ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Πληροφορικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Υλοποίηση ενός ANQP server χρησιμοποιώντας την τεχνολογια Hotspot 2.0»

Ντάκου Ιωάννα

ΜΠΣΠ13075

UNIVERSITY OF PIRAEUS

Department of Informatics

THESIS

«Implementing an ANQP server over a hotspot 2.0 infrastructure»

«Ntakou Ioanna»

Abstract

Extensive use of Wi-Fi technology creates many hotspots that carry a lot of information, which should be provided to the users. The Access Network Query Protocol (ANQP) helps a user to select a network before associating with it. This paper examines a client – server connection based on IEEE 802.11u protocol and analyzes the results of ANQP queries. An ANQP server is implemented according to a System Design, which will allow performing tests locally and also remotely, using a database.

Περίληψη

Η εκτεταμένη χρήση της τεχνολογίας Wi-Fi έχει ως αποτέλεσμα τη δημιουργία πολλών hotspot τα οποία περιέχουν αρκετή πληροφορία που είναι ανάγκη να μεταφερθεί στους χρήστες. Το Access Network Query Protocol (ANQP) δίναι τη δυνατότητα στο χρήστη να επιλέξει ένα δίκτυο προτού συνδεθεί με αυτό.Αυτή η έρευνα εξετάζει μία σύνδεση client – server βασισμάνη στο πρωτόκολλο IEEE 802.11u και αναλύει τα αποτλέσματα που παράγονται από ANQP queries. Υλοποιείται ένας ANQP server σύμφωνα με ένα συγκεκριμένο σχεδιασμό συστήματος, που θα επιτρέψει την διεξαγωγή ελέγχων του δικτύου τοπικά αλλά και απομακρυσμένα με τη χρήση μίας βάσης δεδομένων

Acknowledgements

I would like to acknowledge the University Carlos III de Madrid for having me as an Erasmus Student, in order to complete my master Thesis. Moreover, my Professor Dr. D. Vergados in the University of Piraeus for recommending me for this research and giving me opportunity. My professor in Madrid, Dr. Albert Banchs for making me part of the research team and his assistant professor Antonio de la Oliva for giving me all the necessary equipment and guiding me through the whole process. Moreover I would like to acknowledge the Erasmus+ Program for funding my studies and made this possible.

.

November 2015

Ntakou Ioanna

Contents

Table of Figures

Table of Tables

CHAPTER 1: Concept and Technological Background

1.1 Motivation

Wi-Fi networking is spreading the last years extensively due to the repeated use of smartphones and tablets. Users can see and interact with many Wi-Fi networks, but a major problem remains. How can the device select the right network? It is very common that the device doesn't recognize available networks and cannot identify the necessary security credentials. It is mandatory to explore methods that will allow having access to a network's characteristics and managing this information in order to achieve the best Wi-Fi connection for the user.

The idea that was the starting point of this research was initially originated from IEEE 802.11u protocol and how we can interact with it. It was clear from the start that having a hotspot with many characteristics will trigger the need to handle what kind of information is sent to the user. In that way we could control the networks seen by it. The first step towards this goal was to implement the access to an external database, so that all the elements could be identified and modified. This concept generated a number of implementation so that a final system design will be sufficient to the initial idea.

1.2 Software Define Networking

The concept of this research is partly based in the Software Define Networking (SDN) architecture for network programmability, which promises a simplify method for network management. The SDN architecture is:

- Directly programmable
- Agile
- Programmatically configured

- Open standards-based and vendor-neutral

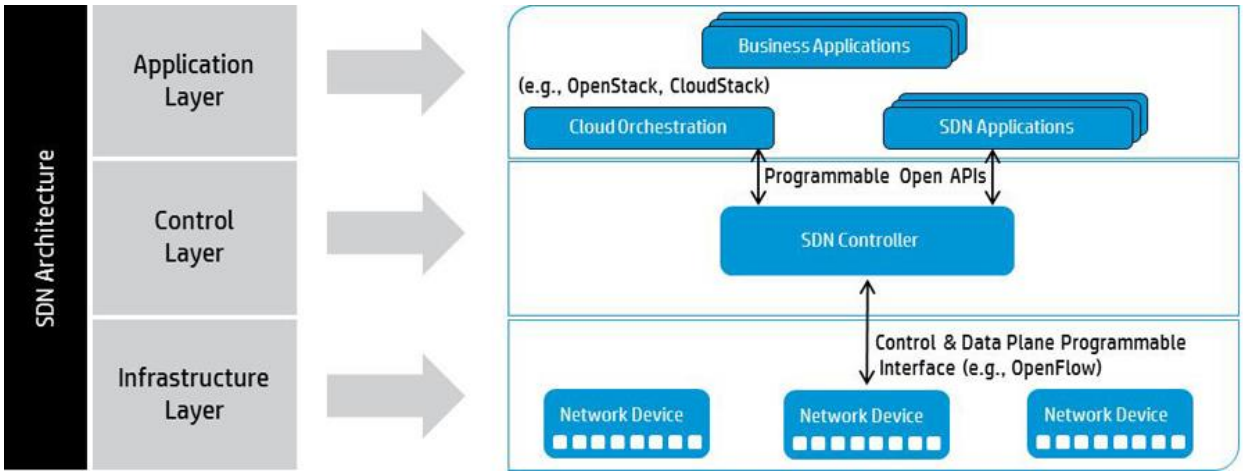

Figure 1 Software Define Network Architecture

The Openflow protocol is the foundation element of the SDN architecture, but for this research it is not used. (1) The idea is that another middleware will be managing the communication between the Controller and the Access Point.

1.3 Hotspot 2.0

Hotspot 2.0 (HS 2.0) derived from Wi-Fi Certified Passpoint (2) is a standard for accessing Wi-Fi and it was developed by the Wi-Fi Alliance and the Wireless Broadband Association. It enables users to automatically and securely establish a connection with a Wi-Fi network. Due to HS 2.0 technology, selecting a roaming partner, exchanging credentials and securely downloading the user's email can be performed from a mobile device or a tablet. Hotspot 2.0 basic specifications introduce several new features to existed hotspots: network discovery and selection, seamless network access, secure authentication and

connectivity. The basic characteristics of a Hotspot 2.0 service provider Wi-Fi network include (2):

- An IEEE 802.11u-enabled Wi-Fi network including capabilities of the hotspot
- An IEEE 802.1x-enabled Wi-Fi network with WPA2-Enterprise encryption.
- Facilitator for Wi-Fi roaming and home operator billing

The following figure shows the states a mobile device is for the network discovery and selection.

- ✓ **Discovery:** A network scan is performed in order to find available APs and information about them.
- ✓ **Registration:** the device sets up a new account with the hotspot operator. A check is made whether the device already has credentials for the hotspot and in that case this state is evanescent.
- ✓ **Provisioning:** credential information and policy information are established for the mobile device.
- ✓ **Access:** the mobile device has been successfully associated and authenticated and the user has access to the subscribed services.

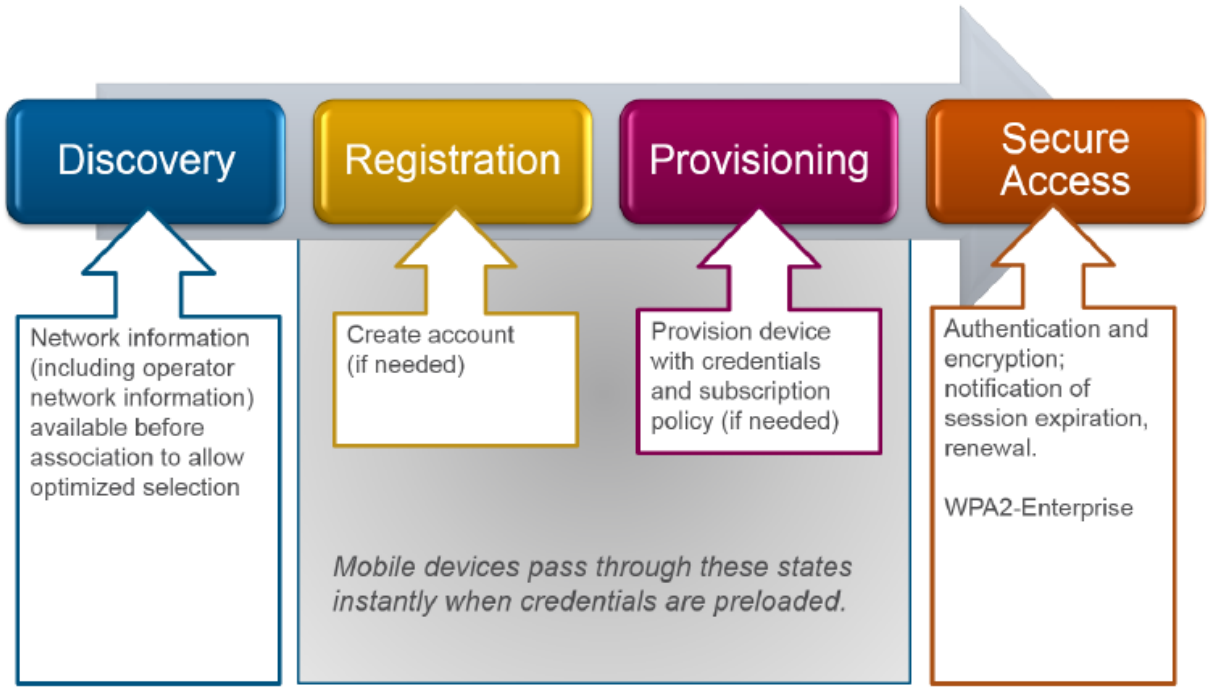

Figure 2 Network discovery and selection (2)

1.4 IEEE 802.11u protocol

IEEE 802.11u (3) is an amendment to the IEEE 802.11 medium access control (MAC), that provides connection to external networks using common wireless devices such as smartphones and tablets. An 802.11u-capable device can take advantage of Hotspot 2.0, also called Wi-Fi Certified Passpoint (2), suitable for public-access Wi-Fi. Hotspot 2.0 enables cellular-like roaming among Wi-Fi networks and between Wi-Fi and cellular networks. If a subscriber is in range of at least one Wi-Fi network, the device will automatically select a network and connect to it.

Another notable features of 802.11u include the transmission of pre-connection information to prospective users defining the type of network that is offered (private, free public, chargeable public, emergency) as well as the venue type (educational, residential, business, vehicular), roaming consortium information, the ability to receive messages from

the Emergency Alert System (EAS) and the ability to contact emergency services when necessary. Network discovery and selection is facilitated by the Access Network Query Protocol (ANQP), which comprises elements defining the services offered by an access point. Thus enabling a device to discover an appropriate network for any requirement it may have.

The main extensions to the MAC layer are (3):

- the Generic Advertisement Service (GAS) that enables a communication of a Station with an Access Point before an actual association
- additional information elements (IEs) for the Beacon frame and other management frame types
- a QoS mapping of external QoS control parameters to the QoS parameters of 802.11
- a MAC Service Data Unit (MSDU) rate limiting function to enforce the resource utilization limit if indicated by the destination Station
- the support of emergency services, i.e. allow a Station without proper security credentials to still place an emergency call

A basic component of the IEEE 802.11u protocol is the Generic Advertisement Service (GAS). GAS is a request-response protocol, which provides L2 transport mechanism between a wireless client and a server in the network prior to authentication. It provides functionality that enables STAs to discover the availability of information related to desired network services, e.g., information about services such as provided in an IBSS, local access services, available Subscription Service Providers (SSP) and/or SSPNs or other external networks. GAS uses a generic container to advertise network services' information over an IEEE 802.11 network. Public Action frames are used to transport this information. (3) In this research this component is crucial and will be modified in a way that the hotspot's data can be extracted.

1.5 Access Network Query Protocol

Access Network Query Protocol (ANQP) provides a range of information, such as IP address type and availability, roaming partners accessible through a hotspot, and the Extensible Authentication Protocol (EAP) method supported for authentication, for a query and response protocol. The ANQP Information Elements (IEs) provide additional data that can be sent from an IAP to the client to identify the IAP's network and service provider. If a client requests this information through a GAS query, the hotspot AP sends the ANQP capability list in the GAS Initial Response frame indicating support for the following IEs:

IEEE 802.11 ANQP Information Elements

- Venue Name: Provides zero or more venue names associated with the BSS to support the user's selection.
- Network Authentication Type: Provides a list of authentication types carrying additional information like support for online enrollment or redirection URL.
- Roaming Consortium: Provides a list of information about the Roaming Consortium and/or SSPs whose networks are accessible via this AP.
- IP Address Type Availability: Provides STA with the information about the availability of IP address version and type that could be allocated to the STA after successful association.
- NAI Realm: Provides a list of network access identifier (NAI) realms corresponding to SSPs or other entities whose networks or services are accessible via this AP; optionally amended by the list of EAP Method, which are supported by the SSPs.
- 3GPP Cellular Network: Contains cellular information such as network advertisement information e.g., network codes and country codes to assist a 3GPP non-AP STA in selecting an AP to access 3GPP networks.

- Domain Name: Provides a list of one or more domain names of the entity operating the IEEE 802.11 access network.

Hotspot 2.0 ANQP Information Elements

• HS Query list: Provides a list of identifiers of HS 2.0 ANQP elements for which the requesting mobile device is querying in a HS ANQP Query.

• HS Capability list: Provides a list of information/ capabilities that has been configured on an AP. The HS Capability list element is returned in response to a GAS Query Request.

• Operator Friendly Name: Zero or more operator names operating the IEEE 802.11 AN.

• WAN Metrics: Information about the WAN link connecting an IEEE 802.11 AN and the Internet.

• Connection Capability: Provides connection status of the most commonly used communications protocols and ports.

• NAI Home Realm Query: Used by the STA to determine if the NAI realms for which it has security credentials are realms corresponding to SPs or other entities whose networks or services are accessible via this BSS

• Operating Class Indication: Provides information on the groups of channels in the frequency band(s) the Wi-Fi access network is using

An advertisement server in the Hotspot Operator's network contains ANQP-elements or information that can be used to derive the required ANQP-elements. The information in the ANQP server can be obtained by the Access Network Query Protocol. An ANQP server can be co-located with an AP or in an external device. Throughout this specification where the text describes an ANQP-element as being provided by an AP, it is to be understood that the source of the message is an ANQP Server (4).

ANQP information transparently transmitted over the Generic Advertisement Service Protocol, a service that provides over-the-air transportation for frames of higher-layer advertisements between Wi-Fi stations (802.11 Stations) or between a server in an external network and a station. ANQP requests are passed in the Query Request field of GAS Initial Request and GAS Comeback Request frames, and ANQP answers - in the Query Response field of GAS Initial Response and GAS Comeback Response frames (5). Figure 3 shows a communications flow between a WLAN AP and a WLAN device (6).

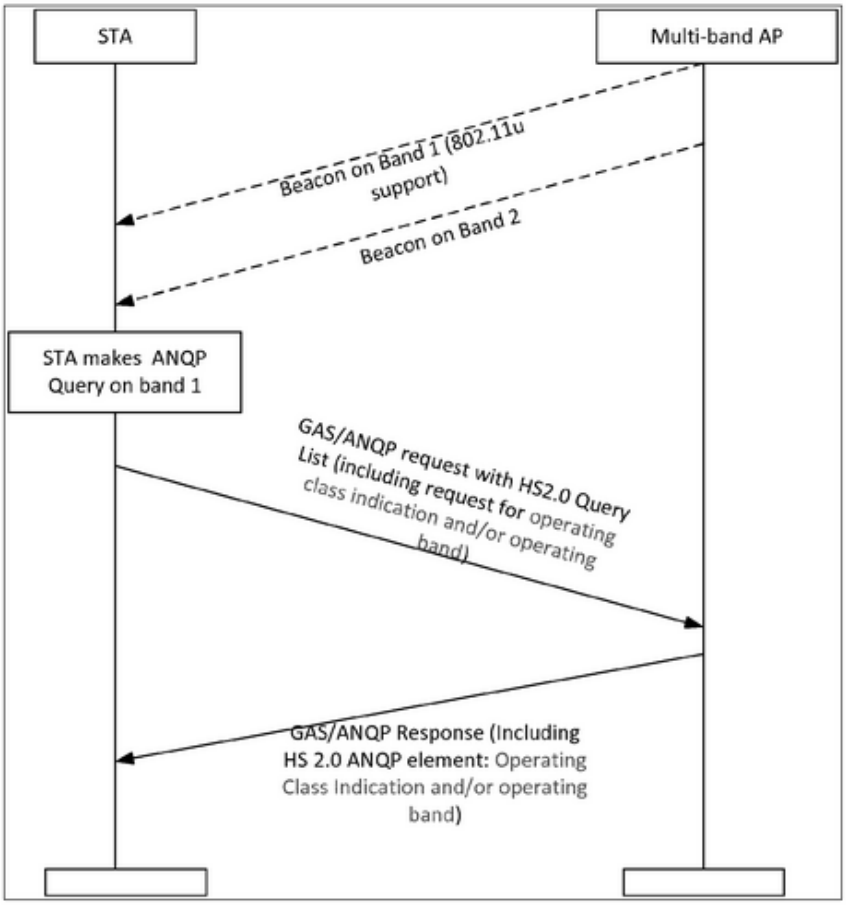

Figure 3 STA and AP basic communication

CHAPTER 2: System Design

2.1 Hostapd

Hostapd is a user space deamon for access points and authentication servers. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP Authenticators, RADIUS client, EAP server, and RADIUS authentication server. To communicate with a kernel driver hostapd has to use some interface. All new cfg80211 (and mac80211) based drivers that implement AP functionality are supported using nl80211 interface. For old kernel drivers hostapd contains separated drivers. As far a Linux is concerned, there are 3 other drivers you can use:

- **HostAP**
- **madwifi**
- **prism54**

Hostapd will be the Access Point of this Design Model and some specific characteristics will be added. Hostapd (version 2.3) is going to be installed in the Desktop where there is also the atheros antenna, in order to create an Access Point. In order to create a desirable access point, hostapd gives the option of configuring certain characteristics. The configuration file of hostpad is going to use a formation according to 802.11u protocol and some basic requirements that are needed in order to be compliant with the specifications set.

To begin with, the configuration included some specific ANQP, so that the necessary queries will be supported. The queries will be intercepted and asked later to the database. Those are:

- **Venue Name** : Provides a venue name to the hotspot
- **Roaming Consortium** : list of information about the Roaming Consortium and/or SSPs whose networks are accessible via this AP
- **Network Authentication Type** : indicates what type of network authentication is used in the network.
- **3GPP** : provides Cellular Network information

- **Domain Name** : provides a list of one or more domain names of the entity operating the IEEE 802.11 access network
- **IP Address Type Availability** : provides STA with the information about the availability of IP address version and type
- **WAN metrics :** provides information about the WAN link that connects the hotspot to the Internet.

The complete configuration file can be found in Appendix (hostapd.conf).

These features are helping enabling 802.11u and permit committing basic ANQP queries for the values that are being set. Therefore, clients can receive general information about the network and then associate with it. When a client scans for networks, the hotspot made using the hostapd daemon, will provide an SSID based profile that will establish the specific network available to the users. It is expected that in order for a user to associate to the hotspot a certain authentication will be needed, which is implemented according to the Hotspot 2.0 technology demands. In the following chapters the authentication that is used for this system design will be explained.

Once hostapd is ready and includes all the required amendments, a client can associate, establish a connection and extract all the ANQP elements. In this experiment a STA will be created in order to achieve that.

2.2 Station

Wpa_supplicant is a wpa supplicant for Linux BSD, Mac, and Windows with support for WPA and WPA2 authentication. It is suitable for both desktop/laptop computers and embedded systems. Supplicant is the IEEE 802.1X/WPA component that is used in the client stations It implements key negotiation with WPA Authenticator and it controls the roaming and IEEE 802.11 authentication/association of the WLAN driver. Moreover, it is an open source

client that can be used as a station (STA), in terms of connecting into a wireless network and reacting with it. It requires a basic configuration in order to associate with a hotspot and it supports 802.11u implementation. Also it includes a basic interface, wpa_cli, which will help sending ANQP queries to the hotspot.

In order to interact with wpa_supplicant, a text-based frontend program is needed, wpa_cli. It is used to query current status, change configuration, trigger events and request interactive user input. It supports two modes: interactive and command line. Both modes share the same command set and the main difference is in interactive mode, providing access to unsolicited messages (event messages, username/password requests).

In this system design, Wpa_supplicant is going to be installed in the laptop, in order to create the station (STA) and connect to the hotspot. Wpa_supplicant is going to be configured in a way that it implements the IEEE 802.11u protocol and a WPA2 Enterprise Authentication, like EAP-TLS, as it is required in the hotspot and will be explained in the following chapters.

Once an SSID-based profile is made the STA can immediately authenticate and connect with the hotspot. Through wpa_cli interface an ANQP fetch can be initialized from the interworking enabled network.

Wpa_supplicant gives the option to perform ANQP queries without associating. To do that there shouldn't be defined any network on the configuration file and try interworking through wpa_cli control interface. But in this project, once we use hotspot 2.0 technology, the complete authentication and connection is done.

The configuration file of wpa_supplicant is made according to the settings set to the hotspot. The complete file can be found in the Appendix (wpa_supplicant.conf).

2.3 Communication between Hotspot and Station

The connection between the Access Point and the client will be layer 2, based on the MAC address of a wireless antenna in the server. Layer 2 refers to the Data link layer of the Open Systems Interconnection (OSI). The data link layer is concerned with moving data across the physical links in the network. In a network, the switch is a device that redirects data messages at the layer 2 level using the destination Media Access Control (MAC) address to determine where to direct the message. (7)

The Data-Link layer contains two sublayers that are described in the IEEE-802 LAN standards:

- Media Access Control (MAC) sublayer
- Logical Link Control (LLC) sublayer

The data Link layer ensures that an initial connection has been set up and that incoming data has been received successfully by analyzing bit patterns at special places in the frames.

In the specific System Design there is a typical client server communication. Hotspot is the server and the station is the client. The hotspot has a MAC address from the Atheros Wireless Antenna. This is the one the station will use to establish the connection and recognize the SSID based profile of the network.

Wpa_supplicant provides commands in order to perform queries and test if any ANQP elements are declared. Each ANQP element is requested with a specific id.
Some of them are shown in table 1 (8).

ANQP Element	ID
Venue Name	258
Roaming Consortium	261
Network Authentication Type	260
3GPP	264
Domain Name	268

IP Address Type Availability	262

Table 1 ID of ANQP Elements

2.3.1 Authentication

Hotspot 2.0 (HS2), an approach to public access Wi-Fi, is based on the Wi-Fi Alliance technical specification of the same name and 802.11u. Because of the secure authentication that is needed, it is necessary for this technology to implement an authentication method according to WPA2 Enterprise, which is typically used with a customer- premise RADIUS server.

WPA2 is a security protocol and certification program to secure wireless computer networks. Though more complicated to set up, it offers individualized and centralized control over access to a Wi-Fi network. Users are assigned login credentials they must present when connection to a network and never deal with the actual encryption keys.

There are different EAP types supported to create this kind of authentication. In this experiment EAP TLS (Transport Layer Security) is used with self-signed certificates for the server and the client using RADIUS server for authentication which is supported in hostapd. This method is one of the most secure, but requires more implementation and maintenance. Both client and server validation is done via SSL certificates. Instead of providing a username and password when connecting, end-user devices or computers must have a SSL certificate file loaded into the client. SSL is an acronym for Secure Sockets Layer, an encryption technology that was created by Netscape**.** SSL creates an encrypted connection between a web server and a visitors' web browser allowing for private information to be transmitted without the problems of eavesdropping, data tampering, or message forgery.

In this system design, wpa_supplicant is going to be configured in a way that it implements a WPA2 Enterprise Authentication, like EAP-TLS, as it is required in the hotspot. To do so, a client certificate will be needed for RADIUS Authentication Server and the encryption methods are declared.

When a STA tries to associate with an AP a typical communication is performed. Figure 4 represents the basic packet flow between the client and the hotspot during a successful association attempt. The authentication process is shown and then an ANQP query request and response. At this point the values that are returned are coming from the configuration file, as they were set.

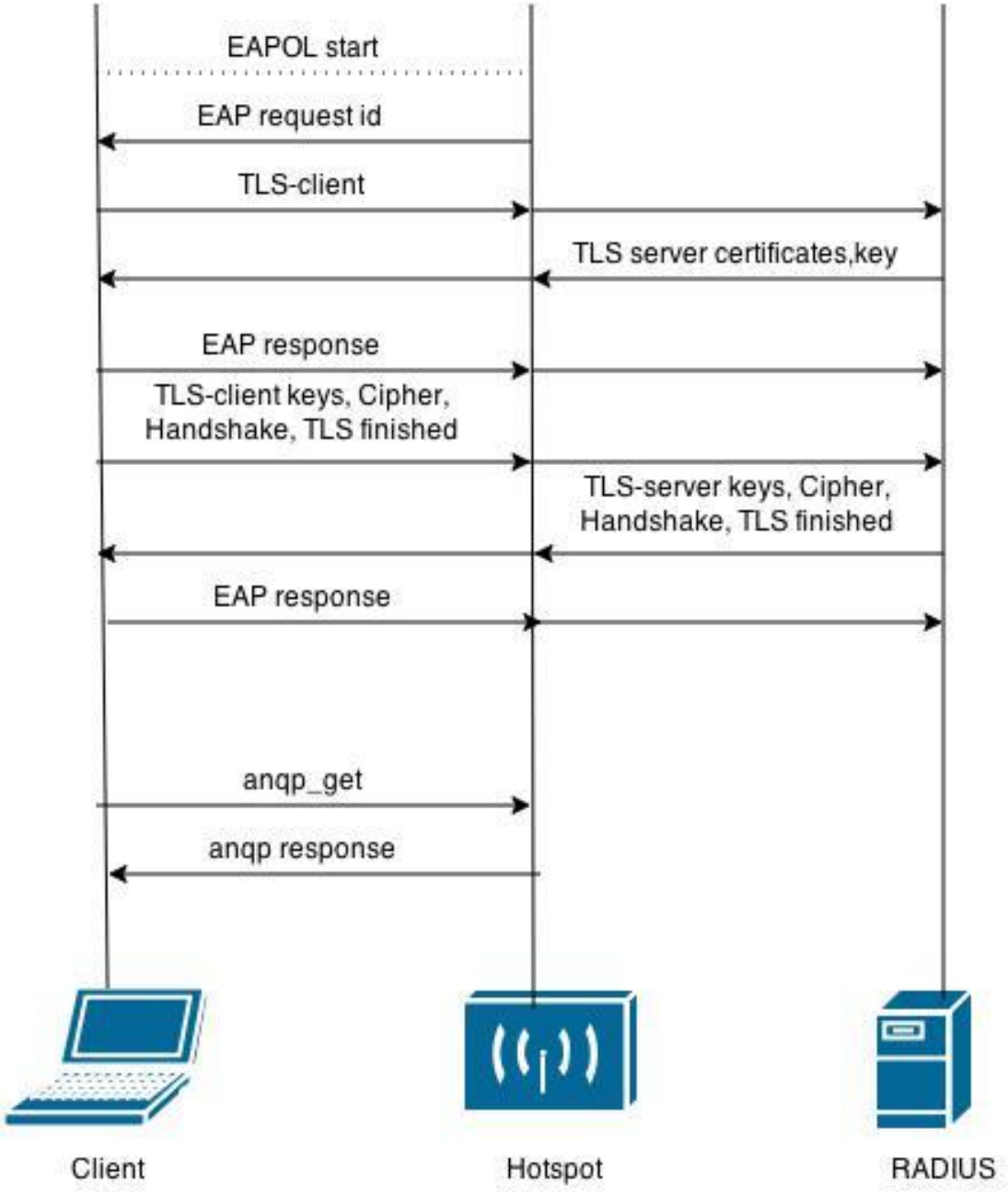

Figure 4 Authentication and Connection

2.4 ANQP Server with a Database

This System Design modifies the basic model described above according to the way ANQP elements will be exported and sent to the client. The idea is that all ANQP values are stored in a database and must be derived and sent to the client from there, when a query is made.

First of all, a simple database in mysql server stores all the necessary values of the ANQP elements. Each element is a table, which besides the value also includes an indexed mac address and an id. The database provides values for the following elements:

- Venue Name
- Roaming Consortium
- Network Authentication Type
- 3GPP
- Domain Name
- IP Address Type Availability
- WAN metrics

Some tables were created with the following format:

```
CREATE TABLE Table_Name (
id int NOT NULL AUTO_INCREMENT,
mac VARCHAR(17) DEFAULT '00:20:a6:ca:a4:d6',
v_name VARCHAR(100),
PRIMARY KEY (id)
);
CREATE UNIQUE INDEX mac_index
ON Table_Name (mac);
INSERT INTO Table_Name (mac,v_name) values ('00:20:a6:ca:a4:d6','value');
```

Table 2 Database format

In order to achieve a connection with hostapd and the database a middleware is needed. This middleware gives the desirable output from the query requested. To achieve that, a function is implemented in C that includes mysql functionality and interacts with a part of

hostapd specifically with the gas server. The sqlfunc.h is actually a SQL handler for hostapd that takes as input an SQL query and returns a single column value. The basic algorithm for its implementation is in Table 3 and the complete file can be found in Appendix (sqlfunc.h):

```
send_to_handler(query);
send_to_handler(String query){
        connect_with_sql();
        sql_query(query);
        response = get_sql_response();
        return response;
}
```

Table 3 Handler implementation

The server is also defined in this function, in this case "localhost", but can be changed so that it can work remotely with another server. That will enable us to have all the information in a centralized or distributed location and will give the opportunity to retrieve data from anywhere in the world. Thus, it will enable us to have unlimited hotspots with similar configuration with an easy way of managing the ANQP information. A future, more detailed implementation can use values for the database connection from a configuration file.

Hostapd needs to be modified in order to recognize SQL library. So, in the Makefile (/hostapd-2.3/hostapd/Makefile) there needs to be the following lines:

```
CFLAGS += „mysql_config –cflags“
LIBS += „mysql_config –libs“
```

The next changes will be in the gas_serv.c file (hostapd-2.3/src/ap). This file includes the code needed for the GAS server of the hotspot. Here we include the handler:

```
#include "sqlfunc.h"
```

Of course it is mandatory to adjust hostapd gas_serv.c file, so that the right sql queries will be made for each element and the responses will be forwarded. Gas_serv.c file contains functions, which initially derives data from the configuration file. The following function are

located inside the gas_serv.c file:

Static void anqp_add_venue_name(struct hostapd_data *hapd, struct wpabuf *buf)
Static void anqp_add_network_auth_type(struct hostapd_data *hapd, struct wpabuf *buf)
Static void anqp_add_roaming_consortium(struct hostapd_data *hapd, struct wpabuf *buf)
Static void anqp_add_ip_addr_type_availability(struct hostapd_data *hapd, struct wpabuf *buf)
Static void anqp_add_3gpp_cellural_network(struct hostapd_data *hapd, struct wpabuf *buf)
Static void anqp_add_domain_name(struct hostapd_data *hapd, struct wpabuf *buf)
Static void anqp_add_wan_metrics(struct hostapd_data *hapd, struct wpabuf *buf)

Table 4 ANQP methods in gas_serv.c file

These functions are implemented for each ANQP element and for the requirements of this research they have been modified using the following algorithm. The complete functions can be found in Appendix (gas_serv.c). The basic algorithm is in Table 5.

```
if (exists_in_database){
send_the_database_value();
}else{
run_initial_code();
}
```

Table 5 ANQP methods implementation

After the adjustments the specific functions send a query to the database, using sqlfunc.h, and get a response from the ANQP Database. The response is the ANQP values set in the database that subsequently will be sent to the client.

According to the above description some new components are added in the System Design. Except from the client, the hotspot and the authentication server, an ANQP Database requires, also a Handler. The main role of the Handler is to set the connection between the hotspot and the Database. All these components are shown in Figure 5.

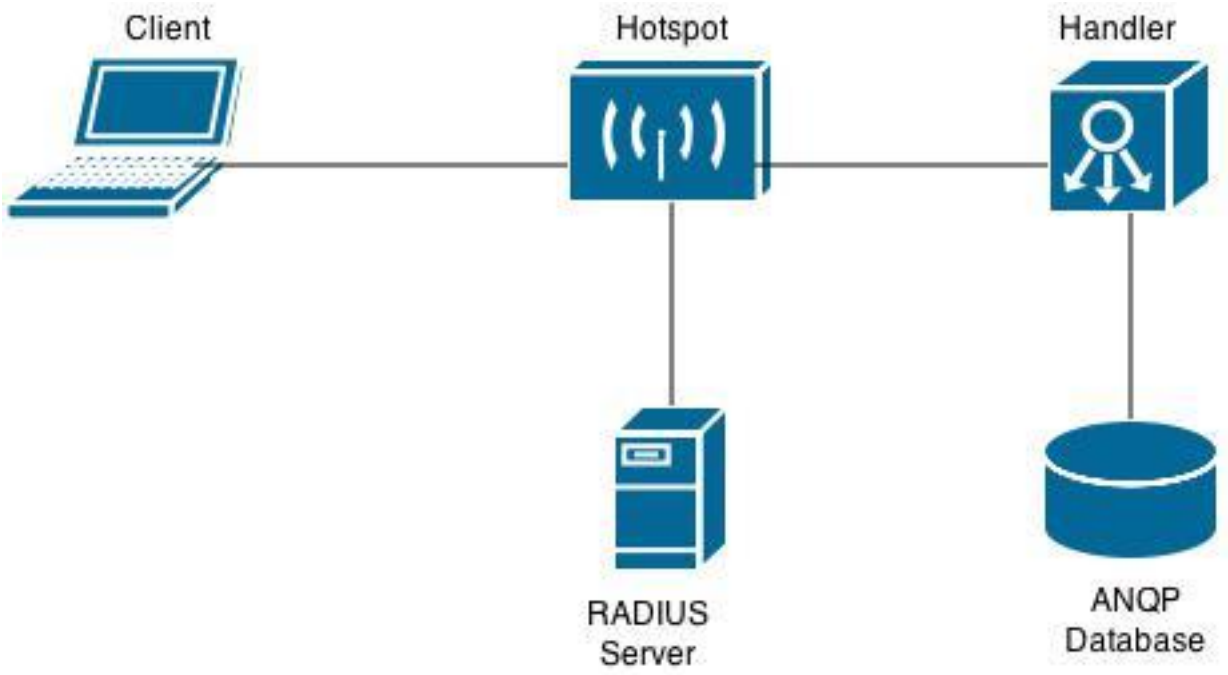

Figure 5 System Model Components

After these modifications the communication between the client and the Access Point differs from the initial. Figure 6 shows that communication including the database connection. The part of the authentication is not included as it is similar to the initial.

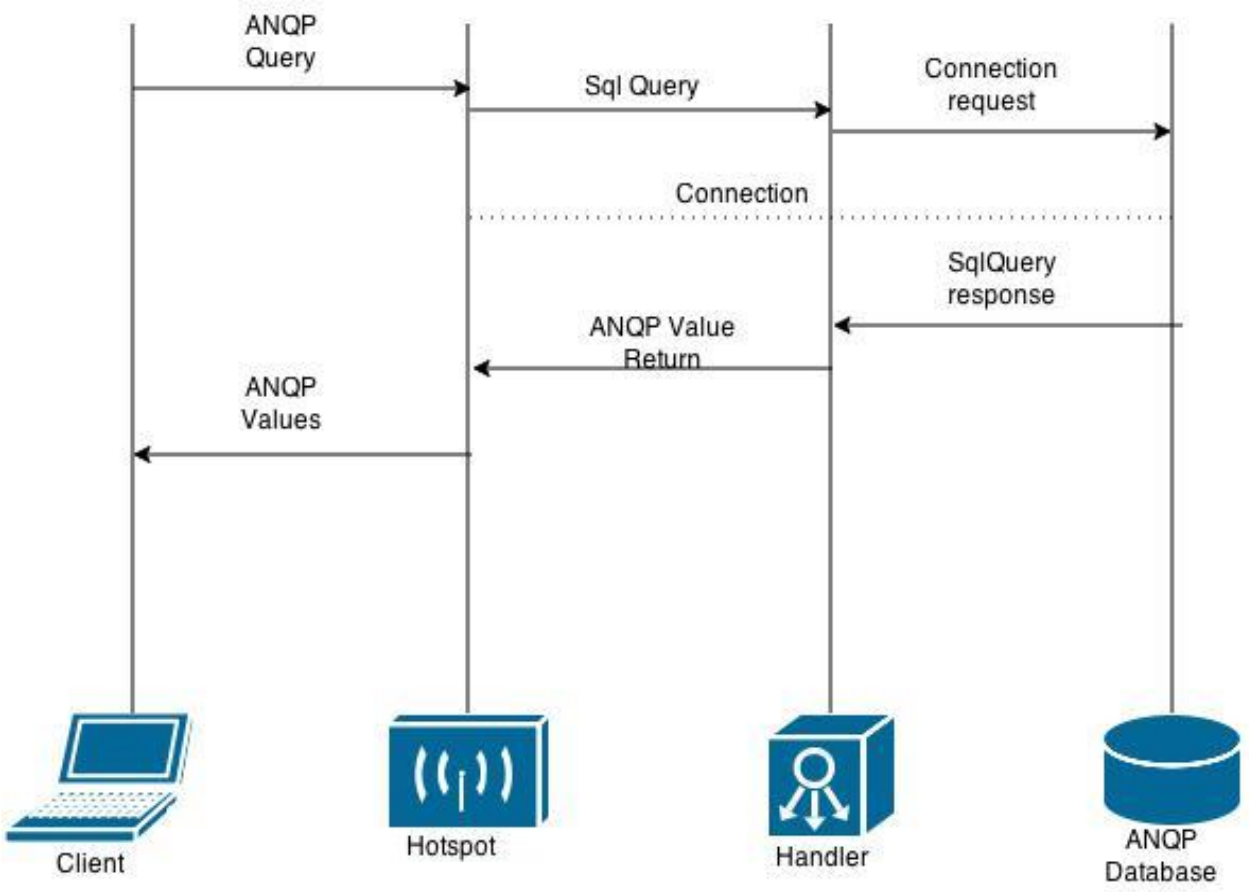

Figure 6 System Components communication

CHAPTER 3: Validation

3.1 Performing ANQP queries

This research main purpose is to test how an Access Point and a client work with IEEE 802.11u protocol in terms of performing ANQP queries. ANQP queries are mandatory in order to extract all the ANQP elements and help the user to select a hotspot. The examination will start first locally, which means there will be only two components: the client and the hotspot. An assumption can be made here, that the results will be derived from the configuration file of the hotspot.

The basic elements that are needed in order to perform an ANQP query are:

- The MAC Address
- The ANQP ID of the specific ANQP element
- The wpa_cli interface of wpa_supplicant

In order to check if the hotspot supports a certain ANQP element, the following format of an ANQP query id performed:

```
>anqp_get <MAC Address> <id>
```

For example if the venue name of the hotspot needs to be derived:

```
> anqp_get 00:20:a6:ca:a4:d6 258
```

With the following result:

```
OK
<3>GAS-QUERY-START addr=00:20:a6:ca:a4:d6 dialog_token=4 freq=2412
<3>GAS-QUERY-DONE addr=00:20:a6:ca:a4:d6 dialog_token=4 freq=2412
status_code=0
result=SUCCESS
<3>RX-ANQP 00:20:a6:ca:a4:d6 Venue Name
```

Table 6 ANQP query results for a specific element

In this way we can see all the ANQP elements in the hotspot, as long as there is an id. This shows that the hotspot has indeed a venue name but without presenting the actual value.

3.2 Results

All the elements included in the hotspot are requested with an anqp_fetch command.

Wpa_ supplicant returns the actual values of the elements in a hexadecimal format with the bss command. The format of this is:

bss <MAC Address>

The results from this, when hostapd is extracting data locally, from the configuration file are in Table 7:

```
bss 00:20:a6:ca:a4:d6
> id=1
bssid=00:20:a6:ca:a4:d6
freq=2412
beacon_int=100
capabilities=0x0411
qual=0
noise=0
level=-35
tsf=0000000010852027
age=692
ie=000474657374010882848b960c1218240301012a010432043048606c30140100000fac0
40100000fac040100000fac010c007f080000008000400040 6b09d407010020a6caa4d66c02
7f006f0700052233445566dd180050f2020101000003a4000027a4000042435e0062322f00d
d07506f9a10140000
flags=[WPA2-EAP-CCMP][ESS][HS20]
ssid=test
anqp_venue_name=07010d656e67746573742076656e7565
anqp_network_auth_type=000000
anqp_roaming_consortium=052233445566
anqp_ip_addr_type_availability=14
anqp_3gpp=000600040142f419
anqp_domain_name=08746573742e636f6d
hs20_wan_metrics=01401f0000e803000050f0b80b
```

Table 7 ANQP query results for all ANQP elements in a local response

All ANQP elements defined in the configuration file are shown here with a value given. Because of the client, the results are hexadecimal values and need to be converted.

Subsequently, the same ANQP query will be performed to the System Design that is implemented in this research, with the ANQP server. This time the results will be derived from the database, in the form they were stored there.

The result looks very similar but this time the ANQP elements get the values that was set in the database. So, when all the values are requested the result is the following:

```
bss 00:20:a6:ca:a4:d6
> id=2
bssid=00:20:a6:ca:a4:d6
freq=2412
beacon_int=100
capabilities=0x0411
qual=0
noise=0
level=-38
tsf=0000000005167759
age=37
ie=000474657374010882848b960c1218240301012a010432043048606c30140100000fac0
40100000fac040100000fac010c007f0800000080004000406b09d407010020a6caa4d66c02
7f006f0700052233445566dd180050f2020101000003a4000027a4000042435e0062322f00d
d07506f9a10140000
flags=[WPA2-EAP-CCMP][ESS][HS20]
ssid=test
anqp_venue_name=07010e656e673a746573742056656e7565
anqp_network_auth_type=41636365074616e6365206f66207465726d7320616e6420636f
6e646974696f6e73
anqp_roaming_consortium=0a32323333343435353636
anqp_ip_addr_type_availability=41646472657373207479706206e6f7420617661696c61
626c65
anqp_3gpp=3234342c3931
anqp_domain_name=746573742e636f6d
hs20_wan_metrics=30313a383030303a313030303a38303a3234303a33303030
```

Table 8 ANQP query results for all ANQP elements for a remote response

3.3 Comparison

It is obvious that a convert needs to be made, in order to see what actual values are returned in each system design. To do that a simple converter is used from hexadecimal values to string. Only the modified System with the ANQP server suggested in this research returns actual string values that where set in the database. In the following table the conversion is shown:

ANQP Element	Hex Value	String Value
Venue Name	07010e656e673a746573742056656e7565	Eng:test Venue
Network Authentication Type	416363657074616e6365206f662074657 26d7320616e6420636f6e646974696f6e73	Acceptance of terms and conditions
Roaming Consortium	0a32323333343435353636	2233445566
IP Address Type	416464726573732074797 065206e6f7420617661696c61626c65	Address type not available
3 GPP	3234342c3931	244,91
Domain Name	746573742e636f6d	Test.com
WAN Metrics	30313a383030303a313030 303a38303a3234303a33303030	01:8000:1000:80:240:3000

Table 9 Conversion of hexadecimal values to String values

It is certain that with the System Design suggested handling the ANQP elements is more clear and accurate, because the values can be set with a more adaptive way. It is, by all means, more efficient to handle a simple database and extract the necessary data from there. Moreover it can model many different Wi-Fi hotspots, according to the needs of each one.

To compare the ANQP response time of local and remote a sample of measurements was collected for both approaches. The measurements are made from the time a query begins until

the time the results are shown and the operation is completed. The measurements are gathered in boxplots (Figure 7) so the comparison can be clearer. Box plots are useful for identifying outliers and for comparing distributions.

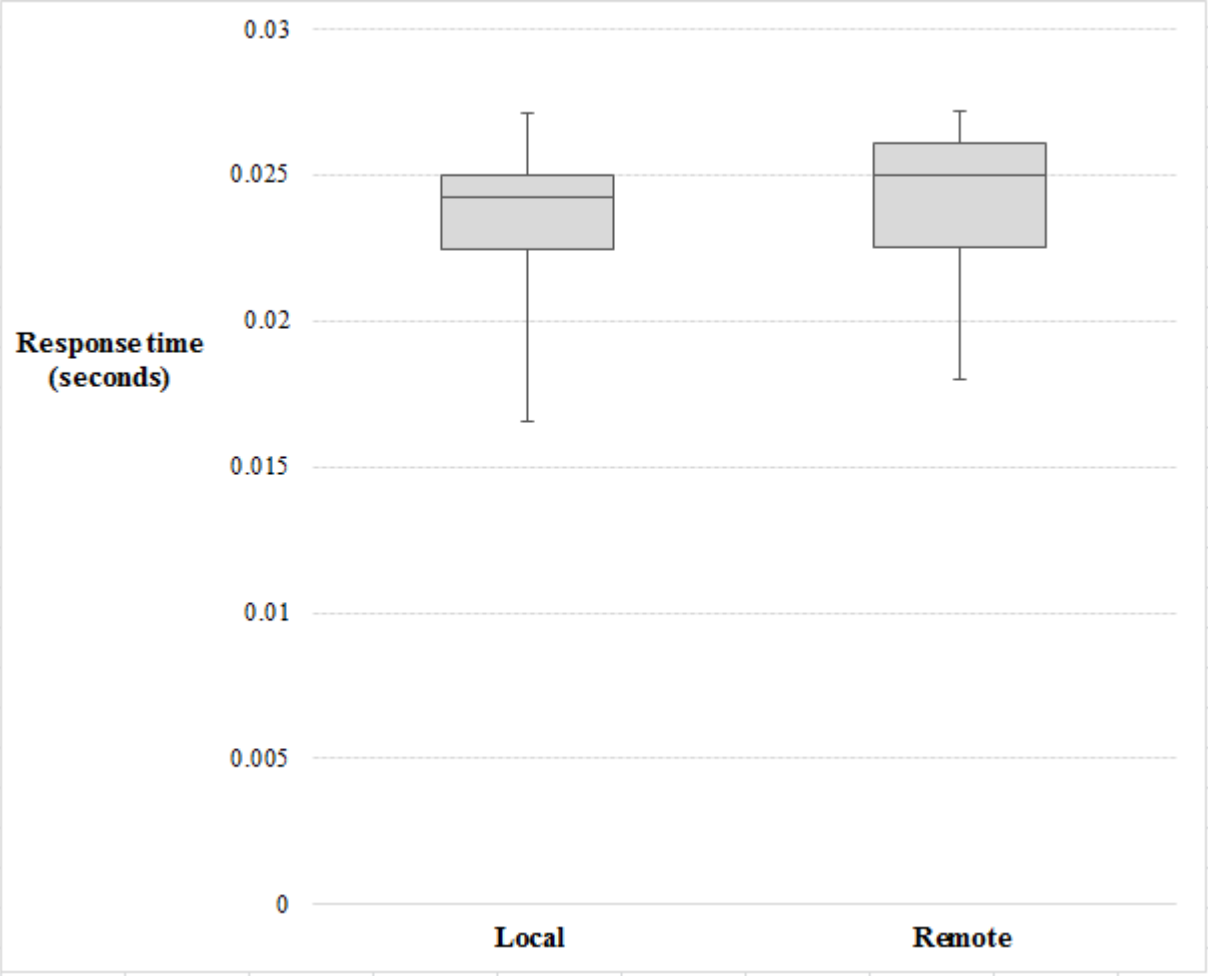

Figure 7 Boxplots of local and remote response

Response	Local	Remote
Population size	20	20
Median	0.2426817	0.24986417

Minimum	0.016530175	0.017982245
Maximum	0.027138857	0.027198012

Table 10 Local and Remote response time

It is clear in the Figure 9 and 10 that the local response and the remote response have a difference of one hop. It is expected that with a remote SQL server the response time will significantly increase, but this delay will occur from the network connection delay that exists in every TCP/IP connection over the World-Wide-Web). This delay can be from 1 millisecond to some hundred milliseconds, always depending of the connection and not in the implementation, thus we decided to not take this delay in account. Of course we need to consider that having a more complex database will surely have an effect on the response time, but unless we have millions of entries this effect will be minimal. In that case, probably it will be mandatory to improve the performance of the database with certain methods for optimization.

Conclusion

This study analyzes an ANQP server implementation based on a design model, in which ANQP queries are requested from a client to a hotspot, that support 802.11u and hotspot 2.0. The responses can be sent locally from the initial configuration file or remotely from a database with ANQP elements. We have shown that even with a database connection, the results will be delivered on time, without any significant delay, thus it if feasible to use an SQL database and an SQL handler to retrieve the ANQP values.

Future Work

The specific research can be the inception for further experimenting and testing in the specific field. First of all the System model can be modified so that hostapd can send directly

the ANQP query to a Controller. This Controller will form the equal SQL query for the database. Probably a Python API can be used in order to establish that kind of communication.

Regarding the testing of the response time, some more specimens must be taken to have a better view of the System Model. Moreover it would be interesting to exam the response time between each component. Having those results will help understand in which steps there are longer delays.

Finally, it would be very interesting to see this System working on a different client. For example, using an actual smartphone in order to receive real results from a hotspot.

APPENDIX

Hostapd.conf

```
##### hostapd configuration file ##################################################
# Empty lines and lines starting with # are ignored

# AP netdevice name (without 'ap' postfix, i.e., wlan0 uses wlan0ap for
# management frames); ath0 for madwifi
interface=wlan0

# hostapd event logger configuration
#
# Two output method: syslog and stdout (only usable if not forking to
# background).
#
# Module bitfield (ORed bitfield of modules that will be logged; -1 = all
# modules):
# bit 0 (1) = IEEE 802.11
# bit 1 (2) = IEEE 802.1X
# bit 2 (4) = RADIUS
# bit 3 (8) = WPA
# bit 4 (16) = driver interface
# bit 5 (32) = IAPP
# bit 6 (64) = MLME
#
# Levels (minimum value for logged events):
#  0 = verbose debugging
#  1 = debugging
#  2 = informational messages
#  3 = notification
#  4 = warning
#
logger_syslog=-1
```

logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2
debug=4
Interface for separate control program. If this is specified, hostapd
will create this directory and a UNIX domain socket for listening to requests
from external programs (CLI/GUI, etc.) for status information and
configuration. The socket file will be named based on the interface name, so
multiple hostapd processes/interfaces can be run at the same time if more
than one interface is used.
/var/run/hostapd is the recommended directory for sockets and by default,
hostapd_cli will use it when trying to connect with hostapd.
ctrl_interface=/var/run/hostapd

Access control for the control interface can be configured by setting the
directory to allow only members of a group to use sockets. This way, it is
possible to run hostapd as root (since it needs to change network
configuration and open raw sockets) and still allow GUI/CLI components to be
run as non-root users. However, since the control interface can be used to
change the network configuration, this access needs to be protected in many
cases. By default, hostapd is configured to use gid 0 (root). If you
want to allow non-root users to use the contron interface, add a new group
and change this value to match with that group. Add users that should have
control interface access to this group.
#
This variable can be a group name or gid.
#ctrl_interface_group=wheel
ctrl_interface_group=0

IEEE 802.11 related configuration

SSID to be used in IEEE 802.11 management frames
ssid=test
Alternative formats for configuring SSID
(double quoted string, hexdump, printf-escaped string)
#ssid2="test"
#ssid2=74657374
#ssid2=P"hello\nthere"

UTF-8 SSID: Whether the SSID is to be interpreted using UTF-8 encoding
#utf8_ssid=1

Country code (ISO/IEC 3166-1). Used to set regulatory domain.
Set as needed to indicate country in which device is operating.
This can limit available channels and transmit power.
#country_code=US

Operation mode (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g,
ad = IEEE 802.11ad (60 GHz); a/g options are used with IEEE 802.11n, too, to
specify band)
Default: IEEE 802.11b
hw_mode=g

Channel number (IEEE 802.11)
(default: 0, i.e., not set)
Please note that some drivers do not use this value from hostapd and the
channel will need to be configured separately with iwconfig.
#
If CONFIG_ACS build option is enabled, the channel can be selected
automatically at run time by setting channel=acs_survey or channel=0, both of

```
# which will enable the ACS survey based algorithm.
channel=1

# Beacon interval in kus (1.024 ms) (default: 100; range 15..65535)
beacon_int=100

# DTIM (delivery traffic information message) period (range 1..255):
# number of beacons between DTIMs (1 = every beacon includes DTIM element)
# (default: 2)
dtim_period=2

# Maximum number of stations allowed in station table. New stations will be
# rejected after the station table is full. IEEE 802.11 has a limit of 2007
# different association IDs, so this number should not be larger than that.
# (default: 2007)
max_num_sta=255

# Fragmentation threshold; 2346 = disabled (default); range 256..2346
# If this field is not included in hostapd.conf, hostapd will not control
# fragmentation threshold and 'iwconfig wlan# frag <val>' can be used to set
# it.
fragm_threshold=2346

# Station MAC address -based authentication
# Please note that this kind of access control requires a driver that uses
# hostapd to take care of management frame processing and as such, this can be
# used with driver=hostap or driver=nl80211, but not with driver=madwifi.
# 0 = accept unless in deny list
# 1 = deny unless in accept list
# 2 = use external RADIUS server (accept/deny lists are searched first)
macaddr_acl=0

# IEEE 802.11 specifies two authentication algorithms. hostapd can be
# configured to allow both of these or only one. Open system authentication
# should be used with IEEE 802.1X.
# Bit fields of allowed authentication algorithms:
# bit 0 = Open System Authentication
# bit 1 = Shared Key Authentication (requires WEP)
auth_algs=3

# Send empty SSID in beacons and ignore probe request frames that do not
# specify full SSID, i.e., require stations to know SSID.
# default: disabled (0)
# 1 = send empty (length=0) SSID in beacon and ignore probe request for
#     broadcast SSID
# 2 = clear SSID (ASCII 0), but keep the original length (this may be required
#     with some clients that do not support empty SSID) and ignore probe
#     requests for broadcast SSID
ignore_broadcast_ssid=0

# Default WMM parameters (IEEE 802.11 draft; 11-03-0504-03-000e):
# for 802.11a or 802.11g networks
# These parameters are sent to WMM clients when they associate.
# The parameters will be used by WMM clients for frames transmitted to the
# access point.
#
# note - txop_limit is in units of 32microseconds
# note - acm is admission control mandatory flag. 0 = admission control not
# required, 1 = mandatory
# note - here cwMin and cmMax are in exponent form. the actual cw value used
# will be (2^n)-1 where n is the value given here
```

#
wmm_enabled=1
#
WMM-PS Unscheduled Automatic Power Save Delivery [U-APSD]
Enable this flag if U-APSD supported outside hostapd (eg., Firmware/driver)
#uapsd_advertisement_enabled=1
#
Low priority / AC_BK = background
wmm_ac_bk_cwmin=4
wmm_ac_bk_cwmax=10
wmm_ac_bk_aifs=7
wmm_ac_bk_txop_limit=0
wmm_ac_bk_acm=0
Note: for IEEE 802.11b mode: cWmin=5 cWmax=10
#
Normal priority / AC_BE = best effort
.
Note: for IEEE 802.11b mode: cWmin=5 cWmax=7
#
High priority / AC_VI = video
wmm_ac_vi_aifs=2
wmm_ac_vi_cwmin=3
wmm_ac_vi_cwmax=4
wmm_ac_vi_txop_limit=94
wmm_ac_vi_acm=0
Note: for IEEE 802.11b mode: cWmin=4 cWmax=5 txop_limit=188
#
Highest priority / AC_VO = voice
wmm_ac_vo_aifs=2
wmm_ac_vo_cwmin=2
wmm_ac_vo_cwmax=3
wmm_ac_vo_txop_limit=47
wmm_ac_vo_acm=0
Note: for IEEE 802.11b mode: cWmin=3 cWmax=4 burst=102

Static WEP key configuration
#
The key number to use when transmitting.
It must be between 0 and 3, and the corresponding key must be set.
default: not set
#wep_default_key=0
The WEP keys to use.
A key may be a quoted string or unquoted hexadecimal digits.
The key length should be 5, 13, or 16 characters, or 10, 26, or 32
digits, depending on whether 40-bit (64-bit), 104-bit (128-bit), or
128-bit (152-bit) WEP is used.
Only the default key must be supplied; the others are optional.
default: not set
#wep_key0=123456789a
#wep_key1="vwxyz"
#wep_key2=0102030405060708090a0b0c0d
#wep_key3=".2.4.6.8.0.23"

IEEE 802.1X-2004 related configuration

Require IEEE 802.1X authorization
ieee8021x=1

EAPOL-Key index workaround (set bit7) for WinXP Supplicant (needed only if
only broadcast keys are used)
eapol_key_index_workaround=0

Integrated EAP server

Optionally, hostapd can be configured to use an integrated EAP server
to process EAP authentication locally without need for an external RADIUS
server. This functionality can be used both as a local authentication server
for IEEE 802.1X/EAPOL and as a RADIUS server for other devices.

Use integrated EAP server instead of external RADIUS authentication
server. This is also needed if hostapd is configured to act as a RADIUS
authentication server.
eap_server=1

Path for EAP server user database
If SQLite support is included, this can be set to "sqlite:/path/to/sqlite.db"
to use SQLite database instead of a text file.
eap_user_file=/etc/hostapd.eap_user

CA certificate (PEM or DER file) for EAP-TLS/PEAP/TTLS
ca_cert=/CA2/ca.pem

Server certificate (PEM or DER file) for EAP-TLS/PEAP/TTLS
server_cert=/CA2/server/server.pem

Private key matching with the server certificate for EAP-TLS/PEAP/TTLS
This may point to the same file as server_cert if both certificate and key
are included in a single file. PKCS#12 (PFX) file (.p12/.pfx) can also be
used by commenting out server_cert and specifying the PFX file as the
private_key.
private_key=/CA2/server/server.key

Passphrase for private key
private_key_passwd=netcom;

RADIUS client configuration
for IEEE 802.1X with external Authentication Server, IEEE 802.11
authentication with external ACL for MAC addresses, and accounting

The own IP address of the access point (used as NAS-IP-Address)
own_ip_addr=127.0.0.1

RADIUS authentication server configuration

hostapd can be used as a RADIUS authentication server for other hosts. This
requires that the integrated EAP server is also enabled and both
authentication services are sharing the same configuration.

File name of the RADIUS clients configuration for the RADIUS server. If this
commented out, RADIUS server is disabled.
radius_server_clients=/etc/hostapd.radius_clients

The UDP port number for the RADIUS authentication server
radius_server_auth_port=1812

WPA/IEEE 802.11i configuration

Enable WPA. Setting this variable configures the AP to require WPA (either
WPA-PSK or WPA-RADIUS/EAP based on other configuration). For WPA-PSK, either

```
# wpa_psk or wpa_passphrase must be set and wpa_key_mgmt must include WPA-PSK.
# Instead of wpa_psk / wpa_passphrase, wpa_psk_radius might suffice.
# For WPA-RADIUS/EAP, ieee8021x must be set (but without dynamic WEP keys),
# RADIUS authentication server must be configured, and WPA-EAP must be included
# in wpa_key_mgmt.
# This field is a bit field that can be used to enable WPA (IEEE 802.11i/D3.0)
# and/or WPA2 (full IEEE 802.11i/RSN):
# bit0 = WPA
# bit1 = IEEE 802.11i/RSN (WPA2) (dot11RSNAEnabled)
#wpa=1
wpa=2

# Set of accepted key management algorithms (WPA-PSK, WPA-EAP, or both). The
# entries are separated with a space. WPA-PSK-SHA256 and WPA-EAP-SHA256 can be
# added to enable SHA256-based stronger algorithms.
# (dot11RSNAConfigAuthenticationSuitesTable)
#wpa_key_mgmt=WPA-PSK WPA-EAP
wpa_key_mgmt=WPA-EAP

# Set of accepted cipher suites (encryption algorithms) for pairwise keys
# (unicast packets). This is a space separated list of algorithms:
# CCMP = AES in Counter mode with CBC-MAC [RFC 3610, IEEE 802.11i/D7.0]
# TKIP = Temporal Key Integrity Protocol [IEEE 802.11i/D7.0]
# Group cipher suite (encryption algorithm for broadcast and multicast frames)
# is automatically selected based on this configuration. If only CCMP is
# allowed as the pairwise cipher, group cipher will also be CCMP. Otherwise,
# TKIP will be used as the group cipher.
# (dot11RSNAConfigPairwiseCiphersTable)
# Pairwise cipher for WPA (v1) (default: TKIP)
#wpa_pairwise=TKIP CCMP
wpa_pairwise=CCMP

# Pairwise cipher for RSN/WPA2 (default: use wpa_pairwise value)
rsn_pairwise=CCMP

##### IEEE 802.11u-2011 ###############################################################

# Enable Interworking service
interworking=1

# Access Network Type
# 0 = Private network
# 1 = Private network with guest access
# 2 = Chargeable public network
# 3 = Free public network
# 4 = Personal device network
# 5 = Emergency services only network
# 14 = Test or experimental
# 15 = Wildcard
access_network_type=4

# Whether the network provides connectivity to the Internet
# 0 = Unspecified
# 1 = Network provides connectivity to the Internet
internet=1

# Additional Step Required for Access
# Note: This is only used with open network, i.e., ASRA shall ne set to 0 if
# RSN is used.
asra=0
```

```
# Emergency services reachable
esr=1

# Unauthenticated emergency service accessible
uesa=1

# Venue Info (optional)
# The available values are defined in IEEE Std 802.11u-2011, 7.3.1.34.
# Example values (group,type):
# 0,0 = Unspecified
# 1,7 = Convention Center
# 1,13 = Coffee Shop
# 2,0 = Unspecified Business
# 7,1  Private Residence
venue_group=7
venue_type=1

# Homogeneous ESS identifier (optional; dot11HESSID)
# If set, this shall be identifical to one of the BSSIDs in the homogeneous
# ESS and this shall be set to the same value across all BSSs in homogeneous
# ESS.
hessid=00:20:a6:ca:a4:d6

# Roaming Consortium List
# Arbitrary number of Roaming Consortium OIs can be configured with each line
# adding a new OI to the list. The first three entries are available through
# Beacon and Probe Response frames. Any additional entry will be available only
# through ANQP queries. Each OI is between 3 and 15 octets and is configured as
# a hexstring.
#roaming_consortium=021122
roaming_consortium=2233445566

# Venue Name information
# This parameter can be used to configure one or more Venue Name Duples for
# Venue Name ANQP information. Each entry has a two or three character language
# code (ISO-639) separated by colon from the venue name string.
# Note that venue_group and venue_type have to be set for Venue Name
# information to be complete.
#venue_name=eng:Example venue
#venue_name=fin:Esimerkkipaikka
# Alternative format for language:value strings:
# (double quoted string, printf-escaped string)
venue_name=P"eng:test venue"

# Network Authentication Type
# This parameter indicates what type of network authentication is used in the
# network.
# format: <network auth type indicator (1-octet hex str)> [redirect URL]
# Network Authentication Type Indicator values:
# 00 = Acceptance of terms and conditions
# 01 = On-line enrollment supported
# 02 = http/https redirection
# 03 = DNS redirection
network_auth_type=00
#network_auth_type=02http://www.example.com/redirect/me/here/

# IP Address Type Availability
# format: <1-octet encoded value as hex str>
# (ipv4_type & 0x3f) << 2 | (ipv6_type & 0x3)
# ipv4_type:
# 0 = Address type not available
```

```
# 1 = Public IPv4 address available
# 2 = Port-restricted IPv4 address available
# 3 = Single NATed private IPv4 address available
# 4 = Double NATed private IPv4 address available
# 5 = Port-restricted IPv4 address and single NATed IPv4 address available
# 6 = Port-restricted IPv4 address and double NATed IPv4 address available
# 7 = Availability of the address type is not known
# ipv6_type:
# 0 = Address type not available
# 1 = Address type available
# 2 = Availability of the address type not known
ipaddr_type_availability=14

# Domain Name
# format: <variable-octet str>[,<variable-octet str>]
#domain_name=example.com,another.example.com,yet-another.example.com
domain_name=test.com

# 3GPP Cellular Network information
# format: <MCC1,MNC1>[;<MCC2,MNC2>][;...]
#anqp_3gpp_cell_net=244,91;310,026;234,56
anqp_3gpp_cell_net=244,91

# NAI Realm information
# One or more realm can be advertised. Each nai_realm line adds a new realm to
# the set. These parameters provide information for stations using Interworking
# network selection to allow automatic connection to a network based on
# credentials.
# format: <encoding>,<NAI Realm(s)>[,<EAP Method 1>][,<EAP Method 2>][,...]
# encoding:
#       0 = Realm formatted in accordance with IETF RFC 4282
#       1 = UTF-8 formatted character string that is not formatted in
#           accordance with IETF RFC 4282
# NAI Realm(s): Semi-colon delimited NAI Realm(s)
# EAP Method: <EAP Method>[:<[AuthParam1:Val1]>][<[AuthParam2:Val2]>][...]
# EAP Method types, see:
# http://www.iana.org/assignments/eap-numbers/eap-numbers.xhtml#eap-numbers-4
# AuthParam (Table 8-188 in IEEE Std 802.11-2012):
# ID 2 = Non-EAP Inner Authentication Type
#       1 = PAP, 2 = CHAP, 3 = MSCHAP, 4 = MSCHAPV2
# ID 3 = Inner authentication EAP Method Type
# ID 5 = Credential Type
#       1 = SIM, 2 = USIM, 3 = NFC Secure Element, 4 = Hardware Token,
#       5 = Softoken, 6 = Certificate, 7 = username/password, 9 = Anonymous,
#       10 = Vendor Specific
#nai_realm=0,example.com;example.net
# EAP methods EAP-TLS with certificate and EAP-TTLS/MSCHAPv2 with
# username/password
#nai_realm=0,example.org,13[5:6],21[2:4][5:7]

##### Hotspot 2.0 #############################################################

# Enable Hotspot 2.0 support
hs20=1

# Disable Downstream Group-Addressed Forwarding (DGAF)
# This can be used to configure a network where no group-addressed frames are
# allowed. The AP will not forward any group-address frames to the stations and
# random GTKs are issued for each station to prevent associated stations from
# forging such frames to other stations in the BSS.
#disable_dgaf=1
```

```
# WAN Metrics
# format: <WAN Info>:<DL Speed>:<UL Speed>:<DL Load>:<UL Load>:<LMD>
# WAN Info: B0-B1: Link Status, B2: Symmetric Link, B3: At Capabity
#   (encoded as two hex digits)
#   Link Status: 1 = Link up, 2 = Link down, 3 = Link in test state
# Downlink Speed: Estimate of WAN backhaul link current downlink speed in kbps;
#           1..4294967295; 0 = unknown
# Uplink Speed: Estimate of WAN backhaul link current uplink speed in kbps
#           1..4294967295; 0 = unknown
# Downlink Load: Current load of downlink WAN connection (scaled to 255 = 100%)
# Uplink Load: Current load of uplink WAN connection (scaled to 255 = 100%)
# Load Measurement Duration: Duration for measuring downlink/uplink load in
# tenths of a second (1..65535); 0 if load cannot be determined
hs20_wan_metrics=01:8000:1000:80:240:3000
```

wpa_supplicant.conf

```
##### Example wpa_supplicant configuration file ###################################
#
# This file describes configuration file format and lists all available option.
# Please also take a look at simpler configuration examples in 'examples'
# subdirectory.
#
# Empty lines and lines starting with # are ignored

# NOTE! This file may contain password information and should probably be made
# readable only by root user on multiuser systems.

# Note: All file paths in this configuration file should use full (absolute,
# not relative to working directory) path in order to allow working directory
# to be changed. This can happen if wpa_supplicant is run in the background.

# Whether to allow wpa_supplicant to update (overwrite) configuration
#
# This option can be used to allow wpa_supplicant to overwrite configuration
# file whenever configuration is changed (e.g., new network block is added with
# wpa_cli or wpa_gui, or a password is changed). This is required for
# wpa_cli/wpa_gui to be able to store the configuration changes permanently.
# Please note that overwriting configuration file will remove the comments from
# it.
update_config=1

# global configuration (shared by all network blocks)
#
# Parameters for the control interface. If this is specified, wpa_supplicant
# will open a control interface that is available for external programs to
# manage wpa_supplicant. The meaning of this string depends on which control
# interface mechanism is used. For all cases, the existence of this parameter
# in configuration is used to determine whether the control interface is
# enabled.
#
# For UNIX domain sockets (default on Linux and BSD): This is a directory that
```

will be created for UNIX domain sockets for listening to requests from
external programs (CLI/GUI, etc.) for status information and configuration.
The socket file will be named based on the interface name, so multiple
wpa_supplicant processes can be run at the same time if more than one
interface is used.
/var/run/wpa_supplicant is the recommended directory for sockets and by
default, wpa_cli will use it when trying to connect with wpa_supplicant.
#
Access control for the control interface can be configured by setting the
directory to allow only members of a group to use sockets. This way, it is
possible to run wpa_supplicant as root (since it needs to change network
configuration and open raw sockets) and still allow GUI/CLI components to be
run as non-root users. However, since the control interface can be used to
change the network configuration, this access needs to be protected in many
cases. By default, wpa_supplicant is configured to use gid 0 (root). If you
want to allow non-root users to use the control interface, add a new group
and change this value to match with that group. Add users that should have
control interface access to this group. If this variable is commented out or
not included in the configuration file, group will not be changed from the
value it got by default when the directory or socket was created.
#
When configuring both the directory and group, use following format:
DIR=/var/run/wpa_supplicant GROUP=wheel
DIR=/var/run/wpa_supplicant GROUP=0
(group can be either group name or gid)
#
For UDP connections (default on Windows): The value will be ignored. This
variable is just used to select that the control interface is to be created.
The value can be set to, e.g., udp (ctrl_interface=udp)
#
For Windows Named Pipe: This value can be used to set the security descriptor
for controlling access to the control interface. Security descriptor can be
set using Security Descriptor String Format (see http://msdn.microsoft.com/
library/default.asp?url=/library/en-us/secauthz/security/
security_descriptor_string_format.asp). The descriptor string needs to be
prefixed with SDDL=. For example, ctrl_interface=SDDL=D: would set an empty
DACL (which will reject all connections). See README-Windows.txt for more
information about SDDL string format.
#
ctrl_interface=DIR=/var/run/wpa_supplicant

IEEE 802.1X/EAPOL version
wpa_supplicant is implemented based on IEEE Std 802.1X-2004 which defines
EAPOL version 2. However, there are many APs that do not handle the new
version number correctly (they seem to drop the frames completely). In order
to make wpa_supplicant interoperate with these APs, the version number is set
to 1 by default. This configuration value can be used to set it to the new
version (2).
Note: When using MACsec, eapol_version shall be set to 3, which is
defined in IEEE Std 802.1X-2010.
eapol_version=1

```
# EAP fast re-authentication
# By default, fast re-authentication is enabled for all EAP methods that
# support it. This variable can be used to disable fast re-authentication.
# Normally, there is no need to disable this.
fast_reauth=1

# Interworking (IEEE 802.11u)

# Enable Interworking
interworking=1

# Homogenous ESS identifier
# If this is set, scans will be used to request response only from BSSes
# belonging to the specified Homogeneous ESS. This is used only if interworking
# is enabled.
hessid=00:20:a6:ca:a4:d6

# Automatic network selection behavior
# 0 = do not automatically go through Interworking network selection
#     (i.e., require explicit interworking_select command for this; default)
# 1 = perform Interworking network selection if one or more
#     credentials have been configured and scan did not find a
#     matching network block
auto_interworking=1

# Hotspot 2.0
hs20=1

network={
        ssid="test"
        priority=1
        identity="user"
        key_mgmt=WPA-EAP
        proto=RSN
        pairwise=CCMP
        group=CCMP
        eap=TLS
        ca_cert="/home/netcom/Desktop/ca2/ca.pem"
        private_key="/home/netcom/Desktop/ca2/client.p12"
        private_key_passwd="netcom;"

}

cred={

        realm="test.com"
        domain="test.com"
        roaming_consortium=2233445566

}
```

Sqlfunc.h

```
#include <stdio.h>
#include <mysql.h>
#include <string.h>
/* Get_From_Sql(const char *query,char *out)
 * \input query : gets the query for the SQL
 * \input out : pointer to  the variable of the output
 * the function gets the SQLquery and the out variable and returned the string of the input column.
 * the copy is done using memcpy(*dest,*src,size) function (copies size*x bytes from src to dest)
*/
void Get_From_Sql(const char *query,char *out)
{
 MYSQL *con;
 MYSQL_RES *res;//ResultSet
 MYSQL_ROW row;
 //future work get inits from conf file
 char *server = "localhost";
 char *user = "root";
 char *pass = "toor";
 char *db = "11u";
 //end of future work
 int size = 1;
 con = mysql_init(NULL);

 if (con == NULL)
 {
   out[0] =  '';
 }

 if (!mysql_real_connect(con, server, user, pass, db, 0, NULL, 0))
 {
    mysql_close(con);
    out[0] =  '';
 }

 if (mysql_query(con,query))
 {
    mysql_close(con);
    out[0] =  '';
 }

 res = mysql_use_result(con);
 while((row = mysql_fetch_row(res)) != NULL){
  memcpy(out,row[0],strlen(row[0]));
  size = strlen(row[0]);
 }
 out[size] = '\0';
 //memory optimization for the garbage collector
 mysql_free_result(res);
 mysql_close(con);
}
```

Gas_serv.c

```
static void anqp_add_venue_name(struct hostapd_data *hapd, struct wpabuf *buf)
{
	if (hapd->conf->venue_name) {
		u8 *len;
		unsigned int i;
		len = gas_anqp_add_element(buf, ANQP_VENUE_NAME);
		wpabuf_put_u8(buf, hapd->conf->venue_group);
		wpabuf_put_u8(buf, hapd->conf->venue_type);

		//start of changes
		//can we find the mac from the query ? if yes then
		//char query[64];
		//sprintf(query,"select v_name from Venue_Name where mac = '%s'\0",VARIABLE);
		char *query = "select v_name from Venue_Name where mac = '00:20:a6:ca:a4:d6'\0";
		char out[101];
		Get_From_Sql(query,out);
		if (out[0] != ' '){
			int l_name_len = 0;
			//rest no -3
			l_name_len = strlen(out) - 3;
			//rest no lang
			char l_lang[3];
			char l_name[l_name_len];
			//rest no memcpy(l_lang,&out,3);
			memcpy(l_lang,&out,3);
			//rest &out[0]
			memcpy(l_name,&out[3],l_name_len);
			wpabuf_put_u8(buf, 3 + l_name_len);
			wpabuf_put_data(buf,l_lang, 3);
			wpabuf_put_data(buf, l_name,l_name_len);
			gas_anqp_set_element_len(buf, len);
		}else{
			for (i = 0; i < hapd->conf->venue_name_count; i++) {
				struct hostapd_lang_string *vn;
				vn = &hapd->conf->venue_name[i];
				wpabuf_put_u8(buf, 3 + vn->name_len);
				wpabuf_put_data(buf, vn->lang, 3);
				wpabuf_put_data(buf, vn->name, vn->name_len);
			}
			gas_anqp_set_element_len(buf, len);
		}
	}
}


static void anqp_add_network_auth_type(struct hostapd_data *hapd,
				       struct wpabuf *buf)
{
```

```
	if (hapd->conf->network_auth_type) {

		//start of changes
		//can we find the mac from the query ? if yes then
		//char query[64];
		//sprintf(query,"select v_name from Venue_Name where mac = '%s'\0",VARIABLE);
		char *query = "select v_name from Authentication where mac = '00:20:a6:ca:a4:d6'\0";
		char out[101];
		Get_From_Sql(query,out);
		if (out[0] != ' '){
			int network_auth_type_len = 0;
			network_auth_type_len = strlen(out);
			char network_auth_type[network_auth_type_len];
			memcpy(network_auth_type,&out[0],network_auth_type_len);

			wpabuf_put_le16(buf, ANQP_NETWORK_AUTH_TYPE);
			wpabuf_put_le16(buf, network_auth_type_len);
			wpabuf_put_data(buf, network_auth_type, network_auth_type_len);
		}else{
			wpabuf_put_le16(buf, ANQP_NETWORK_AUTH_TYPE);
			wpabuf_put_le16(buf, hapd->conf->network_auth_type_len);
			wpabuf_put_data(buf, hapd->conf->network_auth_type,hapd->conf->network_auth_type_len);
		}

	}
}


static void anqp_add_roaming_consortium(struct hostapd_data *hapd,
                                        struct wpabuf *buf)
{
	unsigned int i;
	u8 *len;

	len = gas_anqp_add_element(buf, ANQP_ROAMING_CONSORTIUM);
	//start of changes
	//can we find the mac from the query ? if yes then
	//char query[64];
	//sprintf(query,"select v_name from Venue_Name where mac = '%s'\0",VARIABLE);
	char *query = "select v_name from Roaming_Consortium where mac = '00:20:a6:ca:a4:d6'\0";
	char out[101];
	Get_From_Sql(query,out);
	if (out[0] != ' '){
		int roaming_consortium_len = 0;
		roaming_consortium_len = strlen(out);
		char roaming_consortium[roaming_consortium_len];
		memcpy(roaming_consortium,&out[0],roaming_consortium_len);
		wpabuf_put_u8(buf, roaming_consortium_len);
		wpabuf_put_data(buf, roaming_consortium, roaming_consortium_len);
		gas_anqp_set_element_len(buf, len);

	}else{

		for (i = 0; i < hapd->conf->roaming_consortium_count; i++) {
```

```
                        struct hostapd_roaming_consortium *rc;
                        rc = &hapd->conf->roaming_consortium[i];
                        wpabuf_put_u8(buf, rc->len);
                        wpabuf_put_data(buf, rc->oi, rc->len);
                }
        gas_anqp_set_element_len(buf, len);
        }
}


static void anqp_add_ip_addr_type_availability(struct hostapd_data *hapd,
                                               struct wpabuf *buf)
{
        if (hapd->conf->ipaddr_type_configured) {
                //start of changes
                //can we find the mac from the query ? if yes then
                //char query[64];
                //sprintf(query,"select v_name from Venue_Name where mac = '%s'\0",VARIABLE);
                char *query = "select v_name from ip_address where mac = '00:20:a6:ca:a4:d6'\0";
                char out[101];
                Get_From_Sql(query,out);
                if (out[0] != ' '){
                        int ipaddr_type_availability_len = 0;
                        ipaddr_type_availability_len = strlen(out);
                        char ipaddr_type_availability[ipaddr_type_availability_len];
                        memcpy(ipaddr_type_availability,&out[0],ipaddr_type_availability_len);

                        wpabuf_put_le16(buf, ANQP_IP_ADDR_TYPE_AVAILABILITY);
                        wpabuf_put_le16(buf, ipaddr_type_availability_len);
                        wpabuf_put_data(buf, ipaddr_type_availability,ipaddr_type_availability_len);


                }else{

                        wpabuf_put_le16(buf, ANQP_IP_ADDR_TYPE_AVAILABILITY);
                        wpabuf_put_le16(buf, 1);
                        wpabuf_put_u8(buf, hapd->conf->ipaddr_type_availability);
                }
        }
}

static void anqp_add_3gpp_cellular_network(struct hostapd_data *hapd,
                                           struct wpabuf *buf)
{
        if (hapd->conf->anqp_3gpp_cell_net) {
                //start of changes
                //can we find the mac from the query ? if yes then
                //char query[64];
                //sprintf(query,"select v_name from Venue_Name where mac = '%s'\0",VARIABLE);
                char *query = "select v_name from 3gpp where mac = '00:20:a6:ca:a4:d6'\0";
                char out[101];
                Get_From_Sql(query,out);
                if (out[0] != ' '){
                        int anqp_3gpp_cell_net_len = 0;
```

```
                        anqp_3gpp_cell_net_len = strlen(out);
                        char anqp_3gpp_cell_net[anqp_3gpp_cell_net_len];
                        memcpy(anqp_3gpp_cell_net,&out[0],anqp_3gpp_cell_net_len);
                        wpabuf_put_le16(buf, ANQP_3GPP_CELLULAR_NETWORK);
                        wpabuf_put_le16(buf,anqp_3gpp_cell_net_len);
                        wpabuf_put_data(buf, anqp_3gpp_cell_net,anqp_3gpp_cell_net_len);

                }else{
                        wpabuf_put_le16(buf, ANQP_3GPP_CELLULAR_NETWORK);
                        wpabuf_put_le16(buf,hapd->conf->anqp_3gpp_cell_net_len);
                        wpabuf_put_data(buf, hapd->conf->anqp_3gpp_cell_net,hapd->conf->anqp_3gpp_cell_net_len);
                }
        }
}


static void anqp_add_domain_name(struct hostapd_data *hapd, struct wpabuf *buf)
{
        if (hapd->conf->domain_name) {
                //start of changes
                //can we find the mac from the query ? if yes then
                //char query[64];
                //sprintf(query,"select v_name from Venue_Name where mac = '%s'\0",VARIABLE);
                char *query = "select v_name from Domain where mac = '00:20:a6:ca:a4:d6'\0";
                char out[101];
                Get_From_Sql(query,out);
                if (out[0] != ' '){
                        int domain_name_len = 0;
                        domain_name_len = strlen(out);
                        char domain_name[domain_name_len];
                        memcpy(domain_name,&out[0],domain_name_len);
                        wpabuf_put_le16(buf, ANQP_DOMAIN_NAME);
                        wpabuf_put_le16(buf, domain_name_len);
                        wpabuf_put_data(buf, domain_name,domain_name_len);
                }else{
                        wpabuf_put_le16(buf, ANQP_DOMAIN_NAME);
                        wpabuf_put_le16(buf, hapd->conf->domain_name_len);
                        wpabuf_put_data(buf, hapd->conf->domain_name,hapd->conf->domain_name_len);
                }
        }
}

static void anqp_add_wan_metrics(struct hostapd_data *hapd,
                                 struct wpabuf *buf)
{
        if (hapd->conf->hs20_wan_metrics) {
                u8 *len = gas_anqp_add_element(buf, ANQP_VENDOR_SPECIFIC);
                //start of changes
                //can we find the mac from the query ? if yes then
                //char query[64];
                //sprintf(query,"select v_name from Venue_Name where mac = '%s'\0",VARIABLE);
                char *query = "select v_name from Wan_Metrics where mac = '00:20:a6:ca:a4:d6'\0";
                char out[101];
                Get_From_Sql(query,out);
```

```
                if (out[0] != ' '){
                        int hs20_wan_metrics_len = 0;
                        hs20_wan_metrics_len = strlen(out);
                        char hs20_wan_metrics[hs20_wan_metrics_len];
                        memcpy(hs20_wan_metrics,&out[0],hs20_wan_metrics_len);

                        wpabuf_put_be24(buf, OUI_WFA);
                        wpabuf_put_u8(buf, HS20_ANQP_OUI_TYPE);
                        wpabuf_put_u8(buf, HS20_STYPE_WAN_METRICS);
                        wpabuf_put_u8(buf, 0); /* Reserved */
                        wpabuf_put_data(buf, hs20_wan_metrics, hs20_wan_metrics_len);
                        gas_anqp_set_element_len(buf, len);

                }else{

                        wpabuf_put_be24(buf, OUI_WFA);
                        wpabuf_put_u8(buf, HS20_ANQP_OUI_TYPE);
                        wpabuf_put_u8(buf, HS20_STYPE_WAN_METRICS);
                        wpabuf_put_u8(buf, 0); /* Reserved */
                        wpabuf_put_data(buf, hapd->conf->hs20_wan_metrics, 13);
                        gas_anqp_set_element_len(buf, len);
                }
        }
}
```

Bibliography

1. **Foundation, Open Networking.** *SDN Architecture Overview.* s.l. : ONF TR-504, 2014.

2. **Group, Wi-Fi Alliance® Hotspot 2.0 Technical Task.** *Wi-Fi CERTIFIED Passpoint™ (Release 1) 6 Deployment Guidelines.* October 2012.

3. **IEEE.** *IEEE 802.11u Wireless LAN Medium Access Control and Physical Layer Specifications, Amendment 9: Interworking with External Networks.* 2011.

4. **Zhixian Xiang, John Kaippallimalil, Hinghung Anthony Chan, Khosrow Tony Saboorian.** *System and Method for ANDSF Enhancement with ANQP Server Capability . US20130272287 A1* October 17, 2013. Application.

5. **Lavrukhin, Vladimir.** *An Overhead Analysis of Access Network Query Protocol (ANQP) in Hotspot 2.0 Wi-Fi Networks.* Russia : IEEE , 2013.

6. **Leo Patanapongpibul, Assen GOLAUP.** *Operating band support for a Wireless Local Area Network .* 2014.

7. **George Calcev, Lin Cai, Xingxin ZHANG.** *System and Method for Efficient Access Network Query Protocol (ANQP) Discovery of Multiple Access Points (APs). US 20140112325 A1* Apr 14, 2014. Application.

8. **Ruckus.** *How Interworking Works: A Detailed Look at 802.11u and Hotspot 2.0 Mechanisms.* s.l. : Ruckus Wireless, Inc, 2013.

9. **Bin Chen, George Calcev, Hanan Ahmed, Kaidi Huang.** *System and Method for Common Attributes in HESSID and the Associated Queries . US 20140126563 A1* May 8, 2014.

10. **Angelo Centonza, Robert Hancock, Eleanor Hepworth, Stephen McCann.** *Method of controlling information requests . US 20100146272 A1* June 10, 2010.

11. **Rohil, Vishal Gupta and Mukesh Kumar.** *Enhancing Wi-Fi with IEEE 802.11u for mobile data offloading.* 1,2Department of Computer Science and Information Systems, BITS Pilani, Pilani, India : International Journal of Mobile Network Communications & Telematics (IJMNCT), 2012.

12. **J. Arkko, B. Aboba, J. Korhonen, Ed. TeliaSonera F.Barri.** *Network Discovery and Selection Problem.* 2008.