ASSOCIATION MECHANISMS IN WIRELESS SDN NETWORKS

by

Alexandros-Stylianos Valantasis

Department of Electrical and Computer Engineering

UNIVERSITY OF THESSALY

VOLOS 2017,GREECE

Title Thesis:

“ASSOCIATION MECHANISMS IN WIRELESS SDN NETWORKS”

"Μηχανισμοί Σύνδεσης σε Ασύρματα Δίκτυα καθορισμένα από Λογισμικό"

Author:

Alexandros-Stylianos Valantasis

Supervisors:

Athanasios Korakis, Assistant Professor

Antonios Argyriou, Assistant Professor

Presented to the Department of Electrical and Computer Engineering of
The University of Thessaly at Volos in Partial Fulfillment
of the Requirements for the Degree of

DIPLOMA OF SCIENCE IN COMPUTER AND COMMUNICATION
ENGINEERING

THE UNIVERSITY OF THESSALY

SEPTEMBER 2017

Declaration of Authorship

I, Alexandros-Stylianos Valantasis, hereby certify that this thesis titled, ”*Association Mechanisms in Wireless SDN Networks”* has been composed by me and is based on my work, unless stated otherwise.

The research was carried out wholly during the candidacy for the graduate degree of Diploma of Science in Computer and Communication Engineering at the University of Thessaly, Department of Electrical and Computer Engineering,Volos,Greece.

I have documented every source and material I have quoted or consulted at the References section.

Dedicated to my Family & Friends

Ευχαριστίες

Με την περάτωση αυτής της εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα αναπληρωτή καθηγητή του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, κ. Αθανάσιο Κοράκη για την εμπιστοσύνη που έδειξε στο πρόσωπό μου αλλά και την πίστη του στις δυνατότητές μου, προσφέροντας μου την ευκαιρία να ασχοληθώ με το συγκεκριμένο θέμα και να αποκτήσω ουσιαστικά εφόδια μέσα από αυτήν τη διαδικασία.
Επίσης, θα ήθελα να ευχαριστήσω τους εργαζόμενους στο Εθνικό Κέντρο Έρευνας και Τεχνολογικής Ανάπτυξης (ΕΚΕΤΑ) Βόλου, οι οποίοι με βοήθησαν να ξεπεράσω τυχόν δυσκολίες που προέκυψαν.
Ιδιαίτερη ευχαριστία θα ήθελα να απευθύνω στον κ. Κωνσταντίνο Χούμα για τις ουσιώδεις υποδείξεις και παρεμβάσεις, καθώς και για την καθημερινή του ενασχόληση και καθοδήγηση που βοήθησαν σε μεγάλο βαθμό στην εκπόνηση της παρούσας εργασίας.
Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου και τους φίλους μου, στους ανθρώπους που με τους όποιους πέρασα χαρούμενες και λυπημένες στιγμές και μου παρείχαν ουσιαστική υποστήριξη σε αυτό το ταξίδι της γνώσης μέχρι και το τέλος.

Περίληψη

Στη σημερινή εποχή, ο αριθμός των χρηστών με πρόσβαση στο διαδίκτυο (Internet) μέσω ασύρματων τοπικών δικτύων (WLANs) IEEE 802.11 αυξάνεται καθημερινά. Ο μεγάλος αριθμός εισερχομένων χρηστών, που καθορίζουν τους πελάτες, θέλει να μπορεί να διασυνδέεται ανά πάσα στιγμή με το κατάλληλο Access Point, με συνέπεια η πολυπλοκότητα του δικτύου να αυξάνεται και οι διαχειριστές του δικτύου να μην μπορούν να ικανοποιήσουν αποτελεσματικά τις ανάγκες του κάθε χρήστη. Για να αποφευχθεί η παραπάνω πολυπλοκότητα, είναι ουσιαστικό να υπάρχει ένας μηχανισμός εξισορρόπησης του φορτίου (load balance) που υπάρχει στο δίκτυο.
Πολλές επιστημονικές προσεγγίσεις έχουν γίνει για το προαναφερθέν θέμα, εξετάζοντας κάθε φόρα διαφορετικούς μηχανισμούς σύνδεσης (association mechanisms) σε WiFi δίκτυα. Ωστόσο, οι περισσότεροι από αυτούς είτε είναι ξεπερασμένοι είτε δεν εισάγουν ένα αποτελεσματικό μηχανισμό που να περιλαμβάνει την έννοια του Software Defined Networking(SDN).
Για το σκοπό αυτό, η συγκεκριμένη διπλωματική εργασία, αξιοποιώντας την πρωτοποριακή αρχιτεκτονική του SDN, η οποία διαχωρίζει το επίπεδο ελέγχου (control plane) από το επίπεδο δεδομένων (data plane) αλλά και τις δυνατότητες του OpenFlow πρωτοκόλλου, εισάγει έναν αξιόπιστο αλγόριθμο για τη σύνδεση των εισερχόμενων χρηστών με το καλύτερο διαθέσιμο Access Point του δικτύου.
Ο αλγόριθμος που παρουσιάζεται εξετάζει την κίνηση του φόρτου μέσα στο δίκτυο, μέσω διαφόρων στατιστικών στοιχείων που αποστέλλονται από τα Access Points στον ελεγκτή του SDN. Τέλος, ο αλγόριθμος υπολογίζει μια τιμή, το διαθέσιμο εύρος ζώνης δια-μεταφοράς (available throughput bandwidth) του κάθε Access Point που μπορεί να παρέχει στον εισερχόμενο χρήστη.
Ο παραπάνω μηχανισμός παρουσιάζεται (demonstrate) μέσω μιας διαδικτυακού τύπου διεπαφής (web-based network interface) όπου ο χρήστης μπορεί εύκολα να παρατηρήσει τα χαρακτηριστικά του δικτύου και να αξιολογήσει την αποδοτικότητα του αλγορίθμου.

Abstract

Nowadays, the number of users accessing the Internet via IEEE 802.11 WLANs (Wireless Local Area Networks) and associate with Access Points is increasing day by day. These big amounts of incoming users, who define the client devices, amplify the network complexity; as a result the network administrators can not satisfy efficiently each user’s needs.

To avoid the above complexity, it is vital to balance the load of the entire network. Many scientific approaches have been introduced regarding the aforementioned topic, considering different mechanisms of WiFi association, but, most of them are out of date or do not introduce an efficient mechanism which includes the SDN concept. For that purpose, this thesis, taking advantage of the SDN innovative architecture, to separate control plane from data plane, and OpenFlow abilities, introduces a reliable algorithm for the association of the incoming user to the best available Access Point.The algorithm presented makes a load traffic examination via various statistics Access Points sends to the SDN controller and, finally, calculates the available throughput bandwidth of each Access Point can provide to the incoming user. All the aforementioned mechanism is demonstrated via a network web based interface where the network administrator can easily observe the network characteristics and evaluate the efficiency of the implemented algorithm.

Table of Contents

List of Figures & Tables

1 Introduction

Today, the number of users accessing the Internet via IEEE 802.11 WLANs (Wireless Local Area Networks) is increasing day by day. These big amounts of incoming users, who define the client devices, amplify the network complexity; as a result the network administrators can not satisfy efficiently each user's needs.

For that purpose, many scientific approaches have been made in the topic of load balancing and how the clients would efficient associate to Access Points in a WiFi network, with only a few having addressed a reliable mechanism including the SDN concept.

With SDN (Software Defined Network) being an innovative networking paradigm with great potentials and high flexibility in managing and deploying network services, because of its innovative architecture to separate the control plane from data plane, this thesis will take advantage of these skills to introduce a reliable association algorithm in a WiFi network.

1.1 Organization

The rest of this dissertation is organized as follows:

In *chapter 2*, we present basic background information about the wireless networks and the association process, such as a comprehensive overview of Software Defined Networks and OpenFlow architecture. Also, *chapter 2* refers to a variety of scientific researches related with the area of association in traditional WiFi networks and examines the advantages and disadvantages of each one. Finally, *chapter 2* introduces some pioneer researches in the aforementioned topic including the SDN concept.

In *chapter 3*, details about the algorithm implementation and design are provided. Specifically, *chapter 3* elaborates the scenario for the evaluation of the algorithm, the whole algorithm logic, the basic metrics included in the mathematical formula and also the programming operation of the basic components of the algorithm.Moreover, *chapter 3* presents the tools, which participated in the construction of the algorithm and some general information about each of them.

Finally, in *chapter 4* we present the demonstrated algorithm process via a network web interface and how this process is done; in the last *chapter 5*, we summarize and conclude this thesis and propose directions for future research.

2 Background

This thesis introduces an algorithm associating users to best available APs (Access Points) in a WiFi network taking advantage of SDN and OpenFlow opportunities. For that reason, in this chapter an inclusive overview of SDN, OpenFlow, IEEE 802.11(WiFi) architecture, such as association in WiFi are presented.

Especially, this chapter is divided in 2.A and 2.B. In 2.A, the wireless networks and association process are introduced, while related works which had been made in the research topic of association in a WiFi, with and without the SDN, are briefly presented.

In 2.B, we analyze the basic architecture of SDN and OpenFlow, basic components of them such as OpenFlow Switch and OpenFlow Controllers, and finally we describe the entire relationship between these innovative concepts.

2.A Wireless Networks & Association Techniques

2.A.1 WLAN and association process

Today, the number of users accessing the Internet via IEEE 802.11 WLANs (Wireless Local Area Networks) is increasing day by day. The users, who define the client devices, can vary from tablets and laptops to smartphones. According to the *Juniper Research report* [1], the volume of data traffic generated by mobile devices will hit nearly 197,000 petabytes (PB) by 2019 that mainly offloaded to WiFi. In order to serve this growing number of users with different characteristics, WLANs must have many Access Points, which provide the functionality needed to satisfy these heterogeneous clients.

A typical wireless local network (WLAN) has three basic components: Access Point (AP), Wireless Stations(STA) and Basic service sets (BSS).The wireless station can be any device being able to communicate using the 802.11 standard (WiFi) and can vary from laptop, smart phone, workstation to printers and scanners.The Access Point is a wireless device with two functionalities. The first one is that it acts as a network platform for connections between WLANs or to a wired LAN while the second one is that it acts as a relay between stations attached to the same AP.

Finally, the basic service set (BSS) is the logical component of wireless architecture, in contrast with the other two (wireless station and the access point) which are both physical components. In general, the BSS is a set of wireless stations controlled by a single management function and has two configuration options. In an IBSS, the stations communicate directly to one another without the need for an access point. In a BSS infrastructure, there is a connection to the wired network. An extended service set (ESS) is a set of infrastructure BSSs that appears as a single BSS. This is important for connection redundancy but some security issues arise that need to be addressed.

The most important procedure in a wireless local network is the connection of stations with the APs. The selection of the AP that a WLAN station connects with must be done prudently as it determines the performance of the station. In the nomenclature of IEEE 802.11, such AP selection procedure is referred to as association. In WiFi, the association process is divided into three steps: scanning, authentication and association.

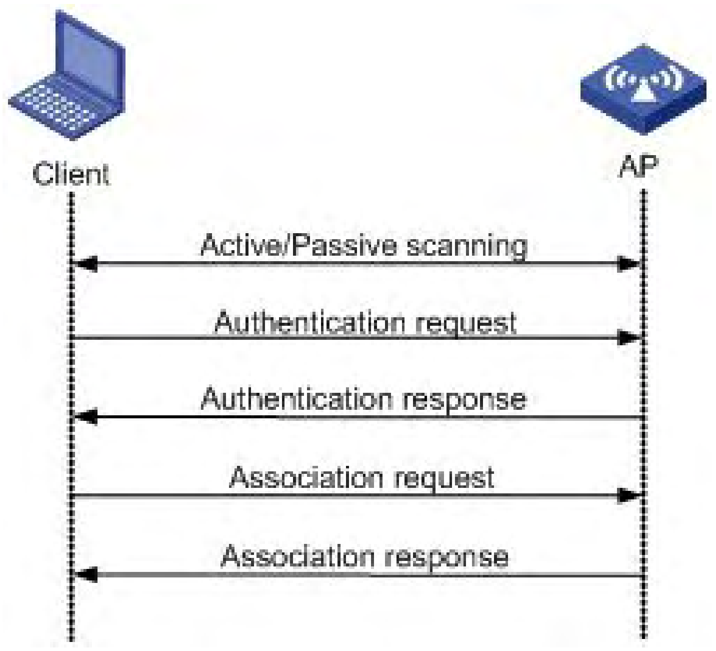


Figure 2.1 WiFi Association Process

Scanning

The main purpose of scanning is the selection of the appropriate AP to be associated with, in all available channels. Scanning can be active or passive.In passive mode, the station finds APs by listening to periodic beacon frames. The station using active scanning broadcasts a probe request frame on each channel and may receive multiple probe response frames from different APs working on the same channel.Typically, the majority of WLANs operate with active scanning for association.

Authentication

After scanning process, the station sends an authentication request to the selected AP for connection. In case of no password set on the AP (just like the open Wi-Fi found in public places), the AP replies with an authentication request at once and association begins. When the WiFi is password-protected, the AP sends an authentication reply message to the host. Then, the host must enter the right password and send this encrypted information back to the AP. Finally, the AP decrypts this message and if the correct password is inserted, the AP sends a successful authentication message to the host and the authentication procedure is finished.

Association

Following the authentication, the host sends an association request message to the AP.Then, the AP responds with an association reply message including its MAC address.At this point the connection is established and the host can exchange data with the AP.

2.A.2 Related works in WiFi association mechanisms

As we have already mentioned, the most crucial procedure in a wireless local network is association between Stations and Access Points.The challenging problem in this procedure is how to find the appropriate Access Point for efficient association with the station.

RSSI Approach

The most commonly used metric for the association between access points and stations is the received signal power from an AP, known as RSSI (Received Signal Strength Indication). After scanning, a station chooses the AP from which it receives frames with the highest RSSI. However – as the field literature stresses – such an RSSI based association is not necessarily the most appropriate association mechanism. In addition, the RSSI based association might result in unbalanced and not the best available throughput among BSSs (Basic Service Sets)[2-3]. Therefore, a station associated with the highest RSSI AP might suffer from low throughput that is caused by the overloaded bandwidth utilization in that BSS. For these reasons, an alternative association mechanism is considering signal to interference and noise (SINR) per connection, as well as asymmetric traffic, which proposed in [4]. Although this approach considered uplink channel conditions as well, thus offering a significant improvement, it was not able to lead to the best available throughput performance[15].

AP Load Approach

Another association strategy is to examine the load of all available Access Points and choose the appropriate one. The authors in [5], explained a selection mechanism that estimates the load of AP based on momentary measurements of the transmission rate and the fraction of time an AP obtains the channel for its transmissions. However, this technique has the disadvantage of considering only downlink traffic and therefore supposes that channel conflict is only among APs.
An alternative projects belonging in the *AP Load Approach* field are [2],[6]. In that case the AP load is treated a metric reflecting the AP's weakness to satisfy the requirements of its associated stations.Another approach followed in [7], consider association decisions on a stat called airtime cost, which regards both uplink and downlink traffic as well as AP load.The aforementioned share the common characteristic of considering the AP load estimation influenced by transmissions only of associated users.[15]

AP Load Approach including Neighbor nodes

Similar to the above the author in [8] introduces an AP load assumption with a differentiation that influences an AP load metric not only by the associated stations but also from other neighboring stations and their transmissions. Thus, the authors consider AP load over all neighboring nodes. The main disadvantage of this assumption is that it is not taking into account the importance of rate adaptation mechanisms because transmissions rates are fixed.[15]

Traffic Intensity Approach

The main difference from the above approaches is that it takes into consideration non-saturated traffic, which means that users transmit and receive not all the times. The author [9] introduces a new approach, a *Traffic Intensity,* in which APs should assign an activity level estimator to their associated STAs based on observations of their traffic intensity. Nevertheless, this approach fails to characterize the traffic intensity of neighboring nodes that belong to adjacent cells, although these contend for channel usage or even interfere with transmissions in the cell under consideration.[15]

Available Bandwidth Approach

In this approach, the author [10] introduced a new association metric, mentioned as EVA (Estimated aVailable bAndwidth). With EVA, stations can find the appropriate AP that provides the maximum achievable throughput among scanned APs. EVA is designed to calculate the available bandwidth on a channel with respect to a station that is going to join a WLAN (Wireless Local Area Network). For the estimation of the available bandwidth, EVA estimator considers the level of conflict on a BSS by calculating collision probability and channel idle ratio based on channel state. After searching all accessible channels for available APs, a station with the EVA estimator chooses the AP that provides the largest EVA. The main disadvantage of EVA association approach is that it has not the functionality to manage hidden node terminals and their effect, which appear very often in dense WLANs.

Hidden Node Terminals Approach

As we have already mentioned, the variety of association approaches does not give much attention to the effect of hidden node terminals. For that purpose, the author in [11] proposed a metric that includes contention and interference as well. This approach focuses on the effect of interfering nodes and uses a factor that captures the error probability due to collisions, considering it as a value proportional only to the number of STAs associated to each AP and STAs that belong to neighboring cells and operate on the same channel. The disadvantage of this assumption is that it does not consider APs transmitting on downlink as potential interference. In addition, this approach is not able to distinguish between nodes that just contend for channel usage and nodes that appear hidden.[15]

2.A.3 Related works in WiFi association with SDN

All the approaches mentioned above have advantages and disadvantages. However, their main disadvantage is the scope of approaching the problem of WiFi association. All these techniques revolve around finding a solution and making changes in the driver. These changes are carried out in the physical and data link layer of OSI model and not in the above layers as desirable.

Because of this, a small change in the network policy may need all the devices to be configured, something increasing the complexibility of the entire network and making it very difficult to researchers to introduce new network protocols. Additionally, most of these solutions are generally proprietary,which makes it difficult to extend their functionality and improve their flexibility such spectrum efficiency and QoS requirements can be considered.

As we have already mentioned, the majority of WiFi association approaches allow clients and not the infrastructure to make AP association decisions, not always evaluated as the optimal ones. For all these reasons, it is imperative

to bring up a new approach to the problem of WiFi assignation.This can “satisfied” via SDN.
Software-Defined Networking (SDN) has emerged as an open, efficient and flexible network management concept for large networks. By decoupling the control plane from the data plane, SDN can centralize network management operations in a single entity, often referred to as a controller.Due to its flexibility, the SDN concept is also currently being adopted for wireless network management, including WiFi networks.

There has been a number of projects that tried to extend SDN to wireless networks,especially WiFi, and implemented QoS management and efficient resource allocation in them.Contributions such as OpenRoads[12], OpenSDWN[13], EmPOWER[14] and Odin[16] build new mechanisms on top of OpenFlow in order to support mobility, virtualization, and Service Set IDentifier (SSID) management. However, we are interested in projects using SDN to address the problem of AP selection.[48]

In [47], the author presents a version of AP selection approach based on SDN. Specifically, he have introduced the FF concept, which is an algorithm that calculates and assigns a performance metric *Fittingness Factor* to each AP, to allow the controller to associate the most suitable AP to a device.[48]

Building on the previous project, the author in [48], proposes a more dynamic AP selection approach, in which the controller managing the WiFi network selects the most suitable AP for a specific application.This approach is based on an algorithm that calculates and assigns a performance metric to each AP, *called Fittingness Factor (FF)*, which is a function addressing the suitability of the available spectrum resources to the application requirements.Before assigning an AP, the algorithm calculates another parameter, called *Network Fittingness Factor* that takes into account the QoS requirements of a wireless user joining the network, the current network capacity, and the quality of the connectivity provided to the remaining wireless users.[48]

In [49], the authors propose the use of a dynamic AP selection algorithm implemented in a SDN-based framework. In this work, the devices receive network resource-related statistics from the SDN controller, which guide the client device to associate itself with the best available AP.This association is based on the received statistics that jointly consider the network load in terms of the AP bandwidth and RSSI value.[48]

Finally in [50], a new approach called meSDN is introduced. The meSDN project presents an innovative idea and considers not the down-link transmission (from AP to clients), which can be controlled using OpenFlow rules, but the uplink transmission (traffic originating from client to AP).The main idea of meSDN is that requires participation of client devices. Now, the OpenFlow rules in an AP can be controlled by the client before traffic arrives in the AP.

As mentioned above, a very popular AP association approach in WiFi is to examine the load of APs. Even though many scientific approaches have been made in the aforementioned topic, only a few have addressed the issue of network load balance including the SDN concept. For that reason, this thesis proposes an algorithm which makes load balance of APs in software defined wireless network. In particular, the thesis examines the load of OpenFlow APs with a metric of throughput and associates the incoming stations to the appropriate APs, according to the available bandwidth each AP can provide.

2.B Software Defined Networking

2.B.1 Traditional Architectures and Limitations

Network devices as we know it today show limitations due to their conventional architecture. These devices have software which controls the network. Additionally, they deal not only with data plane, but also with control plane using networking information in order to generate the forwarding table used by data plane to route packets accordingly. In the traditional way, the two planes are combined in one device itself using standard protocol. Thus, traditional network devices operate autonomously with different characteristics, capabilities, management interfaces and policies definition. Therefore, network configuration is manual and each device has to be configured separately [17]. Even though proprietary solutions exist to facilitate complex network management, these only work in homogeneous networks and according to traditional networking paradigms.[51]

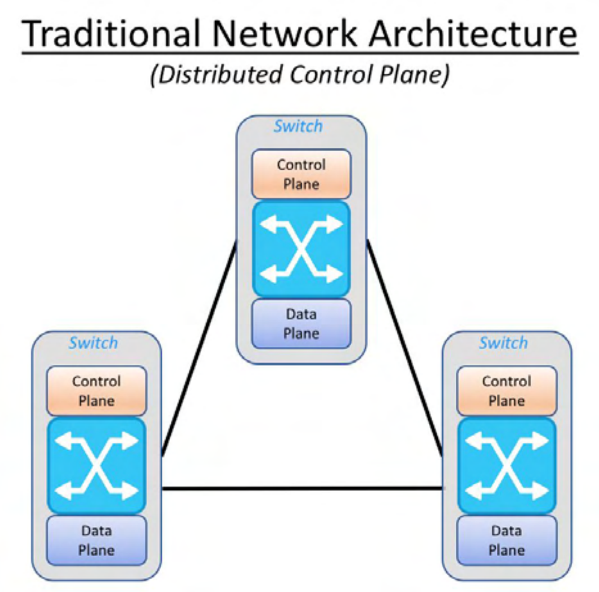


Figure 2.2: Architecture of traditional network

The evolution of telecommunications, software and hardware technology is rapid.Although evolution is visible, the only technology area that is visibly stagnated is networking.The reason for such stagnation is the fact that it is a closed system, meaning only vendors of the network devices have access to device configuration,preventing the change of device characteristics.This does not allow the implementation of new ideas that may arise by the network research community or by new requirements of network operators and limiting the scope for developing and implementing innovations.[51]

2.B.2 Software Defined Networking

Software-Defined Networking (SDN)[18] was introduced in 2010. This innovative networking architecture overlays the limitations of management and controls the network, while, also serves as the networking paradigm which aims to provide the entire research network community a fast and simple method to test new technologies and ideas.

SDN proposes a new architecture, where decouples the network control and management from the data plane. This separation is a departure from the traditional way and is the state-of-art concept who makes the network easily programmable.

Due to this separation,we can abstract the complex process from the repetitive forwarding process.The complex process can be automated and managed separately in a centralized SDN controller.

The basic idea of SDN architecture is that it decouples the control plane from data plane. The above process has as result the creation of an open interface between the two planes.The control plane,now, located outside of the network devices (router, switch) in a controller which manages the forwarding tables of network devices. The data plane forwards the data according to its flow (forwarding) tables, which previously control plane (controller) had inserted.This approach makes the implementation of new networking management techniques much easier, since a new routing protocol can be tested without requiring conversions of the network device configuration.[19]

There are three different ways of control and data plane separation.

The first approach is the *strictly centralized control.* Here, a controller manages all the forwarding elements in the system, retains a global view of the entire network and can easily ensure that the network is in a consistent, optimal configuration. It is the simpler, agent-less solution, since no extra functionality needs to be installed on network nodes. Nevertheless, as usual centralized logic, it can lead to single point of failure.

The second approach is the *logically-centralized control*, where network devices have partial functionality embedded in them and act as intermediary across the first and third approach.

Lastly, the third approach is the *fully distributed control*. In this approach, a local controller runs on each compute node and manages the forwarding element directly (and locally). Thus, the control plane becomes distributed across the network. However, the virtual network topology needs to be synchronized across all the local controllers. This is accomplished by using a distributed database. The main advantage is that fully distributed control achieves significantly better scalability, highly-available by design and no single-point-of-failure against centralized control.Nonetheless, no global view of the network is available, while extra computing must be done on the local host.

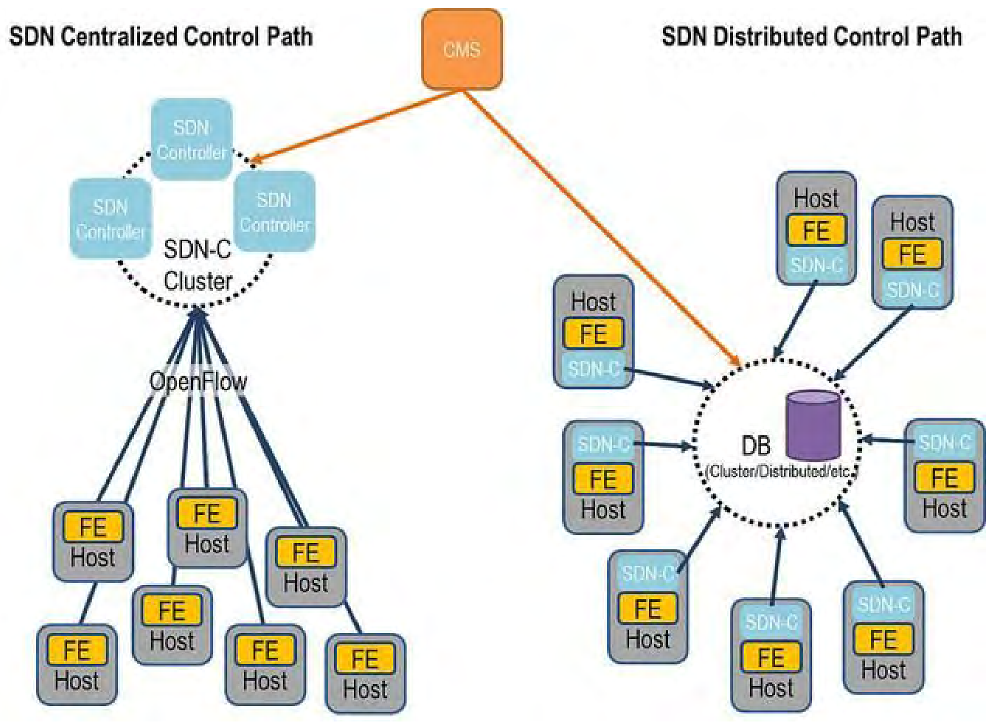

SDN Centralized Control Path
CMS
SDN Distributed Control Path
SDN Controller
SDN Controller
SDN Controller
SDN-C Cluster
OpenFlow
FE
Host
SDN-C
DB
(Cluster/Distributed/etc.)

Figure 2.3: SDN Centralized vs Distributed control path

SDN architecture consists of three basic layers: the application layer, the control layer and the infrastructure layer. In addition, there are Application Programming Interfaces (APIs), called Northbound and Southbound APIs, which provide the essential communication tools between these layers.

Infrastructure Layer: It is at the lowest level in the SDN architecture as it's the physical layer that contains all the hardware for forwarding packets at line rate. Software runs on these hardware devices providing a control data plane interface (Southbound API).

Southbound API: The Southbound API handles the communication between the control layer and the hardware devices inside the infrastructure layer with a standard protocol.

Control Layer: The most important entity of the SDN architecture. Control layer contains the controller who has the basic logic and strategy for the network. It has the functionality to communicate with the network devices in the lower layer (Infrastructure) via Southbound API and inform them of how to forward their packets. Also control layer can communicate with the upper layer (Application) through Northbound API via Rest calls.

Application layer: This layer contains all the actual software applications, where all features and services are defined. Applications want to know all the information about the network devices and topology in order to make decisions according to the changes in the network. For that reason, it uses functions of the Northbound API and provides variety of network functionalities e.g. loadbalancers, monitoring, routing.

Northbound API: The controller provides an API to programmers to communicate with the lower layer. This bidirectional communication between applications and control planes enables support for switching, routing, firewall,etc. as it abstracts the forwarding device. The majority of the SDN controllers interface via Rest APIs or APIs in JAVA and PYTHON.

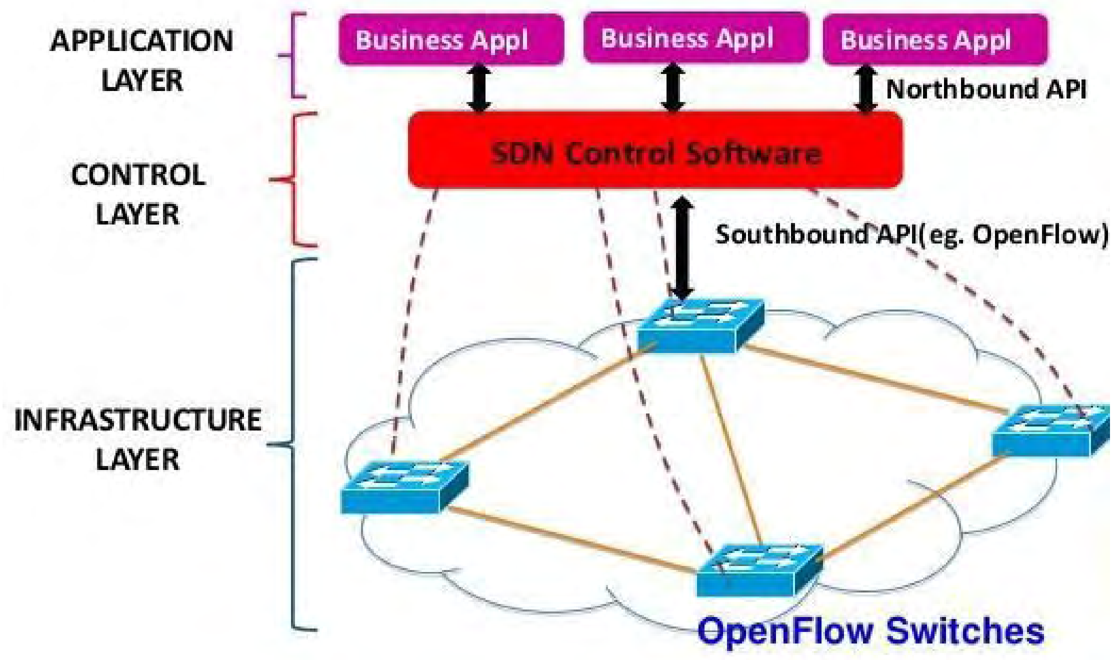


Figure 2.4: SDN Architecture

Taking the aforementioned into account, SDN brings a boost in networking technology due to the innovative notion of controlling and management of the network via programming.

2.B.3 SDN Protocol-OpenFlow

OpenFlow[20] is the most popular standard for the connection of the control plane and the data plane. It is defined as a communication protocol enabling access to the forwarding plane of a network device through the network.This protocol permits the process of the packet forwarding to be specified not by the network devices, but, by software. Also, it was developed by engineers from Stanford and California University in 2007 and the calibration process is being managed by the Open Network Foundation (ONF).

OpenFlow Architecture

The architecture OpenFlow is designed in such a way as to separate the data plane and the control plane, while it executes control, based on packet's information.

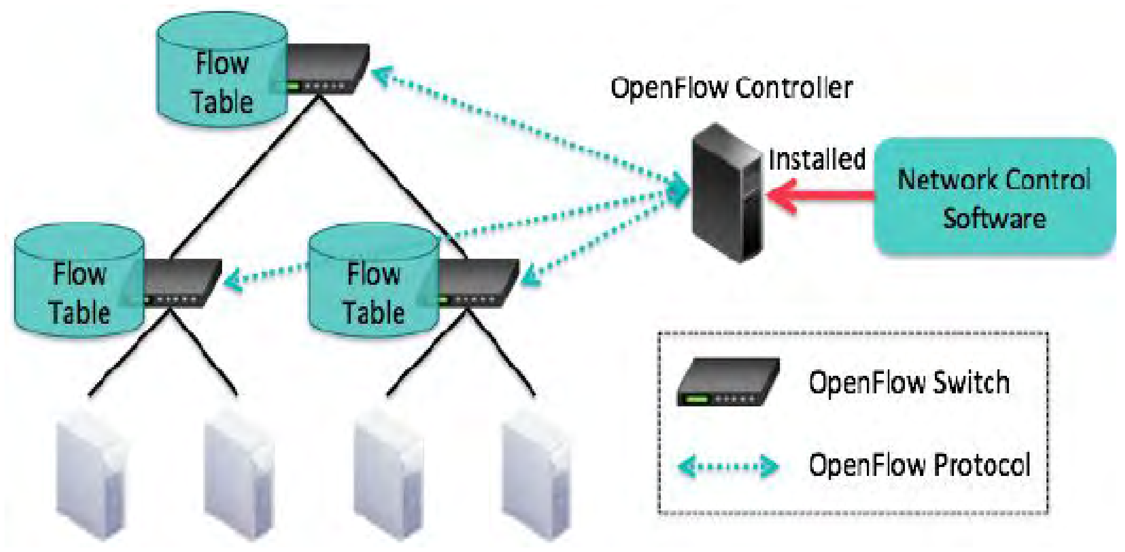


Figure 2.5: OpenFlow Architecture

From figure 2.4, OpenFlow network includes an OpenFlow Controller, OpenFlow Switches and the OpenFlow protocol for the communication of the above.

OpenFlow Connections

Communication between the switch and the controller is provided through a secure and standard Transport Layer Security (TLS) or TCP connection with default TCP port number 6633.

In the beginning, there is an exchange of credentials between the switch and the controller. When a connection is established, a symmetric OFPT_HELLO message is sent, either from controller or the switch, containing the OpenFlow version.In case of connection interruption between the switch and controller, the switch must immediately enter one of the following modes; it can drop packets meant for the controller or act as a legacy switch.[38]

Flow Table

A Flow table[22] of a switch has all the information need the switch know for the process of forwarding the packets. It contains prioritized rules of match and action condition.The match condition judges which rule of the flow table of this switch applies to the packet and then the action condition determines how the packet should be managed. As packets arrive, the switch matches the rules of its flow table to the headers of the packets and if there is a match, then a specific action is chosen. In case of no rules-matching, the packet is sent immediately to the controller via an OpenFlow message. This notification is called a packet-in event. When the controller receives this packet-in event, it can add a new rule to the flow table to handle this type of packet.[38]

OpenFlow Flow Entry

As seen in figure 2.5 each entry in the flow table of an OpenFlow switch is divided in three parts: Rule, Action and Statistics.

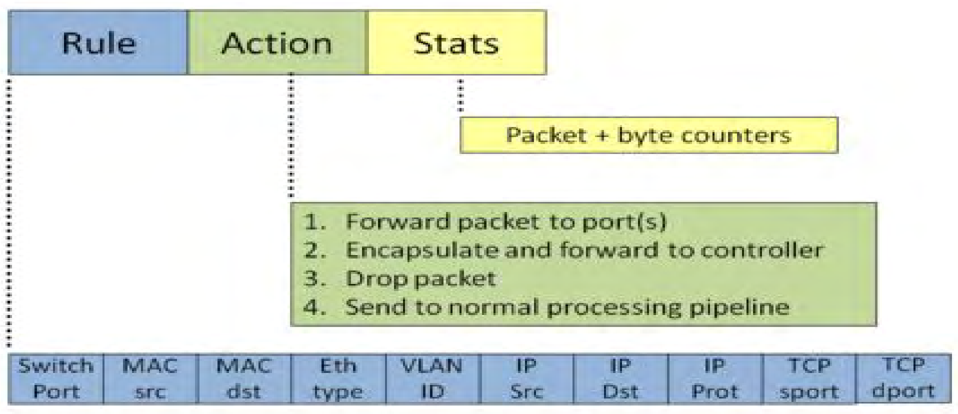


Figure 2.6 OpenFlow Flow entry

- Rule: It is the part matching the frames of the flows and specifying which packets have to be handled by this entry.

- Action: Every time a packet matches a rule, a specific action must be performed. For that reason,first, the action has to be defined.The basic actions can be defined such forwarding to a number of ports, forwarding to the controller,dropping the frame, and modifying frame fields.[19]

- Statistics: Whenever a flow rule is matched, the switch informs some basic counters. These can be frame counters, which indicate how many times a specific flow is matched. Also,there are flow, port and queue counters, which keep statistics of the packets handled.

OpenFlow Message Types

The OpenFlow protocol defines three basic type of messages.The controller-to-switch, the symmetric and the asynchronous.

Controller-to-switch: This kind of messages is initiated by the controller to manage or inspect switch’s state. These appear with the following type:

- Features: Controller requests information about the switch’s features.
- Configuration: Controller sets and queries configuration parameters of the switch.
- Modify-State: Controller manages the state of the switches, by adding / deleting and modifying flow entries.
- Read-State: Controller collects statistics from the flow table of the switch.
- Send-Packet: Controller sends a packet out of a specified port on the switch.
- Barrier: Used by the controller to ensure message dependencies have been met.

Symmetric: This type of messages is initiated either by the switch or the controller and sent without solicitation.These appear with the following type:

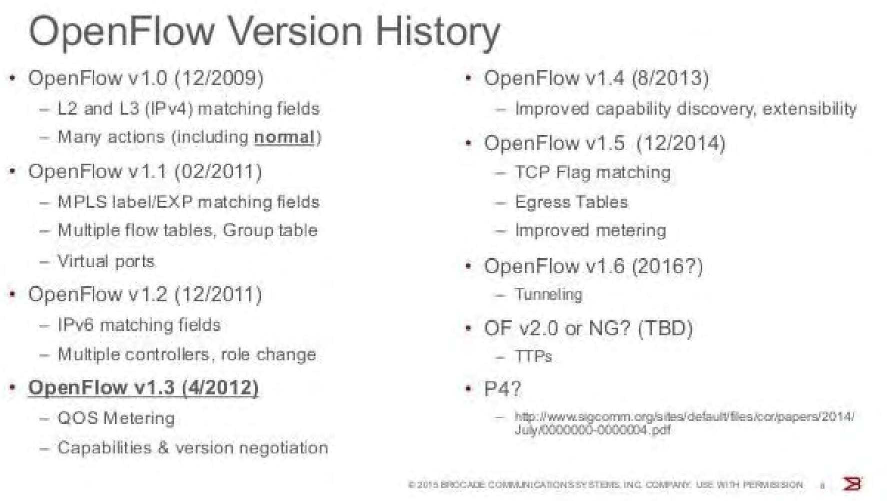
- Hello: Hello messages are exchanged between the switch and controller upon connection startup.
- Echo: Echo request/reply messages can be sent from either the switch or the controller, and must return an echo reply.
- Vendor: Vendor messages provide a standard way for OF switches to offer additional functionality.

Asynchronous: This type is initiated by the switch and used to inform the controller about the update of network events. These appear with the following type:

- Packet-in: For all packets that do not have a matching flow entry, a packet-in event is sent to the controller.
- Flow-Removed: When a flow entry expires (its duration exceeds the given timeout), this event is sent to the controller.
- Port-status: Switch sends port-status messages to the controller when port configuration state changes.

OpenFlow Versions

OpenFlow has an active and continuously development with vital upgrades. As indicated in the figure 2.6, the life of OpenFLow versions began in 2009 with OpenFlow version 1.0 and continues until today with the upcoming OpenFlow version 2.0.



OpenFlow Version History

- OpenFlow v1.0 (12/2009)
  - L2 and L3 (IPv4) matching fields
  - Many actions (including **normal**)
- OpenFlow v1.1 (02/2011)
  - MPLS label/EXP matching fields
  - Multiple flow tables, Group table
  - Virtual ports
- OpenFlow v1.2 (12/2011)
  - IPv6 matching fields
  - Multiple controllers, role change
- **OpenFlow v1.3 (4/2012)**
  - QOS Metering
  - Capabilities & version negotiation
- OpenFlow v1.4 (8/2013)
  - Improved capability discovery, extensibility
- OpenFlow v1.5 (12/2014)
  - TCP Flag matching
  - Egress Tables
  - Improved metering
- OpenFlow v1.6 (2016?)
  - Tunneling
- OF v2.0 or NG? (TBD)
  - TTPs
- P4?
  - http://www.sigcomm.org/sites/default/files/ccr/papers/2014/July/0000000-0000004.pdf

© 2015 BROCADE COMMUNICATIONS SYSTEMS, INC. COMPANY. USE WITH PERMISSION. 8

Figure 2.7 OpenFlow Version History

This thesis has been developed with OpenFlow version 1.3

OpenFlow Switch

OpenFlow switch is a basic component in the OpenFlow network. It is the network device that supports the OpenFlow protocol and forwards the packets across the SDN. An OpenFlow switch usually contains one or two flow tables, a group table, and a channel, where it can communicate with the SDN controller for inserting or deleting flow entries in the flow table.[19]

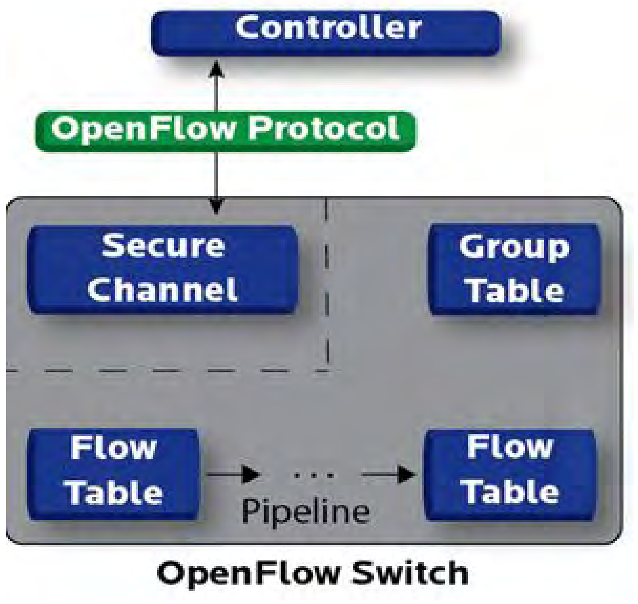


Figure 2.8: OpenFlow Switch

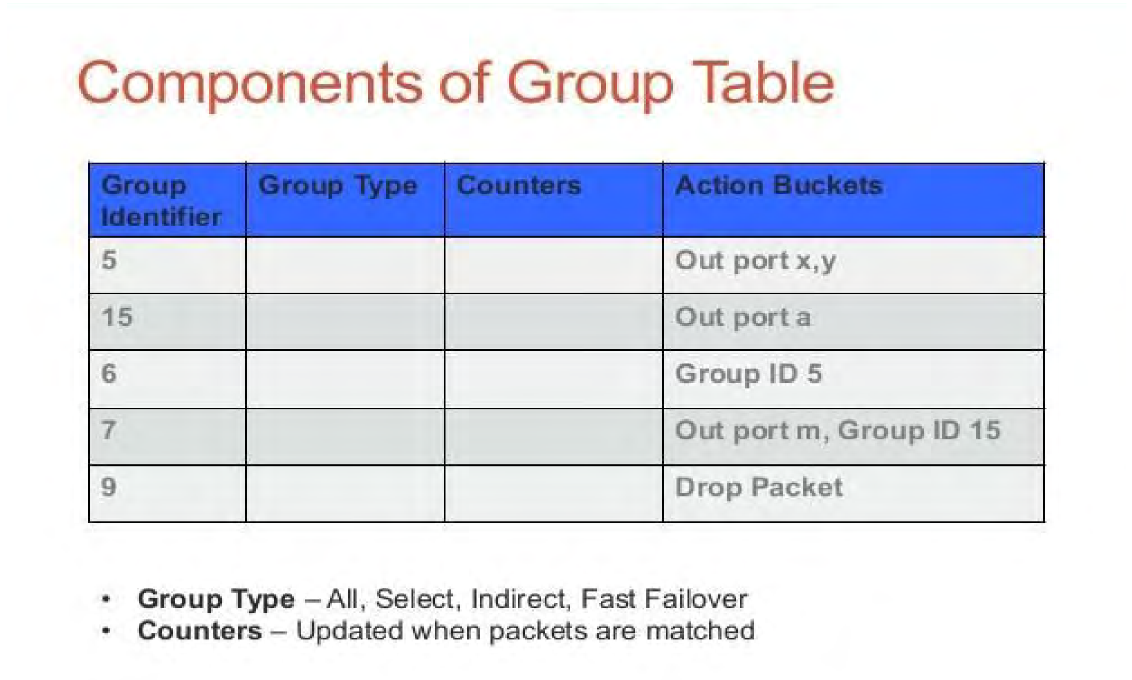
The basic component of OpenFlow switch is the flow table. As we have already mentioned, it contains a set of flow entries. Each one consists of match fields, counters and a set of instructions to apply on the matching packets.

An OpenFlow switch also maintains various traffic statistics, including per-port byte and packet counters, and per-flow table entry byte and packet counters. In addition, each flow table entry can be configured with both hard and soft time outs.

Each Flow entry contains:

- **Match fields:** it is the field matching the incoming packets.
- **Priority:** matching priority of the flow entry.
- **Counters:** it contains metrics for the mated packets.
- **Instructions:** to modify the action set or pipeline processing.
- **Timeouts:** maximum amount of time before flow expiration.
- **Cookie:** data value chosen by the Controller. Used by Controller.

Another basic component of OpenFlow switch is the group table. A group table consists of group entries, which contains a number of flow entries all sharing a common attribute.



Components of Group Table

Group Identifier	Group Type	Counters	Action Buckets
5			Out port x,y
15			Out port a
6			Group ID 5
7			Out port m, Group ID 15
9			Drop Packet

- **Group Type** – All, Select, Indirect, Fast Failover
- **Counters** – Updated when packets are matched

Figure 2.9: Components of Group Table

Each group entry consists of:

- **Group Identifier:** integer for group identification.

- **Group Type:** Group types can marked as *Required* or *Optional.* If a group marked as *Required* ,can have two types available. The **all** and the **indirect.** The *Optional* groups also can have two types available. The **select** and the **fast failover.**

- **Counters:** it contains metrics for the mated packets.

- **Action Buckets:** each bucket contain a list of actions to execute.

Finally, in the OpenFlow switch a critical process is how the packets associate with the flow table.This is determined in OpenFlow pipeline process.

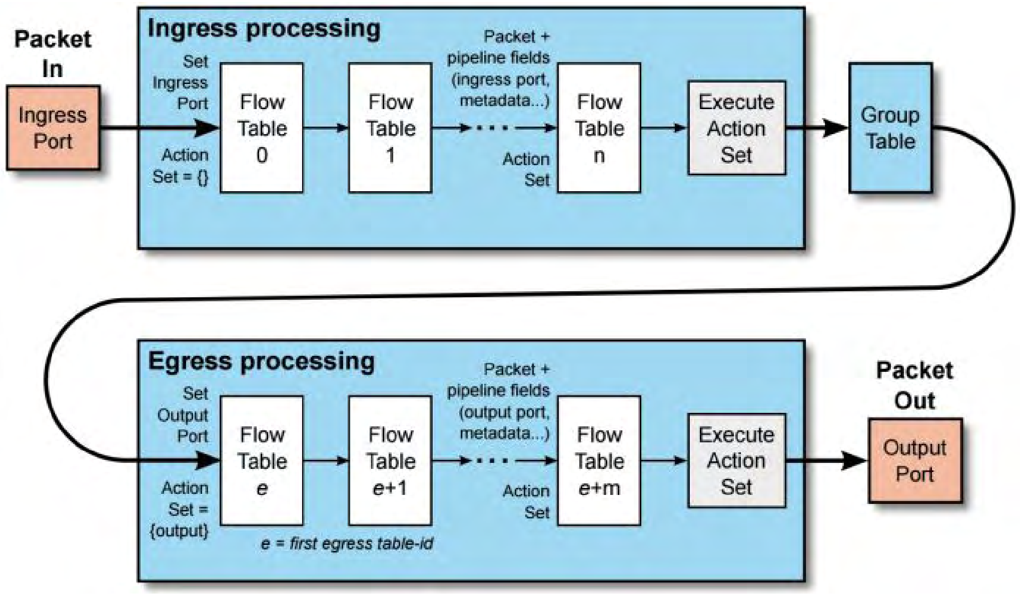


Figure 2.10: Pipeline Process

As we can see from the *Figure 2.10*, the flow table has numbers starting from zero. The process starts when a new packet arrives in the OpenFlow switch.

Then, if the packet matches a flow entry, the flow table executes the instructions which are stored in the appropriate flow entry. These instructions can be sent from the packet to another flow table according to the Goto instruction or to terminate the pipeline process and the packet is processed in proportion to the associated actions.

If the packet has no match with any flow entry of the flow table, the packet is then inclined if the flow table has no table-miss flow entry. If not, the flow table has a table-miss flow entry. Then the packet is processed according to the table-miss configurations.[19]

OpenFlow switches divided in two categories, commercial and software.

Commercial switches are physical devices with hardware to support OpenFlow. Typical commercial switches are HP [23], Pica8 [24], and NEC [25].

Contrary to commercial switches, software switches have the ability to use and install in every type of hardware. Typical commercial switches are Open-WRT[26], and OpenVSwitch[27].

In this thesis we will use the OpenVSwitch.

OpenFlow Controller

The controller is the brain of the network which uses OpenFlow protocol to interact with the switches. It can manage, control and administrate the flow tables.

Also, all the SDN controllers are written in software and are an independent application running in a dedicated server. Also they are flexible and dynamic, because they are easily programmable in hardware.

As we said above, it has the whole functionality to communicate with the application layer (via northbound api) to define the applications needs for managing effectively the switch's behavior (via southbound api).

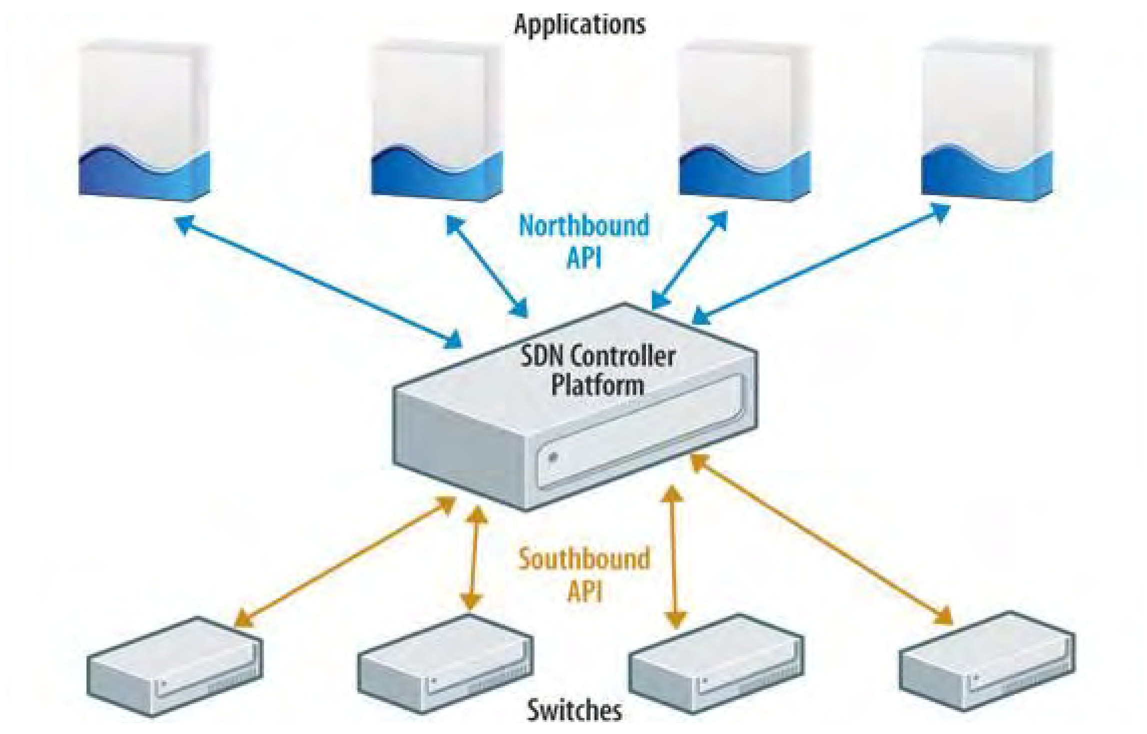


Figure 2.11: SDN Controller

The switch's behavior is managed, by controlling its flow tables. Flow tables can be inserted with rules formerly (pro-active) or after a packet arrives (react-active).

- **Reactive installation:** When a new flow arrives to the switch, the switch examines if there is a match for the specific flow in the flow table. If the flow is not matched, the switch makes an OpenFlow packet-in packet message and sends it to controller for instructions. The reactive installation process,considers the OpenFlow controller’s command for the creation of a rule in the flow table.[19]

- **Proactive installation:** To avoid the above technique,which demands the controller reacting to packets, the proactive installation obliges the OpenFlow controller to generate the flow tables ahead of time for all traffic matches that could come into the switch. By pre-defining all the flows in the flow table, the packet-in event never happens, as a result the effacement of the latency communicating with the controller on every flow.[19]

Nowadays, there is a variety of SDN controllers with different characteristics and functionalities.They can vary from the programming language to software performance. A few popular open source controllers are Pox [29], Nox [28],Floodlight[30], OpenDayLight [33], Ryu[34] and Cisco APIC [35], HP VAN[36], VMware NSX [37], Trema[32], Onos[31] are some popular commercial ones.

- **NOX [28]:** NOX is an open-source platform, which is written in C++. Also, it provides an OpenFlow 1.0 API.

- **POX[29]:** It consists in a NOX implementation written in Python. It is suitable for research and academic purposes, because it is easy to start with. The main advantage of NOX is that it is suitable for the configuration of big complex networks and controller needs to be fast. A disadvantage of POX is that it does not support multithreading.[19]

- **Floodlight[30]:** Floodlight is a controller based on Java, or Jython. Floodlight was developed by David Erickson.It supports multithreading and was developed so as to work with an increasing number of network devices that support OpenFlow. The main advantage is that can it be used in a heterogeneous network, because it can have a mix of OpenFlow and non-OpenFlow devices inside the network.[19]

- **OpenDayLight [33]:** It is an open-source SDN controller dated back in 2013 and is widely used not only for academic purposes but also and by industry members with the goal to make SDN more diaphanous and to act as basis for Network Function Virtualization (NFV). It is written in Java and is a highly available and scalable controller,suitable for heterogeneous networks.[19]

- **ONOS [31]:** Developed in 2014, ONOS (Open Network Operating System) is an open source project, created by The Linux Foundation. The purpose of the project was to create a high performance software-defined networking (SDN) operating system.[19]

- **Trema [32] :**Trema is a full-included framework developed by NEC and based on the Ruby and C. The Trema framework supports only the version 1.0 OpenFlow and it is also used as an emulator for OpenFlow networks. [19]

3

Design & Implementation

As we mentioned, the problem of efficient association to Access Points (APs) in WiFi is critical for the networking research community. Even though many scientific approaches have been made in the aforementioned topic, only a few have addressed an efficient mechanism for association including the SDN concept.

For that reason, this thesis introduces an algorithm associating users to best available APs in a WiFi network while taking advantage of SDN and OpenFlow opportunities. Especially, an SDN controller examines periodically the load traffic of all Wifi APs and associates every incoming user to the best available. This load traffic examination is done via various statistics APs send to the SDN controller, which collects them and calculates the available throughput bandwidth of each AP can provide to the incoming user.

The rest of the chapter is organized as follows: In 3.1 the basic implementation scenario is explained, where the algorithm is based on. In 3.2 the principles and the construction of the algorithm is introduced, while in 3.3 the construction tools used are presented.

3.1 Scenario

To examine the algorithm efficiency, a basic scenario is introduced as shown in *Figure 3.1*

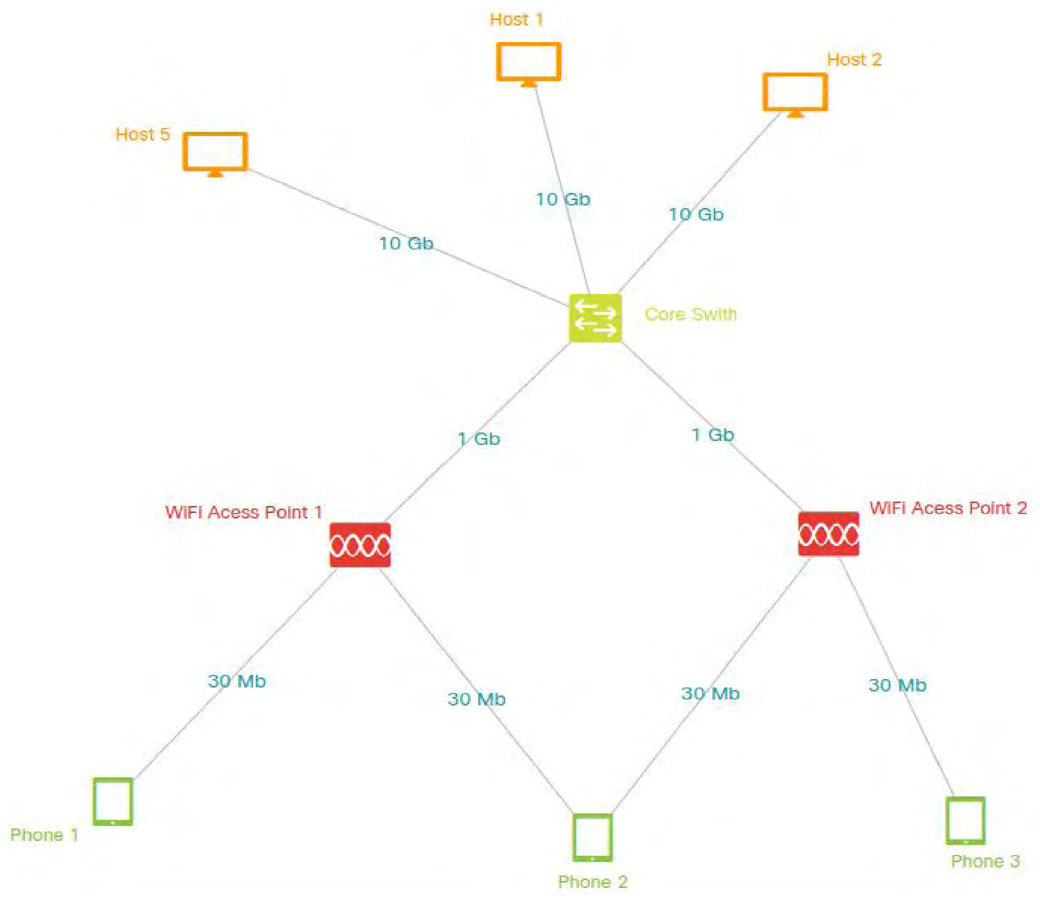


Figure 3.1: Basic Scenario

Scenario Components

As we can see in the scenario topology, the following components appear: two WiFi Access Points, three cellphones and one core switch which is used as a gateway to the Internet and --for this scenario-- is connected with three virtual hosts.

The two WiFi Access Points are actually two OpenFlow Switches and specifically two OpenVSwitch, which communicate via OpenFlow protocol with an SDN Ryu Controller.In the scenario, the two WiFi APs have wired and wireless interfaces.The wired interface is responsible for the wired connection between the AP and the core switch,while the wireless interfaces are responsible for the wireless connection with the cellphones. Similar to the concept of APs, the core switch is an OpenFlow Switch, which connects with the same SDN controller and its setup had been made in Mininet Emulator and in NITOS testbed physical nodes as it did for the APs.The core switch is connected with three Hosts via wired interfaces provided by Mininet.

Scenario

At the beginning of the experiment, the first cellphone (phone 1) comes and wants to associate with an AP. Because of its network position, it had only the AP 1 available for connection. Also, at that moment, the load traffic in the AP 1 is zero, because no connection is established, so the SDN controller associates the phone 1 to the AP 1. During the above process, a new cellphone (phone 3) comes and wants to associate with an AP.Similarly, because of its position, it had only the AP 2 available for connection, so the SDN controller associates the phone 3 to the AP 2.

When the two connections are established, the down-link transmission of data began, from hosts to cellphone 1 and 3. The above transmissions create load traffic in each AP.

The algorithm is designed when a new cellphone (phone 2) comes and wants to connect to an AP. But now, phone 2 has two APs available for association,because of its position in the network.In this case, the SDN controller examines the load of the two available APs, by calculating the throughput of the existing flows in each AP.Finally, the AP with the most available free bandwidth calculated by controller will be associated with phone 2.

3.2 Algorithm

Algorithm Explain

The basic concept of the algorithm is to efficiently manage the load traffic of all WiFi APs and associate the incoming user with the most appropriate.

The algorithm implementation is included in an SDN Ryu Controller, which is connected with all available OpenFlow Switches (two WiFi APs, one core switch).

The algorithm estimates three basic metrics for the calculation of the load traffic of APs.

First, calculate the **current throughput load of AP**, second the **wireless link capacity** and last it estimates the **available Bandwidth** an AP can provide from the abovementioned metrics.

Current throughput load of AP

The controller inquires periodically after 10 seconds, the flow statistics from the two Wifi APs. These statistics are obtained from APs flow tables and reflect the load traffic of AP. When the controller acquired these statistics, the algorithm starts the calculation with the below formula

$$throughput = \sum_{0}^{i} ((byte_i(t) - byte_i(t-10)) \times 8)\, bits \div 10\, sec$$

,where i is the number of flows in each AP's flow table, except the flow with destination the cellphone 2.

As we can see, the algorithm takes a statistic, the counted *byte* ,and subtracts it from the previous counted *byte*. The result is multiplied by 8, because we want the *throughput* to be in bits and finally is divided by 10, because 10 is the period of measurement. The result represents the current down-link throughput of all flows in an AP in bits/second. Because we want to examine the load traffic of the AP, we must include in the above formula all the existing flows of down-link transmission in each AP.

Specifically, in AP 1, the formula includes all the existing flows which manage down-link traffic from all available Hosts(host1,host2,host5) to cellphone 1. Similarly, in AP 2 we concentrate in all existing flows which manage down-link traffic from all available Hosts (host1,host2,host5) to cellphone 3.

As we can observe, the formula did not include down-link flows with destination the cellphone 2, because as we described above in the scenario,the cellphone 2 is the target phone we want to find the best available AP to associate. So we do not include flows to phone 2 in our measurement, because they will affect the final throughput calculated measurement.

The above process calculating the throughput of all down-link flows in the two APs in bits/second and for that reason is called *current throughput load of AP.*

Capacity of Wireless link

In addition to the calculation of *current throughput load of AP,* the algorithm examines the real *capacity of the wireless link* of each AP. We want to know the exact capacity of wireless link because we concentrate in the down-link transmission between APs and cellphones.

For the examination of *capacity of the wireless link* we tried a variety of techniques.First, we want to get this information from the wireless driver of the APs, but the wireless driver did not offer this kind of information. Finally, the measurement of real capacity of wireless link is done with the Iperf tool between two physical nodes, which is wirelessly connected in NITOS testbed. We observed that the maximum bandwidth the wireless link can provide between two NITOS nodes, is approximately in a range between 25-30M bits/second. So, we assumed that the upper bound of a NITOS wireless link is 30 Mbit/sec and for that reason the maximum capacity of the wireless link between an AP and a cellphone is 25-30 Mbit/sec.

Available Bandwidth

Having managed the *current throughput load of AP* and the *capacity of the wireless link* the algorithm with a simple subtraction calculate the *available bandwidth*

available bandwidth = capacity of the wireless link - current throughput load of AP

which represent the load condition of the wireless link.

The SDN controller, knowing every moment the load condition of the wireless link of each AP can associate the cellphone 2 with the AP with the maximum *available bandwidth.*

Algorithm Programming Components

The entire algorithm logic is included in the SDN Ryu Controller. We will present some basic parts of the programming code, where the algorithm is based on. For the programmable part of the algorithm we used functions provided by OpenFlow protocol to obtain our metrics.

As mentioned above, the controller inquires periodically after 10 second flow statistics especially from the two Wifi APs. In particular, the controller calls a function

def send_flow_stats_request(self, datapath) every 10 seconds.

The function *send_flow_stats_request* has as definition the variable *datapath*, which is the address of the two WiFi APs.

The controller send a *OFPFlowStatsRequest* to the specified by *datapath* AP.This message request all the information for flows in flow table and specific request information for flows matching

OFPMatch(eth_type=ether_types.ETH_TYPE_IP).

This means that the controller requests information for all the IP flows in the two APs flow tables.

The answer to the above request comes asynchronous to the controller. The controller can receive it with

@set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER).

So, when controller receives an *OFPFlowStatsReply* message from an AP, it calls a function named *def flow_stats_reply_handler(self, ev),* which handles the above message.

The function *flow_stats_reply_handler(self, ev)* first examines from which of the two possible APs it received the reply message.

Next, from the variable *ev it* takes all information about the flows and more specifically it gathers and sums the counted byte statistics matched for all IP flows with destination the cellphone 1 or cellphone 3, according to the APs.

```
def flow_stats_reply_handler(self, ev):
msg = ev.msg
body = ev.msg.body
for stat in ev.msg.body:
ipv4_addr =stat.match['ipv4_dst']
if ipv4_addr != HOST2_IPADDR:
self.byte_count=stat.byte_count+ self.byte_count
```

The aggregation of counted byte for the specified flows is included in the variable

self.byte_count

For the calculation of the current throughput load of APs we used the above formula. So,

```
throughpout_flow=float((self.byte_count - self.last_flow_metric)*8)/10
self.last_flow_metric_s2=self.byte_count
```

Where the *self.byte_count* variable is subtracted with the previous calculated metric (*self.last_flow_metric*) and multiplied by 8, because we want the throughput to be in bits and finally is divided by 10, because 10 is the period of measurement. The final result represents the current throughput of all flows in a AP in bits/second.

At last, we stored the *self.byte_count* variable for our next throughput calculation, when a new *OFPFlowStatsReply* message arrives.

The above process calculates the *current throughput load* of the two Aps. For the association technique, first, the two *throughput* are compared and the AP with the minimum *throughput* is the best available option to associate the cellphone 2.

As we can see, we compare the two throughput with

```
if self.throughpout_flow_list[2]<=self.throughpout_flow_list[3]:
```

If the *current throughput load* of AP 1 is less than AP 2 we associate the cellphone 2 to the AP 1, by changing the flow table of the core switch.

```
match =
datapath.ofproto_parser.OFPMatch(in_port=3,eth_type=ether_types.ETH_TYPE_ARP,
arp_tpa=HOST2_IPADDR)
actions=[datapath.ofproto_parser.OFPActionOutput(1)]
self.add_flow( self.datapaths[1],1, match, actions , ofproto.OFPFC_MODIFY )
```

```
match =
datapath.ofproto_parser.OFPMatch(in_port=3,eth_type=ether_types.ETH_TYPE_IP,
ipv4_dst=HOST2_IPADDR)
actions=[datapath.ofproto_parser.OFPActionOutput(1)]
self.add_flow( self.datapaths[1],1, match, actions , ofproto.OFPFC_MODIFY)
```

By adding the above python commands, the controller sends a message to the core switch (self.datapaths[1]) and changes the flows in its flow table.The controller adds a new flow rules in core switch's flow table, which indicates that when a Host (host1,host2,host5) wants to communicate with the cellphone 2, the only way is by sending the data traffic in the port(1), namely the AP 1.

A similar process took place if *the current throughput load* of AP 2 was less than AP 1. Then we associated the cellphone 2 by changing the flow table of the core switch. The main difference, as we can see below, is that the controller adds a new flow rule in core switch's flow table, which indicates that when a Host (host1,host2,host5) wants to communicate with the cellphone 2 the only way is by sending the data traffic not in the port(1) but in the port(2), namely AP 2.

```
match =
datapath.ofproto_parser.OFPMatch(in_port=3,eth_type=ether_types.ETH_TYPE_ARP,
arp_tpa=HOST2_IPADDR)
actions=[datapath.ofproto_parser.OFPActionOutput(2)]
self.add_flow( self.datapaths[1],1, match, actions , ofproto.OFPFC_MODIFY )
```

```
match =
datapath.ofproto_parser.OFPMatch(in_port=3        ,eth_type=ether_types.ETH_TYPE_IP,
ipv4_dst=HOST2_IPADDR)
actions=[datapath.ofproto_parser.OFPActionOutput(2)]
self.add_flow( self.datapaths[1],1, match, actions , ofproto.OFPFC_MODIFY)
```

3.3 Tools Used

As previously described, the two Wifi Access Points and the core switch are actually two OpenFlow Switches. The setup of core switch had been made in Mininet Emulator and in NITOS testbed physical nodes while the two WiFi Access Points setup based only in NITOS.All these components communicate via OpenFlow protocol with an SDN Ryu Controller.

3.3.1 Ryu Controller

The SDN controller used in this thesis is a Ryu [34] controller. All the programming logic and the algorithm implementation is included in the Ryu controller.

Ryu is one of the most popular SDN controllers in the industry. Ryu is based on Python and can support multi-threading.The main advantage of Ryu is that provides software components with well defined API that make it easy for developers to create new network management and control applications. It currently supports a variety of southbound protocols such as OpenFlow, OF-Config,etc.About OpenFlow, Ryu supports fully 1.0, 1.2, 1.3, 1.4, 1.5 and Nicira Extensions. All of the code is freely available under the Apache 2.0 license.Last,Ryu have a main executable called Ryu manager where it listens to a specific IP address and on port 6633, the standard OpenFlow port.[38]

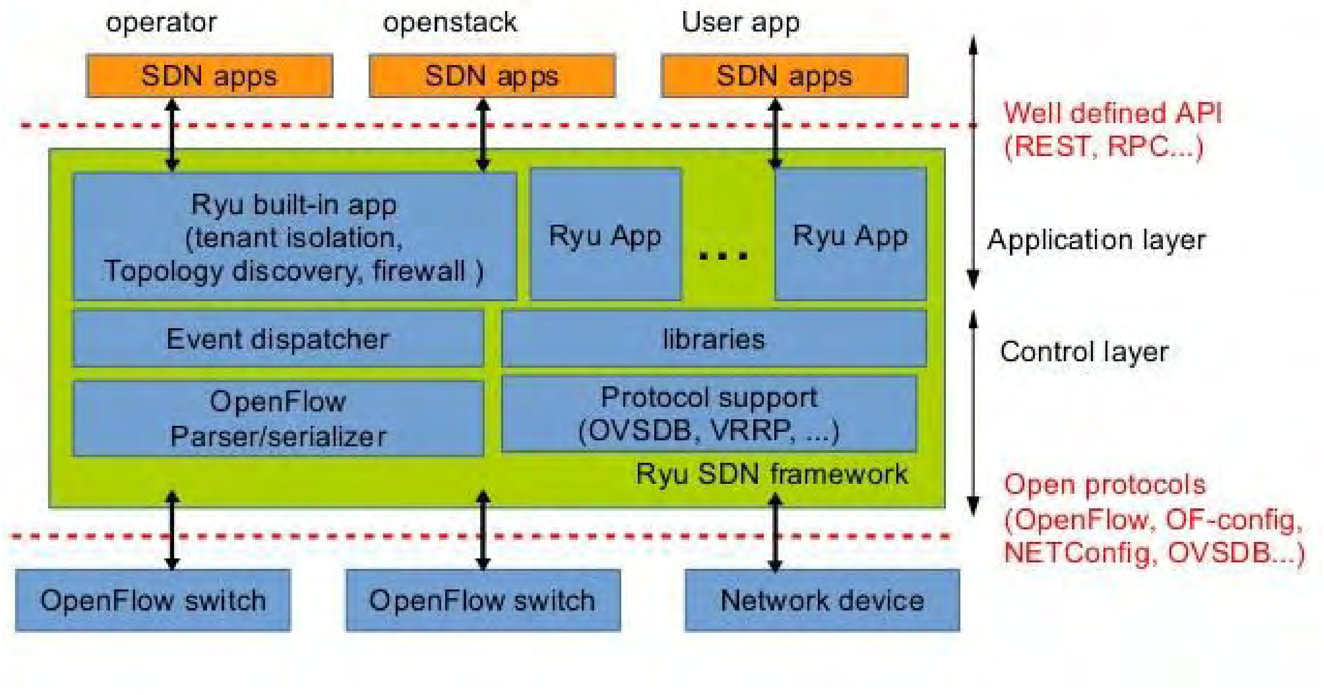

operator
openstack
User app
SDN apps
SDN apps
SDN apps
Well defined API (REST, RPC...)
Ryu built-in app (tenant isolation, Topology discovery, firewall)
Ryu App
...
Ryu App
Application layer
Event dispatcher
libraries
Control layer
OpenFlow Parser/serializer
Protocol support (OVSDB, VRRP, ...)
Ryu SDN framework
Open protocols (OpenFlow, OF-config, NETConfig, OVSDB...)
OpenFlow switch
OpenFlow switch
Network device

Figure 3.2: Ryu Architecture

3.3.2 Open vSwitch

The three basic components in our scenario (two WiFi APs, one core switch) are OpenFlow Switches and specifically Open Vswitch.

The first OpenFlow switch is Open vSwitch(OVS)[39]. It is purely a software based implementation which can use OpenFlow protocol to transmit flow table data from controller to the switching hardware. It allows programmatic control and vendor independent management interfaces. It is an alternative to Linux native bridges and VLAN interfaces.

There are two basic components inside Open vSwitch. The first has the control of the management layer and called ovs-vswitchd and the second one has the control of the forwarding plane and called OVS kernel module. Ovs-vswitchd is in the user space and communicates with the user and collects packet information and flow routes while kernel module talks to the NIC for collecting packets and it is in the kernel space.[38]

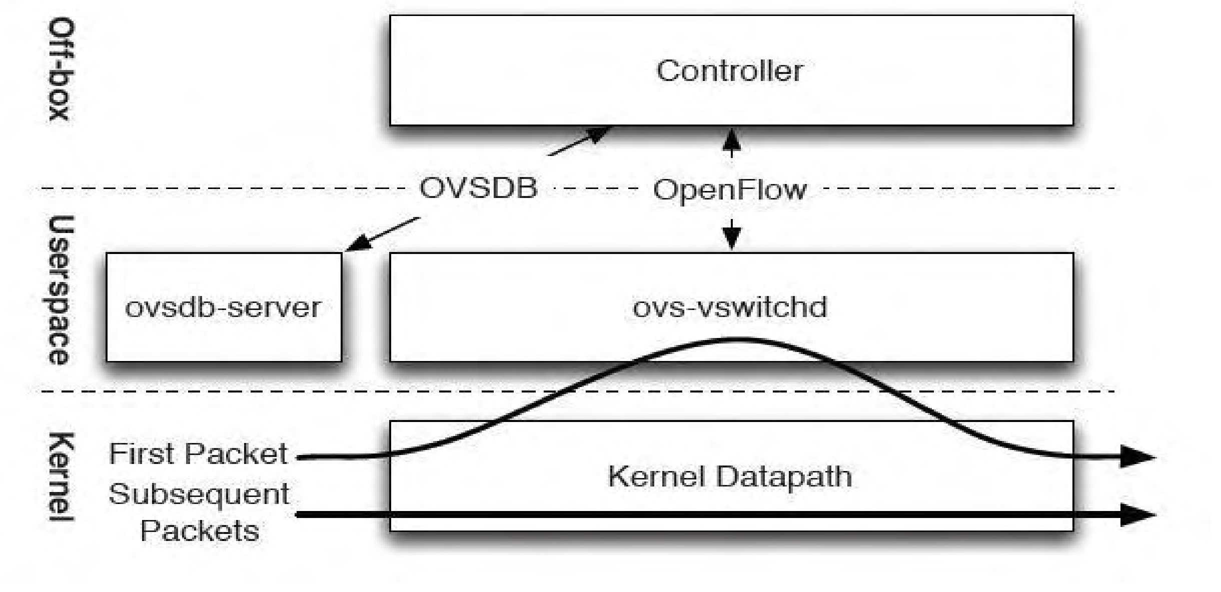


Figure 3.3: OpenVSwitch

As we have mentioned, the setup of the WiFi APs had been made in the physical nodes of NITOS testbed. For the connection of the WiFi physical NITOS APs with the SDN controller, the WiFi APs must act as OpenFlow Switch and especially Open Vswitch.

For the above configuration, two bash scripts were created.

```
ovs-vsctl add-br s3
ovs-vsctl set bridge s3 protocols=OpenFlow13
ovs-vsctl set-controller s3 tcp: $localhost address
ovs-vsctl add-port s3 eth1 -- set Interface eth1ofport=1
ovs-vsctl add-port s3 wlan0.sta2 -- set Interface wlan0.sta2 ofport=2
ovs-vsctl add-port s3 wlan0.sta1 -- set Interface wlan0.sta1 ofport=3
```

The above commands created an OpenFlow switch in the WiFi AP 2 ,which has protocol version OpenFlow 1.3 and three ports one for the wired connection to the core switch and two for the wireless connection to cellphone 2 and 3.

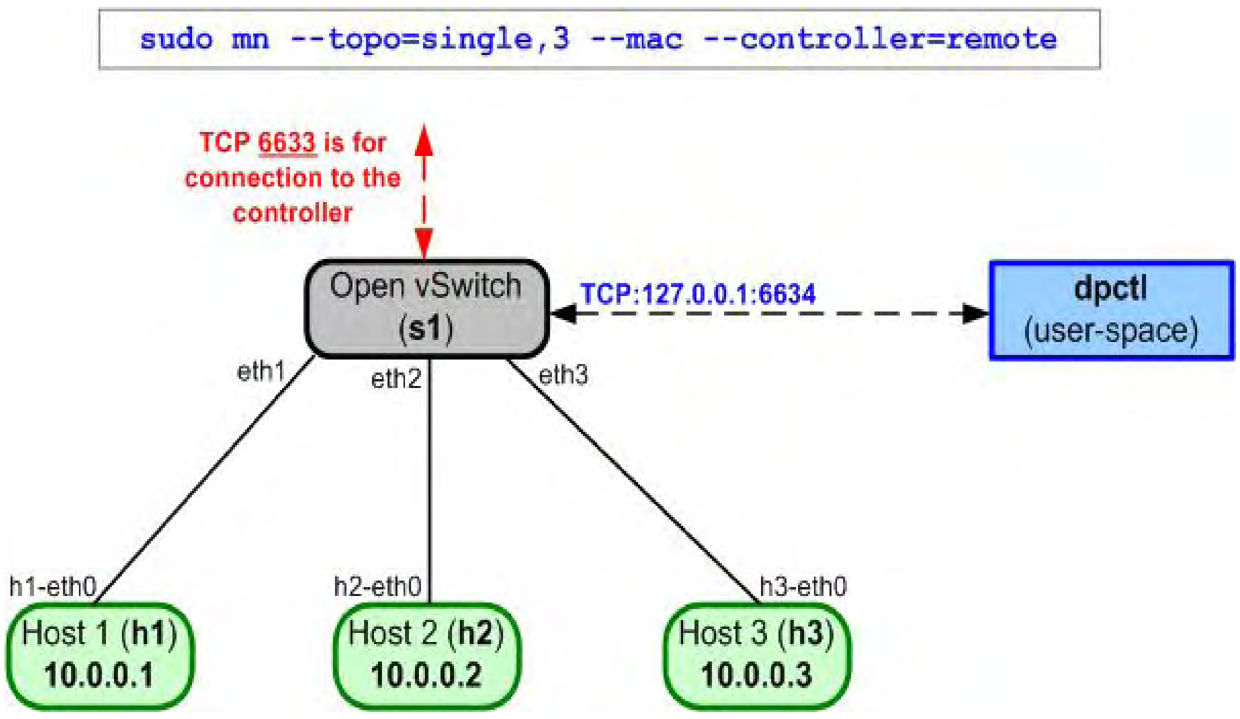
Similarly the other script setup the WiFi AP 1 acting as OpenFlow Switch.

3.3.3 Mininet

The other basic component of the scenario is the core switch. The core switch is an OpenFlow Switch, which connect with an SDN controller and via three virtual hosts.The setup had been made in Mininet Emulator.

Mininet is a network emulator based in Python. Its main task is to establish virtual hosts, switches and links, as a result in this virtual network topology new techniques can be easily tested, before moving in a real network system. Also, it supports multiple software based OpenFlow switches, custom topologies and provides a Python API for programmability of the network.The virtual components utilize the actual Linux network applications including the kernel and network stack for emulating devices.

Mininet can be either installed on a Linux system or it can be run in a Virtual Machine via VirtualBox[38].



```
sudo mn --topo=single,3 --mac --controller=remote
```

Figure 3.4: Mininet Emulator

For the above configuration of core switch,a python mininet script is written. Especially with the following commands,

```
core switch = net.addSwitch( 'core switch' , dpid='1', protocols=["OpenFlow13"])

host 1 = net.addHost( 'host 1', ip='192.168.1.1/24', mac='00:00:00:00:00:01' )
host 2= net.addHost( 'host 2', ip='192.168.1.2/24', mac='00:00:00:00:00:02' )
host 3 = net.addHost( 'host 3', ip='192.168.1.3/24', mac='00:00:00:00:00:03' )
```

the core switch and the three virtual hosts are created and then with

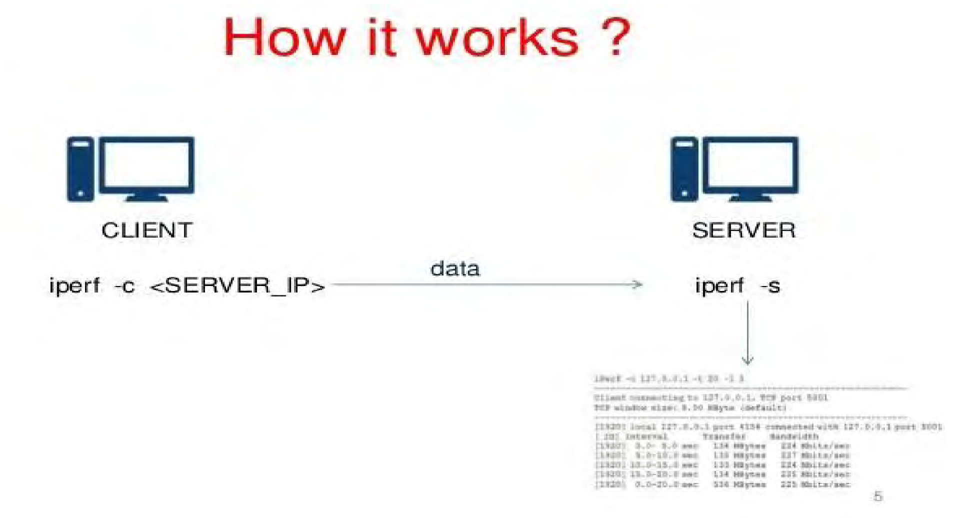
```
controller = net.addController( 'controller', ip=CONTROLLER_IP, port=6633)

net.get('core switch').start([controller])
```

the core switch is connected and communicates with the Ryu controller.

3.3.4 Network Benchmarking Tool

As we can imagine, all the algorithm implementation has purpose when the hosts started to transmit data to the cellphones.The transmitted data will be generated by a network tool called iperf[40].

Iperf [40] is a network testing tool, because it creates TCP and UDP data streams. The main functionality is that can generate traffic between two nodes and measure the throughput,bandwidth and the quality of a network link,considering clients and servers functionalities.[38]



```
iPerf -c 127.0.0.1 -t 20 -i 5
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 127.0.0.1 port 4138 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[1920]  0.0- 5.0 sec   134 MBytes   224 Mbits/sec
[1920]  5.0-10.0 sec   135 MBytes   227 Mbits/sec
[1920] 10.0-15.0 sec   133 MBytes   224 Mbits/sec
[1920] 15.0-20.0 sec   134 MBytes   225 Mbits/sec
[1920]  0.0-20.0 sec   536 MBytes   225 Mbits/sec
```

Figure 3.5: Iperf Tool

In this thesis, first, iperf with UDP stream traffic is used for the evaluation of the algorithm. Nevertheless, because of the wireless link between the APs and the cellphones, we had no guaranteed that the bits sent will be received by the cellphones. To tackle this problem, we had to use iperf with TCP stream traffic. But with TCP stream, we could not generate a specific amount of bits in our stream. Finally, we used iperf3[41], which gave us the opportunity to generate specific amount of data stream bits and the confirmation that anything sent would be successfully delivered to the cellphones.

3.3.5 NITOS Testbed

During the development of the implementation, two environments for evaluation and testing were used: Mininet emulator and the NITOS wireless testbed.

First all the implementation was based on Mininet, in sense that all the basic components such as the two APs, the core switch and the three hosts and cellphones were configured in Mininet and not in NITOS. Thus, it was easy to test our algorithm without extra work for further configuration. But, in the last stage of the implementation, the wireless concept and the setup of WiFi Access Points for association with the cellphones had to be introduced. Thus,the setup of the the core switch, the two Wifi APs and the three cellphones as the wireless connection between them implemented in the physical nodes of NITOS testbed.

NITOS (Network Implementation Testbed using Open Source code) is a wireless testbed managed by the Network Implementation Testbed Laboratory (NITlab) of the Electrical and Computer Engineering Department at the University of Thessaly in collaboration with the Center of Research and Technology Hellas (CERTH).The NITOS testbed gives researchers the opportunity to perform experiments and test their implementations in real-time environments.It consists of wireless nodes based on open source software.

The NITOS facilities are the NITOS outdoor testbed, the CERTH indoor testbed,and the Tholos indoor testbed.The control and Management Framework (OMF) is being used in order to control and manage the testbed.For this thesis, we will use six nodes from the Tholos indoor testbed.

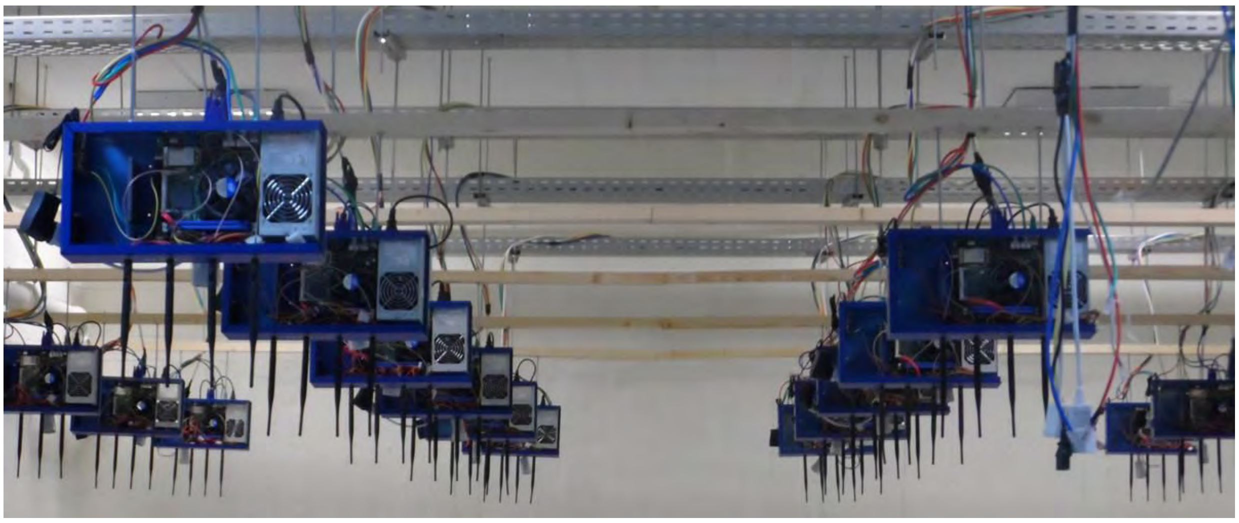


Figure 3.6: Tholos Indoor testbed

The Tholos indoor testbed is an isolated physical of nodes environment at the University of Thessaly's campus building. It consists of 40 Icarus nodes which contain multiple heterogeneous interfaces making the user capable of performing several realistic scenarios.

The WiFi ICARUS nodes have been developed by NITlab team. The basic manufacture characteristics of Icarus nodes are that they are equipped with 802.11a/b/g and 802.11a/b/g/n wireless interfaces and feature new generation intel 4-core cpu's and new generation solid state drives.

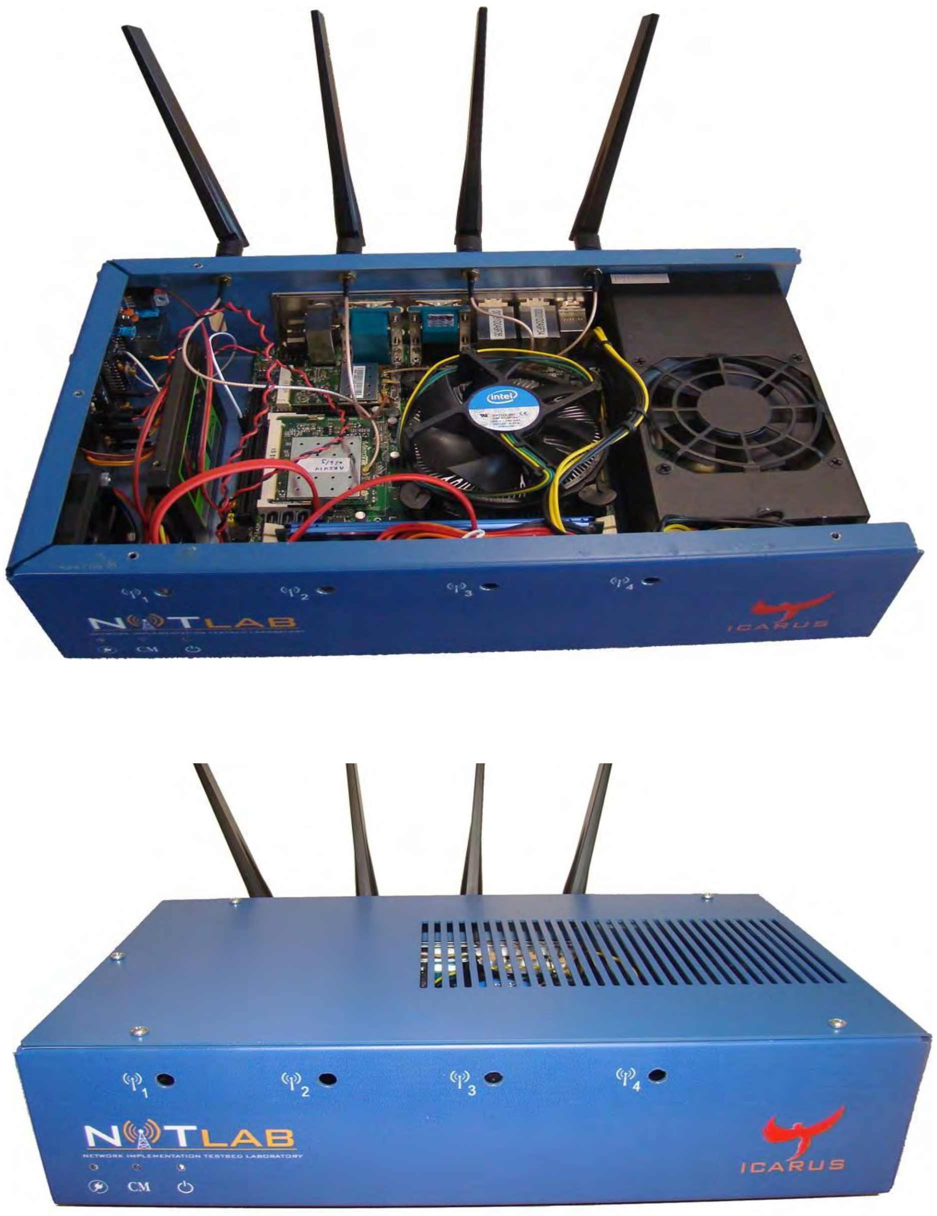


Figure 3.7: Icarus Node

The specification of Icarus Nodes can been included in the below table.

Specifications:

Motherboard	Features two Gigabit network interfaces and supports two wireless interfaces
CPU	Intel® Core™ i7-2600 Processor, 8M Cache, at 3.40 GHz
RAM	8G DDR3
Wireless Interfaces	Atheros 802.11a/b/g & Atheros 802.11a/b/g/n (MIMO)
Chassis Manager card	NITlab CM card
Storage	Solid state drive
Power Supply	350 Watt mini-ATX
Antennas	Multi-band 5dbi , operates both on 2.4Ghz & 5Ghz
Pigtails	High quality pigtails (UFL to RP-SMA)

Table 1: Icarus Nodes Specifications

As mentioned above, we used six Icarus nodes for our final implemented scenario.

The two Icarus nodes have been configured as WiFi Access Points, via two bash scripts,and they have created two different WLANs. Also, the rest of the nodes have been configured as stations (cellphones), which have been wireless connected in the two WiFi APs, while in the last one have been implemented the core switch which functionality based in mininet.

4

Demo

After the successfully implementation of the algorithm, which has as result the optimal association between cellphones and Access Points, this chapter of this thesis is based on demonstration of the above process. Specifically, in chapter 4.1 the basic components and the techniques for the construction of demo are presented, while in 4.2 the basic scenario is demonstrated.

4.1 Demo Construction

For the demonstration of the project,we had to create a real-time network web interface. This interface has the functionality to communicate successfully with the Ryu controller and obtain various metrics. This functionality makes the web interface act as a real-time interface,so that a user can easily verify basic elements of the algorithm when the scenario is running.

First, the programming language used for the deployment of the above web application was Html5,JavaScript and CSS.

HTML5[42] is a revision of HTML, the standard computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is the fifth and current major version of the HTML standard and was adopted by the new working group of the World Wide Web Consortium in 2007. One of the biggest differences between HTML5 and previous versions of the standard is that older versions of HTML require proprietary plugins and APIs. HTML5 provides one common interface to make loading elements easier.

Cascading Style Sheets (**CSS**)[43] is a style language used for describing the presentation of a document.Most often used to set the visual style of web pages and user interfaces written in HTML. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.

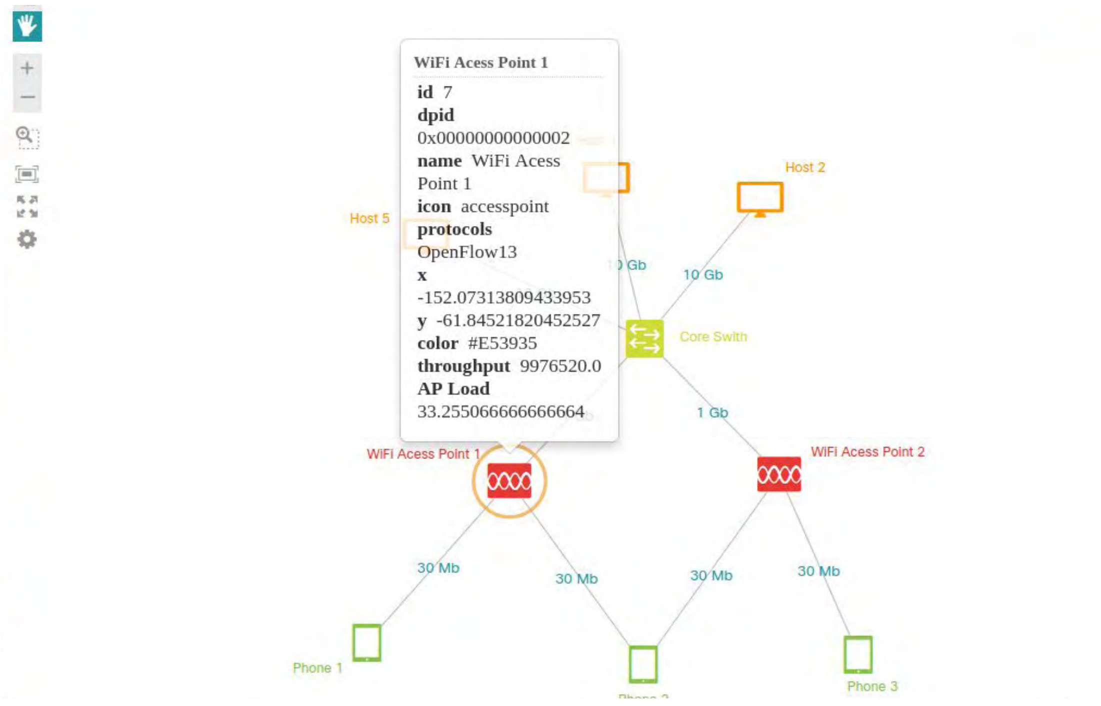
JavaScript [44] is a high level, object based, and interpreted programming or script language from Netscape. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web.It is used to make web pages interactive and provide online programs,including video games. Nowadays,the majority of websites employ it, and all modern web browsers support it without the need for further plug ins.It is a multi-paradigm language and supports different programming styles.

Beyond the above three programming languages,the network real-time web interface, is based on Cisco NeXt UI toolkit[45].

NeXt UI toolkit is an HTML5/JavaScript based toolkit for network web application, which provides high performance and quality framework and network centric topology component. Also, NeXt features MVVP, OOP and DOM manipulation, while it renders network topologies and enables user interaction with them through event listeners.

The basic advantage of NeXt UI is that the network topology is represented as a JavaScript object, essentially consisting of *nodes and links.* So, the developer can very easily construct a new network topology from beginning.

In this thesis all the network configured with NeXt UI toolkit, via two JavaScript scripts. The first contains all the available information for the configuration of the nodes and links in the topology. Specifically, via the first script, in every node is configured the name, the type (phone, access point, switch), the IP address and other basic elements.


WiFi Acess Point 1

id 7
dpid 0x00000000000002
name WiFi Acess Point 1
icon accesspoint
protocols OpenFlow13
x -152.07313809433953
y -61.84521820452527
color #E53935
throughput 9976520.0
AP Load 33.255066666666664

Figure 4.1: Nodes Configuration

Also with the same script all the links of the topology are configured.

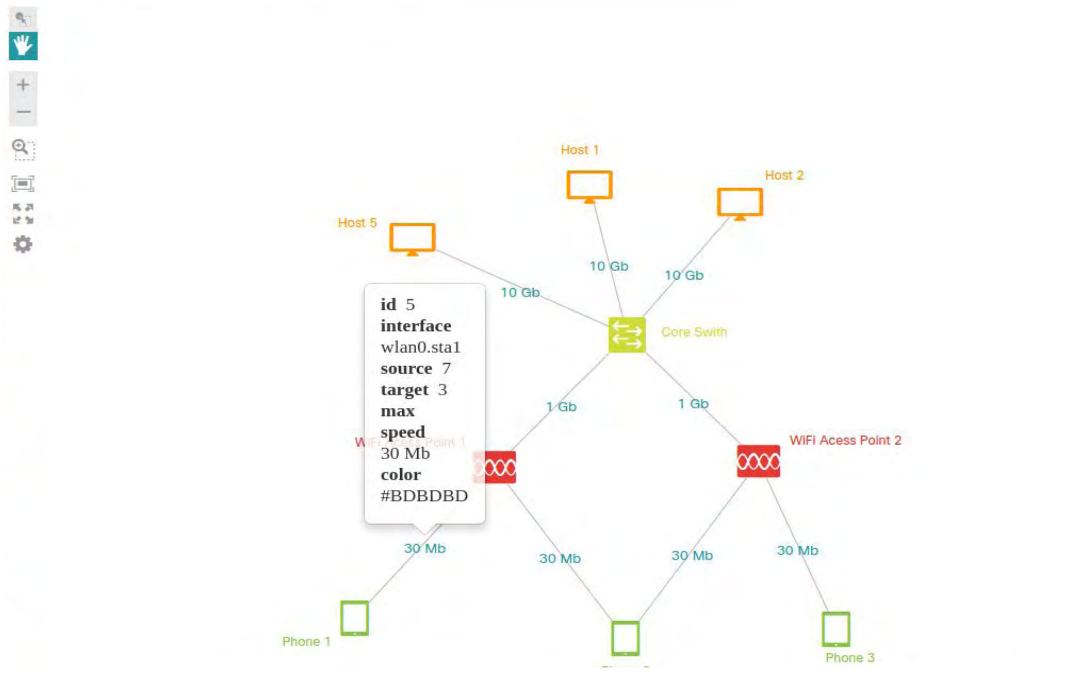

id 5
interface wlan0.sta1
source 7
target 3
max speed 30 Mb
color #BDBDBD

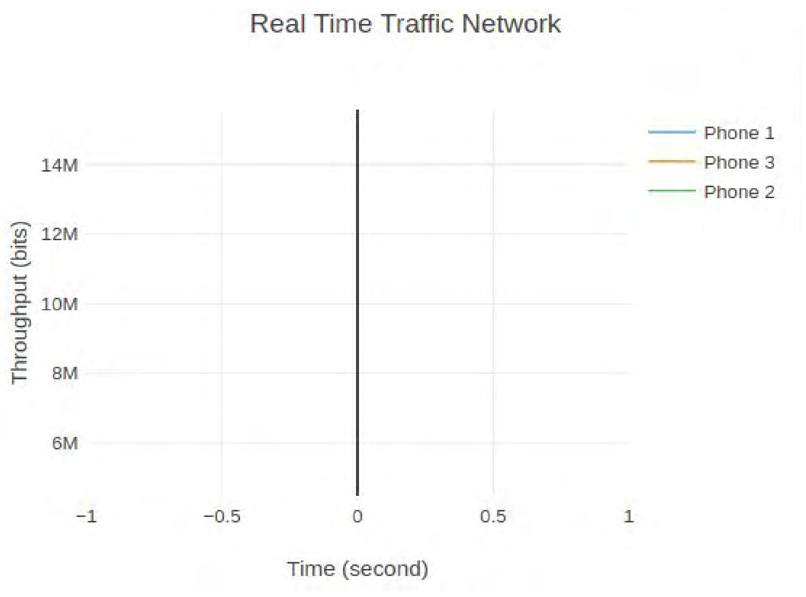
Figure 4.2: Links Configured

As mentioned, with the above script, all the network topology is configured. But, without the second JavaScript script the network topology it would only be a static snapshot in the web page with no functionality, real-time measurements from Ryu controller and interaction with the user.

To turn the above network topology into “life”, we had to introduce a connection mechanism between the Ryu controller and the web interface.The purpose was, that in case of a change in the Ryu controller and especially in the association algorithm, this will immediately reflect on the web interface. The statistic we wanted to see real-time in our demo was the available throughput in each WiFi APs.To establish the above real-time connection, a local CSV file was used.

Specifically, the Ryu controller writes in a CSV file the available throughput of each AP after 10 seconds and the web application read from the CSV file after 3 seconds.Thus, any change occurring in the AP’s throughput,displays real-time in the web page.

Also, the web interface,expect the real-time network topology contains and a real-time traffic graph for the all transmitted flows inside the network.



Graph 4.3: Graph of Traffic in Network

For the above implementation,we used the the JavaScript Graphic library plotly.js.Plotly.js[46] is a free open source interactive, high-level, Javascript graphing library, which is built on d3.js and webgl and supports over 20 types of interactive charts, including 3D charts, statistical graphs, and SVG maps.With plotly library we manage to create a real-time graph with the throughput statistics of all flows the Ryu controller handles.That, gives the opportunity to the user to examine the entire traffic in the network and observe how the implemented association algorithm handles this traffic.

4.2 Demonstration of Scenario

In this chapter we will present what happens in the web interface when a basic scenario is running. We will show the steps via snapshots, which appeared in the user.

Step 1

First, as we mentioned in chapter 3.1 in the scenario explanation, the cellphone 1 comes and associates with AP 1, the cellphone 3 associate with the AP 2 and after that the cellphone 2 had the option to associate with one of the two available APs.

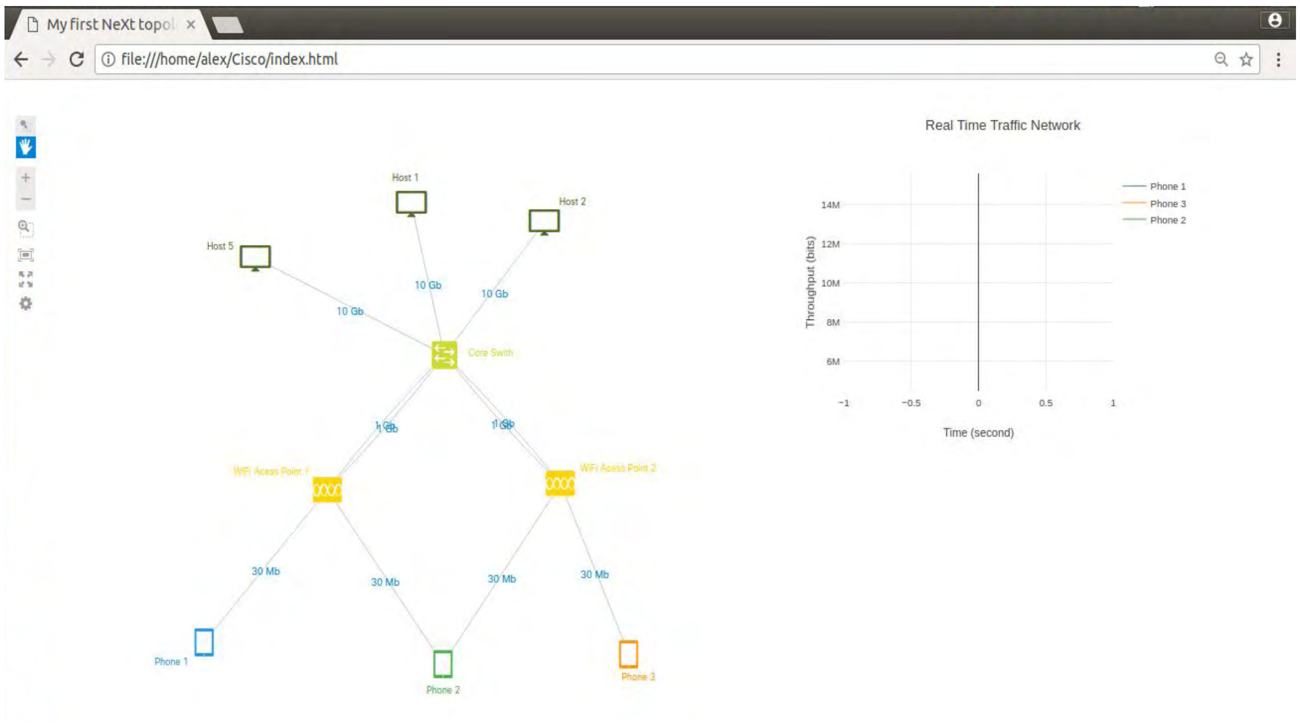
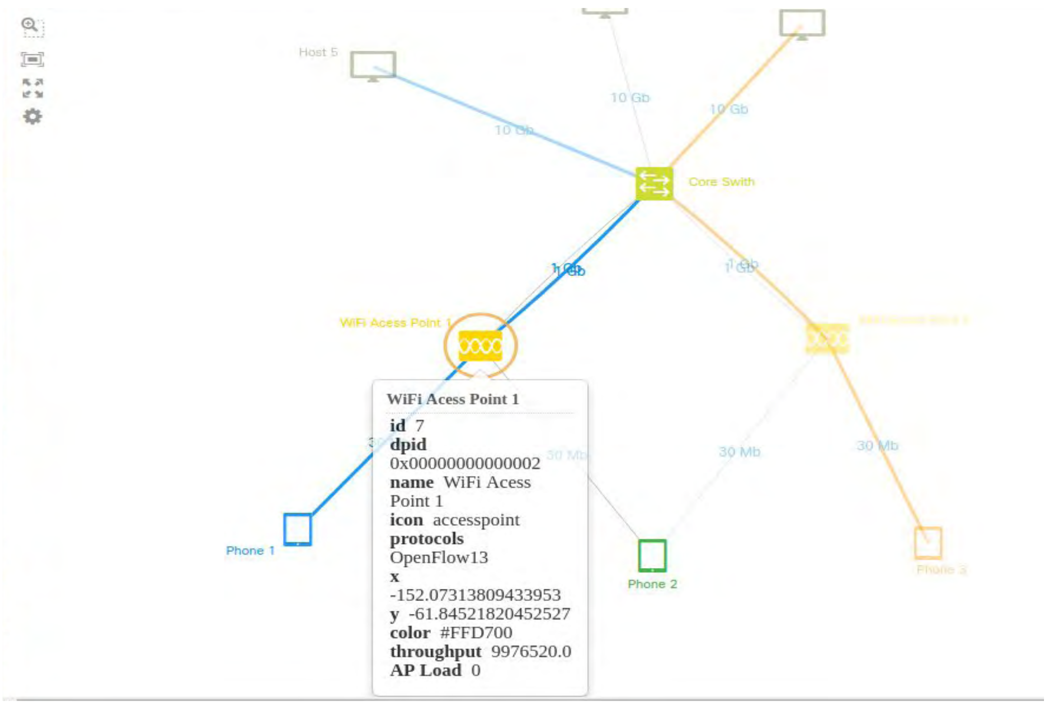


Figure 4.4: Start of the Scenario

The figure above is the network topology without any transmission flows between the hosts and the cellphone. As we can see, there is only the static topology with the information about all the network components and at right a graph which represents the traffic in the network,which is in this case is zero.

Step 2

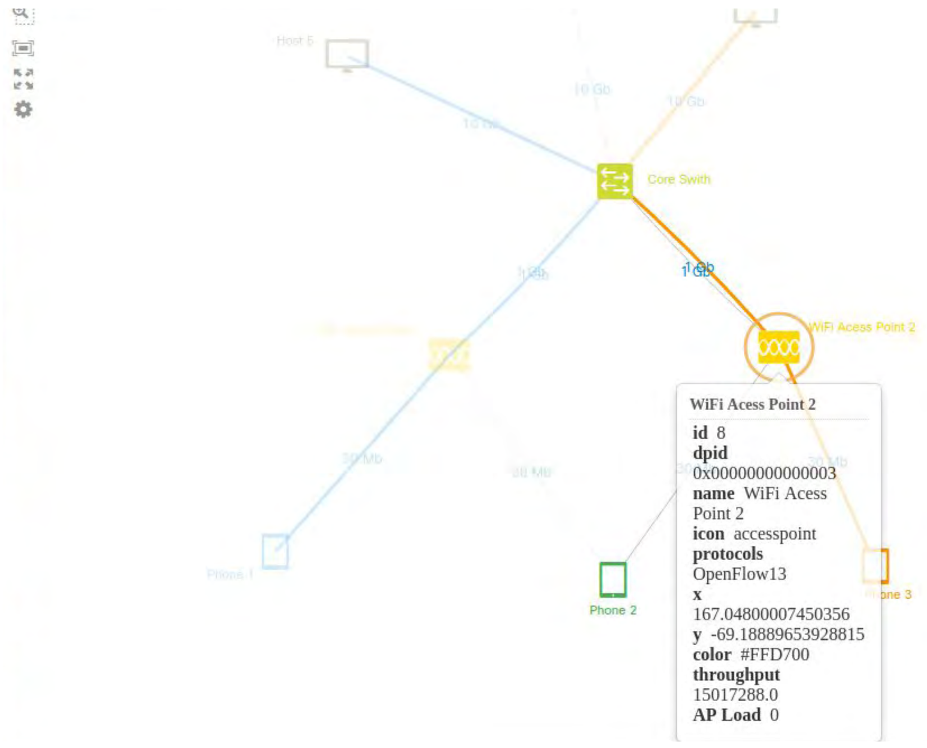
At the next moment, following the basic scenario, the transmission of data between the Hosts starts --in our case host 5, and phone 1. Similarly, a new transmission starts between host 2 and phone 3.Specifically, the flow from Host 5 to phone 1 has throughput rate of 10 Mbits/second, while the flow from Host 2 to phone 3 has rate of 15 Mbits/second. Because the transmission between the Host 5 and phone 1 is performed through the AP
1, this information reflects in the AP 1 statistics .



WiFi Acess Point 1
id 7
dpid 0x00000000000002
name WiFi Acess Point 1
icon accesspoint
protocols OpenFlow13
x -152.07313809433953
y -61.84521820452527
color #FFD700
throughput 9976520.0
AP Load 0

Figure 4.5: Host5 → Phone 1 Transmission

As we can see the WiFi AP 1 has throughput rate 9976520 or 10 Mbits/sec. Similarly, the Wifi AP 2 has throughput rate of 15017288 or 15 Mbits/sec as we can see below.



WiFi Acess Point 2
id 8
dpid 0x0000000000000003
name WiFi Acess Point 2
icon accesspoint
protocols OpenFlow13
x 167.04800007450356
y -69.18889653928815
color #FFD700
throughput 15017288.0
AP Load 0

Figure 4.6: Host2 → Phone 3 Transmission

The final snapshot of the two available flows and the throughput rate of it are presented below. With the blue color appears the flow between the Host 5 and phone 1 with throughput rate of 10 Mbits/second as we can see in the graph at the the right. With the orange color appears the flow between the Host 6 and phone 3 with throughput rate of 15 Mbits/second.

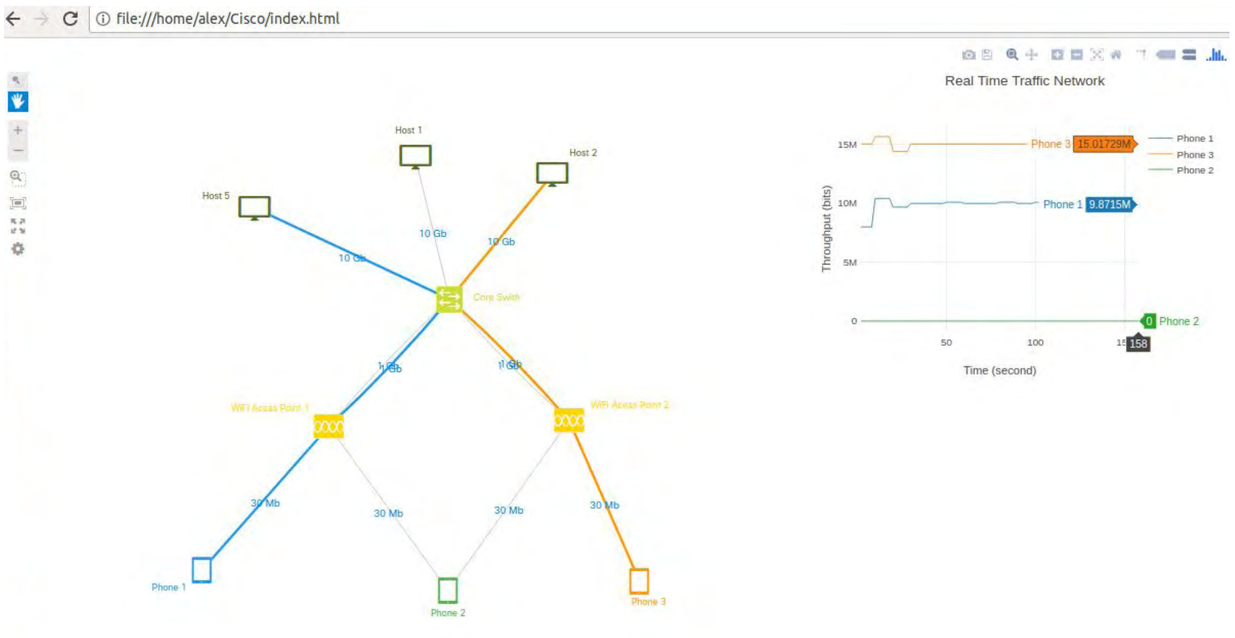


Figure 4.7: Snapshot of Step2

Step 3

After the transmission of data with destination the cellphone 1 and 3, a flow started between the host 1 and phone 2.This flow is represented with green color and has two available options to transmit the data. In this case data transmission is done through AP 1 because of the throughput of AP 1 is less than the throughput of AP 2.

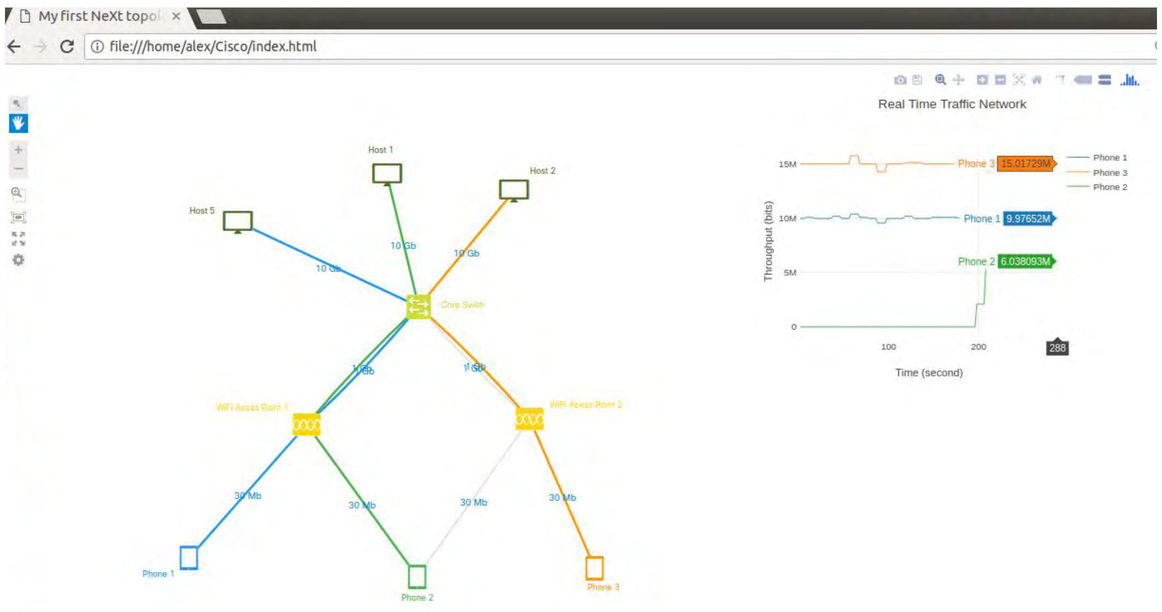


Figure 4.8: Snapshot of Step 3

Step 4

Assuming that the transmission rate of the blue flow changed from 10 Mbits/second to 20 Mbits/second, then the association algorithm would decide that the best available AP is the AP 2.So, the green flow would go through the AP 2.

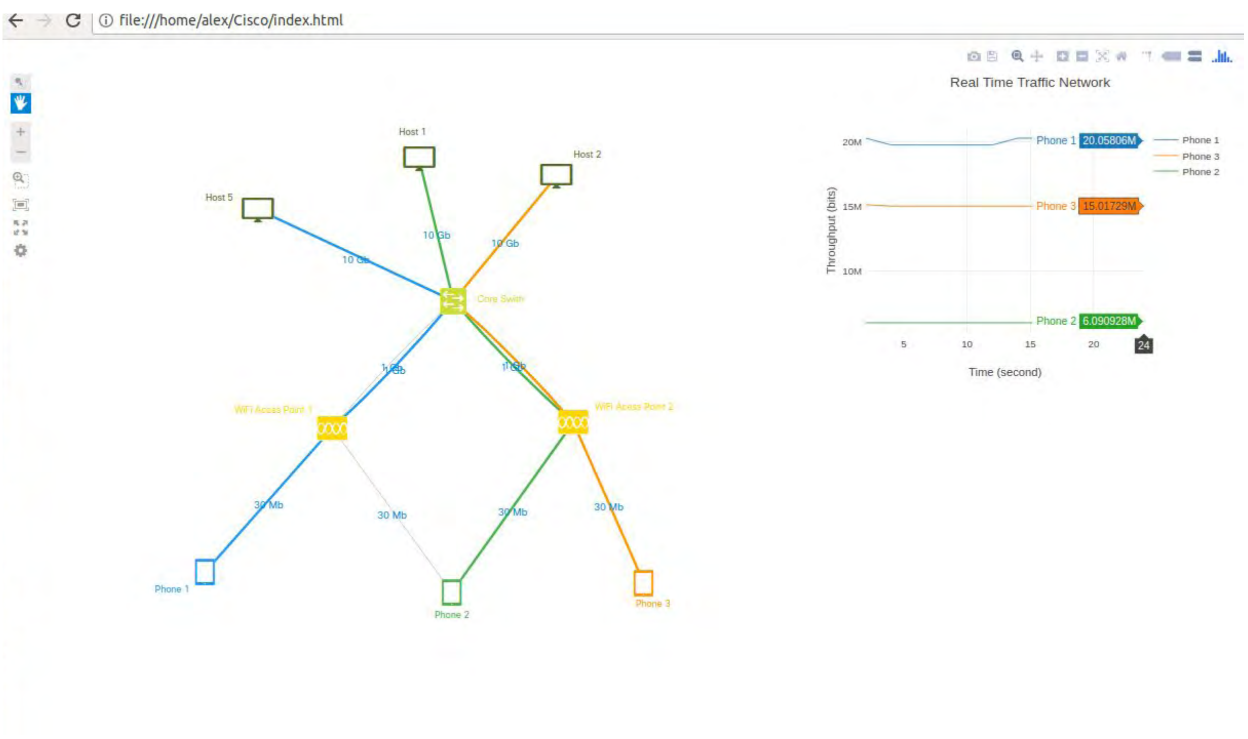


Figure 4.9: Snapshot of Step 4

5

Conclusion

The problem of efficiently association the users with Access Points in WiFi is critical and many scientific approaches have been made in the aforementioned topic. But, only a few have addressed a reliable mechanism for association including the SDN concept. Taking advantage of the global network view of SDN and OpenFlow functionalities,this thesis proposes a load balance solution for association the incoming users to the best available APs. The solution presented in this thesis implements an algorithm,who examines periodically the down-link load traffic of all Wifi APs and considering the capacity of wireless link,associates every incoming user to the best available. This load traffic examination is done via various statistics APs send to the SDN controller, which collects them, calculates the available throughput bandwidth of each AP can provide to the incoming user.

5.1 Future Work

A variety of scientific directions can be introduced as future work of this thesis.

First of all,the thesis can be improved by changing the stability of the scenario into a more dynamic approach. Specifically, the implemented algorithm can be changed and have functionality in a dynamic network, where new stations (cell phones) with different characteristics will come and leave during the experiment process.

Moreover, another approach it would be to take advantage of the SDN ability to manage a heterogeneous network and transform one of the two WiFi Access Points in a LTE Access Point.This transformation will make the entire thesis more realistic in the needs of daily routine.

Also, considering the significant raise of the IOT (Internet Of Things)concept,which is based on a wireless network of sensors,is critical to find an approach of managing the efficient communication between the sensors.This management can be done via SDN.So,an another scientific direction for the improvement of this thesis would be the replace of the two WiFi access points with two LoRa sensors and examine efficient mechanisms of communication between the sensors.

Last, another scientific direction for approaching the problem of load balancing in a wireless network is to combine the SDN concept with Machine Learning algorithms and Artificial Neural networks,so that better decisions will be made.

REFERENCES

[1] “Mobile data to hit 197K PB by 2019, mainly offloaded to Wi-Fi”, Available:https://www.mobileeurope.co.uk/press-wire/mobile-data-traffic-to-hit-197-000-petabytes-by-2019-mainly-offloaded-to-wi-fi

[2] Y. Bejerano, S.-J. Han, and L. Li, “Fairness and Load Balancing in Wireless LANs Using Association Control,” in Proc. ACM MobiCom’04 , Philadelphia, PA, USA, Sept. 2004, pp. 315–329.

[3] Y. Bejerano and Randeep S. B. Mifi: “A framework for fairness and qos assurance for current ieee 802.11 networks with multiple access points”.In IEEE/ACM Trans. Netw., 14(4):849–862, 2006

[4] T. Korakis, O. Ercetin, S. Krishnamurthy, L. Tassiulas, and S. Tripathi, “Link Quality based Association Mechanism in IEEE 802.11h CompliantWireless LANs,” in Proc. RAWNET’06 , Boston, MA, Apr. 2006, pp. 725–730.

[5] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki and C. Diot. “Measurement-based self organization of interfering 802.11 wireless access networks”. In Proceedings of INFOCOM, pages 1451–1459. IEEE,2007.

[6] O. Ekici and A. Yongacoglu. “A novel association algorithm for congestion relief in ieee 802.11 wlans”. In Proceedings of IWCMC, pages 725–730, New York, NY, USA, 2006. ACM.

[7] G. Athanasiou, T. Korakis, O. Ercetin, and L. Tassiulas. “Dynamic crosslayer association in 802.11-based mesh networks”. In Proceedings of INFOCOM , pages 2090–2098. IEEE, 2007.

[8] K. Sundaresan and K. Papagiannaki. “The need for cross-layer information in access point selection algorithms”. In Proceedings of SIGCOMM, pages 257–262, New York, NY, USA, 2006. ACM.

[9] M. Abusubaih and A. Wolisz. “An optimal station association policy for multirate ieee 802.11 wireless lans”. In Proceedings of MSWiM , pages 117–123, New York, NY, USA, 2007. ACM.

[10] Heeyoung Lee , Seongkwan Kim , Okhwan Lee , Sunghyun Choi , Sung-Ju Lee“Available Bandwidth-Based Association in IEEE 802.11 Wireless LANs”. IN Proceeding School of Electrical Engineering & INMC, Seoul National University, Seoul, 151-744, Korea and Multimedia Communications & Networking Lab, Hewlett-Packard Laboratories, Palo Alto, CA 94304.

[11] M. Abusubaih and A. Wolisz. Interference-aware decentralized access point selection policy for multi-rate ieee 802.11 wireless lans.In Proceedings of PIMRC, pages 1–6. IEEE, 2008

[12]K.-K. Yap, et al., “OpenRoads: Empowering Research in Mobile Networks”, ACM SIGCOMM Computer Communication, Vol. 40 Issue 1, Jan. 2010.

[13] J. Schulz-Zander, et al., “OpenSDWN: programmatic control over home and enterprise WiFi”, ACM SIGCOMM Symposium on Software Defined Networking Research, Santa Clara, CA, 14-17 Mar. 2016.

[14]R. Riggio, T. Rasheed, and F. Granell, “EmPOWER: A Testbed for Network Function Virtualization Research and Experimentation”, IEEE SDN for Future Networks and Services (SDN4FNS), Trento, Italy 11-13 Nov. 2013.

[15] Stratos Keranidis ,Thanasis Korakis,Iordanis Koutsopoulos,Leandros Tassiulas “Contention and Traffic Load-aware Association in IEEE 802.11 WLANs:Algorithms and Implementation”

[16]J. Schulz-Zander, L. Suresh, N. Sarrar, and A, Feldmann, “Programmatic Orchestration of WiFi Networks,” USENIX Symposium on Networked Systems Design and Implementation (NSDI), Philadelphia, PA, USA, 19-20 June. 201

[17] Das, S., Parulkar, G., McKeown, N.: Unifying packet and circuit switched networks. In: GLOBE-COM Workshops, 2009 IEEE. pp. 1–6. Department of Electrical Engineering, Stanford University (December 2009)

[18] G. Ganger, J. Wilkes, W. USENIX Association, G. ACM Special Interest Group in Operating Systems., and A. S. ACM Digital Library.,Proceedings of the 9th USENIX Conference on File and Stroage Technologies. USENIX Association, 2011.

[19]“Implementing a traffic engineering service in SDN”, Athanassios Xirofotos ,Department of Electrical and Computer Engineering,Volos Thessaly 2017

[20]T. A. H. B. G. P. L. P. N. McKeown, "Openflow: Enabling innovation in campus networks," in ACM SIGCOMM Computer Communications Review , New York, 2008.

[22]ONF, “OpenFlow Switch Specification Version: 1.2.0,” Current, vol. 0. pp. 1–312, 2011

[23]“Hp 3800 switch series (2011),” 2011. [Online]. Available: http://h17007.www1.hp.com/us/en/networking/products/ switches.

[24] “Pica8 (2011),” 2011 .[Online]. Available: http://www.pica8.com/open-switching/open-switching-overview.php

[25]“Watanabe, H.: Nec programmableflow-univerge pf5820. Tech. rep., NEC,” 2012.

[26] “Yiakoumis, Y.: Pantou : Openflow 1.0 for openwrt,” 2011. [Online]. Available: http://www.openflow.org/wk/index.php/OpenFlow_1.0_for_OpeWRT

[27]“Open vSwitch: Production quality, multilayer open virtual switch,” 2013. [Online]. Available: http://openvswitch.org/ features/.

[28]“McCauley,M.: About NOX,” 2012. [Online]. Available: http://www.noxrepo.org/nox/about-nox/.

[29]“McCauley, M.: About POX,” 2012. [Online]. Available: http://www.noxrepo.org/pox/about-pox/.

[30]“Floodlight Is an Open SDN Controller,” 2013. [Online]. Available: http://www.projectfloodlight.org.

[31] “The Open Network Operating System (ONOS) is a software defined networking (SDN) OS,” 2014. [Online]. Available: http://onosproject.org/.

[32] “Shimonishi, H.: Trema : Full-stack openflow framework in ruby and c,” 2009.

[33] Z. K. Khattak, M. Awais, and A. Iqbal, “Performance evaluation of OpenDaylight SDN controller,” in Proceedings of the International Conference on Parallel and Distributed Systems-ICPADS, 2014, vol. 2015–April, pp. 671–676.

[34]“Component-based software defined networking framework”,Build SDN Agilely Available: https://osrg.github.io/ryu/

[35] “Cisco Application Policy Infrastructure Controller (APIC)” ,Available: https://www.cisco.com/c/en/us/products/cloud-systems-management/application-policy-infrastructure-controller-apic/index.html

[36] “HP Virtual Application Networks (VAN) SDN Controller” ,Available: https://www.sdxcentral.com/products/hp-virtual-application-networks-van-sdn-controller/

[37] “VMware NSX Network Virtualization and Security Platform” , Available: https://www.vmware.com/uk/products/nsx.html

[38]”SOFTWARE DEFINED LOAD BALANCING OVER AN OPENFLOW-ENABLED NETWORK” by DEEPAK VERMA University of Texas at Arlington May 2017

[39]J. P. T. K. E. J. J. A. Z. J. R. J. G. A. W. J. S. P. S. K. A. M. C. B. Pfaff, "The Designand Implementation of Open vSwitch," inUSENIX/ACM Symposium on NetworkedSystems Design and Implementation, 2015.

[40]"iPerf-The ultimate speed test tool for TCP, UDP and SCTP," [Online]. Available:https://iperf.fr/.

[41] “Invoking iperf3 — iperf3 3.2 documentation - ESnet Software”,Available: http://software.es.net/iperf/invoking.html

[43] CSS,Available: https://en.wikipedia.org/wiki/CSS

[44] JavaScript,Available: https://en.wikipedia.org/wiki/JavaScript

[45] Cisco DevNet NeXt UI toolkit, Available: https://developer.cisco.com/site/neXt/

[46] The open source JavaScript graphing library that powers Plotly,
Available: https://plot.ly/javascript/

[47]A. Raschellà, F. Bouhafs, M. Seyedebrahimi, M. Mackay, Q. Shi, "A Centralized Framework for Smart Access Point Selection based on the Fittingness Factor", International Conference on Telecommunications (ICT), Thessaloniki, Greece, 16-18 May, 2016.

[48]Alessandro Raschellà, Faycal Bouhafs, Mirghiasaldin Seyedebrahimi, Michael Mackay, Qi Shi, "Quality of Service Oriented Access Point Selection Framework for Large WiFi Networks"

[49] K. Sood, S. Liu, S. Yu, Y. Xiang, "Dynamic Access Point Association Using Software Defined Networking", International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 18-20 Nov. 2015.

[50]J. Lee, M. Uddin, J. Tourrilhes, S. Sen, S. Banerjee, M. Arndt, K.-H.Kim, and T. Nadeem, "mesdn: Mobile extension of sdn," in Proceedings of the Fifth International Workshop on Mobile Cloud Computing &#38; Services, ser. MCS '14. New York, NY, USA: ACM, 2014, pp. 7–14.[Online]. Available: http://doi.acm.org/10.1145/2609908.2609948

[51] Nádia Pires Gonçalves ,"A Testbed for research and development of SDN applications
using OpenFlow."