DESIGN AND IMPLEMENTATION OF 802.11 CONTROL API AND EXPERIMENTAL EVALUATION OF CHANNEL SELECTION ALGORITHMS

BY PAVAN KULKARNI

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Dipankar Raychaudhuri

and approved by

_____

_____

_____

New Brunswick, New Jersey

October, 2016

ABSTRACT OF THE THESIS

DESIGN AND IMPLEMENTATION OF 802.11 CONTROL API AND EXPERIMENTAL EVALUATION OF CHANNEL SELECTION ALGORITHMS

by Pavan Kulkarni

Thesis Director: Professor Dipankar Raychaudhuri

This thesis presents design and implementation of an HTTP based control API on top of the 802.11 software access point(hostapd) along with experimental evaluation of channel coordination algorithms using the framework developed. The hostapd_cli, a front end program to interact with hostapd is modified to control the channel via HTTP interface through CSA(Channel Switch Announcement) commands. The time taken to switch the channel through this framework is studied experimentally. The API is then used to enable experimental evaluation of radio resource management algorithms running on a central controller. The control framework has been implemented on the ORBIT testbed using a set of 802.11 nodes with specified topology controlled by a logically centralized algorithm running on a server machine. The setup is used to study candidate channel assignment algorithms in terms of metrics such as total system throughput. Throughput analysis for different channel coordination algorithms was carried out using iperf to compare their performance. The performance of WiFi Automatic Channel Selection(ACS), a survey based channel selection algorithm, is compared with alternative graph coloring algorithms including HSum and HMinMax. The

experimental results show that the graph coloring algorithms HSum and HMinMax perform better when compared to ACS in terms of overall system throughput. The ORBIT nodes were grouped to form two different networks and performance of inter-network and intra-network cooperation was studied by controlling the channel through a logically centralized controller. The results show increased performance of inter-network cooperation when compared to intra-network cooperation in terms of overall system throughput.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Professor Dipankar Raychaudhuri for his support and invaluable guidance through this project. It has been a great learning experience and pleasure working under his supervision. I am very grateful to Ivan Seskar and Prof. Roy Yates for their valuable insights during weekly meetings. I would like to thank Parishad Karimi and Francesco Branzino for all the discussions we had that has helped structure most of the work. I am grateful to my family for their constant support and encouragement. Lastly, I would like to thank my friends at WINLAB and Rutgers University for making this journey a pleasant one.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The IEEE 802.11 working group for Wireless LAN was founded in 1992 under the chairmanship of Vic Hayes [18]. In 1993 Henrik Sjdin proposed the idea of Public Access Local Wireless Networks at the NetWorld+Interop conference in The Moscone Center in San Francisco [16]. While he did not use the term hotspot, this is considered the first mention of the concept. In 1994 funded by the National Science Foundation, Dr. Alex Hills of Carnegie Mellon begins to implement "Wireless Andrew" a wireless research initiative which originally provided coverage to 7 buildings on campus. The next milestone came when Australia's Commonwealth Scientific and Industrial Research Organization (CSIRO) patented a technique for reducing multipath interference of radio signals transmitted for computer networking and it was named the 802.11a standard [14]. Eventually in 1997 the first 802.11 standard was released which provided up to 2 Mbps link speeds. Then the term "hotspot" was coined and MobileStar became the first company to provide WiFi hotspots in public places like airports, hotels and it signed contracts with companies like Starbucks, American Airlines etc. The WiFi Alliance, a nonprofit trade association working for universal compatibility and quality user experience was also formed. Then 802.11b standard was released for WiFi operating at 11Mbps link speeds on the 2.4GHz frequency. In the later years a series of 802.11 standards were released which supported efficient transmission and better speed. The list of 802.11 technologies is summarized in the table below:

IEEE Standard	Frequency	Speed	Transmission range	Access Method
802.11	2.4GHz	1 to 2Mbps	20 feet indoors	CSMA/CA
802.11a	5GHz	Up to 54Mbps	20 to 75 feet indoors	CSMA/CA
802.11b	2.4GHz	Up to 11Mbps	Up to 150 feet indoors	CSMA/CA
802.11g	2.4GHz	Up to 54Mbps	Up to 150 feet indoors	CSMA/CA
802.11n	2.4GHz/5GHz	Up to 600Mbps	175+ feet indoors	CSMA/CA

Table 1.1: IEEE 802.11 standards

According to study conducted by iPass [10] a commercial WiFi network provider, by 2015 there were more than 50 million worldwide public hotspots, 80% increase since 2013. The number of hotspots is expected to grow rapidly and to hit 340 million global hotspots by 2018. With this exponential growth, WiFi is emerging as a truly global and roamable network.

1.1 WiFi Channels in 2.4GHz band

802.11 workgroup currently documents use of WiFi in five distinct frequency ranges: 2.4GHz, 3.6GHz, 4.9GHz. 5GHz and 5.9GHz bands. Technologies like 802.11b/g/n uses 2.4GHz band and it is most widely used WiFi frequency band. There are 14 channels in this band spaced 5MHz apart. As the protocol requires 16.25MHz to 22MHz channel separation, adjacent channels overlap. The below figure shows the graphical representation of 2.4GHz band channel overlapping.

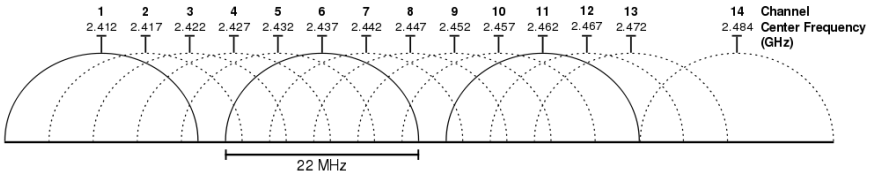

Figure 1.1: 2.4GHz WiFi Channels

As seen in the above figure the adjacent channels overlap which creates interference

when data is transmitted through them. With 22MHz bandwidth for each channel, 2.4GHz band give three channels that do not overlap - Channels 1, 6 and 11. Transmitting data through these channels will create close to zero interference.

Co-channel interference is not much of a problem unless there are multiple clients transmitting in the same channel. The is so because WiFi undergoes carrier sensing and backoff mechanism when it detects channel is busy. This time sharing mechanism does not affect the throughput severely when there are limited number of clients associated with the Access Point. Adjacent-Channel interference on the other hand is where you run into problems and channel selection becomes critical [15]. These channel related interference can be reduced or eliminated by selecting proper WiFi channels for a network.

1.2 WiFi Access Point Controllers

With increasing density of WLAN access points it becomes practically impossible to configure each and every access point manually. Apart from that, manual configuration of access points does not help in improving the system efficiency. Channel assignment is important factor for throughput performance of a WLAN system. A WLAN system should have least number of overlapping channels for better throughput performance. One another important factor is load balancing. This cannot be done by stand alone access points. Controller based access points can also balance the load keeping into account the overall system performance. Consider there are two access points with overlapping coverage as shown in the figure below

.

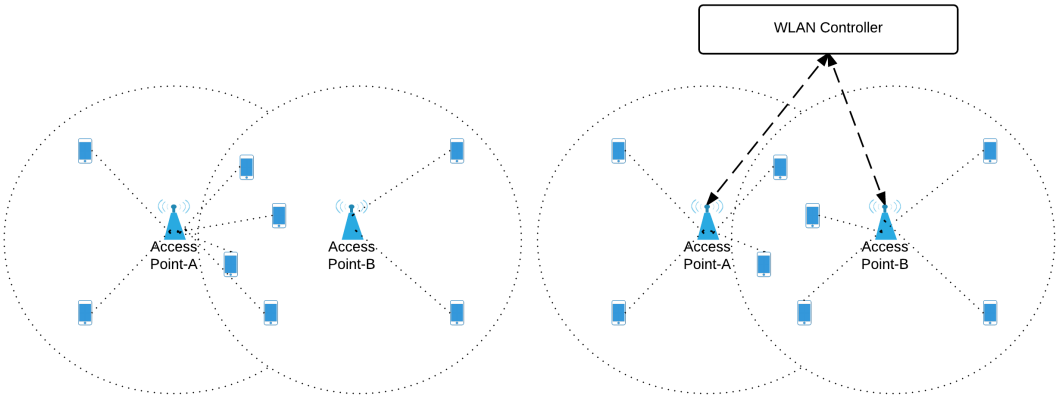

Figure 1.2: UE association with Access Points with and without controller

Suppose six clients are associated with AP-A and two clients associated with the AP-B. Considering each client gives same amount of load to the access point, AP-A is heavily loaded. In case of stand alone access points, they do not communicate which result in load imbalance and thus performance is degraded. If there is a controller in the system, it will have load information of both the access points. Thus it will reduce the load on AP-A by associating few clients in the overlapping region to AP-B as shown in the figure above.

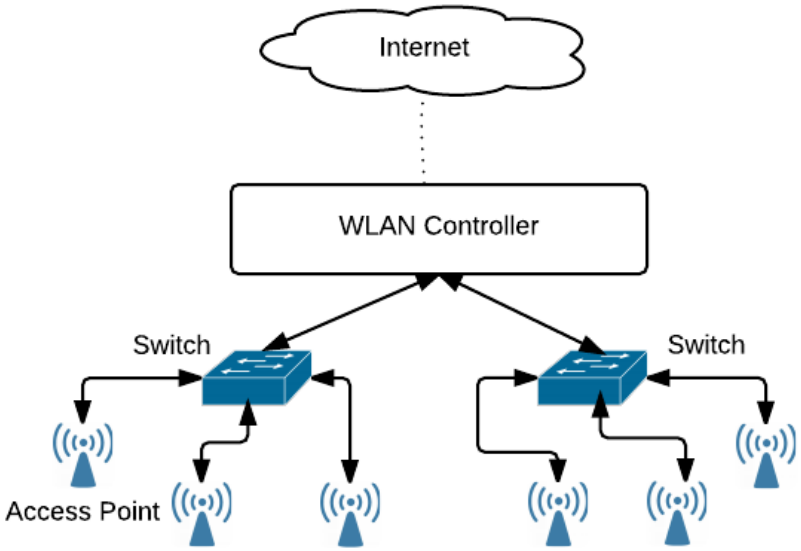

Figure 1.3: WiFi Access Point Controller Architecture

Figure 1.3 shows a typical WLAN controller architecture. As shown in the architecture above, each access point is connected to the controller through a switch. The controller has overall picture of the radio environment by maintaining a database of Access Point's operating channel, transmit power, location etc. Through these information the controller constructs a radio map of the wireless environment. A suitable algorithm is chosen and based on that new parameters are generated.

Chapter 2

Evolution of Wi-Fi and Related Work in Channel Co-ordination

2.1 Hostapd

Hostapd(Host access point daemon) is a software access point capable of converting any Wireless Interface card to an Access Point [12]. The latest version of hostapd supports madwifi and mac80211 based drivers. For the evaluation purpose we use mac80211 drivers. The below table shows the list of technology hostapd supports.

Technology	Band	Year	Max Speed
802.11a	5GHz	1999	54Mbps
802.11b	2.4GHz	1999	11Mbps
802.11g	2.4GHz	2003	54Mbps
802.11n	2.4GHz and 5GHz	2009	600Mbps
802.11ac	5GHz	2013	54Mbps

Table 2.1: List of technologies used by hostapd

As hostapd is a software access point, it has its own capabilities and drawbacks. It can create multiple APs on the same card which have different SSIDs. Hostapd can create one AP on one card and another AP on the second card all within the single instance of the hostapd [12]. Although hostapd can create multiple APs on the same card, all the APs should operate in the same channel. If there is a channel change, both the APs have to change the channel. Another drawback is assigning IP to the AP and the clients connected to the AP. This has to be done through DHCP server. In our evaluation this is done using ifconfig commands.

2.1.1 Hostapd configuration file

The hostapd configuration file specifies the list of configuration parameters which it uses to start the access point. The configuration parameters include interface to use, SSID name, channel number, type of technology to use etc. The figure below shows a sample configuration file.

Figure 2.1: Sample hostapd configuration file

As shown above, hostapd uses the parameters specified in the configuration file to start the access point. Capabilities like type of driver being used and technology need to be specified carefully considering the capabilities of the driver in the system. In case of 2.4GHz band the channel number ranges from 1 to 13, with 1, 6 and 11 being orthogonal channels. This differs based on the technology specified in configuration file.

2.1.2 Hostapd architecture

Hostapd includes IEEE 802.11 access point management (authentication / association), IEEE 802.1X/WPA/WPA2 Authenticator, EAP server, and RADIUS authentication server functionality. It can be built with various configuration option, e.g., a standalone AP management solution or a RADIUS authentication server with support for number of EAP methods. Hostapd is designed in such a way that it is OS independent, has a portable C code for all its functionality [12]. This helps the users to create softAP on different platforms having different functionalities. Hostapd implements a control interface that can be used by external programs to control the operations of the hostap

daemon and to get status information and event notifications. There is a small C library that provides helper functions to facilitate the use of the control interface.

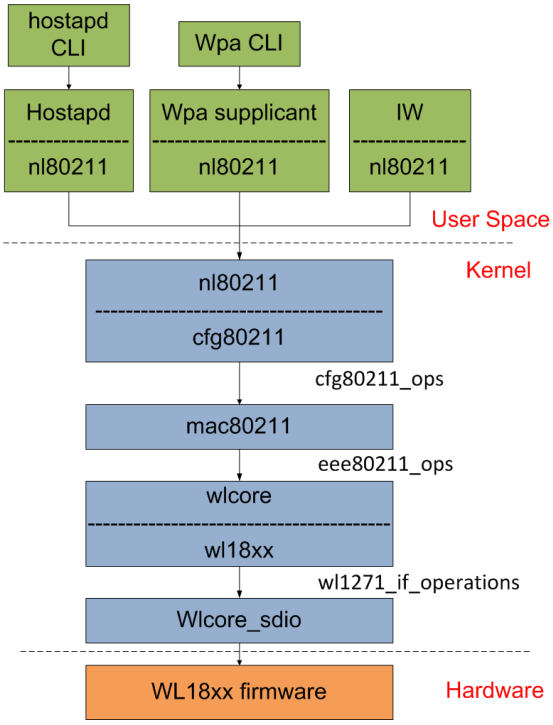

Figure 2.2: hostapd and wpa-supplicant architecture

The hostapd code-base comes with two main components, hostapd and wpa_supplicant as shown in the figure above. Hostapd is used to run the access point and wpa_supplicant is used for the client. Hostapd interacts with driver with the help of driver interface functions. In the figure shown above nl80211 interface function interacts with the nl80211 driver. Hostapd has a control interface which acts as an interface to hostapd_cli. hostapd_cli exposes commands to users through which different hostapd parameters can be controlled. hostapd_cli is further used in our implementation to develop HTTP framework on top of it.

2.2 hostapd_cli

hostapd_cli is a frontend program to interact with hostapd. It is used to query the status of hostapd and set parameters through command line interface. hostapd_cli has two modes, interactive and command line. Both the modes share the common set

of commands. Interactive mode is started when hostapd_cli is executed without any parameters on the command line. Commands are then entered from the controlling terminal in response to the hostapd_cli prompt. In command line mode, the same commands are entered as command line arguments. The table below shows few of the commands supported by hostapd_cli:

Command	Description
status	Reports information pertaining to number of clients connected, channel etc.
interface	Show available interfaces and/or set the current interface.
get_config	Get information present in hostapd configuration file.
chan_switch	Switch channel through CSA(Channel Switch Announcement) mechanism.

Table 2.2: hostapd_cli commands

The chan_switch command is used to develop HTTP framework on top of hostapd. The format of the command is shown below:

chan_switch #beacons #freq

Where #beacons shows the number of beacons after which the channel switch command is sent. #freq is the frequency of the new channel.

2.3 Channel Switch Announcement(CSA)

The Channel Switch Announcement is used by an Access Point in a BSS to advertise when it is changing to a new channel and the channel number of the new channel [2]. The format of the Channel Switch Announcement element is shown below.

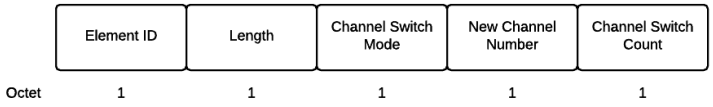

Figure 2.3: CSA Packet Format

- Channel Switch Mode: Indicates any restrictions on transmission until a channel switch. An AP in a BSS sets the Channel Switch Mode field to either 0 or 1 on transmission.
- New Channel Number: set the number of the channel to which the AP is switching.
- Channel Switch Count: this field either is set to the number of TBTTs(Target Beacon Transmission Time) until the AP sending the Channel Switch Announcement element switches to the new channel or is set to 0. A value of 1 indicates that the switch occurs immediately before the next TBTT. A value of 0 indicates that the switch occurs at any time after the frame containing the element is transmitted.

An AP shall inform associated STAs that the AP is moving to a new channel and maintain the association by advertising the switch using Channel Switch Announcement elements in Beacon frames, Probe Response frames, and Channel Switch Announcement frames until the intended channel switch time [2]. The AP may force STAs in the BSS to stop transmissions until the channel switch takes place by setting the Channel Switch Mode field in the Channel Switch Announcement element to 1. The channel switch should be scheduled so that all STAs in the BSS, including STAs in power save mode, have the opportunity to receive at least one Channel Switch Announcement element before the switch. The AP may send the Channel Switch Announcement frame in a BSS without performing a backoff, after determining the WM is idle for one PIFS period. A STA that receives a Channel Switch Announcement element may choose not to perform the specified switch, but to take alternative action. For example, it may choose to move to a different BSS. A STA in a BSS that is not the AP shall not transmit the Channel Switch Announcement element.

2.4 Related Work in Wi-Fi channel coordination

[7] analyses the growing problem of dense wireless networks. It studies the coexistence of low cost residential APs and actively managed service provider APs in overlapping

frequency and time domains. Detailed simulation experimentation is done for increasing ratio of residential APs compared to managed APs and the other way around. The impact on throughput on clients as the ratio of residential APs increase is examined in this paper. A key finding of the paper is that as the density of the managed APs increase, although the overall band utilization increase the managed APs would perform worse compared to residential APs. Liew's MIS model is constructed for the given contention graph and performance of different channel assignment schemes is studied. The simulation results show that the performance of single APs would be comparatively better if a un-managed AP is added into its vicinity rather than adding a managed AP.

Reference [19] studies the existence of WiFi and LTE in unlicensed band. The studies in the paper show that co-existence of WiFi and LTE Unlicensed networks causes significant interference to WiFi as LTE is not designed to sense the channel before transmitting data. This causes WiFi to back off multiple times as LTE continuously transmits data and keep the channel busy. WiFi and LTE throughput characterization is formulated based on the distance and power received from the interfering AP. In the case of WiFi the interfering AP is LTE base station and in the case of LTE, the interfering AP is WiFi base station. Two type of optimization is discussed, Joint Power Control Optimization and Joint Time Division Channel Access Optimization. The Joint Power Control Optimization optimizes the power transmitted by each base station so that the aggregate throughput received is increased. The Joint Time Division Optimization optimizes the time share of each Access Point so that the aggregate throughput received by all the base station is increased. The centralized optimization approach of WiFi and LTE parameters is incorporated in our evaluation of WiFi channel coordination algorithms.

Reference [17] paper presents the design and proof of concept validation of a novel network assisted spectrum coordination service. The paper suggests an overlay network for dissemination of spectrum usage information to otherwise independent radio devices. This is used to execute a decentralized spectrum coexistence policies. A

system architecture for spectrum coordination is suggested which contains in-network Spectrum Gateways(SGs) that connect between themselves and to all wireless devices in the region through a overlay network. The differentiating feature of this architecture is that the spectrum updates from a wireless device are updated to those devices which are under the influence of the wireless device. This is done using a region of interest based geo-routing protocol. Therefore each wireless entity acquires updates from the devices which are in its influence. Finally a distributed coexistence algorithm is suggested having Non-adaptive parameter selection(NAPS), Adaptive Parameter Selection(APS) and Global Coordinated Resource Packing(GCRP). These methods describe the way in which radio parameters are disseminated across the network. Three different association schemes are compared: Least Distance, Intra-Network Optimization and Cooperative Optimization.

Reference [8] proposes a technique to improve the usage of wireless spectrum in the case WLAN environment. A weighted form of graph coloring is proposed that takes into account the interference observed from each of the wireless node in the topology. A scalable distributed algorithm for channel assignment is proposed. The paper also demonstrates that better performance can be achieved by assigning partially overlapping channels to Access Points. The weights are assigned to the edges based on the potential of interference. Higher the weight, higher the interference in that particular edge. So, the channels are assigned based on the weight of the edge. Two different channel assignment algorithms are discussed in this paper, HSum and HMinMax. The HSum algorithm uses the sum of the weights to the adjacent nodes which have the same channel. The channel is assigned based on the minimum sum. HMinMax is based on the maximum weight of the edge for each particular adjacent node. The channel is assigned based on least maximum value. Finally the simulation results for each algorithm is performed and the average system throughput is measured based on the channel assigned to each node. The results also show that HSum performs better in terms of overall system throughput when compared to HMinMax.

Reference [6] proposes a Smart WiFi methodology that improve the spectrum utilization and overall WiFi performance. The paper proposes two different entities, Radio Environment Maps(REM) and Radio Resource Management(RRM). REM enables use of wireless environment data sets such as Access Point location, pathloss models, real time interference levels and statistical channel occupancies. RRM is used for efficient WiFi coordination and optimization that utilizes REM information. REM is generated using Measurement Capable Devices like smart-phones, WiFi access points. These devices are exclusively for measurement purposes and they report the information to REM data storage and acquisition. The REM data can be used by different entities like User Interface program, RRM unit etc. RRM algorithm is proposed that focuses on maximizing the WiFi sum throughput through Signal to Interference plus Noise ratio. Two resource allocation strategies are discussed for optimal WiFi performance. Strategy one discusses appropriately allocating disjoint communication channels to each of the access point so that there is minimal interference. However this can be used where there is significant spectrum under utilization. In the case of dense WiFi network multiple access point in a location have to be assigned same channel. To deal with this, strategy two is proposed. Strategy two discusses assigning overlapping channels to access points by introducing a power optimization algorithm. The algorithm ensures maximum system throughput. The RRM also utilizes historical information of the access points to make the decisions. Here the RRM can be compared to a controller which has overall picture of radio environment.

Reference [9] discusses fairness problems in uncoordinated WiFi deployments. In the case of dense WiFi network few of the access points struggle to get their share of channels which in turn affects the overall system performance. An algorithm called MAXChop is proposed which works on channel hopping technique and uses only orthogonal channels. An uncoordinated deployment is examined where there is no central entity which controls the position of the access point, transmit power, channel assignment and other important factors. The fairness is examined only at the access point level and not between access point and client. The paper suggests that even if a graph

is k-colorable, a good static channel assignment algorithm will have unfairness for few access points. To improve the unfairness in static channel assignment, channel hopping technique is discussed. In order to fairly treat all access point channel hopping technique is used. So, basically time is divided into slots and at the end of each time slot channel is switched. The channel hopping sequence is obtained through the MaxChop algorithm. The algorithm at first obtains the hopping sequence of all the neighboring APs. Based on that it decides its own hopping sequence. The channel hop is performed asynchronously. The results show that channel hopping increases the overall system throughput. The fairness of the APs especially the ones which are affected in static assignment is increased. The UDP throughput of the overall system increased by about 10% although channel switching did reduce the system throughput by 0.5% to 0.6%.

Chapter 3

HTTP framework implementation on hostapd

3.1 GNU Libmicrohttpd

GNU Libmicrohttpd is a C library used to run HTTP server as part of another application [3]. GNU libmicrohttpd can be used as a independent library without any dependencies. The APIs provided by the library are simple to use, well documented and are backward compatible with all the versions of HTTP. Libmicrohttpd is implemented in such a way that the HTTP server runs concurrently with other applications and message exchange between http server and other applications is made simple and supports almost all formats. These features are the main factors to use Libmicrohttpd in the framework developed.

The APIs used for the development of the framework are discussed below.

MHD_start_daemon(MHD_USE_SELECT_INTERNALLY , PORT, NULL, NULL, & answer_to_connection, (void *) PAGE, MHD_OPTION_END): Starts HTTP server on the given port number. answer_to_connection is a callback function which is triggered when there is HTTP request. The arguments have to end with MHD_OPTION_END. The function returns NULL on error and handle to the daemon on success.

MHD_create_response_from_data(size_t sz, void *data, int must_free, int must_copy): Creates a response with size sz and data pointed by the pointer "data". When must_free is set to true the data is erased after the response it sent. When must_copy is set to true, the daemon makes a copy of the data sent in the response. The function returns NULL on error.

3.2 Architecture

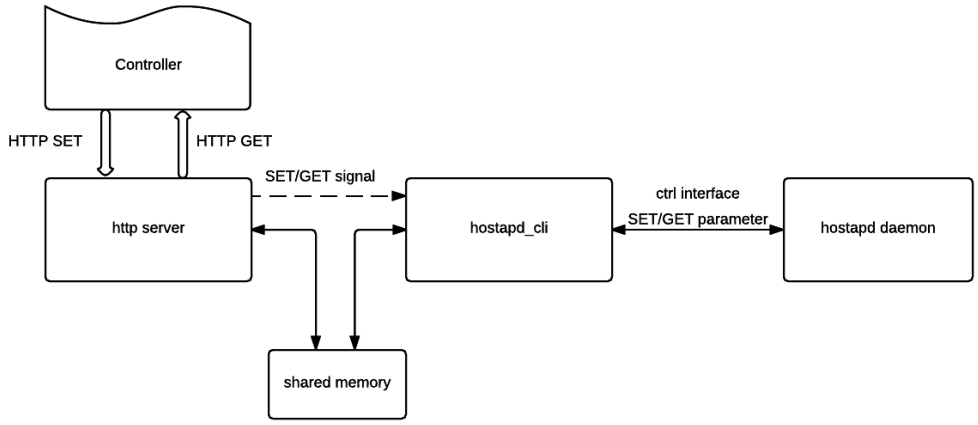

Figure 3.1: Implementation Architecture

Figure 3.1 shows the implementation architecture of HTTP framework to control the channel parameter of hostapd. The hostap daemon running the access point using the wireless interface is executed in each node. This software uses IEEE802.11 drivers to run the access point. The hostapd_cli is a command line interface based front end program for interacting with hostapd. It has set of commands which you can use to change certain parameters in hostapd. For example "interface" command shows available interfaces in the node and sets the interface when multiple interfaces are available. Similarly another command is set_chan which is used to set the channel of access point. This is used only if the driver supports Channel Switch Announcement(CSA) feature as discussed in Section 2.3. The hostapd_cli interacts with hostapd through the control interface. hostapd implements a control interface that can be used by external programs to control the operations of hostap daemon and to get status information and event notifications. There is a small C library, in form of a single C file, wpa_ctrl.c, that provides helper functions to facilitate the use of control interface. External program that needs to communicate with hostapd can use hostapd_cli by linking wpa_ctrl.c. The HTTP server acts as an interface between controller and hostapd_cli. The HTTP server sends the GET and SET signal sent by the controller to the hostapd_cli through a shared memory. The HTTP SET signal received from the controller has parameter

type and value. Once the HTTP SET signal is received from the controller, the values embedded in the request are updated in the shared memory. Once the values are updated, HTTP server sends SIGINT signal to hostapd_cli. hostapd_cli after receiving the SIGINT signal accesses the updated value and perform the related function. A typical message flow within the system for channel switch is shown below

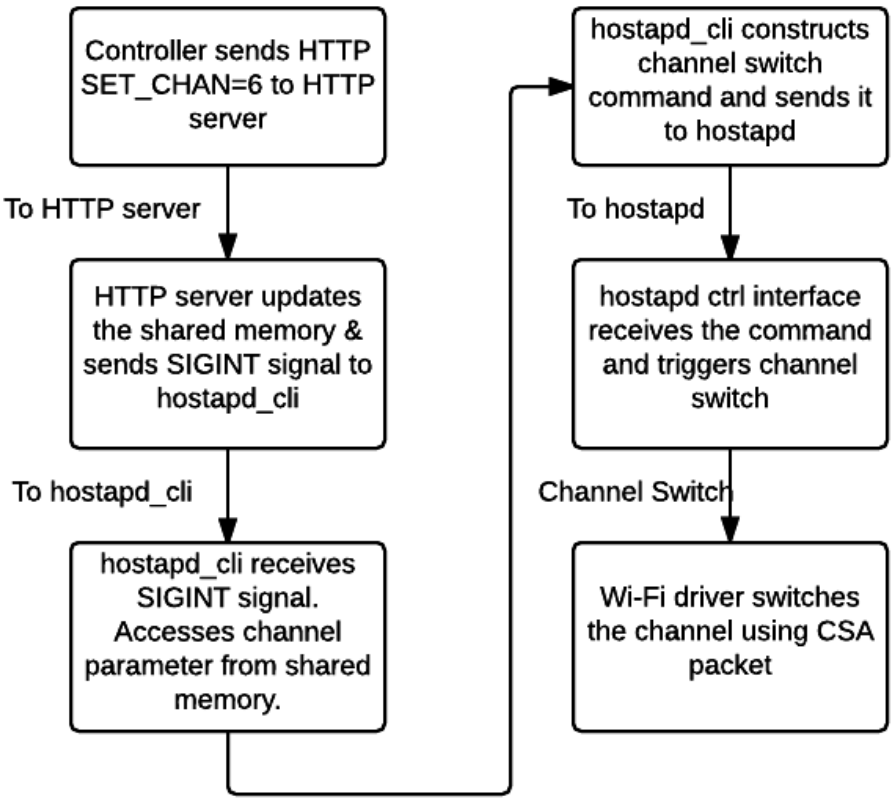

Figure 3.2: Channel Switch Message Flow

The figure above shows how the HTTP channel switch message flows through the framework. At first the controller decides the new channel to which an access point has to change. It frames a HTTP request by embedding new channel number into the HTTP request. This HTTP request is sent to the HTTP server by mentioning appropriate port number. The port number in the HTTP message request and the

port number in which the HTTP server is running has to match. A call back function answer_to_connection is implemented in HTTP server to receive the HTTP request message. Once this message is received by the server it extracts the parameter in the message. First it decides whether the message is GET or SET request. In this example it will be a SET request. Then the function extracts the channel number in the message request. Once the channel number is received the next task is to send the channel information to hostapd_cli. To do this task, shared memory is used. The channel number is shared between HTTP server and hostapd_cli by using shared memory. This is explained in the Figure 3.1. hostapd_cli accesses the updated channel through the shared memory. hostapd_cli is a software on top of hostapd which is developed to control different parameters like channel power etc. This software gives command line interface to the user to control the parameters through set of commands. The software has been modified to remove the command line interface feature and to act as an interface between HTTP server and hostapd_cli. The channel number is accessed and the appropriate command is generated. The command generated is sent to hostapd through wpa control interface. Finally wpa control interface triggers a channel switch using Channel Switch Announcement packets. Client connected to the access point receives this packet and internally switches the channel.

Chapter 4

Experimental Approach and Algorithmic design

Channel planning for the access points especially in public places where there are numerous access points is very crucial. Efficient planning of channel assignment is in direct correlation with the overall system throughput. The HTTP framework developed exposes APIs to GET and SET the channel of an access point through hostapd_cli interface. ORBIT is a Radio Grid testbed to test scalable wireless protocols. The ORBIT infrastructure is used to perform the experimentation. The prime advantage of using the ORBIT testbed is that the experimentation can be carried out in real world indoor wireless environment. The other advantage is that the experiment can be scaled to many nodes and the results can be obtained for different topologies.

4.1 ORBIT test bed setup

The ORBIT test bed is a indoor radio grid setup at WINLAB, Rutgers University [13]. It is used as a emulator for controlled indoor wireless experimentation. The radio grid consists of 20x20 wireless nodes having 802.11a/b/g wireless cards with around 1 meter spacing in between. The users of the node can run their own OS image and software packages on top of the operating system. The users also have access to node console and the console logs. For example, users can run their own network layer protocols or new application software to test a specific network application. The following figure shows the indoor grid set up in orbit test bed.

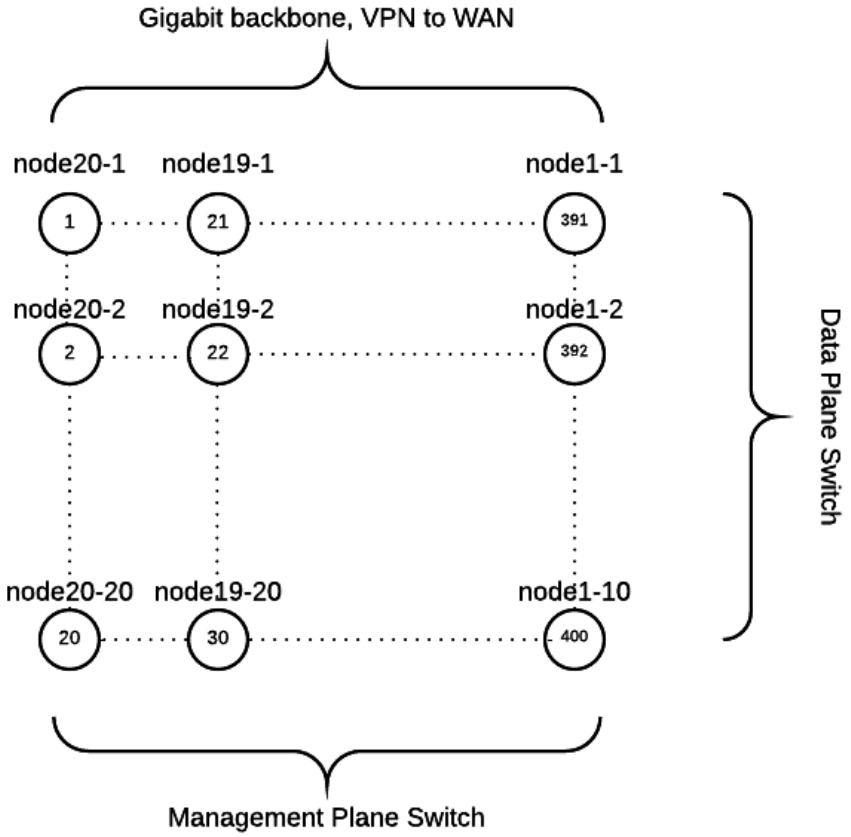

Figure 4.1: Indoor Grid setup in ORBIT testbed

The grid contains ORBIT radio nodes which serve as a primary platform for user experimentation. It contains 20x20 nodes arranged in rectangular position as shown in the figure above. Each row has 20 nodes aligned next to each other. The distance between the adjacent node is close to 1.13 meter. Each node is a computing machine with re-loadable user Operating System and have WLAN interfaces. The instrumentation subsystem is used to measure radio measurements and is also used to generate artificial subsystem. There are two types of support servers. The front-end server support the web services and back end servers is used for experimentation and data storage.

The ORBIT testbed has Management/Control software and user level application software to control the Radio nodes. The Management/Control software contains Node

Handler, Collection Server and Disk Loading Server. The Node handler distributes the experimental script to the nodes through multicast. The user has the power to group the nodes and run the experiments simultaneously on all the nodes. The nodes in the group receive the script and run the experiments. The collection server collects the results obtained through experiment. The nodes after running the script, collect the statistics and send the results to multicast server through multicast channel. The disk loading server is used to load the user image on hard disk on nodes. An image can be loaded on multiple nodes through a multicast channel. The radio nodes software contains Node Agent, ORBIT Measurement Library(OML) and Libmac. Node agent resides in nodes and receive commands from Node Handler. It is responsible to run and stop the application and report the status of the experiment to the Node Handler. The ORBIT Measurement Library defines a data structure for sending/receiving, encoding/decoding measurement data that are exchanged. Libmac is a user space C library to control MAC layer parameters like Tx power, Channel Setting etc. It is developed on top of device drivers present in the node.

4.2 Need for channel assignment algorithms

WiFi works on carrier sensing mechanism. Each access point scans its wireless environment to see if the frequency in which it is intended to operate on is busy or free. If it senses that a particular channel is busy it backs off and scans the wireless environment again. This repeats till the channel becomes free. Basically WiFi carrier sensing is composed of two distinct functions, Clear Channel Assessment(CCA) and Network Allocation Vector(NAV). Clear Channel Assessment is based on physical carrier sensing which listens to received energy on radio interface. CCA is defined in IEEE 802.11-2007 as part of Physical Medium Dependent(PMD) and Physical Layer Convergence Protocol(PLCP) layer [11]. NAV is virtual carrier sensing where the channel is kept busy for mandatory amount of frames. The atheros ath9k driver uses the CCA mechanism for carrier sensing. Clear Channel Assessment is composed of two related functions, carrier sense(CS) and energy detection(ED). Carrier sense is the ability of an access

point to receive and decode a incoming WiFi signal preamble. In addition CCA must report the channel as BUSY for the length of received frame indicated in the frame's PLCP length field. PLCP header has either one of the following information: the time required to transmit the frame of the number of the octets present in the frame. Based on this information the access point is backed off for a particular amount of time. The PLCP frame format is shown in the below figure.

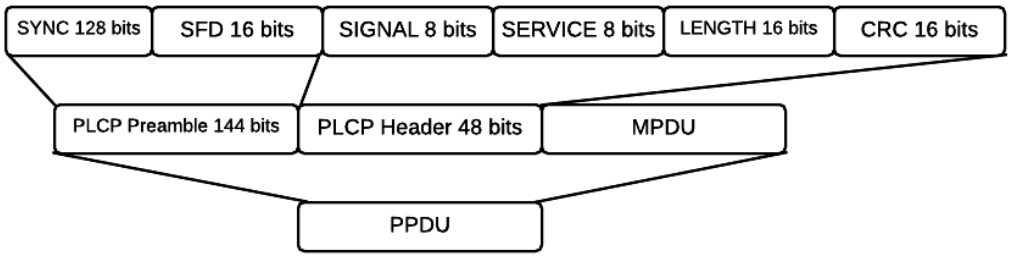

Figure 4.2: PLCP Frame Format

PLCP header contains fixed data rate so that the station receiving that packet can decode the information [5]. Energy detection is referred to ability of the access point to detect non WiFi signals. The signal may involve signals from interference sources and unidentifiable WiFi transmissions that have been corrupted or no longer can be decoded. ED may involve power threshold below which the channel is not termed busy. As discussed above if the channel is busy there is significant impact on the throughput of the access point which is trying to transmit data in the same channel. So, uncoordinated WLANs can be greatly affected by interference from neighboring WLANs. There has to be proper channel planning based on the radio map as channel assignment has direct correlation with the overall system throughput.

4.3 Channel assignment algorithms

There are different approaches to the channel assignment based on how the algorithms are executed. The algorithms can be broadly classified as centralized and distributed.

The centralized algorithm works on a central controller. The central controller accesses the WiFi parameters of all the access points it is controlling. It has overall picture of the access points and their radio interference region. The controller after receiving the parameters of the underlying access points executes the algorithm. The parameters are accessed through an interface between Controller and Access Points. The channel assignments algorithms used for the evaluation are discussed below.

4.3.1 Automatic Channel Selection algorithm

Automatic Channel Selection(ACS) is a software package that is inbuilt with hostapd [1]. The algorithm enables the interfaces to automatically figure out the channel of operation based on the radio environment. ACS is a survey based algorithm that uses nl80211 survey API command to query interface for channel active time, channel busy time, channel tx time and noise. The active time is the amount of time interface has spent in the channel and the busy time is the amount time the interface has found that the channel is busy and could not initiate communication. The tx time is the amount of time the interface has spent transmitting data in the channel. A parameter known as interference factor is calculated for each channel to determine which channel to select. The interference factor is the ratio of amount of time the interface has observed the channel busy to amount of time the interface has spent transmitting data. This corresponds to:

$$(busy\_time - tx\_time)/(active\_time - tx\_time) * 2^{(chan\_nf - band\_min\_nf)}$$

The power 2 reflects the way power decreases with distance. Minimum noise floor is the lowest recorded noise in all the channels. There are factors that are considered along with interference factor to decide the channel. The target operational bandwidth is one of them. For eg: 20MHz band in 2.4GHz channel overlap and 20MHz band in 5GHz band do not overlap. These factors are considered to select the channel.
Automatic channel selection is triggered by enabling ACS in hostapd config file. The .config file is edited as follows to enable ACS.

$$CONFIG\_ACS = y$$

To make hostapd perform ACS during run time, channel number in hostapd.conf file has to be set to 0 or acs as follows

$$channel = 0$$

or

$$channel = acs\_survey$$

Once the hostapd is started with this configuration the ACS algorithm is triggered before the driver is initialized. After execution of algorithm the channel is decided and this channel number is used to start the access point. Once ACS is executed at the start the channel is constant for the life time of the access point.

4.3.2 HSum Algorithm

HSum is a channel coloring algorithm and it works well in the region where multiple wireless devices share a limited RF spectrum. The algorithm can be implemented in centralized as well as in distributed fashion. Before we go into details of the algorithm, we shall discuss the method of calculating the weights between nodes to run the algorithm. Let us consider a simple four node graph as shown below:

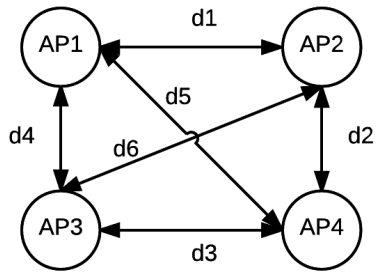

Figure 4.3: A simple topology to illustrate the calculation of weight of an edge

Let the distance between each pair of access point be d1, d2 upto d6 as shown in the figure above. Here we assume that all the access points transmit with the same power. The received signal strength is calculated based on the distance between the access points. The path-loss increases with the increase in distance between the nodes. Following path-loss model is used to calculate the received power [20].

$$L = 20logf + Nlogd + P_f(n) - 28$$

where,

L = total pathloss in dB.

f = frequency of transmission in MHz.

d = Distance in meters.

N = The distance power loss coefficient.

n = Number of floors between transmitter and receiver.

$P_f(n)$ = Floor loss penetration factor.

The distance power loss coefficient for 2.4GHz frequency is 28 and and floor loss coefficient for single floor is 15. Based on the above formulation the path-loss for each access point from the remaining access point is calculated. The received power in dB is determined based on the pathloss calculated and transmit power. Considering there are N access points, each access point receives signals from N-1 access points. Therefore each access point perform N-1 pathloss calculations. Total pathloss calculations performed in the overall system is $N * (N - 1)$. Therefore pathloss calculations in a system is proportional to N^2 where N is total number of access points.

The algorithm has two steps, the initialization step and the optimization step. In the initialization step all the access points are either assigned a random channel or LCCS(Least Congested Channel Search) is performed to assign the channels. Then in the optimization step the algorithm is executed. The algorithm can be explained in following manner:

Notations:

k: Number of available channels.

$N(ap_i)$: Set of neighbors of ap_i.

$C(ap_i)$: Channel assignment for ap_i.

Initialization step:

- *LCCS*: Run hostapd's ACS algorithm for the initial channel assignment.
- $(C(ap_i)) \rightarrow c_i$: Assign channel c_i to AP i.

Optimization steps:

- $sum_i = \sum_{n=1}^{i} w_i \quad \forall\ i \in c_i$: Add the weights of all the edges which have the vertices (Access Points) operating in the same channel.
- $min(sum_i)\ \forall i$: Calculate the minimum of the summations calculated in the previous step.
- Assign the channel which has minimum summation.

Based on the summation of the weights of the edges operating in same channel the new channel is decided. At the start a random node is chosen and the above steps are performed. After completion of the above mentioned algorithmic steps next random node is chosen. This is followed untill all the nodes in the graph are covered. In the example graph shown in Figure 4.3 one of the ways in which algorithm progresses might be AP1, AP2, AP3 and AP4. The channels are assigned in a serial fashion and an iteration is complete once all the nodes in the graph are covered.

4.3.3 HMinMax Algorithm

The algorithm has two steps, initialization step and optimization step. In the initialization step all the access points are either assigned a random channel or LCCS(Least Congested Channel Search) is performed to assign the channels. Then in the optimization step the algorithm is executed. The following steps explains the algorithm:

Notations:

k: Number of available channels.

$N(ap_i)$: Set of neighbors of ap_i.

$C(ap_i)$: Channel assignment for ap_i.

Initialization step:

- *LCCS*: Run hostapd's ACS algorithm for the initial channel assignment.
- $(C(ap_i)) \rightarrow c_i$: Assign channel c_i to AP i.

Optimization steps:

- $max_i = \max_{n=1}^{i} w_i \ \ \forall\ i \in c_i$: Find the maximum among all the edges which have the vertices (Access Points) operating in the same channel.
- $min(max_i)\ \forall i$: Calculate the minimum of the summations calculated in the previous step.
- Assign the channel which has minimum summation.

Based on the maximum value in the weights of the edges operating in same channel the new channel is decided. Similar to HSum algorithm discussed above, at the start a random node is chosen and the above steps are performed. After completion of the algorithmic steps next random node is chosen. This is followed until all the nodes in the graph are covered. In the example graph shown in figure 4.3 one of the ways in which algorithm progresses might be AP1, AP2, AP3 and AP4. The channels are assigned in a serial fashion and an iteration is complete once all the nodes in the graph are covered.

4.4 Controller design and implementation in ORBIT test bed

ORBIT console is used to develop controller which controls multiple nodes in the grid. The nodes are grouped to form a logical network. A set of tasks are simultaneously executed for the nodes that are grouped. An overview of the controller architecture is as shown below.

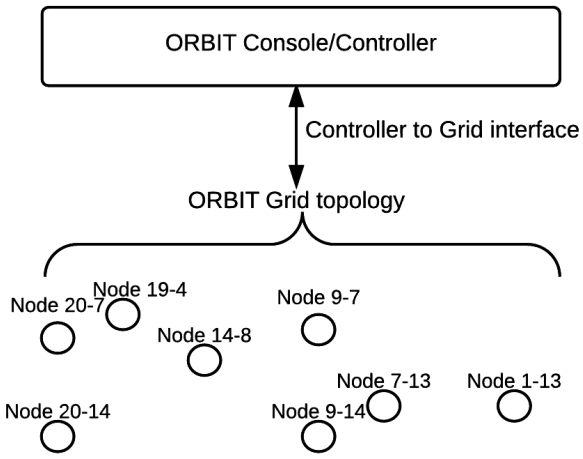

Figure 4.4: Overview of Controller Design in ORBIT testbed

The controller implemented in the console interacts with ORBIT nodes through an interface which connects application server to grid through a dataplane switch. A typical controller experiment having eight nodes to measure a throughput performance when channel of few of the nodes are switched is explained below.

- *Define Topology:* Choose appropriate nodes in the grid to perform experiment. The factors considered in choosing a node can be position of the node, its driver capabilities etc. The position has to be chosen appropriately so that the experiment depicts a real world scenario. For the channel switch through CSA(Channel Switch Announcement) mechanism each node has to be equipped with ath9k driver. In the experiments we perform, the client is collocated in the same node where the modified hostapd is triggered. So, each node should support two wireless interfaces WLAN0 and WLAN1.

- *Define Group:* Set of nodes are grouped to form a logical network. *defGroup* a OMF command is used to logically place the nodes in a network to perform synchronous operations.

- *Load Drivers:* After the groups are defined appropriate drivers are loaded onto

the nodes. ath9k driver is used which supports Channel Switch Announcement mechanism. iwlwifi driver is used in the second interface WLAN1 which acts as a client.

- *Configuration changes:* Configuration files are edited using sed functions for the initial channel assignment and unique access point names.

- *Throughput test using iperf and IEEE 802.11 control API:* Iperf server is executed in the access point and iperf client is executed in the clients. Iperf client server connection is established by appropriately specifying the IP addresses. Throughput measurements are performed by sending HTTP GET/SET messages to appropriate nodes.

Chapter 5

Evaluation

Experiments were conducted to evaluate the algorithms discussed in the previous section using the HTTP framework developed. These evaluations were conducted using the ORBIT facility at WINLAB [4]. The 400 node indoor grid was used for the experimentation. Topologies are created in the grid by selecting different nodes. The nodes are selected based on their location so that the evaluation is realistic. Also there are restrictions on the driver capabilities of nodes. As discussed earlier the nodes are used in such a way that both access point and client connected to it reside in the same node. Each node has two WLAN interfaces, these two interfaces are used to create the access point and client. The access point was created in the WLAN0 interface having ath9k driver and client connecting the access point is created in WLAN1 interface having iwlwifi driver. Each client is connected to the access point through the wireless channel. Iperf is used to communicate between access point and client. Iperf server is started at the access point running at the interface WLAN0 and iperf client is started at the interface WLAN1. With iperf client acting as a sender of data and iperf server acting as a receiver of data, uplink throughput measurements are conducted in all the experiments. Throughput measurement is done by conducting iperf traffic measurement between client and server. We need to keep note of one point in this setup. If both iperf server and client are running in the same machine, by default the traffic is directed towards the client through the internal system. But for the experimental evaluation the throughput measurement has to be done through the external wireless interface. To achieve this, routing configurations are done within the nodes so that traffic between interface WLAN0 and WLAN1 is passed through the external wireless interface.

5.1 Evaluation of Channel Switch Time using the 802.11 Control API

The HTTP framework developed which is discussed in Chapter 3 uses hostapd_cli software to switch channel without bringing down the interface. A unique HTTP URL which contains the new channel is sent to the HTTP server. The server receives the URL and extracts the channel number present in it. The new channel number is shared to hostpd_cli software. Using this channel hostapd_cli sends a channel switch command to hostapd. Channel Switch Announcement(CSA) mechanism discussed in section 2.3 is used to switch channel. The client connected to the access point receive the CSA packed and switches the channel. All these steps happen without loss in connectivity. In this section we study the performance of the hostapd channel switch through CSA mechanism.

Two nodes in ORBIT testbed were used to perform this experiment. Initially both nodes are assigned the same channel and the access points are started. Throughput performance of the nodes is monitored using the iperf server and client setup. Iperf server is started at the access point. The client connected to the access point runs the iperf client. Iperf communication between access point and the client is triggered once the connection between access point and client is established. The channel is switched using the API and throughput measurements are done. The figure below shows the varying throughput as the channel is switched.

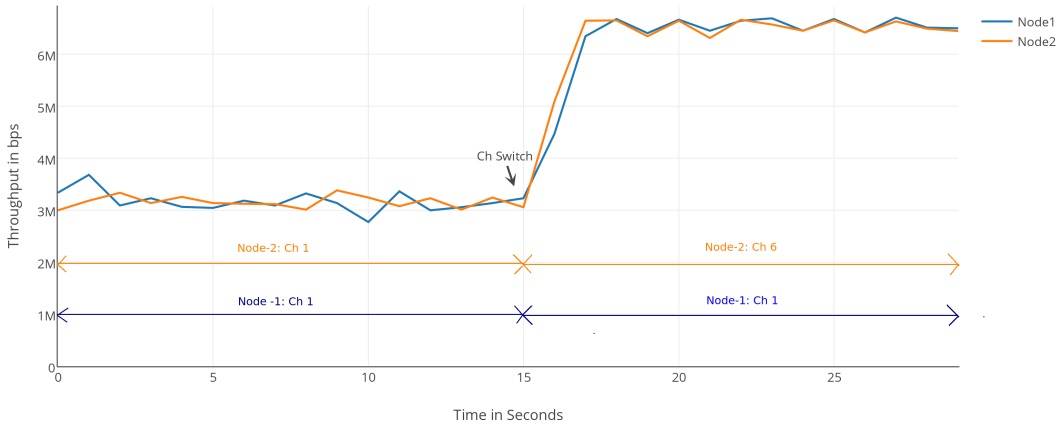

Figure 5.1: Iperf throughput measurement for channel switch using 802.11 control API

As shown in the figure above Node-1 and Node-2 start their respective access points with channel 1. Between 0 to 15 seconds both nodes are operating in same channel. Channel switch is triggered using 802.11 control API just after 15th second. Node-2 is changed to channel 6 during second half of the experiment. Channel 6 is chosen because it is orthogonal to channel 1 and nodes operating simultaneously in these channels have close to zero interference. The sudden peak in throughput is seen after 15^{th} second due to channel switch. The iperf traffic was generated using UDP packets. Average throughput of both the nodes before channel switch is 3.358Mbps and average throughput of the nodes after channel switch is 6.478Mbps.

5.2 Evaluation of Channel Assignment algorithms

Throughput analysis of the channel assignment algorithms discussed in chapter 4 is performed in this section. Three algorithms Automatic Channel Selection(ACS), HSum and HMinMax are compared and the results are analyzed.

5.2.1 Topology for evaluation of channel assignment algorithms

The experimental setup for the throughput analyses involve setting up a topology in ORBIT testbed by choosing appropriate nodes. The nodes are selected based on their position and capability. Location of nodes are chosen so that it imitates a real world scenario. The topology used in the grid for the experimentation is shown below:

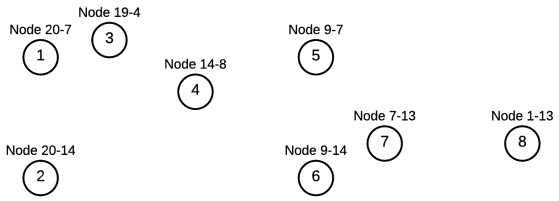

Figure 5.2: Grid topology used for the experimentation

The figure above shows the relative positioning of the nodes in the topology. The nodes are numbered 1 through 8. Also the position of the node in grid is specified by specifying two dimensional names. For eg. node 20-7 is in 20^{th} column and 7^{th} row in the grid.

5.2.2 Throughput analyses of Channel Assignment algorithms

Channel assignment algorithm discussed in Chapter 4 are compared by setting up the topology discussed in the section above. Each node in the grid has a collocated Access point and a client. We note that as the experiment was conducted in ORBIT grid testbed facility, there were other access points operational in the same frequency band. The interference obtained from those access points is also considered in executing the algorithm. Iperf server client communication setup is used to measure the throughput. The below table summarizes the throughput results obtained.

Algorithm	ACS	HMinMax	HSum
Average System Throughput	7.872Mbps	8.13Mbps	8.94Mbps
15 Percentile Throughput	0.95Mbps	1.37Mbps	1.1Mbps

Table 5.1: Throughput Analyses of Channel Assignment Algorithms

The results obtained above are averaged over 3 experiments. From the table above it is seen that HMinMax and HSum performed better in term of over all system throughput when compared to hostapd's Automatic Channel Selection algorithm. HSum algorithm performed better compared to HMinMax which comes from the fact that HSum algorithm considers the sum interfering signals where as HMinMax algorithm considers only maximum of the interfering signals. From these arguments we can conclude that HSum has better picture of interfering access points when compared to HMinMax algorithm. When it comes to 15 percentile throughput HMinMax and HSum performed better when compared to ACS algorithm. This can be attributed to the fact that ACS only considers the interface's active and busy time in the channel to calculate the interference and come up with the least interfering channel. HMinMax and HSum considers the overall interference environment by considering received signal index from each access point. Contrary to overall system throughput measurements, HMinMax performed better than HSum algorithm. This can be attributed to the fact that for each node HMinMax considers maximum interfering edge to replace the channel whereas for HSum the sum of all the interfering edges is considered. The channel assignment for each of the algorithm is shown in the table below:

Algorithm	Node20-7	Node20-14	Node19-4	Node14-8	Node9-7	Node9-14	Node7-13	Node1-13
ACS	1	11	6	6	6	6	6	6
HMinMax	11	1	1	6	6	6	11	1
HSum	11	6	1	6	6	1	11	6

Table 5.2: Channel assigned to the nodes for algorithms

The above table shows channel number assigned to nodes for the channel selection algorithms. As we see most of the nodes performing ACS are assigned the same channel. HMinMax and HSum have varied channel assignments. Better performance of HSum and HMinMax algorithm can be attributed to channel assignments of the nodes as interference reduces when the nearby access points are assigned orthogonal channels.

5.3 Comparing Inter-network cooperation with Intra-network cooperation

In this section we compare inter network and intra network cooperation. The following figure shows the eight node topology used. Two different networks are shown in the figure below in two different colors.

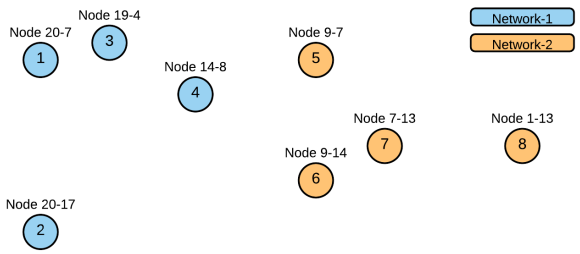

Figure 5.3: Grid topology showing Network-1 and Network-2

The nodes are logically divided into two networks using the *defGroup* functionality in OMF script. In intra-network cooperation each network independently execute the channel coordination algorithm without communicating the interference graph to other network. In inter-network cooperation controller of network-2 gets the interference graph of the nodes present in its network and transfers the interference information to controller in network-1. This is summarized in the figure below.

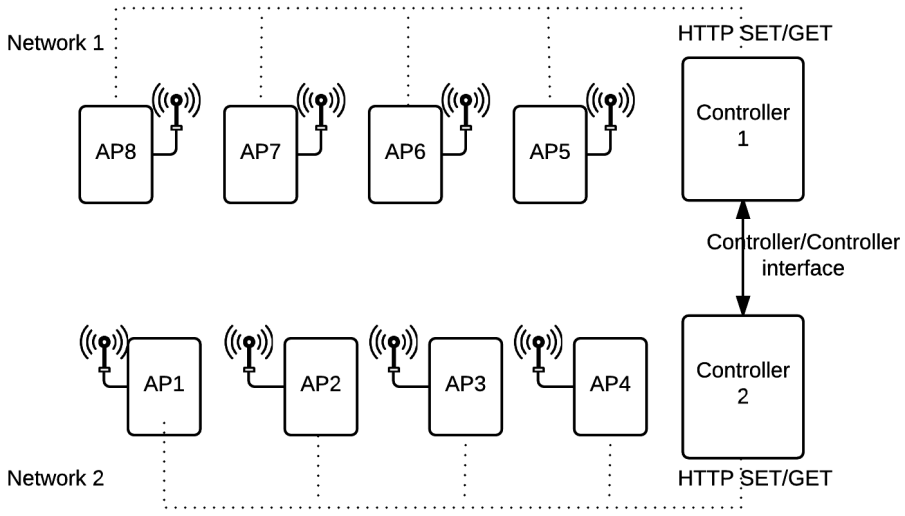

Figure 5.4: Inter-network cooperation mechanism

As shown in the figure controller-1 gets the interference graph of network-1 and controller-2 gets the interference graph of network-2. In the case of inter-network cooperation controller-2 after getting the interference graph for network-2 transfers the information to controller-1. Controller-2 after getting the information executes the channel selection algorithm. Note that two controller infrastructure was emulated in a single ORBIT console script for simplicity purpose.

Throughput measurements were performed comparing intra-network and inter-network cooperation with collocated access point and client as discussed before. Average system throughput for inter-network cooperation was 0.762Mbps and for inter-network cooperation the average system throughput was 1.42Mbps. Better performance of inter-network cooperation is due to the fact that inter-network cooperation considers the interference graph of both the networks before running the algorithm, where as intra-network cooperation considers only its own interference graph. The average system throughput increased by about 86% with inter-network cooperation. This result shows the importance of inter-network cooperation especially in regions where density of access points is high.

Note that ACS is used for initial channel assignment. Although the nodes are logically

grouped to form two different networks, ACS considers nodes in both the networks for channel assignment. The main aim of this section is to study the comparison of intra-network cooperation with inter-network cooperation. With all the nodes in carrier sense range, each node can extract the beacons from every other node in our experimental topology. Thus in the case on intra-network cooperation, even though each node in Network-A is in carrier sense of every other node in Network-B the algorithm was separately executed for each network.

Chapter 6

Conclusion and Future Work

This work motivates the use of seamless channel switching to improve WLAN system performance. It emphasizes channel switching ability of hostapd's CSA mechanism through a control API developed using HTTP infrastructure on top of hostapd. The ability of an access point to switch channels without bringing down the interface using the API developed is stressed upon. Three different channel selection algorithms are compared using throughput measurements. Through the results obtained, it was found that the channel coloring algorithms: HMinMax and HSum performed better than the hostapd's default channel selection algorithm ACS in term of average system throughput. This result shows the performance benefits of channel coloring algorithms as they use interference graph of the WLAN system. Next inter-network and intra-network cooperation was studied for channel cooperation. The results show the benefits of inter-network cooperation with better system throughput when compared to intra-network cooperation.

6.1 Future Work

6.1.1 Distributed Channel Selection Mechanism

The work in this thesis discusses the implementation of channel selection algorithm in a centralized manner. A mechanism to execute the algorithm in a distributed fashion is yet to be explored. Based on the interference graph, each node in the topology could decide its channel by independently running the distributed algorithm. The interference graph used to run the algorithm can either be provided by a centralized entity or the node itself could generate the interference graph based on different scanning

mechanisms.

6.1.2 Algorithm for Access Points in Carrier Sense range and Interference range

In the experiments performed all the nodes were in Carrier Sense range i.e they were able to extract the information in the becons received by all the nodes. Whereas nodes present in interference range will not be able to extract the beacons i.e they will not able to perform CSMA mechanism. There is a need for channel selection algorithm for the nodes present in carrier sense range and interference range. An algorithm for nodes in carrier sense and interference range needs to be formulated.

References

[1] Automatic channel selection(acs). Available: https://wireless.wiki.kernel.org/en/users/documentation/acs.

[2] Channel switch announcement - csa. Available: https://mrncciew.com/2014/10/29/cwap-channel-switch-announcement/.

[3] Gnu operating system. Available: https://www.gnu.org/software/libmicrohttpd/.

[4] Orbit testbed - winlab. Available: http://www.orbit-lab.org/.

[5] Wifi carrier sense. Available: http://www.revolutionwifi.net/revolutionwifi/2011/03/understanding-wi-fi-carrier-sense.html.

[6] Valentin Rakovic, Daniel Denkovski, Vladimir Atanasovski and Liljana Gavrilovska. Radio resource management based on radio environmental maps: Case of smart-wifi. *International Conference on Telecommunications (ICT)*, June. 2016.

[7] Akash Baid and Dipankar Raychaudhuri. Understanding channel selection dynamics in dense wi-fi networks. *IEEE Communications Magazine*, January. 2015.

[8] Arunesh Mishra, Suman Banerjee and William Arbaugh. Weighted coloring based channel assignment for wlans. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9:19–31, July. 2005.

[9] Arunesh Mishra, Vivek Shrivastava, Dheeraj Agarwal, Suman Banerjee and Samrat Ganguly. Distributed channel management in uncoordinated wireless environments. *MobiCom*, Sept. 2006.

[10] iPass. Wi-fi growth map, 2014. Available: https://www.ipass.com/wifi-growth-map/.

[11] Liqun Fu, Soung Chang Liew and Jianwei Huang. Safe carrier sensing range in csma network under physical interference model. *IEEE Transactions on Mobile Computing*, Jan. 2009.

[12] Gentoo Linux. Hostapd. Available: https://wiki.gentoo.org/wiki/Hostapd.

[13] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. *Wireless Communications and Networking Conference*, March. 2005.

[14] J.D. O'Sullivan, G.R. Daniels, T.M.P. Percival, D.I. Ostry, and J.F. Deane. Wireless lan, January 23 1996. US Patent 5,487,069.

[15] Vangelis Angelakis, Stefanos Papadakis and Vasilios A. Siris. Adjacent channel interference in 802.11a is harmful: Testbed validation of a simple quantification model. *IEEE Communications Magazine*, 49, March. 2011.

[16] Qualcomm. The wi-fi evolution an integral part of the wireless landscape, 2014. Available: `https://www.qualcomm.com/products/vive/evolution`.

[17] Dipankar Raychaudhuri and Akash Baid. Nascor: Network assisted spectrum coordination service for coexistence between heterogeneous radio systems. *The Institute of Electronics, Information and Communication Engineers*, Jan. 2014.

[18] Bhavneet Sidhu, Hardeep Singh and Amit Chhabra. Emerging wireless standards - wifi, zigbee and wimax. *World Academy of Science, Engineering and Technology*, Nov 2007.

[19] Shweta Sagari, Samuel Baysting, Dola Saha, Ivan Seskar, Wade Trappe and Dipankar Raychaudhuri. Coordinated dynamic spectrum management of lte-u and wi-fi networks. *IEEE International Symposium*, Dec. 2015.

[20] International Telecommunication Union. Propagation data and prediction models for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 900 mhz to 100 ghz. *ITU Radio communication Assembly*, 1997.