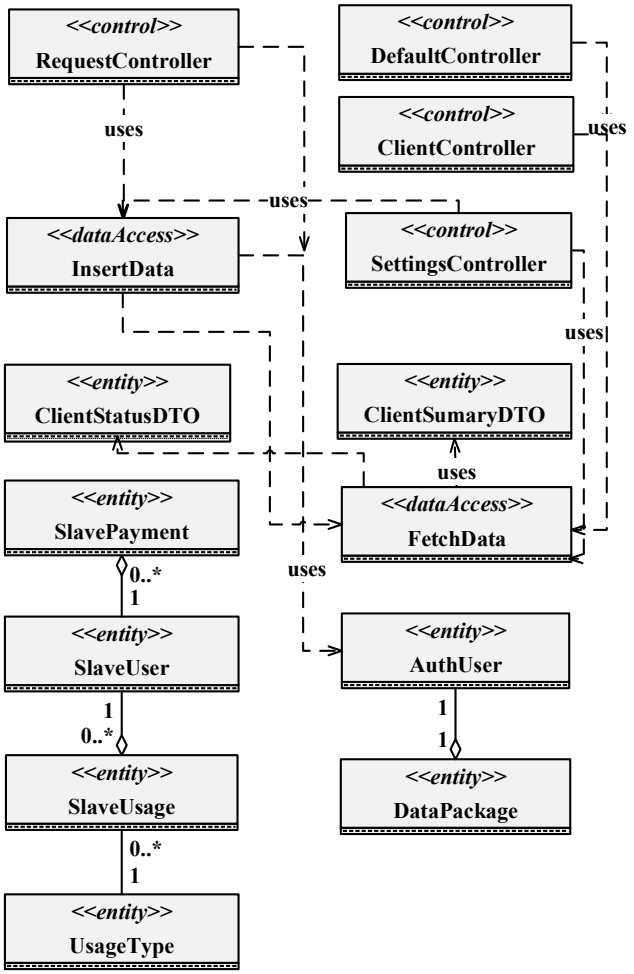

Fig. 2. Abstract class diagram for Quota web application sub system

Figure 2 shows abstract class diagram with all methods and attributes are removed for the simplicity. SlaveUser, AuthUser, SlaveUsage, UsageType, SlavePayment and DataPackage classes represent real world entities; Client user, System admin, Usage record of a user, Usage record type (e.g. Day or Night), Payment of a client user and internet service provider's data package respectively. ClientSummaryDTO and ClientStatusDTO are data transfer object classes which are used for data retrieval [10]. AuthUser has its own set of SlaveUsers and set of UsageTypes but only one active DataPackage. SlaveUser can have multiple payments including past records and set of SlaveUsage records.

Remaining classes are Controller classes which helps to follow Model View Controller (MCV) architectural pattern. All the utility classes and framework border classes are eliminated for the simplicity.

B. Process view

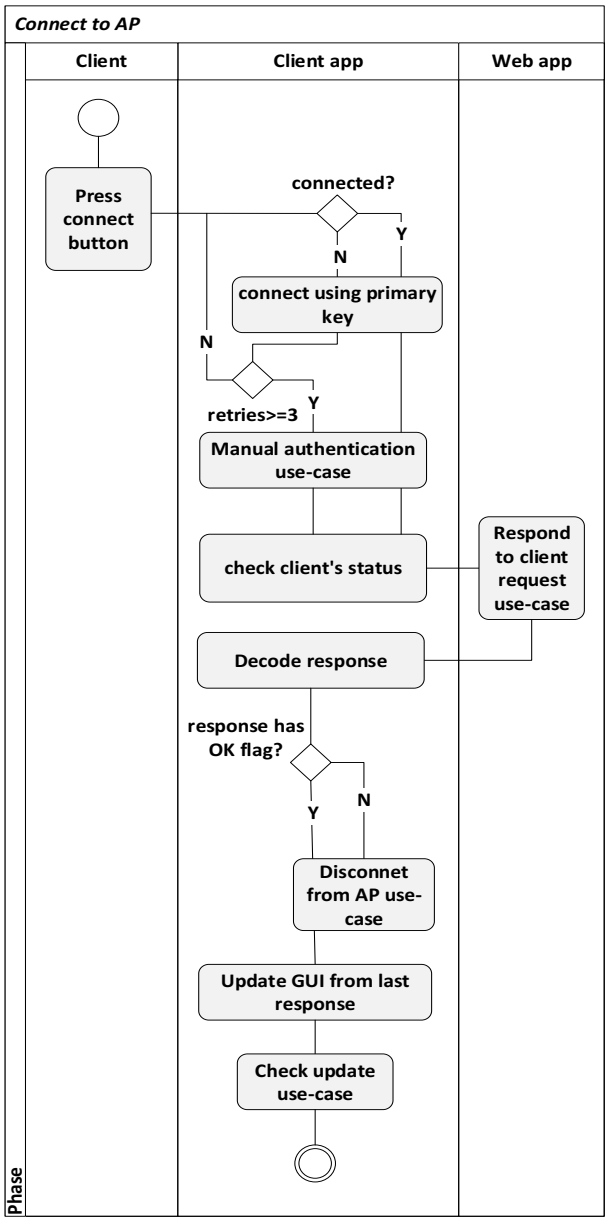

Fig. 3. Activity diagram for Connect to AP use case

Figure 3 the activity diagram corresponds to Connect to access point use case. Client user trigger the use case by clicking the connect button in client app. The app tries up to 3 time to connect. If it fails, app assume that passkey may have corrupted or invalid. Then it prompts for passkey. Then system admin enters the passkey. If connection is established, app tries to retrieve client's status by uploading pending usage data if any. If it was successful, app checks for updates and keep the connection. If client's state says client is not eligible for consuming data, app disconnects the connection immediately.

V. System Implementation

A. Implementation precedure

System is developed using Rational Unified Process known as RUP software development methodology. It is an iterative process framework. Tools used for reporting were Microsoft Word for word processing, Microsoft Visio for diagrams and Microsoft Project is used to create project schedule. PhpStrom 10 and VisualStudio 2015 IDEs were used for server application and client application development respectively.

Git was used as version controlling tool and Symfony2 framework was used for server application implementation. HWIOauth bundle and FOSUserBundle was used respectively for OAuth signing and User management. MySQL, SQLWorkBench and phpmyadmin were used to manage database instances. PHPUnit was used as the main testing framework for server application while VisualStudio test projects were used to test UI and client app [12]. Further, an AWS EC2 instance was maintained with RDP access to test the client app [13].

The development was divided into 4 main phases according to RUP [14]. There were 5 iterations in implementation phase. First iteration delivered a reliable usage monitoring system. Second iteration delivered a central server which can be used to persist client usage. Next iteration both client app and server application implemented to work together to sync relevant information. In 4th iteration user acceptance was enhanced. Final iteration delivered alpha release of the product.

B. Algorithm

```
wifiCard ← getDefaultWifi()
mac ← wifiCard->getMAC()
zone ← getSSID()
initialUsage ← wifiCard->getSentRecieve()/1024.0
pending ← getPending()
lastUsage = 0
uploadThreashold ← 20000

Function Init(){
        Connect(zone)
        res ← checkStatus(mac,zone)
        updateUI(res)
        If (isBlocked(res)) Disconnect()
        If (isNewUser(res)){
                showRegistrationUI()
                Disconnect()
        }
        Repeat mainLoop()
}

Function mainLoop(){
        If (pending > uploadThreashold){
                res ← uploadToServer(mac,zone)
                If (isSuccess(res)) pending = 0
                If (isBlocked(res)) Disconnect()
                updateUI(res)
        }
        kb ← wifiCard->getSentRecieve()/1024.0
        usage ← Kb - lastUsage - initialUsage
        lastUsage ← usage
}
```

Fig. 4. Abstracted psudocode for client app basic functionality

Figure 4 shows the pseudo code that explains the process of client app in a very abstract way. Threads, background processes, error handling and optional features are omitted. All variables defined at the top are public. Init function summarizes the procedure of starting the app. Function getDefaultWi-Fi returns active Wi-Fi adapter object and getSendRecieve fuction returns data usage since device turn on time in bytes. Function onTerminate called whenever the user tries to end the program process. Function getPending retrieves the usage amount pending to be uploaded from last session from local storage. Function setPending persists existing pending value on local storage.

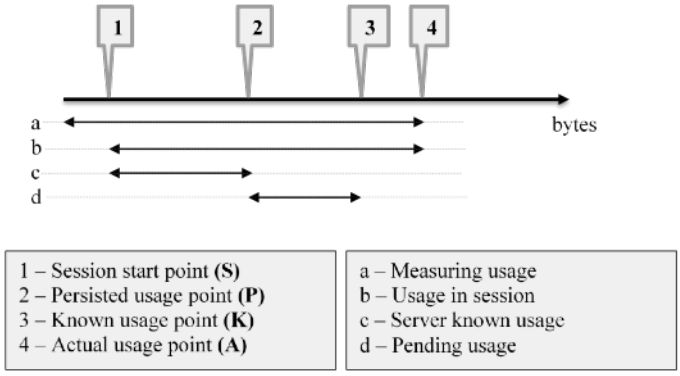

Fig. 5. Usage measuring procedure in windows client app

Figure 5shows the usage measuring procedure. When the client app is started, it reads usage from Wi-Fi interfaces and the session start point S is identified. Time to time, Wi-Fi interface usage is read and the known usage point K coincides with the actual usage point A at that moment. Since sampling has finite frequency these normally stay as distinct points. When usage is updated to the server, the p ersisted usage point P coincides with the known usage point K. Since uploading usage has less frequency, Pending usage (d) may contain more data. Since it is risky, uploading is done when d>T bytes where T is the threshold of pending. Normally it is set to 20Mb.

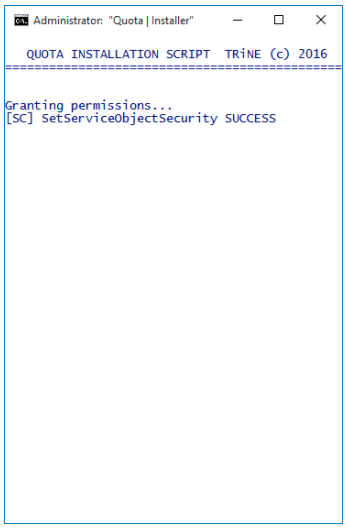

Fig. 6. Windows service installer

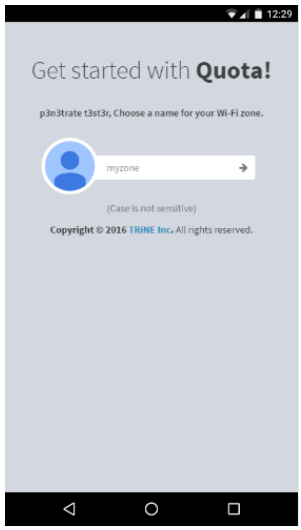

Fig.7. Admin user registration UI viewed in Nexus 5X

C. Main interfaces (UIs)

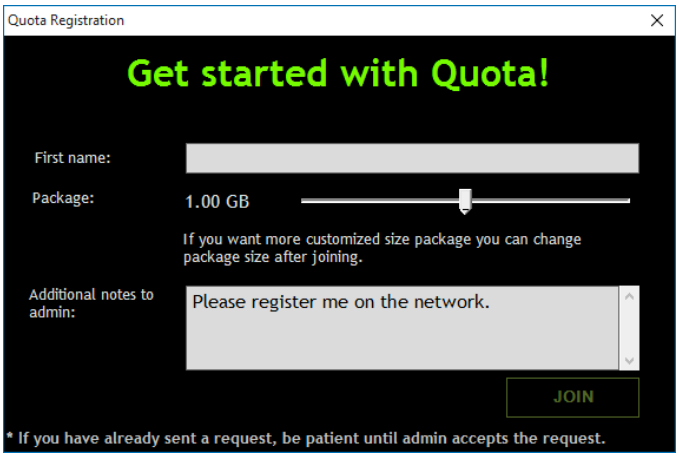

Fig. 8. Registration UI for windows client users

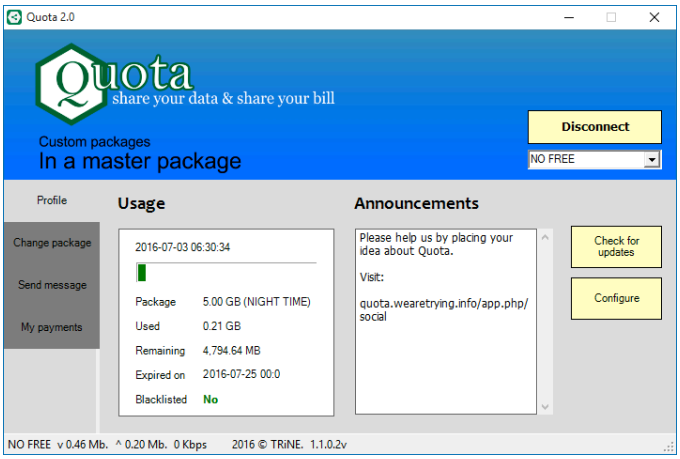

Fig. 9. Dashboard UI for client users

Figure 6 shows windows service installer which runs when windows service is updated. This UI configures the system while indicating progress to the user. Figure 7 is the mobile view of admin user registration UI on Nexus 5X device. It allows Quota admins to choose a unique zone identifier for their shared Wi-Fi network which is called Quota network. Figure 8 is one time GUI which displays when new client user tries to connect to a registered Quota network. User can choose package size there and maximum size a user can choose is limited to non-shared data balance of currently connected Quota network.

Figure 9 is the client user's main Graphical User Interface (GUI) for Windows users which allows users to connect to and disconnect from Wi-Fi network. As well as it provides all other additional functionalities described in requirements specification.

Official website for Quota product can be accessed from Domain Name "quota.wearetrying.info" [1]. Anonymously accessible landing page UI is showed to the user. Users can login to their Quota account using Google and Facebook login buttons [9]. Page is indexed with Open Graph Protocol for better view in Facebook and Google posts [15]. After the login responsive admins' dashboard UI is loaded where all functions for admin users resides. Unauthorized access to this content will result a redirect to landing page with a warning message.

VI. System Testing And Analysis

Unit tests are written for each units and tested along the development. After each iteration functional tests conducted to ensure the correct functionality of the system. For both above, PHPUnit is used with Symfony2 test cases. Then system is tested for integration and user acceptance after system is transferred to AWS EC2 instance [13]. Currently the system is at alpha release. More than 25 real clients and 3 admin users have the benefit of the system and they support the product user acceptance tests by giving feedbacks. Important factor for testing was emulating the system time which is called Clock Mocking. Since the system is sensitive to the time Symfony2 clock mocking is used [16].

Figure 10 shows the test results by PHPUnit and important code fragment used for clock mocking. That is important because there were no working examples found for this task and a discussion was opened for the topic on StackOverflow [17].

```
C:\wamp\www\quota_new\Quota>bin\phpunit -c app

PHPUnit 4.8.26 by Sebastian Bergmann and
contributors.
............

Time: 6.74 seconds

OK (12 tests, 21 assertions)
```

```php
use \Symfony\Bridge\PhpUnit\ClockMock;

public function setUp()
{
    $kernel = static::createKernel();
    $kernel->boot();
    ClockMock::register(__CLASS__);
 }

public function test()
{

    ClockMock::withClockMock(strtotime('DATE'));
        //Time sensitive tests here
    ClockMock::withClockMock(false);
}
```

Fig. 10. (Top) Test results and (Bottom) code working code template for ClockMocking

VII. Conclusion And Future work

Quota system was able to successfully achieve its requirements. In short, system is capable of measuring, persisting and controlling the Wi-Fi usage in a reliable and flexible way. Final product need to have more features than expected to handle unexpected user actions. (E.g. opening two instances of the application, automated bill calculation)

The future plan is to implement the client app to Android and other platforms. That improvement is a must to survive and distribute the product globally. Communication between Windows UI and Service should be using a Transmission Control Protocol (TCP) connection. An authorization header should be web requests. Package types can be generalized using a Quota based system. Languages, regional settings are to be generalized. Automated billing system can be implemented as well for the ease of Quota admin users.

REFERENCES

[1] J. Jayakody, "Quota | Wi-Fi data reselling system," TRiNE, 20 06 2016. [Online]. Available: http://quota.wearetrying.info. [Accessed 20 06 2016].

[2] e. admin, "DD-WRT," embeDD GmbH, [Online]. Available: http://www.dd-wrt.com/site/index. [Accessed 19 06 2016].

[3] Gargoyle, "Gargoyle Router Management Utility," Eric Bishop, [Online]. Available: http://www.gargoyle-router.com/. [Accessed 19 06 2016].

[4] C. Hoffman, "Monitor Bandwidth," How To Greek, [Online]. Available: http://www.howtogeek.com/222740/how-to-the-monitor-the-bandwidth-and-data-usage-of-individual-devices-on-your-network/. [Accessed 19 06 2016].

[5] MSDN, "Introduction to Windows Service Applications," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/d56de412(v=vs.110).aspx. [Accessed 20 04 2015].

[6] MSDN, "Native Wi-Fi," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa816369(v=vs.85).aspx. [Accessed 19 06 2016].

[7] MSDN, "Native Wi-Fi Structures," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/ms706851(v=vs.85).aspx. [Accessed 05 04 2016].

[8] MSDN, "OneClick deploy," [Online]. Available: https://msdn.microsoft.com/en-us/library/t71a733d.aspx. [Accessed 02 06 2016].

[9] Facebook, "Facebook Login," [Online]. Available: https://developers.facebook.com/docs/facebook-login/web. [Accessed 14 05 2016].

[10] D. orm, "Doctrine ORM documentation," Doctrine, [Online]. Available: http://doctrine-orm.readthedocs.io/projects/doctrine-orm/en/latest/reference/dql-doctrine-query-language.html. [Accessed 03 05 2016].

[11] Hardware.Info, "Getting Started With HWIOAuthBundle," Hardware.Info, [Online]. Available: https://github.com/hwi/HWIOAuthBundle/blob/0.4/Resources/doc/index.md. [Accessed 20 05 2016].

[12] VisualStudio, "Testing Tools and Services," Microsoft, [Online]. Available: https://www.visualstudio.com/en-us/features/testing-tools-vs.aspx. [Accessed 05 06 2016].

[13] Amazon, "Amazon Web Services EC2," Amazon , 08 03 2016. [Online]. Available: https://aws.amazon.com/ec2/. [Accessed 14 03 2016].

[14] R. S. Corporation, "Rational Unified Process: Overview," Rational Software Corporation, [Online]. Available: http://sce.uhcl.edu/helm/rationalunifiedprocess/.

[15] O. source, "The Open Graph protocol," Facebook, [Online]. Available: http://ogp.me/. [Accessed 20 06 2016].

[16] Symfony, "Clock mocking and time sensitive tests," Symfony, [Online]. Available: http://symfony.com/blog/new-in-symfony-2-8-clock-mocking-and-time-sensitive-tests. [Accessed 05 06 2016].

[17] J. J. (Trine), "Stackoverflow questions," Stackoverflow, [Online]. Available: http://stackoverflow.com/questions/37509491/symfony2-phpunit-clock-mocking-not-working. [Accessed 20 06 2016].

[18] S. Bergmann, "PHPUnit Manual," Sebastian Bergmann, 05 06 2016. [Online]. Available: https://phpunit.de/manual/current/en/phpunit-book.html. [Accessed 05 06 2016].