A report

on

A Software Defined Network based Controller for Management of Large Scale WLAN

Submitted in partial fulfillment of the requirements
of the degree of

Master of Technology
in *Communication Engineering*

by

Aniruddh Rao K
Roll No. 133079005

Supervisor:
Prof. Abhay Karandikar

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

June, 2016

To Amma & Appa

Dissertation Approval

This dissertation entitled "A Software Defined Network based Controller for management of Large Scale WLAN" by Aniruddh Rao K (Roll No. 133079005) is approved for the degree of Master of Technology, Communication Engineering, Department of Electrical Engineering, IIT Bombay.

Examiners:

[illegible]

[illegible]

Supervisor:

[illegible]

Chairman:

[illegible]

Date: 24 - 6 - 2016

Place: IIT Bombay, Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this thesis.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

[illegible]

Date: 24-6-2016

Aniruddh Rao K

(Roll No. 133079005)

Acknowledgements

I would like to express sincere gratitude to my supervisor, Prof. Abhay Karandikar, for the invaluable guidance, opportunities and infrastructure support provided by him over the last three years.
I would like to thank all my project team mates Mahak, Navneet, Mayank, Akshatha, Nishanth, Ojas, Priyanka, Rohan and Abhishek for their valuable contribution in this project.
I would also like to thank all my fellow Information Networks Lab members, fellow Wireless Networks lab members for their valuable inputs and support during different phases of this project.
I would also like to specially thank all my fellow PC lab members for being so helpful and making lab such a wonderful place to work.

June 2016 **Aniruddh Rao K**

Abstract

Recent growth in demand for Internet data and mobile communications calls for innovation in communication technology to increase data rates, improve coverage and use existing technologies effectively. Deploying Wireless LAN (WLAN) on a large scale is one of the solutions which telecom operators are keen to implement to cater to the increasing demand for high speed data services. Management of large scale deployment of WLAN is a challenge which can be best addressed by centralized management. Standards like CAPWAP and TR-069 provide a handle to centralized control and management of WLAN.

Large scale deployment of WLAN would need Access Points (APs) of different vendors to be managed by a single controller but the existing standards have interoperability issues. The existing standards also have mutiple shortcomings in managing the WLAN effectively. A protocol has been proposed to address these issues and manage WLAN efficiently. It has been taken further to be standardized with TSDSI, Telecom SDO, India.

Also the existing WLAN controllers have very less ability to change network policies and configurations on the go. An emerging technology in the field of networking called Software Defined Networking (SDN) makes modifying the network policies and configurations easier and uniform. It also gives view of entire network and assists in efficient management of the WLAN. An SDN based architecture for dense WLAN controller is proposed in this thesis which would enable dynamic management of the network. The architecture is generic in addressing the management of multiple WLAN technologies. Also as part of this thesis, a SDN based WLAN controller testbed is implemented according to the architecture proposed. The testbed has been setup to manage IEEE 802.11 WLAN deployment.

Contents

List of Figures

Abbreviations

ACS	Automatic Configuration Server
AC	Access Controller
AP	Access Point
CAPWAP	Control And Provisioning of Wireless Access Points
CPE	Customer Premise Equipment
CWMP	CPE WAN Management Protocol
DTLS	Datagram Transport Layer Security
FSM	Finite State Machine
IETF	Internet Engineering Task Force
ISP	Internet Service Provider
MAC	Medium Access Control
ONF	Open Networking Foundation
QOS	Quality Of Service
RFC	Request For Comment
SDN	Software Defined Networking
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
STA	Station
TLS	Trnasport Layer Security
TR069	Technical report 069
TSDSI	Telecommunications Standards Development Society in India
TSP	Telecom Service Provider
UCI	Unified Configuration Interface
UHF	Ultra High Frequency
URL	Uniform Resource Locator

VOIP	Voice Over Internet Protocol
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WTP	Wireless Termination Point

Chapter 1

Introduction

The Government of India has started an initiative called **Digital India** [1] in July 2015 to increase the Internet connectivity and online infrastructure in India. The initiative also includes connecting rural India with high speed networks. In areas with irregular landscapes and sparse population, rolling out optical fiber cable to provide a wired broadband would be difficult and expensive. An alternative to this, as the recent research in the TV UHF band communication suggests, would be deploying TV UHF band wireless network in the backhaul and WiFi Access Points (APs) as access network. It is demonstrated by the testbed at Palghar district, Maharastra by IIT Bombay that wireless broadband can be provided by deploying WiFi access points as access network and using TV UHF to backhaul. WiFi is being deployed on a large scale at public places like bus stands, railway stations, airport etc to cater to increasing demand for Internet connectivity in the urban areas. Google is working to provide WiFi connectivity at 500 railway stations across India. It has already deployed WiFi access points at a few stations and Internet connectivity is provided. A plan has been proposed by the Delhi Government to deploy WiFi through out the capital of India.

Large scale deployment of WiFi is one of the major steps that would be taken to increase Internet connectivity across India. As a consequence, the problem of managing such large deployments arises and it needs to be addressed.

1.1 Central management of large scale WLAN

In a large scale deployment of Wireless LAN (WLAN), thousands of APs are deployed at a site. Configuring each AP individually is a tedious task. Loading each AP individually with new configurations or firmware is time-consuming and may bring in inconsistency due to human error. In order to improve the WLAN performance i.e reduce interference, do load balancing, etc., there is a need for inter AP communication or access to a continuously updated information database about current state of the WLAN. A centralized management scheme addresses these needs. APs communicate to a central entity called controller which would manage the entire WLAN deployment. The controller also manages client connections in the WLAN deployment. To make the WLAN management efficient, configuration of the network has to be updated dynamically.

1.2 SDN paradigm for WLAN management

Software Defined Networking (SDN) is a paradigm that reorganizes the network architecture by separating the forwarding plane and control plane. SDN enables programming network dynamically. It removes computational load from the end routers and places it in a centralized controller. This simplifies the end devices and reduces the network infrastructure cost. SDN also gives the network administrators a complete view of the network and helps in making better routing decisions.
Open Networking Foundation (ONF) defines the SDN architecture [2] as

1. Directly programmable: Network control is directly programmable because it is decoupled from forwarding functions.
2. Agile: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
3. Centrally managed: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
4. Programmatically configured: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN

programs, which they can write themselves because the programs do not depend on proprietary software.

5. Open standards-based and vendor-neutral: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

SDN paradigm is widely used in wired networks to manage flows and routing. In a SDN enabled wired network, the controller uses OpenFlow [3] protocol and updates the flows and routes, thus, removing the computational load on switches. Bringing in SDN to centrally manage a large scale WLAN would make management of WLAN simpler and efficient. It would remove processing load from end access points, making the access points light low cost radio devices and also would enable dynamic control of the flow of traffic.

In summary, it would give all the advantages that a SDN gives in a wired network and in addition enables easier management of WLAN by a centralized controller.

1.3 Organization of this thesis

In this thesis, a detailed study on centralized management of WLAN deployment is presented. Based on the study of existing work in SDN for WLAN management, an architecture for SDN based WLAN controller is proposed. An implementation of testbed for SDN based WLAN management is also presented. In chapter 2, shortcomings of the existing standards CAPWAP [4] and TR069 [5] in managing WLAN are listed and a new extensible protocol for centralized management of WLAN is proposed. In chapter 3, moving towards SDN paradigm, a study of various architectures and implementations of SDN controller for WLAN like OpenRoads [6] (OpenFlow Wireless), Odin [7] and ethanol [8] is presented. Also a SDN controller architecture is proposed for management of large scale WLAN deployment. In chapter 4, an implementation of the testbed of SDN based WLAN controller for IEEE 802.11 WLAN is presented. Configuration and initial setup of the testbed is also explained.

Chapter 2

Central WLAN Management

2.1 Existing standards

There are standard protocols for centralized management of WLAN. Control And Provisioning of Wireless Access Points (CAPWAP) is one such standard protocol defined under IETF. Technical Report 069 (TR069) is another standard. It is a technical specification published by broadband forum and is entitled CPE WAN Management Protocol (CWMP).

2.1.1 CAPWAP

CAPWAP Protocol, is a standard [4] interoperable protocol that enables an Access Controller (AC) to manage a collection of Wireless Terminal Points (WTPs). CAPWAP protocol is defined to be independent of Layer 2 (L2) technology. Traditional protocols for managing WTPs are either manual static configuration via HTTP or proprietary L2 specific or non-existent. CAPWAP assumes a network configuration consisting of multiple WTPs communicating via the IP to an AC. The CAPWAP protocol transport layer carries two types of payload: CAPWAP data messages and CAPWAP control messages. CAPWAP data messages encapsulate forwarded wireless frames. CAPWAP control messages are management messages exchanged between a WTP and AC. RFC for the CAPWAP protocol states the following goals.

1. Centralize the authentication and policy enforcement functions for a wireless network.

2. Remove processing load from WTP by moving it away towards AC except for time critical functions.
3. Provide extensible protocol not bound to a specific technology.

As the CAPWAP protocol is not bound to a specific wireless technology, bindings have been written to support the use of CAPWAP protocol with IEEE802.11 WLANs [9]. Bindings specify the implementation of CAPWAP for a specific wireless technology. CAPWAP protocol supports two modes of operation: split MAC and local MAC. In split MAC mode all the wireless data and management frames are encapsulated via the CAPWAP protocol and exchanged between AC and WTP along with CAPWAP control messages. While in local MAC mode, the data frames are either locally bridged or tunneled as 802.3 frames and management frames are processed by the WTP and then forwarded to the AC.

2.1.2 TR069/CWMP

TR069 [5] is the specification document published by the Broadband Forum that specifies the CPE WAN Management Protocol (CWMP). CWMP is intended for communication between a CPE and Auto-Configuration Server (ACS). CWMP defines a mechanism that encompasses secure auto-configuration of CPE, and also incorporates other CPE management functions into a common framework. It supports a variety of functionalities to manage a collection of CPE which includes:

1. Auto-configuration of CPE and dynamic service provisioning
2. Software/firmware image management of CPE
3. Status and performance monitoring
4. Diagnostics

TR069 defines a SOAP/HTTP based protocol that provides communication between CPE and ACS with typical positioning of ACS and CPE as show in Fig. 2.1. The protocol vastly addresses the issue of automatic configuration and management of different internet access devices like modems, routers, gateways etc by a remote server i.e. ACS.

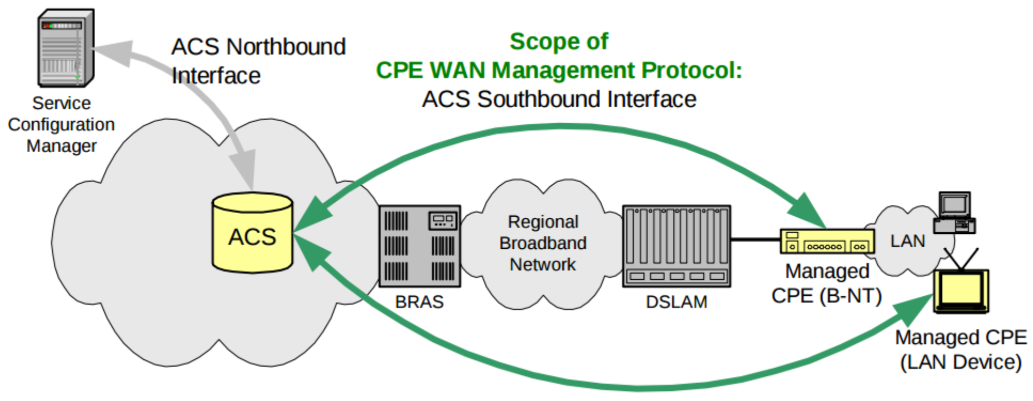

Figure 2.1: Positioning of components in Auto configuration Architecture using CWMP to configure CPE [5]

2.2 Limitations of existing standards

The exisitng standards like CAPWAP and TR069 employ the use of some sort of Controller architecture which helps in managing a large network of Wireless APs. While these standards do spell out a lot of grammar for addressing the management of such networks, there are a few essential gaps which these standards fail to address. The following section enumerates those in the context of CAPWAP and TR069.

2.2.1 Client connection management

The controller should be able to manage client connections to WTP. WTP forwards association and dissociation messages it received from client to controller and the controller responds accordingly. The controller should also be able to send dissociation message exclusively to the WTP, to dissociate a client connected to it.

In CAPWAP, there are two messages defined - Add Station and Delete Station for association and dissociation of clients respectively. However, there is no message defined for authentication of the clients. In TR069, no method is defined for client association or dissociation. Generic method AddObject can be used to manage client connections. However, in case of WTP with multiple radios or broadcasting multiple SSIDs, each radio would be managed by using AddObject, DeleteObject methods as there are no specific methods available to handle multiple radios or WLAN in a WTP. In such a scenario, client connection to one of these WLANs or radios would be managed by creating child objects which is not defined in the scope of TR069.

2.2.2 Roaming

To enable roaming within a WLAN, the controller should be able to dissociate client from a AP or WTP and associate it to another AP or WTP. This process of reassociation should occur with least delay.

In CAPWAP, re-association follows either of these procedures:

1. A full association procedure with all the authentication steps, security and policy message exchanges (to the new WTP) followed by a dissociation procedure (to the old WTP).
2. Association of an STA to a new WTP is assisted by access controller which caches the authentication keys avoiding the 802.1X authentication step and would reduce handover time.

There is no support for IEEE 802.11r in CAPWAP which would allow pre-authentication and facilitate faster handover. Pre-authentication is a procedure where a cleint/STA authenticates itself with all the near by WTPs. TR069 does not have any methods specified for re-association of a client in a WLAN.

2.2.3 Security

The advantages of introducing a controller in the existing architecture should not come at an expense of making the system any less secure. The controller should not introduce vulnerabilities or make the system prone to attacks from malicious elements. The overall security of the system should not be compromised in even the slightest manner.

Section 8 of the IETF, RFC 5418 [10] lists out the vulnerabilities introduced by CAPWAP from a security standpoint. Also, as per Section 4.4 of the same RFC, the various security associations that come into play in establishing a session with a client device and the hierarchical architecture of CAPWAP do not preclude it from introducing vulnerabilities into the network. As defined TR069, a CPE is allowed to discover the ACS either by

1. Locally configuring the CPE with the Uniform Resource Locator (URL) of ACS and usage of Domain Name System (DNS) to resolve the hostname to get to ACS.
2. Use of a Dynamic Host Configuration Protocol (DHCP) server that serves out the URL of the ACS to the CPE.
3. Use of a default URL if not other URL is provided to CPE.

If any adversary manages to modify this URL he gains access to the entire network of devices that was being managed by the ACS.

2.2.4 Connection between controller and access point

A connection is setup between the controller and the access point for communication of various WLAN management messages and exchange of statistics. This should be a reliable connection and a secured one. CAPWAP sets up a UDP based DTLS connection between controller and AP. DTLS is a secured connection used for purposes like streaming live video or Voice Over IP (VOIP) calls. Using DTLS over TLS is a compromise on reliability to achieve faster transmission of data. On local break out of user data at AP, only management messages are forwarded to the controller which do not include video streaming or VOIP data packets. Exchange of management messages needs to be done reliably for consistency in configuration. The use of DTLS over TLS is subject to question as TLS gives more reliability. TR069 uses TLS to setup connection between controller and CPE, but does not mandate TLS. Even a bare TCP connection can be used. But it compromises on security.

2.2.5 Inter controller communication

For managing a large deployment, we may need more than one controller. A hierarchical set of controllers would be a good approach. All real time activities are managed by near local controller and global activities like authentication are done by a global controller. In such a scenario, there is a need for inter controller communication. Neither CAPWAP nor TR069 specify anything related to inter controller communication.

2.3 Addressing the limitations of existing standards

There is a need to address the limitations in the existing standards for efficient management of WLAN. Either a new protocol should be defined or one of the existing standards needs to be extended. Use cases that are necessary for management of large scale WLAN deployment but not addressed by the existing standards are listed below. Some typical scenarios wherein a large number of APs are deployed are taken into consideration and use cases are listed out. Along with addressing limitations in the existing standard, requirements spelled out by the use cases listed should also be addressed.

2.3.1 Use case i: Interoperability

An ISP has WLAN deployments in various residential apartments and office buildings which are spread across a township. The WLAN provides connectivity to the residents/employees both within and outdoors. A new office building has recently been constructed and needs similar WLAN deployment. The ISP is looking to deploy the same network equipment in the new office but cannot find the same network equipment that is being used in the earlier deployments as those models are no longer being manufactured by the vendor. The ISP invites a quote from a different vendor and ends up installing its equipment in the new location. The ISP would like to lay the network such that the new WLAN deployment is also managed by the controller that is used to manage all other deployments in the township. This use case highlights the need for the interoperability issue to be addressed by the protocol. The protocol developed must enable interoperability i.e. same controller must be able to manage different vendor APs as shown in Fig. 2.2.

2.3.2 Use case ii: Multiple WLAN networks on a physical AP

Community WiFi networks allow service providers to leverage unused capacity on existing WiFi infrastructure to offer WiFi network access to visitors and passersby. An operator can also use this excess capacity to offer services to retail and roaming-partner operators' subscribers. The residential subscribers accessing the network from inside their homes have prioritized access to the WiFi resources. The residential WiFi infrastructure is configured in a manner that allows for a secure and independent access channel to retain

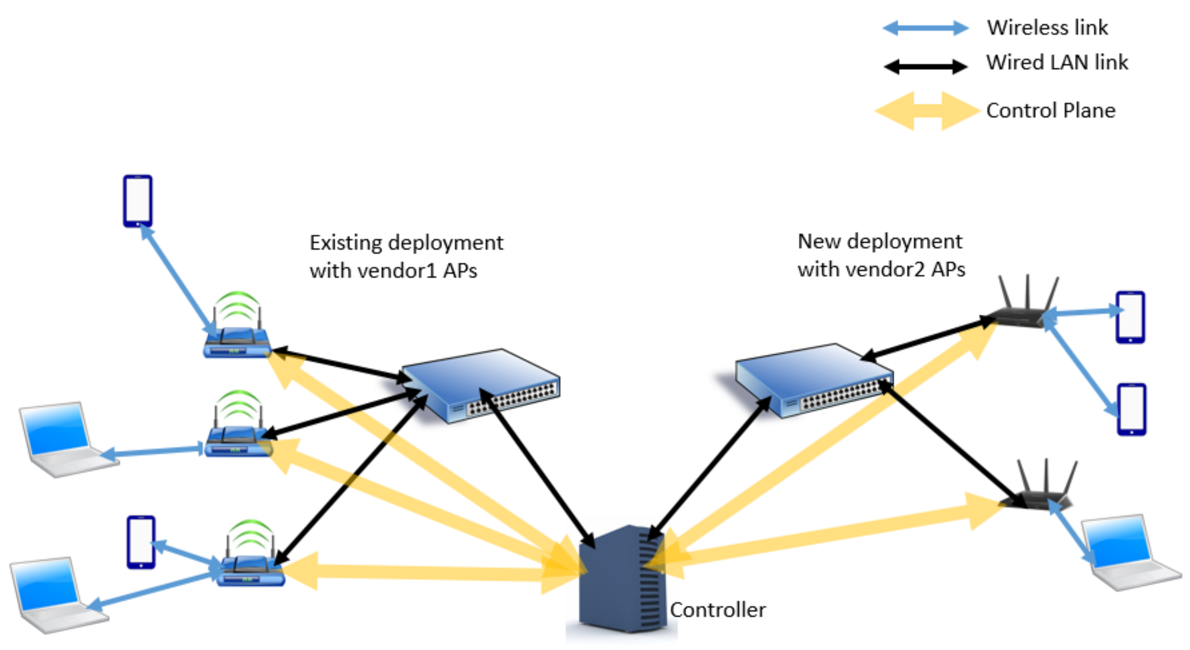
Wireless link
Wired LAN link
Control Plane
Existing deployment with vendor1 APs
New deployment with vendor2 APs
Controller

Figure 2.2: Different model/vendor APs managed by same WLAN controller

service quality, safety, and privacy for both residential and visitor customers. Roaming users are only allowed to use the WiFi network capacity that is not currently used by the subscriber at home.Basically, the wireless AP in the home will provide two networks: a private one for the home owner/subscriber, and a community network for on-the-go subscribers passing through the neighborhood. While the user is at home, all of their WiFi devices e.g. smartphone, tablet, etc., should automatically connect to the private network. When the user travels outside the vicinity of their AP's coverage area, and passes in the range of another AP operated by the same service provider, their client devices will be able to connect to the public network.

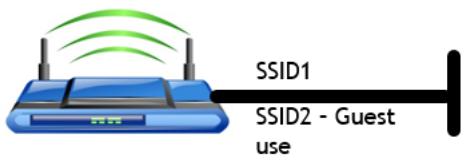

Figure 2.3: Dividing WLAN into multiple virtual networks

As shown in Fig. 2.3, this use case explains the need for the protocol to divide WLAN into multiple networks and manage QOS and access control to users.

2.3.3 Use case iii: Need for inter-controller communication

An Internet Service Provider (ISP) has dense WLAN deployments at multiple locations across a city. Assume, one deployment in bus stand, one in railway station etc. Customers can buy services from ISP and use these services at all the deployments across the city.

1. The ISP provides services to customer according to his subscription. According to the ISP, for each user, the plan features like data rate, data limit etc should be counted across the deployments in the city. Also the user should be authenticated in the same way across these deployments ,i.e., it should be the same globally.
2. The deployments in bus stand and railway station have to cover a wide area and have to cater to large population of users. Within each deployment, the ISP wants to provide seamless roaming with minimum delay due to handover.

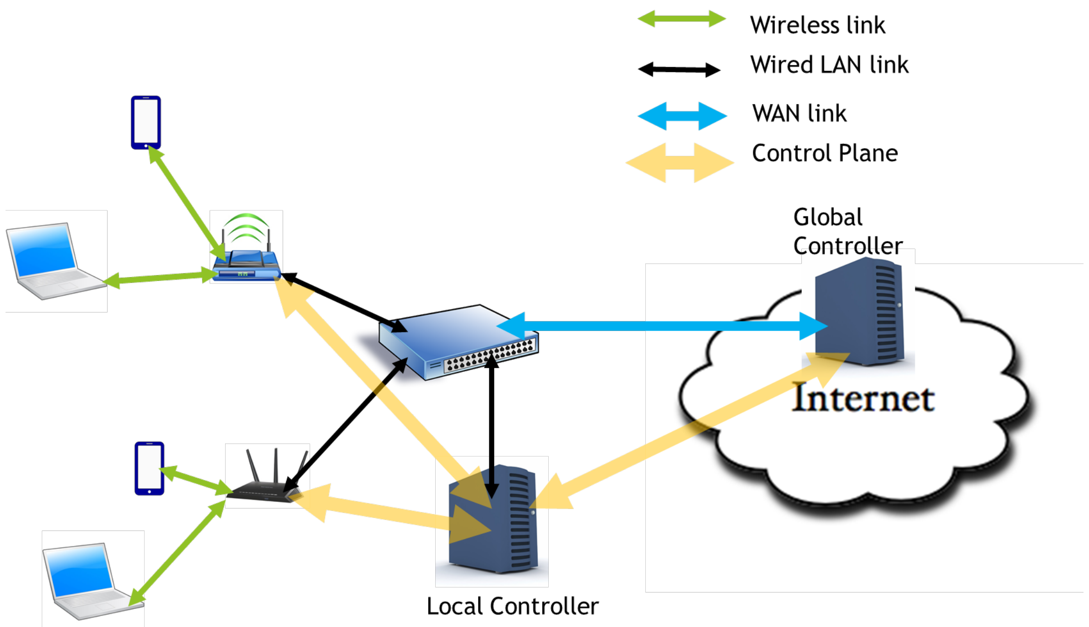

Figure 2.4: Positioning of controllers in hierarchical setup

A hierarchical architecture with one global controller and multiple nearby local controllers per deployment would be a good solution for the above requirements. Global controller can be used for tasks like user authentication, fetching user plan details, configuring user service quality like data rate limiting. While local controller can be used for delay sensitive applications like roaming, load balancing etc. There is a need for inter controller communication between local and global controllers in such a hierarchical architecture. One such deployment is shown in Fig. 2.4. We propose to write extensions to one of the already existing standards to address the limitations of existing standards and requirements spelled out by the use cases listed. TR069 is a remote device configuration protocol while CAPWAP also supports WLAN management in addition to configuring devices remotely. Therefore, we choose to extend CAPWAP to address the existing issues and requirements of the use cases listed.

Chapter 3

SDN based Controller architecture for WLAN management

3.1 SDN paradigm for WLAN management

Most of the enterprise WLAN management solutions are closed and proprietary. The existing standards for WLAN management are not updated with recent feature additions and technology amendments; leading to their implementation by manufacturers in a non standard manner. Thus, using different vendor equipments in a centrally managed WLAN has become almost impossible. This issue of interoperability needs to be addressed. A single controller must be able to manage different vendor APs in a standard manner. The existing standards for management of WLAN mainly address radio configuration and client connection. For addressing access control and QoS management, the controller must have the ability to dynamically configure network. SDN paradigm to manage of WLAN would address these issues. SDN is a network paradigm that relies on the separation of the control and forwarding planes in IP networks. A central entity called controller manages the data flow in the network. According to the definition [2] by ONF, SDN would make WLAN management open standards based, vendor independent and dynamically programmable. SDN in WLAN management would give the following advantages:

- **Diversify the supply chain:** Operating at a level of abstraction, SDN provides standard interfaces for communication. Implementation below the abstraction need not follow standard and can be proprietary. This would also encourage more number of vendors to provide only a standard interface as part of their product.

- **Empower network owners and increase the pace of innovation:** SDN would enable dynamic configuration of the network. Controller can update network policies on the go. SDN would also allow network owners to slice the network into production setup and test setup. The test setup slice could be used to experiment and test any new algorithms without affecting the production setup.

3.2 Study of existing work using SDN in WLAN

A lot of research has been going on in wireless SDN. OpenRoads [6] was a first step in using SDN paradigm to manage wireless networks. Openroads was an attempt to bring in openflow to manage wireless networks. Openroads architecture has three layers.

- Flow layer: In this layer, data flow in the network is managed by OpenFlow [3] and SNMP
- Slicing layer: This layer slices the data path and SNMP configurations for different access points in the network.
- Control layer: Applications at controller decide on how data should flow and configure access points.

Openroads proposes how applications at a central controller can address mobility, load balancing, etc., in a wireless network. Ethanol [8] proposes an architecture where the controller talks to the switches using OpenFlow and to the APs using XML/HTTP based protocol. Ethanol demonstrates features like load balancing, QoS management as applications. Another prototype, Odin [7] proposes a SDN based enterprise WLAN management scheme. The objective of odin is to empower network owners to program and provide WLAN services and features as network applications. Odin controller has a master that speaks to the switches and APs using OpenFlow protocol. Odin addresses mobility efficiently, by creating a light virtual access point per client connected in the network.

3.3 Hierarchical architecture for SDN based controller

Based on the study we propose a generic hierarchical SDN based controller for management of WLAN. Fig. 3.1 shows the typical positioning of controllers in a hierarchical setup. It shows a two level hierarchical setup with a local controller located in the WLAN deployment and a global controller located in the cloud. There can be more than two levels of hierarchy and there can be multiple local controllers within a WLAN deployment. Local controller would manage time critical features like mobility management, localized features like load balancing, etc. Global controller would manage features like global policy management, QoS management, authentication, etc.

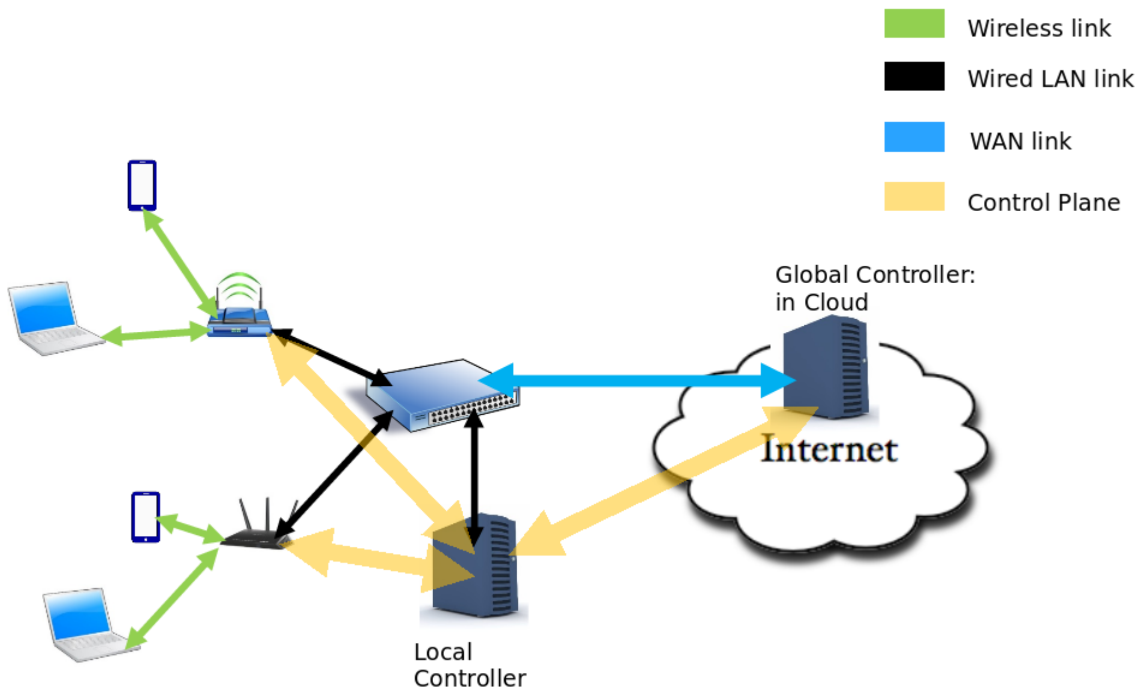

Figure 3.1: Two level hierarchical setup of controllers: A global controller and a local controller

Proposed architecture for the SDN based WLAN management is shown in Fig. 3.2. The controller consists of a *OpenFlow controller module*, a *WLAN controller module*, *North bound interface* and an *API for inter controller communication.* The *OpenFlow controller module* manages OpenFlow switches using standard OpenFlow protocol. The *WLAN controller module* manages wireless APs in the deployment. The protocol for communication between the *WLAN controller module* and wireless APs is independent of wireless technology. Bindings need to be written to support a specific wireless technology. In a network with TV UHF back haul and WLAN deployed as access network, the same protocol can be used to manage TV UHF devices and also WLAN APs. This enables same controller to manage devices of different wireless technology. Wireiless APs must have a WLAN agent that implements the protocol for communication with *WLAN controller module*, parse the messages and take acctions accordingly. Applications implementing features like mobility management, load balancing, etc., are written on top of the controller. *Application plane* communicates with the controller through *north bound interface*. Also a controller can communicate with a peer or with another controller at higher level or lower level in the hierarchy through the *API for inter controller controller communication*. This is needed in a hierarchical setup with multiple local controllers and a global controller.

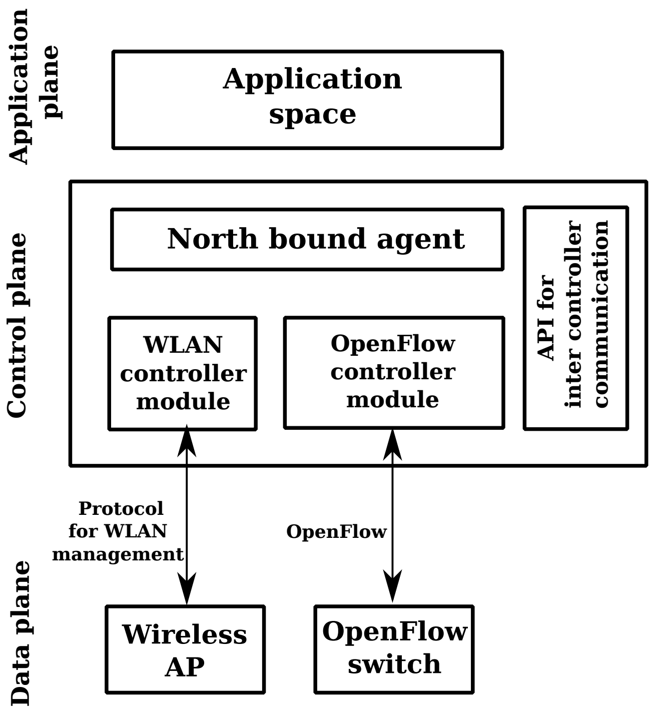

Figure 3.2: Architecture for SDN based WLAN management

Chapter 4

Test bed Implementation: SDN controller for management of WLAN

According to the architecture proposed in chapter 3 for SDN based WLAN management, a testbed is setup to manage the deployment of IEEE 802.11 WLAN. The current test bed contains the implementation of

- Controller on Linux x86 Machine
  - WLAN controller module
  - SDN controller module
    - switch configuration submodule
    - flow controller submodule
  - o North bound interface
- AP on OpenWrt
  - WLAN agent
  - Virtual Switch
    - switch configuration agent
    - flow table update agent

In future, applications would be written on top of the controller modules to implement various WLAN management functionality like load balancing, roaming, etc. The layout of testbed deployment is shown in Fig. 4.1. It has three network entities: Controller, OpenFlow switches and APs.

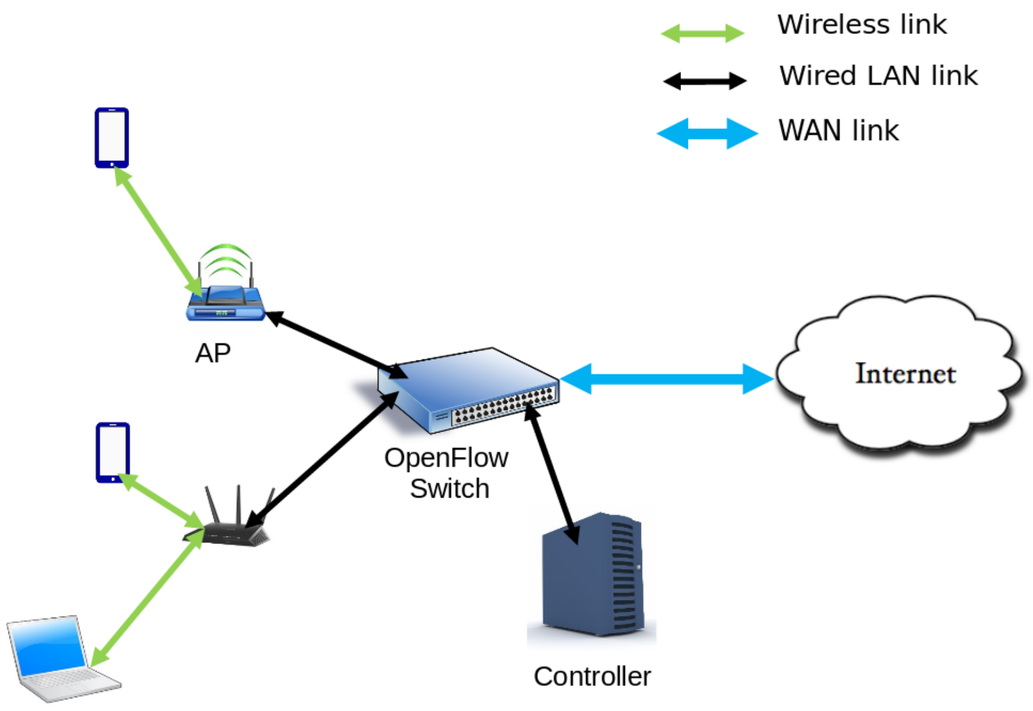

Figure 4.1: Layout of Testbed deployment

4.1 Architecture of the testbed

Detailed architecture of the implementation of the testbed is shown in Fig. 4.2. It shows various modules in the controller, modules in the AP and multiple connections between them. In controller, the *WLAN controller module* is implemented as a user space program in a linux x86 server. *SDN controller module* consists of two submodules: A *switch configuration submodule* which sets up the network interfaces and a *OpenFlow controller submodule* which updates the flow tables. The *OpenFlow controller submodule* is implemented using FloodLight [12] and the *switch configuration submodule* is implemented using OpenVswitch [14](OVS) client. The *WLAN controller module* communicates with wireless AP while the *SDN controller module* communicates with two network entities: OpenFlow switch and wireless AP. Standard OpenFlow switch available in the market is used in the testbed without any modification. Wireless AP is either mikrotik router board [15] or any other standard wireless router with OpenWrt ported on it. OpenWrt [16] is a linux based opensource OS for wireless routers. It supports wireless routers of different make and various other embedded devices. *WLAN agent* and *Virtual switch* are implemented on OpenWrt. *WLAN agent* is implemented as a user space program which configures radio and forwards wireless management frames to controller. *Virtual* *switch* is implemented using OVS. OVS has three main components: OVSDB server, ovswitchd and OVS kernel module which together implement *switch configuration agent* and *flow table update agent.* The *switch configuration submodule* in the controller communicates with OVSDB server to configure the network interfaces. The *flow controller submodule* in the controller communicates with ovswitchd using the OpenFlow protocol to update the flow tables.

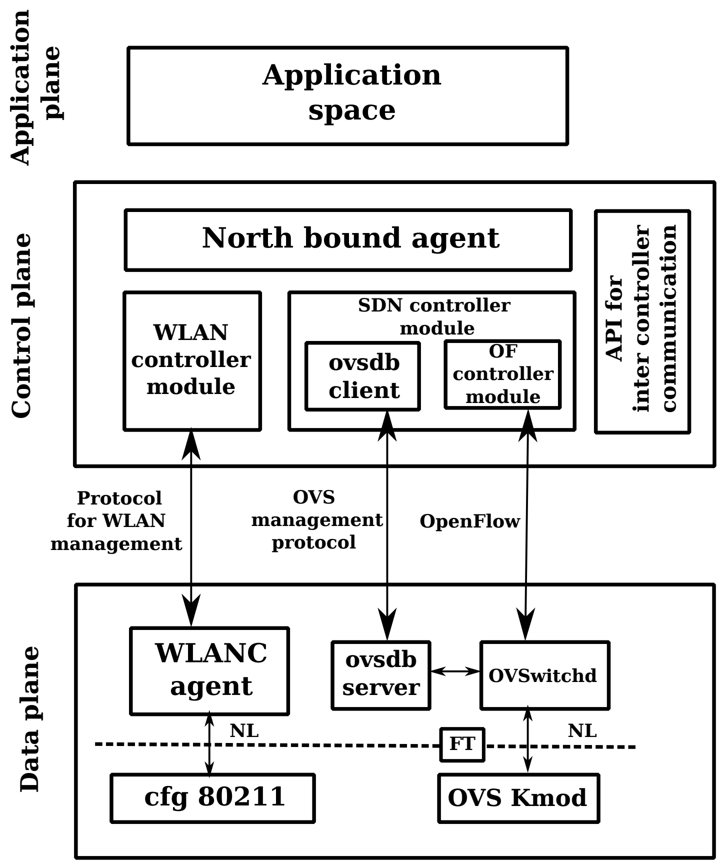

Figure 4.2: Architecture of the testbed Implementation for SDN based WLAN management

4.1.1 WLAN Controller module and WLAN agent

A CAPWAP based WLAN controller module and a WLAN agent on AP is implemented by a team of 6 of us. Our implementation is based on a opensource implementation of CAPWAP called OpenCAPWAP [17]. CAPWAP based controller is implemented as a user space program in linux written in C language. WLAN agent is implemented on AP, as a user space program in OpenWrt. The idea behind porting OpenWrt on to

APs is to manage APs of of different vendors. OpenWrt supports wireless routers of different make and various other embedded devices. According to the terminology of CAPWAP, the WLAN controller is called as Access Controller(AC) and the AP is called as Wireless Terminal Point(WTP). This terminology would be used in the sections related to CAPWAP based controller.

4.1.2 CAPWAP engine

CAPWAP sets up a tunnel between AC and WTP for exchange of management messages and statistics. In our implementation, the engine for setting up CAPWAP tunnel follows the finite state machine as specified by CAPWAP RFC 5415 except that there is no state for firmware update. Firmware update is not addressed in this implementation. The FSM is shown in Fig. 4.3.

As specified by the RFC, both AC and WTP follow the same FSM but some

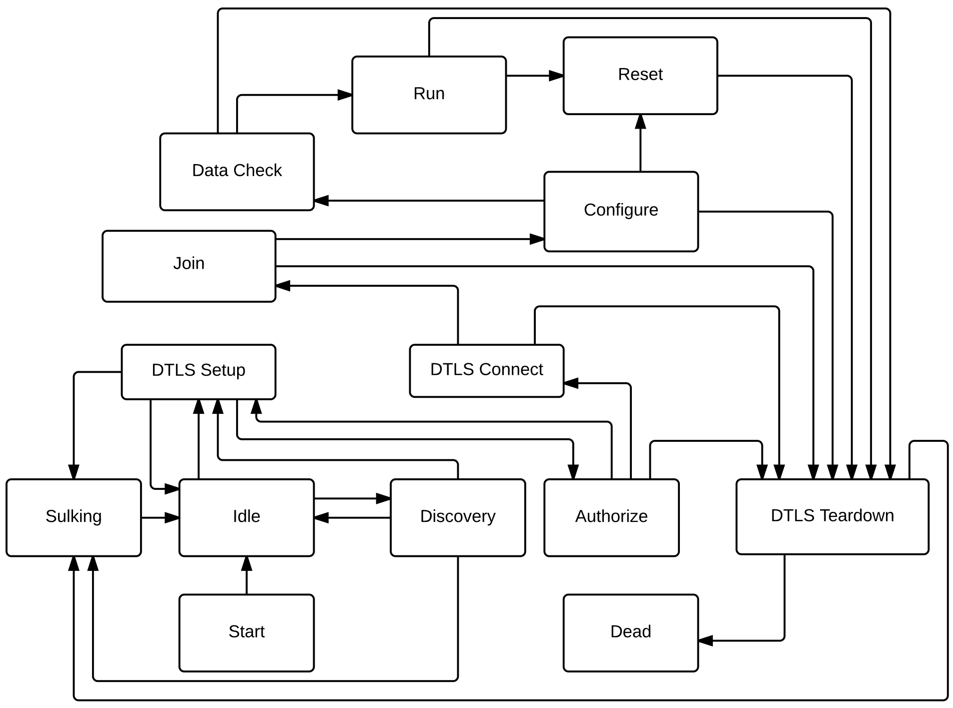

Figure 4.3: Finite State Machine of CAPWAP Protocol

transitions occur in only one of them. The state transitions are different in both AC and WTP as WTP communicates to a single AC while an AC communicates to multiple WTPs. A heart-beat message exchange is setup between AC and WTP to monitor the connectivity of WTP to AC and its availability in the WLAN. The CAPWAP protocol implemented is local MAC mode. All the data frames are locally bridged. All the 802.11 management frames except probe and beacon frames are processed and forwarded to AC in CAPWAP tunnel while all the 802.11 control frames are locally addressed at WTP. Each AC runs three threads:

1. Discovery thread: The AC's discovery thread is responsible for receiving, and responding to discovery request messages from WTP.
2. Listener thread: The AC's Listener thread handles inbound DTLS session establishment requests. Once a DTLS session has been validated, which occurs when the state machine enters the ”Authorize” state, the listener thread creates a WTP session-specific service thread and state context.
3. Service thread: AC's service thread handles the per WTP states and one such state exists per WTP connection. This thread is used for any configuration or communication between AC and WTP. When communication with the WTP is complete, the service thread is terminated and all associated resources are released.

The session starts with WTP in the START mode and after initialization it changes to IDLE mode. In IDLE mode it either connects to a fixed AC if it is explicitly instructed to connect to it or it enters the DISCOVERY phase to find an AC by sending a discovery request. After the DISCOVERY state, it proceeds to DTLS SETUP state where it establishes a secure connection between the access point and the controller. After the DTLS SETUP, it moves to AUTHORIZE state where the DTLS stack needs authorization for the session establishment. After successful authorization it enters the DTLS CONNECT state and after connection establishment, it changes to JOIN state where the WTP and AC communicate with each other and CAPWAP session begins. The AC then sends a successful join response message to the WTP. After receiving the join response message, the WTP is configured through a set of commands in the CONFIGURE state. The WTP reaches the RUN state after confirmation of successful configuration in the DATA CHECK state. From the RUN state, it may be asked to update the configuration in the CONFIGURE state or may lose its connectivity with the AC which will change the state to SULKING. After SULKING state, the WTP goes back to the DISCOVER state after temporary transition to the IDLE state. These state transitions setup a DTLS CAPWAP tunnel between AC and WTP. All configuration updates and exchange of statistics are addressed by the bindings written as applications at both AC and WTP.

IEEE802.11 bindings

Our modified CAPWAP protocol is not bound to a wireless technology and bindings have been written to support a specific wireless technology. RFC 5416 specifies bindings for 802.11 WLAN and a subset of bindings are implemented by us. We follow an architecture where the CAPWAP bindings have been written as modules on top of the controller. Even on the WTP , implementation of bindings are moved away from the main CAPWAP engine as seperate modules. The architecture is shown in Fig. 4.4. This architecture would make addition of new bindings easier as it needs changes only in the binding modules and the main CAPWAP engine would not change.

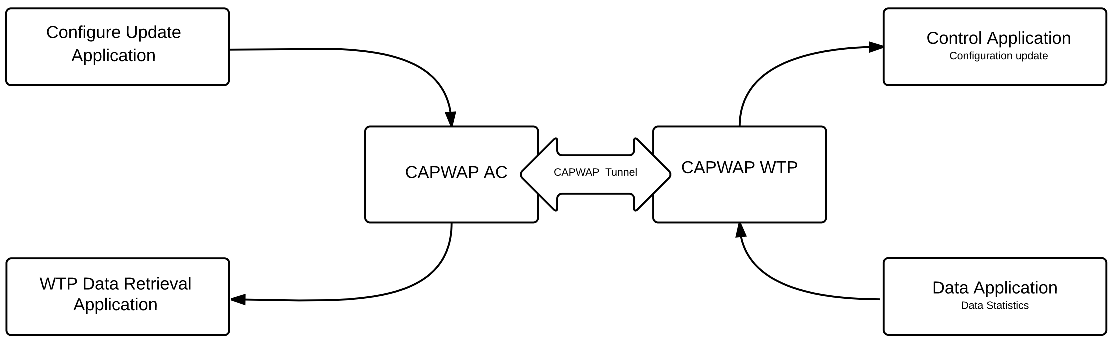

Figure 4.4: Architecture of IEEE 802.11 binding applications

Binding modules at AC

1. Configure update application: This application is used to build and send configuration messages to WTP. The application builds a message according to IEEE802.11 bindings defined in RFC 5416 and sends it to AC. AC encapsulates the message in a CAPWAP header and forwards it to WTP.
2. WTP Data retrieval application: This application will retrieve the information data from different WTPs. AC will receive information data periodically from all the connected WTPs. The data packet will contain WTP-ID to identify which WTP has sent the data. Present implementation only processes the connected Stations information from every WTP.

Binding modules at WTP

1. Control application: This application handles the configuration update requests from AC. WTP on receiving a configuration update message, forwards it to the control application. The control application parses it and updates the configuration.
2. Data application: This application sends various statistics from WTP to AC. The data application periodically fetches statistics from the radio, formats as per the bindings and forwards it to WTP main thread that forwards the data to AC.

Bindings currently implemented

- SET CHANNEL: Configuration message to set channel of operation of WTP. Binding specified in section 6.10 RFC 5416.
- SET TX POWER: Configuration message to set transmit power of WTP. Binding sepcified in section 6.18 RFC 5416.
- STATISTICS: Data sent by WTP to AC. Contains statistics of data forwarded to stations from WTP. Binding specified in section 6.16 RFC 5416

4.1.3 Connection Details: controller to AP and controller to OpenFlow Switch

There would be three connections from the controller to an AP:

1. A DTLS connection between WLAN controller module and WLAN agent in the AP.This connection is setup by the modified CAPWAP protocol for WLAN management.
2. An SSL connection from the OVSDB client in the controller to the OVSDB server in the AP. This connection is setup by OVS management protocol, to configure network interfaces.
3. An SSL connection from the OpenFlow controller module to the ovswitchd in the AP. This connection is setup by the OpenFlow protocol to update the flow tables in the AP.

The controller also communicates with another network entity OpenFlow switch. There would be only two connections from the controller to an OpenFlow switch:

1. An SSL connection from the OVSDB client in the controller to the OVSDB server in the switch. This connection is setup by OVS management protocol, to configure ports and network interfaces.
2. An SSL connection from the OpenFlow controller module to the switch daemon in the OpenFlow switch. This connection is setup by the OpenFlow protocol to update the flow tables in the switch.

The connection between the OVSDB server and the OVSDB client is not a persistent connection while the other two connections are persistent.

4.2 Configuration and setup of the test bed

FloodLight is used as the OpenFlow controller module. Curl commands are written to define flows in the controller. Two types of flows are defined: proactive flows and reactive flows. When a switch gets connected to the OpenFlow controller, all the proactive flows are updated on the switch. Reactive flows are updated in the flow tables of the switch only on the arrival of the first packet matching certain reactive flow defined in the controller. By default, packets with no matching flow in switch's flow tables are sent to the OpenFlow controller module. FloodLight controller listens for connections from switches on port 6633. Switch daemon on AP or OpenFlow switch connects to the FloodLight controller through the OpenFlow protocol.

OVS implements *switch configuration agent* and *flow update agent* in the AP. OVS has three modules:

1. OVSDB server
2. OVS-vswitchd
3. OVS kernel module

OVSDB server listens to connections from the controller on port 6640. OVSDB client on the controller connects to the OVSDB server through OVS management protocol and sends the configurations to be updated. OVSDB server updates the OVSDB with the configurations received. OVS-vswitchd communicates with the OVS-kernel module which configures the interfaces according to the changes in the OVSDB. The switch configuration includes creating bridges, tagging ports with VLAN, etc. OVSDB client also sets the IP address of the OpenFlow controller. OVS-vswitchd picks up the address from the OVSDB and connects to the OpenFlow controller through the OpenFlow protocol. This connection is a secured and persistent connection. OVS-vswitchd updates the flow table in the AP according to the updates received. There are multiple protocols available for configuration of a OpenFlow switch e.g., OFCONFIG by ONF and OVS management protocol. Any one of them can be used to configure the switch depending on the protocols supported by both the controller and the switch.

The WLAN controller module listens on port 1234 for connections from wireless APs. On startup, WLAN agent runs in AP and sends connection request to the WLAN controller. The address of the controller is provided in a configuration file in the WLAN agent. The WLAN controller module and the WLAN agent follow the modified CAPWAP protocol to setup a DTLS connection between them. This is a persistent connection. The WLAN controller module also listens on port 1235 for applications that provide binding specific messages for configuration updates and exchange of statistics.

Chapter 5

Future Work

Applications over SDN controller test bed

North bound applications to be written on top of the SDN controller to implement various algorithms of roaming, load balancing, QoS management, flow management, access control and restrictions etc.

Technical specifictaions for extensions to CAPWAP

As mentioned in chapter 2, extensions have to be written to CAPWAP to address various issues and provide new features in the existing standard. Technical report on gaps has been submitted. Technical specifications have to be written further.

Bibliography

[1] `http://www.digitalindia.gov.in/`

[2] `https://www.opennetworking.org/sdn-resources/sdn-definition`

[3] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 1 :69 - 74, March 2008.

[4] `https://tools.ietf.org/html/rfc5415`

[5] `https://www.broadband-forum.org/technical/download/TR-069.pdf`

[6] Kok-Kiong Yap, Masayoshi Kobayashi, Rob Sherwood, Te-Yuan Huang, Michael Chan, Nikhil Handigol, and Nick McKeown. OpenRoads: empowering research in mobile networks. SIGCOMM Comput. Commun. Rev. 40, 1, 125-126, 2010.

[7] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. Towards programmable enterprise WLANS with Odin. In Proceedings of the first workshop on Hot topics in software defined networks (HotSDN '12). ACM, New York, NY, USA, 115-120, 2012.

[8] Henrique Moura, Gabriel V. C. Bessa, Marco A. M. Vieira, Daniel F. Macedo. Ethanol: Software defined networking for 802.11 Wireless Networks. Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium, 2015.

[9] `https://tools.ietf.org/html/rfc5416`

[10] `https://tools.ietf.org/html/rfc5418`

[11] Nachikethas A. Jagadeesan, Bhaskar Krishnamachari. Software-Defined Networking Paradigms in Wireless Networks: A Survey. ACM Comput. Surv. 47, 2, Article 27, November 2014.

[12] `http://www.projectfloodlight.org/floodlight/`

[13] `https://www.opendaylight.org/`

[14] `http://openvswitch.org/`

[15] `http://routerboard.com/RB433AH`

[16] `http://openwrt.org/`

[17] Massimo Bernaschi, Filippo Cacace, Giulio Iannello, Massimo Vellucci, and Luca Vollero. OpenCAPWAP: An open source CAPWAP implementation for the management and configuration of WiFi hot-spots. Comput. Netw. 53, 2 (February 2009), 217-230, 2009.