MOBILEFLOW: APPLYING SDN TO MOBILITY IN WIRELESS NETWORKS

A Thesis

by

RAGHDAH JAMEEL MAHMOOD AL-SHAIKHLI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Alexander Sprintson
Co-Chair of Committee,	Srinivas Shakkottai
Committee Member,	Radu Stoleru
Head of Department,	Chanan Singh

August 2014

Major Subject: Computer Engineering

ABSTRACT

Wireless technology has become an increasingly popular way for network access. Wireless networks provide efficient, reliable service; supporting a broad range of emerging applications including multimedia streaming and video conferencing.

Currently, there are two dominant technologies for providing wireless network access: cellular broadband networks and wireless local area networks (Wi-Fi). Cellular networks offer ubiquitous coverage, high reliability, and support mobility; yet such networks require expensive specialized equipment and expensive spectrum bands.

In contrast, Wi-Fi networks utilize unlicensed frequency bands; relying on commodity equipment. As a result, Wi-Fi infrastructure operational costs are lower than cellular network costs. Wi-Fi networks however, have limited coverage, do not support mobility, and are less reliable than cellular networks.

Recently, software-defined-networking architectures are gaining interest. The Software-Defined Networking (SDN) approach separates control (forwarding decisions) and data plane (packet processing). This approach provides an abstraction of a network switch and an interface for manipulating this abstraction with clear semantics. The SDN approach enables applications to control underlying network services without knowing the low-level details of specific network equipment. Thus, this approach allows network programming by modifying the behavior of the routers and switches to meet network application requirements.

This thesis introduces a reference architecture that supports user mobility through integration of the SDN technology into Wi-Fi networks. This project then implements a mobility manager application on top of an SDN controller to handle clients' handoff between access points. It proposes an algorithm for mobility prediction, allowing the network operator to minimize packet loss and delays during handoffs. Algorithm validation uses real data traces from the Texas A&M University network. Trace analysis was conducted to extract mobility patterns to build a prediction model which was implemented as an application in the SDN controller.

The approach was tested by measuring packet loss that was decreased by approximately nine times. Collected mobility traces were used to analyze our prediction model performance, whose accuracy reached 65% and 95% when selecting five users with Last-in-First-out scheme with a high- and low-load access point, respectively.

This research lays out groundwork for enhancing the functionality of WiFi networks, including mobility support, while maintaining their advantages in terms of lower cost, flexibility, and user of off-the-shelf components.

DEDICATION

To my family and friends

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor Dr. Alex Sprintson, for all the mentoring he has given me in this work and throughout my entire graduate school experience. During this period, I have grown a tremendous amount both in my technical aptitude and as a person. I would also like to sincerely thank Jasson Casey for his advice. I am very grateful for the help I received from Dr. Alla Kammerdiner, Muxi Yan, Trevor Olsen, Colton Chojnacki, Atin Ruia for this work. Finally, I want to thank Dr. Srinivas Shakkottai and Dr. Radu Stoleru for having served as my committee members.

NOMENCLATURE

TCP	Transport Control Protocol
IP	Internet Protocol
TLS	Transport Layer Security Protocol
SSL	Secure Socket Layer Protocol
SDN	Software Defined Networking
AAA	Authorization, Association, and Accounting
LVAP	Lightweight Virtual Access Point

TABLE OF CONTENTS

LIST OF FIGURES

Page

1. INTRODUCTION

1.1 Motivation

With the advent of smart phones and tablets, more and more people use wireless technology to connect to network services, including surfing the web, making Voice over IP calls, or streaming multimedia. However, with the increasing number of users comes the challenge of how to provide fast, efficient, and reliable service.

There are two dominant types of wireless network access: cellular networks (such as 4G and LTE and Wi-Fi networks. Due to the clients' access patterns, there is a need to support clients' mobility, i.e., provide efficient and reliable service during hand-off events. While cellular networks are very reliable and support clients' mobility, they are very expensive to install and operate. In contrast, Wi-Fi networks are easier to deploy and are less expensive, but they are limited in coverage as well as mobility support, resulting in a large delays and packet losses during mobility events.

Minimizing this delay and packet loss for mobile users is one of the major issues in Wi-Fi networks. A number of protocols were designed to achieve seamless mobility, and minimize the disruption to existing connections during the hand-offs [1] [2] [3]. Unfortunately, the existing solutions have several drawbacks, such as the need to make changes at the mobile device, and, as a result were not widely adopted by the industry.

This thesis makes the following contributions: First, it introduces a layer 2 SDN-based reference architecture that supports user mobility. Second, it proposes a methodology to predict user mobility and to integrate the mobility prediction algorithm in an SDN application. Third, it presents an experimental study that evaluates the performance gains (in terms of minimizing delays and packet losses) achieved by an SDN-based mobility management scheme and the mobility prediction algorithm.

1.2 OpenFlow

OpenFlow [4] is a rich SDN protocol that enables application development and control of feature rich network switches. OpenFlow provides the core switch specifications, configuration, test, and conformance. The OpenFlow data plane needs to support the following operations: field extraction, flow classification, and action application. OpenFlow currently supports only a handful of network protocols, but it is gaining popularity in industry and has attracted a significant attention from the research community.

OpenFlow [5] is an application layer protocol that runs over either TCP or SSL/TLS protocols. It is used for communication between an SDN controller such as NOX, POX, Flood Light, Ryu, and OpenFlow switches. OpenFlow separates the control plane from data plane, as shown in Figure 1, enables a centralized and granular control over the network, and allows services to be developed as applications on top of the controller, which adds a degree of flexibility to the network.

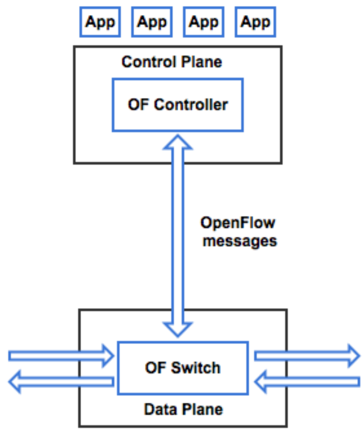

Figure 1: Control and Data planes separation

OpenFlow data plane contains the switch's physical and logical ports where packets are received and forwarded. It also includes the flow tables installed on the switch, the actions included in the flows to be applied to matched packets, and the classifiers, which are the fields used to match packets to flows [6].

Figure 2 depicts the basic OpenFlow datapath. When a switch receives a packet on one of its ports [6], it extracts information such as packet's arrival time and port ID as depicted in Figure 3. The switch then processes the packet's header, extracts the needed information that will be used to match the received packet to a flow entry as shown in Figure 4. OpenFlow uses the following fields from layer 2-3 protocols:

- Ingress port
- Ethernet source address
- Ethernet destination addresses
- Ether type
- VLAN id

- VLAN priority
- IP source address
- IP destination address
- IP protocol
- IP ToS bits
- TCP/UDP source port
- TCP/UDP destination port

  If a match is found, the action associated with the specified flow rule is executed.

These actions include *Forward, drop, modify, and enqueue*.

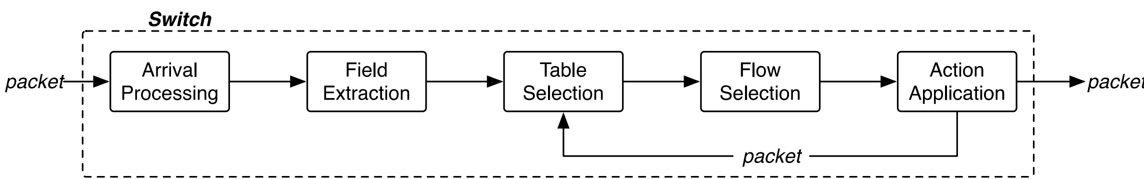

Figure 2: Packet processing in switch
(Courtesy of flowgrammable.org)

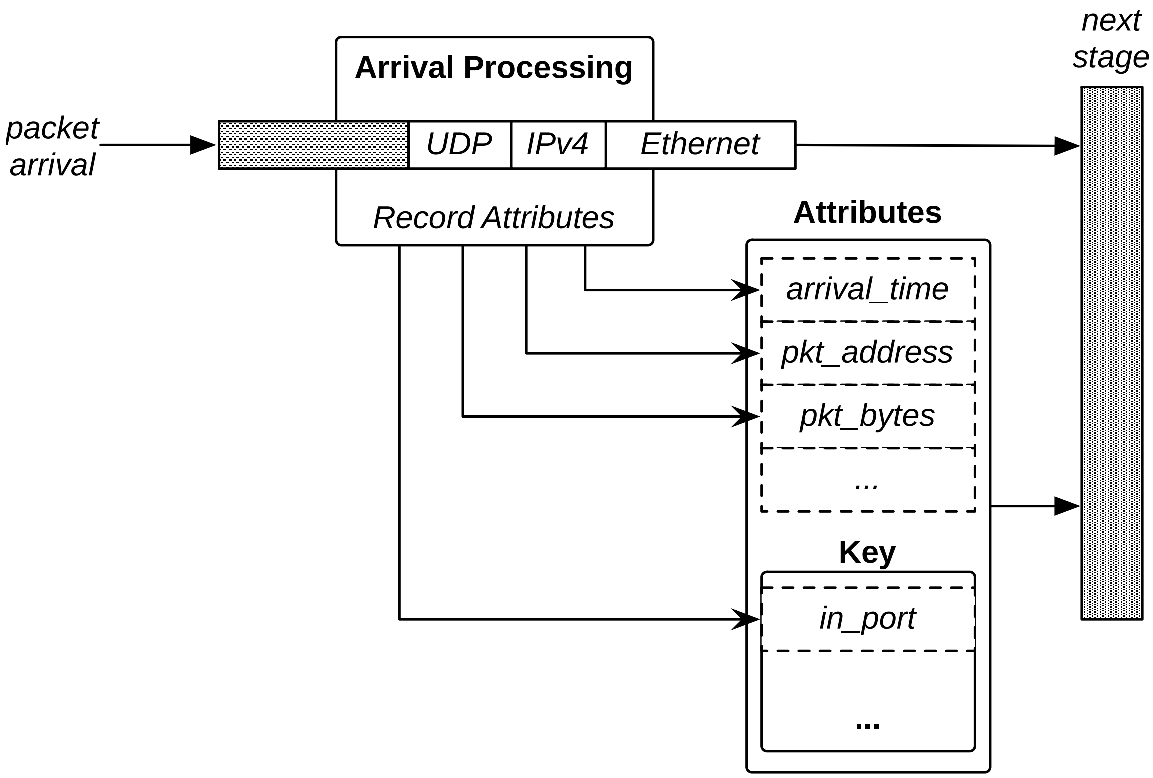

Figure 3: Attributes extraction upon packet arrival at switch
(Courtesy of flowgrammable.org)

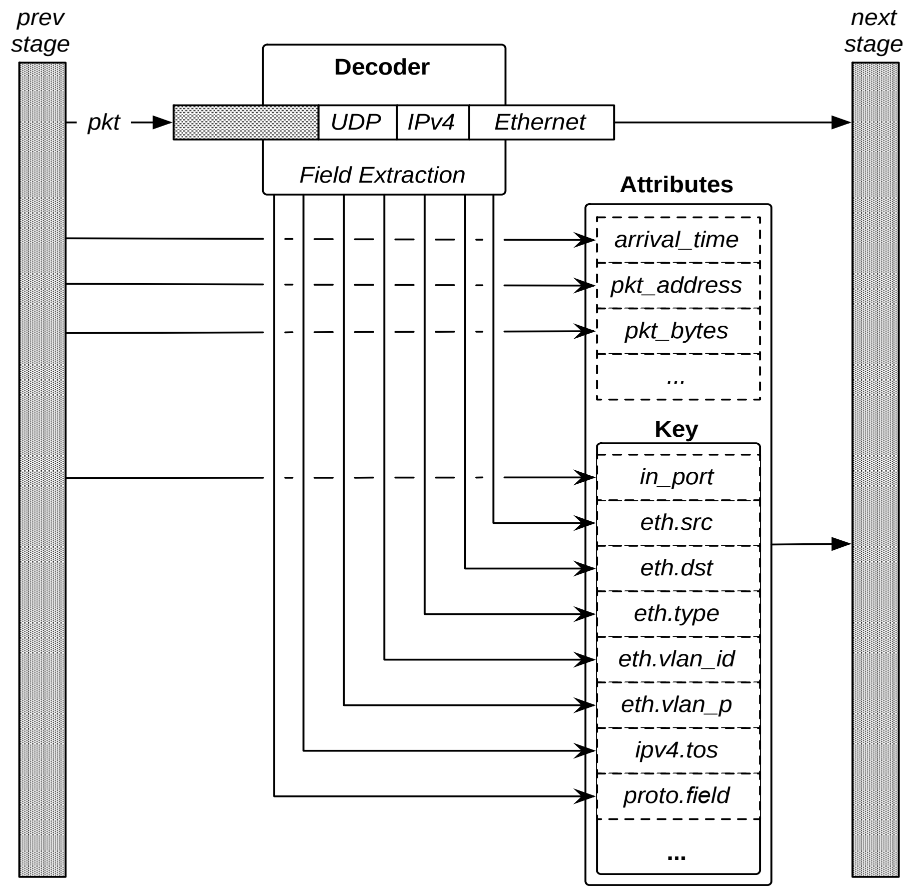

Figure 4: Field Extraction
(Courtesy of flowgrammable.org)

OpenFlow [7] defines a set of messages that controllers use to configure switches to perform the required actions on incoming packets. Some of the messages used between the controller and switches are:

- PacketIn

  PacketIn messages are used to send captured packets from switch to controller. Switch sends packets to the controller either when the action in the matched flow entry specifies sending the packet to the controller or when no match is found.

- PacketOut

  Controller sends PacketOut messages to switch to instruct the switch to perform action on packet; it can either carry the packet or indicate a buffered packet in the switch.

- FlowMod

  Controller can modify flow tables installed on switch by issuing a FlowMod message, which can be used to add, modify, or remove flow entries in the switch.

- PortStatus

  PortStatus message sent from switch to controller to indicate a change in port status. This message is sent when a port has been added, removed or some of the attributes associated with one of the ports has been changed.

- Stats/Multipart Request

  This message is sent from controller to switch to request information about individual flows, flow tables, ports, queues, meter, or group statistics. Upon sending this message controller will await a reply from the switch containing the requested information. This request could span multiple messages, and in that case, all messages will have the same transaction ID.

- Feature Request

  Controller sends this message to inquire about the switch capabilities. It is one of the first messages exchanged between controller and switch upon establishing connection. A response to this message is sent from switch to controller including switch features such as the number of packets that can be buffered in the switch, how many tables the switch can support, and type of connection.

- Experimenter/Vendor

  Either the controller or switch can initiate experimenter message. It is used to extend the OpenFlow protocol capabilities introducing vendor specific functionalities by adding proprietary messages.

1.3 Mobility

When a client connects to a network, it goes through multiple steps as shown in Figure 5. First, it sends a probe request to all access points in range. Upon receiving a reply from an access point, the client sends an authentication request. There, an authentication algorithm could be implemented or open authentication (i.e., no authentication) could be used, e.g., when authentication takes place later with an authentication, authorization and accounting server (AAA), etc. After successful authentication request and reply, the client sends an association request. Upon associating successfully, the client will gain access to the network.

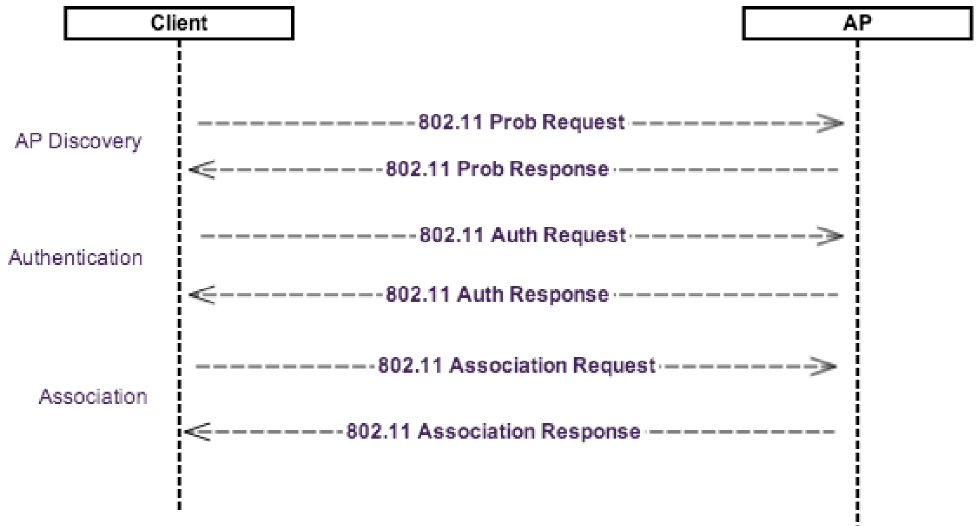

Figure 5: Establishing Connection to AP

In the case of using an authentication server as depicted in Figure 6, the client will be granted limited access to the network, where the AP will only allow the client to exchange traffic with the authentication server.

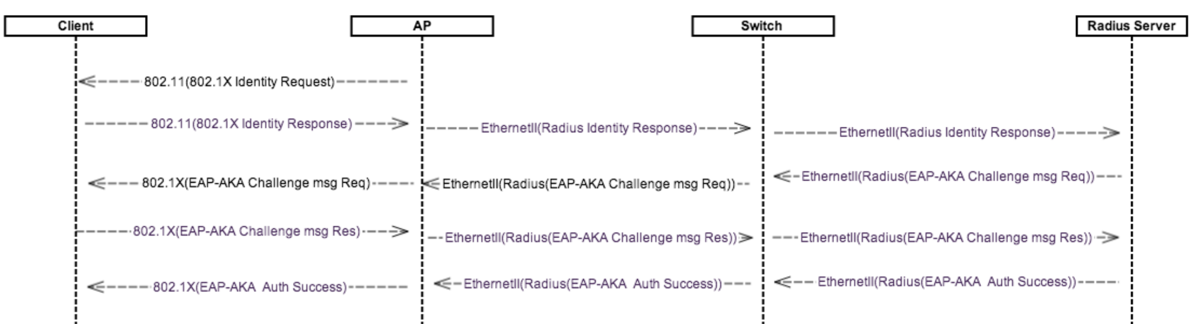

Figure 6: Authentication with AAA server

After successful authentication, client will obtain IP address as depicted in Figure 7, and can send and receive data through the network. Upon moving to a second AP in the network (see Figure 8), the user will run through the steps of authentication, authorization, and accounting again. This process introduces delay and might cause packet losses.

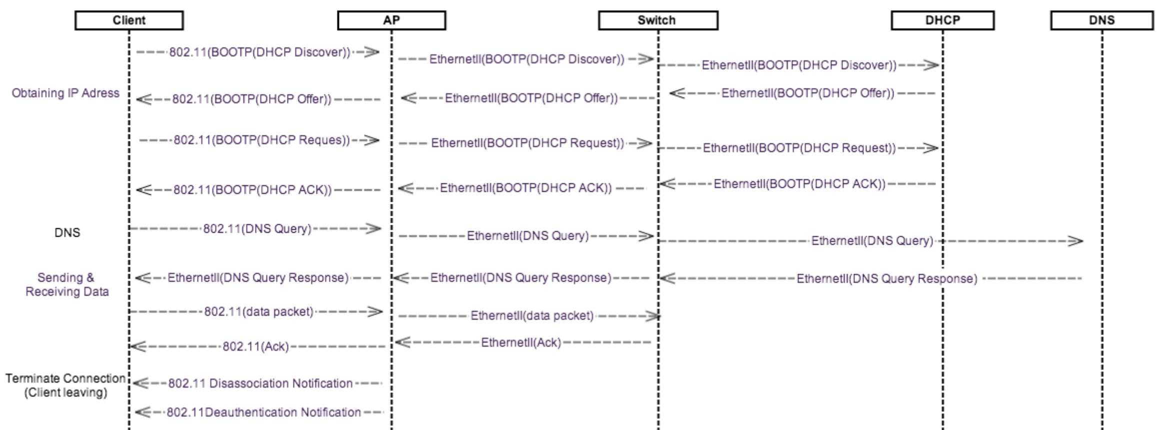

Figure 7: Obtaining IP Address, exchanging data, and terminating connection

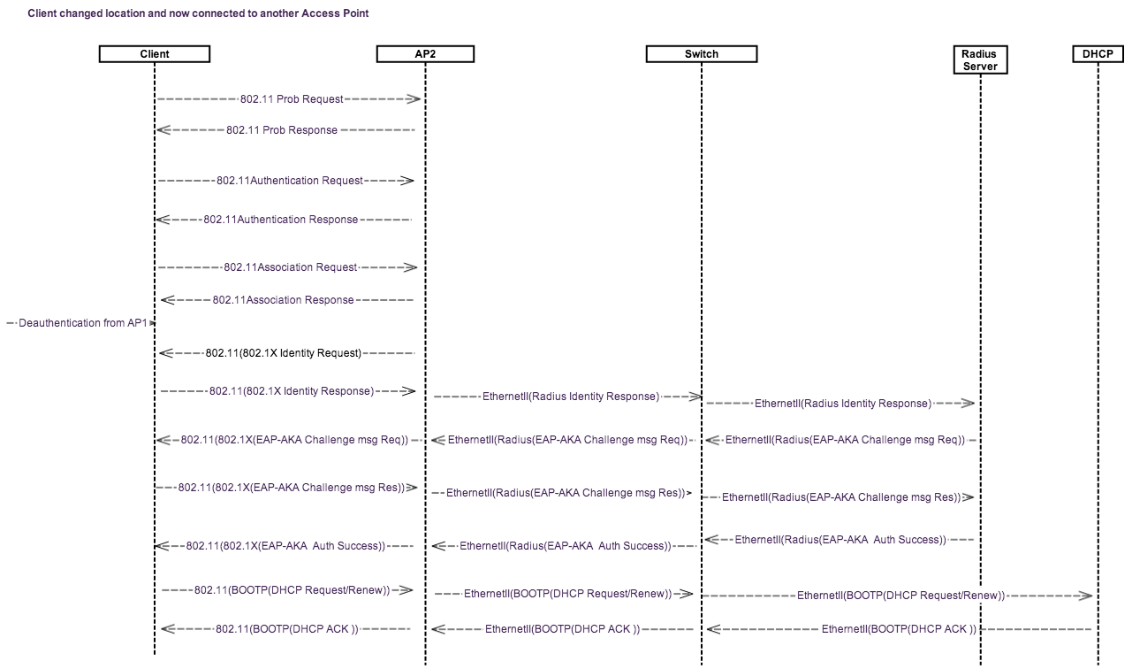

Figure 8: Moving to a new AP

Many of the available technologies tried to address mobility issue. Many of these technologies works on layer three (the Internet layer of the TCP/IP model) solutions. One example is Mobile IP (MIP) [1], where the mobile node (MN) receives a home address (HOA) from a home agent (HA). When the mobile node moves to a new to a new network, MN associates with a foreign agent (FA), receives a second IP address that is called care of address (COA) to use while in the network, and then FA establishes a tunnel with HA to forward packets to MN. MIP however, adds complexity, delay, as well as signaling overhead in the case of too many users and frequent handoffs [2], and it requires the client's involvement by installing software. Extensions to MIP have been introduced to minimize handoff times. Examples include proxy mobile IP (PMIP) [3], which incorporates the MIP functions in the AP. Although PMIP removes the need to install software on the client side, it increases the complexity of the network.

Other technologies for speeding up the handoff process work at layer two. They aim at shortening the time taken by the authentication since it accounts for most of the delay during the handoff process. For example, with 802.11 pre-authentication the user starts the authentication process before leaving the first access point as shown in Figure 9 taking advantage of the fact that in 802.11 clients can authenticate with multiple APs before association with one.

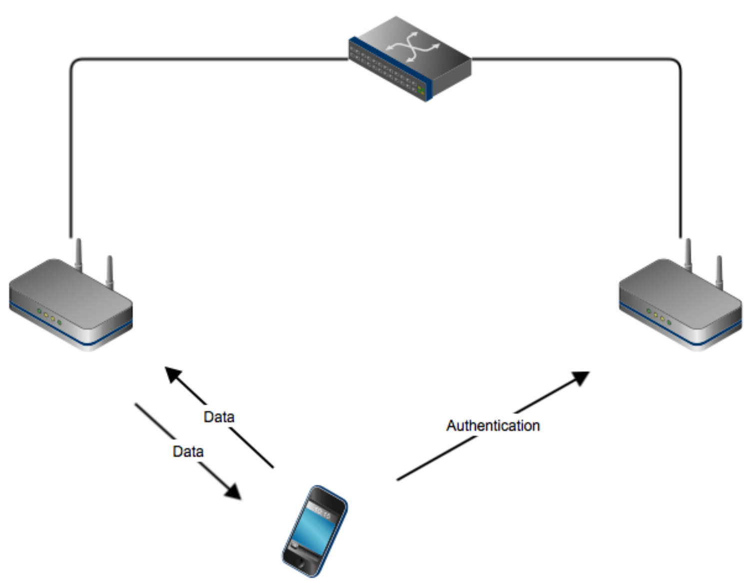

Figure 9: 802.11 Preauthentication

Other schemes employ a make-before-break approach that tries to associate with a second AP before disconnecting from the first one. This approach requires the client to have multiple radios, enabling him/her to connect to multiple access points at the same time.

Another approach is 802.11i preauthentication [8] that minimizes the time associated with 802.11X authentication when there is an authentication server involved. Because client can only associate with one AP at a time, authentication frames exchanged with the new AP is sent through the current AP. An Extensible Authentication Protocol over LAN (EAPOL) frame is initiated from client with Ethertype set to 88C7 to indicate 802.11i preauthentication frame. When an AP receives a frame of this type it will recognize it as an 802.11i preauthentication frame and send it over the network to the second AP. During this operation, the client will continue to exchange data through the first AP.

Our approach also works on the data link layer, but it incorporates the SDN technology as well as a prediction algorithm to find a mobility pattern that will enable a certain degree of accuracy in predicting user's movement.

1.4 Related Work

Previous studies have explored applications of the OpenFlow protocols in wireless networks, [9], [10], and [11]. These papers use OpenRoads, a platform that facilitates innovation in wireless production networks and supports deployment of third-party applications. OpenRoad can be used to construct a working example of architecture as well as applications to support mobility in a Wi-Fi network, and vertical handover between Wi-Fi and WiMAX networks. The architecture and examples presented in [9], [10], and [11] require multiple radios at each mobile device, that allow a user to associate with more than one AP at the same time. The authors were able to test multiple mobility managers, most notably bicast/tricast (N-cast) [12], and Hoolock [13].

N-cast [12] is a mobility manager that uses bi-cast and tri-cast to duplicate traffic flows. The application was built around the assumption that devices will have multiple radios of the same type in the future. N-casting duplicates packets sent to the user device, which enables it to receive packets through multiple radios. With this method, N-cast was able to achieve a seamless handover.

Hoolock [13] is another mobility manager that has been implemented on top of OpenRoad. Hoolock aims to achieve a seamless handover between access points, minimizing latency and packet loss during the transition. Hoolock made the same assumption as N-cast [12] about devices with multiple radios. In their design, one radio will be transmitting data, while the second one monitors the signal power. Upon reaching a predetermined threshold, the client will issue a request to the controller, connecting to the new access point using the second radio. The controller will reroute traffic received through the first AP to go through the second AP. Lastly, after waiting for two RTTs the client will de-associate from the first AP, and continue to receive traffic through the second AP. Therefore, during a mobility event, the client will be using two radios to send and receive packets, thus consuming more power, and will be receiving duplicate packets on both interfaces before the client finally disconnects from the first AP, where packets coming from the second AP will be buffered in the mobile device's interface connected to that AP.

Reference [14] makes another attempt to apply SDN to wireless networks by introducing Odin, an SDN framework that enables network operators to implement services as applications. To that end, one of Odin's goals is to shield the programmer from the complexities of WLAN by using abstraction, so that the programmer will not have to deal with network events such as association, reasocciation, and authentication.

One of the applications available on the Odin platform is a mobility manager, where the metric used is signal strength. To gain access to the network, the user device connects to a Light Weight Virtual Access Point (LVAP), and when it moves, the first physical AP removes the LVAP associated with the client, and the selected physical AP with the strongest received signal creates the LVAP for the user again. This operation eliminates the need for re-association messages, and, at the same time, takes decision making away from the user.

Another reference that uses the idea of virtual access points (VAPs) is [15]. In this paper, Dely et al. introduces CloudMac, a new distributed architecture for wireless networks. The difference between CloudMac and Odin is that Odin's LVAP resides inside physical APs, while CloudMac's VAPs reside in the cloud.

All the previous works showed that using OpenFlow in wireless networks leads to several benefits, including improved performance.

This thesis differs from the other aforementioned works in that it extends the Open flow functionality in order to be able to report events, manage mobility, and enable a centralized control of the network through an OpenFlow application. We demonstrate the capability of our approach by implementing a mobility prediction application that further improves performance by enabling the network to anticipate the mobility events and take proactive actions to support mobility before the actual mobility event occurs. We show that our approach results in a substantial improvement in terms of network performance.

Several prior works have focused on developing predictive models for wireless networks. For example, reference [16] proposes a mobility model that takes into account the characteristics of real wireless LAN environment, and extracts the spatial and temporal parameters, in order to use them in network analysis and building simulation scenarios. The researchers gathered mobility traces using two methods, first using SNMP protocols, and then through syslog messages. This model depends on the association event to determine the client's location. The client goes back and forth between two states (i) on state when associated with an access point and (ii) off state upon leaving the network or when moving to another access point..

The authors in [17] propose a movement prediction system for users of a cellular network using pre-computed patterns using data from OpenMobileNetwork platform. They extended a prediction algorithm proposed by Yavas et al. [18] to be able to preselect patterns based on contextual data such as time of the day. The used the management architecture presented in [19] to obtain this data. Management architecture gathers data from sources such as mobile stations and sends relative data to a central management server. To prove the effectiveness of their method, authors created a visualizer that is a map showing user movement at cellular level. OpenMobileNetwork achieved an accuracy of 30 to 50 percent due to insufficient data; authors argued that obtaining more data will increase the accuracy rate of the platform.

In reference [20] researchers present a mobility model based on real traces collected from the University of Dartmouth campus. The model shows the usage, distribution, hourly arrivals, and departures to and from AP. Since a large number of the trace observations are from computers (74%) rather than hand held devices according to previous study, the study did not depend on individual users' movement. This paper does not model the path taken by the client, nor try to predict the user movement, and it does not provide an evaluation of the model efficiency.

2. DESIGN

We implement our idea by making extensions to OpenFlow, one of the major SDN standards in use today. OpenFlow enables a central controller to configure the behaviors of the switches in the network. We take advantage of this feature of OpenFlow to enable fast mobile station handoff.

2.1 Mobility Support with OpenFlow

We propose the following design of fast mobile station handoff with OpenFlow. The network topology is as shown in Figure 10. The switches in the network are all OpenFlow enabled switches, controlled by the same controller. First, the access points, which are also OpenFlow enabled, send event reports to the controller. Such event reports contain the MLME (MAC Layer Management Entity) events associated with specific mobile station on the access point side, including:

- Probe request
- Authentication
- Deauthentication
- Association
- Reasociation
- Disassociation

The OpenFlow controller makes prediction of each mobile station based on its information of these events associated with the corresponding mobile station. Once the controller determines a handoff is about to happen, it will use OpenFlow protocol to modify the traffic flow whose destination is the mobile station. Such modification will cause the flow to be forwarded in multiple routes, one to the access point with which the mobile station is currently associated, and the other to the access point that is going to be associated with based on the prediction. If the prediction is successful, it will guarantee the minimum packet loss during the handoff.

Our design has certain advantages. First, the IEEE 802.11 driver on the mobile station does not need to be modified at all. Second, the modifications of the access point, OpenFlow switch, and OpenFlow protocol are small.

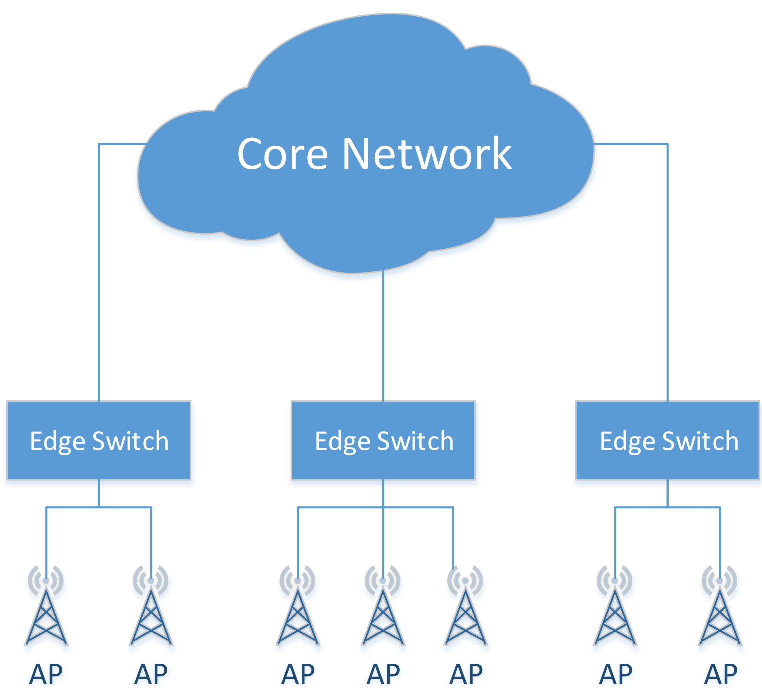

Figure 10: Network Topology

2.2 OpenFlow Extensions

In order to enable our design presented above, we need to make certain extensions to the current OpenFlow standard. This includes extensions on two major parts of OpenFlow: dataplane and OpenFlow protocol. We will describe the extensions in the following subsections.

a) OpenFlow dataplane extension

In common settings, instead of being an independent device, an 802.11 radio interface is usually attached to a switch, which connects the radio interface to other parts of the network with wired links. In our design, the switch to which the radio interface is attached must be an OpenFlow enabled switch. The switch has traditional physical ports supported by OpenFlow, e.g. Ethernet ports. In addition, the radio interface attached to the switch is also considered as a physical port. Each access point created over the radio interface is considered as a logical port attached to the physical port. Each radio interface and access point has its own port configurations. Such design is shown in Figure 11.

In this way, packets from a mobile station will be processed by OpenFlow datapath with the ingress port being the logical point that the mobile station is associated with, and the packets to a mobile station will be forwarded to the corresponding logical port. In this way, we add support of 802.11 packet processing to the OpenFlow datapath.

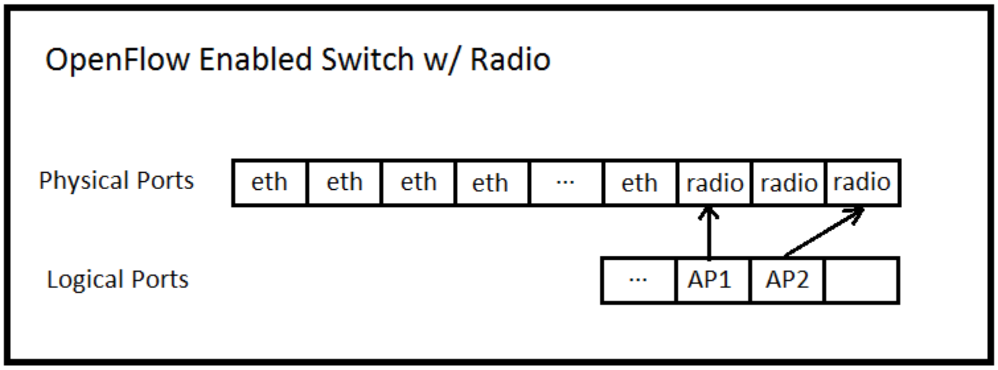

Figure 11: OpenFlow Enabled Switch with radios

b) OpenFlow protocol extension

OpenFlow protocol is used for control plane communication between the controller and the switch. In our design, we need certain extensions to the current OpenFlow protocol. Specifically, we need to define additional messages in OpenFlow protocol. Such extension is made possible by the Experimenter type message in current OpenFlow standard. Experimenter message supports extension on OpenFlow messages by allowing the designer to define the Experimenter ID and Experimenter Type of the new message.

The newly defined messages and their OpenFlow frame formats are described as follows.

- ofp_ap_event_report

This is a message sent from the switch to the controller, which indicates the controller the occurrence of an event associated with certain mobile station. The message structure is shown in Figure 12.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |      type     |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              xid                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Experimenter id                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Experimenter type                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            port_no
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          MAC address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           MAC address          |      Event    |      pad      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: ofp_ap_event_report message structure

- ofp_ap_set_asynchronous / ofp_ap_get_asynchronous

The ofp_ap_set_asynchronous message is sent from the controller to the switch to configure which types of events are of interest to the controller. The switch will not send ofp_ap_event_report of certain type to the controller if these types are not configured to be of interest. ofp_ap_set_asynchronous message is used to acquire the current configuration from the switch. The message structure is shown in Figure 13.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |       type      |             Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              xid                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Experimenter id                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Experimenter type                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            port_no
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Event Of Interest - Equal/Master                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Event Of Interest - Slave                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: ofp_ap_set_asynchronous message structure

- ofp_ap_set_config / ofp_ap_get_config

These two messages sets/gets the current configuration for a radio interface or an access point. The message structure is shown in Figure 14.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |      type     |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               xid                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Experimenter id                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Experimenter type                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               ssid                            
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                           (24 octets)                         .
.                               ...                             .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                ssid                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Radio interface port                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  auth_method  |   auth_group  |      pad      |      pad      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: ofp_ap_set_config message structure

- ofp_ap_set_auth_profile / ofp_ap_get_auth_profile

These two messages are sent from the controller to the switching to add/remove/modify/query an authentication profile. An authentication profile specifies the authentication protocol to be used during the authentication phase of 802.11 mobile station association, as well as the address of the authentication server. The message structure is shown in Figure 15.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |      type     |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              xid                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Experimenter id                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Experimenter type                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| auth_protocol |    command    |              pad              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              name
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                           (24 octets)                         .
.                              ...                              .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                               name                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         server_address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                          (120 octets)                         .
.                              ...                              .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                          server_address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: ofp_ap_set_auth_profile message structure

- ofp_ap_set_auth_group / ofp_ap_get_auth_group

These two messages are sent from the controller to the switch to add/remove/modify/query an authentication group. An authentication group contains a list of authentication profiles to be used. An access point will authenticate a mobile station with each of the profiles listed in its group in order. The message structure is shown in Figure 16.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |      type     |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              xid                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Experimenter id                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Experimenter type                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| auth_protocol |    command    |              pad              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              name
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                           (24 octets)                         .
.                              ...                              .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                               name                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Authentication Profile Name [0] (32 octets)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Authentication Profile Name [1] (32 octets)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: ofp_ap_set_auth_group message structure

3. MOBILITY TRACES ANALYSIS

We gathered mobility traces from the Texas A&M University network. Obtained data from log file generated by Aruba 7210 mobility manager. A sample of an event seen in the log file is shown below in Figure 17

```
Nov 30 07:43:21 2013 aruba2-7210 authmgr[3502]: <522008> <NOTI> <aruba2-7210
10.19.10.8>  User Authentication Successful: username=yykk MAC=04:15:74:e3:65:c9
IP=10.23.10.21 role=tamulink-wpa VLAN=222 AP=ar-1 SSID=tamulink-wpa AAA
profile=TAMU auth method=802.1x auth server=csce-radius-eap-1
```

Figure 17: Authentication with radius server event as shown in the log file

Events in the log include information such as Date and time stamp (e.g., Nov 30 07:43:21 2013), Error location (the specific module location that generated this log e.g., authmgr[3502]). Also an Error number that is unique e.g.,522008 represent an user authentication successful message, Severity level, that specify what type of the logged event, for example NOTIFICATION , the AP Mac and IP address , and Message text , which is the rest of the message.

Other events obtained from the log file are illustrated by Figure 18, where each event includes a list of information such time, error location, which access point involved, etc.

Registering AP

Time	Error location	Error Number	Severity	AP name	AP IP	AP Mac	mode	max clients

Authentication Request

Time	Error location	Error Number	Severity	AP name	AP IP	AP Mac	Client's Mac	Reason

Association Request

Time	Error location	Error Number	Severity	AP name	AP IP	AP Mac	Client's IP	Client's Mac

Association Successful

Time	Error location	Error Number	Severity	AP name	AP IP	AP Mac	Client's IP	Client's Mac

Authentication Successful

Time	Error location	Error Number	Severity	AP name	Client's IP	Client's Mac	username	SSID	VLAN	Auth. Server

Authentication Failed

Time	Error location	Error Number	Severity	Client's Mac	username	Auth. Server

Deauthentication

Time	Error location	Error Number	Severity	AP name	AP IP	AP Mac	Client's Mac	Reason

Disassociation

Time	Error location	Error Number	Severity	AP name	AP IP	AP Mac	Client's IP	Client's Mac

Figure 18: The Chart shows the information included in each of the received events

We examined the log files, first extracting information like Authentication request, Deauthentication, and Disassociation, and then anonymized the data, substituting Mac addresses with IDs. We processed the data sets using statistical analysis tools like Stata and R, and collected statistics in order to understand the current data.

We calculated the usage percentage for the number of users per Access Point, and computed dwell times for users. We identified an event as a mobility event if the transition between two access points took less than two seconds. In the data, we observed that the majority of users tend to stay connected to one access point for less than 100 seconds. As shown in Figure 19, users with small dwell times are more likely to move to another AP. On the other hand, users who end up leaving the network shown to have a different distribution as shown in the dwell time histogram in Figure 20.

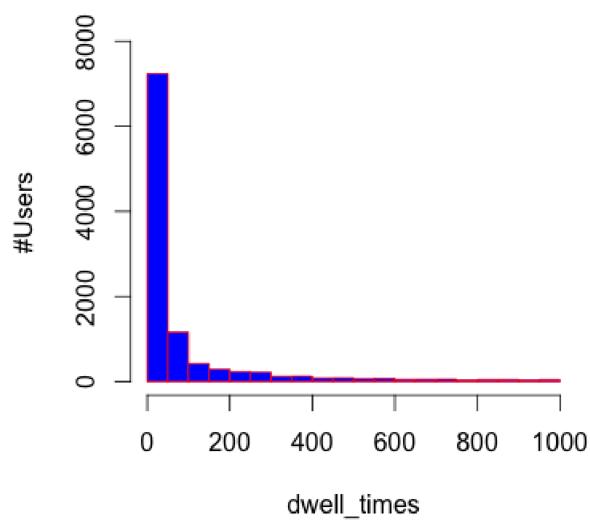

Figure 19: Dwell time histogram for moving user

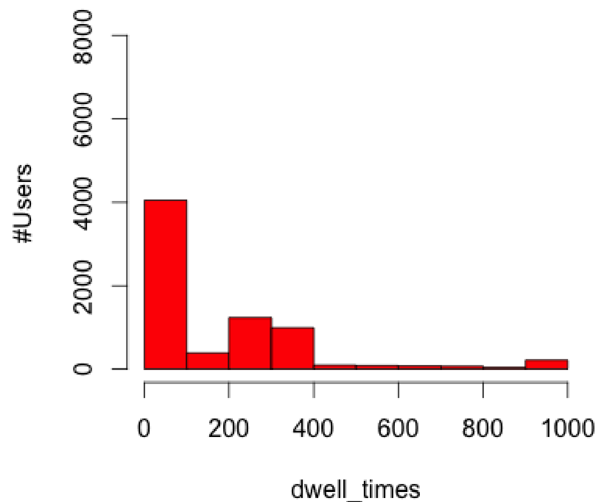

Figure 20: Dwell time histogram for users who end up leaving the system

We implemented a selective algorithm that sorts users connected to the AP based on their connection time, and then selects users that are most likely to move next from this sorted list. In order to keep the model generalized, we did not use our knowledge of APs locations as a factor in the prediction model. With our selective algorithm we started by applying a First in First out (FIFO) approach, where a user who joined first is the one most likely to move out first. We tested this prediction scheme using the collected mobility data to compare performance. We assigned a score for each prediction result, where we added a plus one in case the prediction was correct, and minus one in case of a wrong prediction. This method showed a low accuracy level when tested on an AP with high and average load. We compared the results when picking different number of users each time (from one to five) who are most likely to move as shown in Figures 21, 22, and 23. Applying this method yielded good prediction results only when tested on AP with less number of users. As the number of users connected to the AP grows, prediction accuracy dropped significantly.

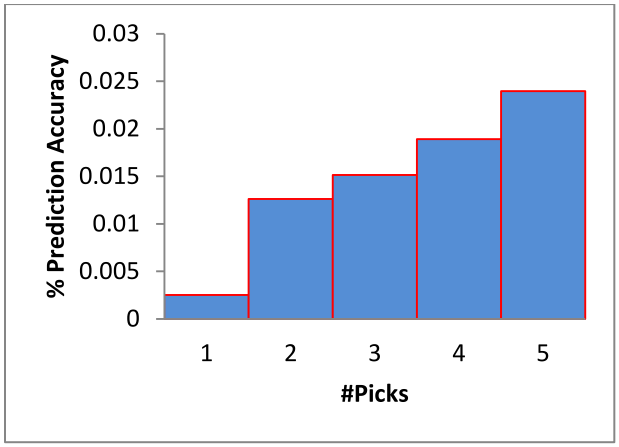

Figure 21: Prediction accuracy increases when increasing the number of users selected from a high load AP with First In First Out approach.

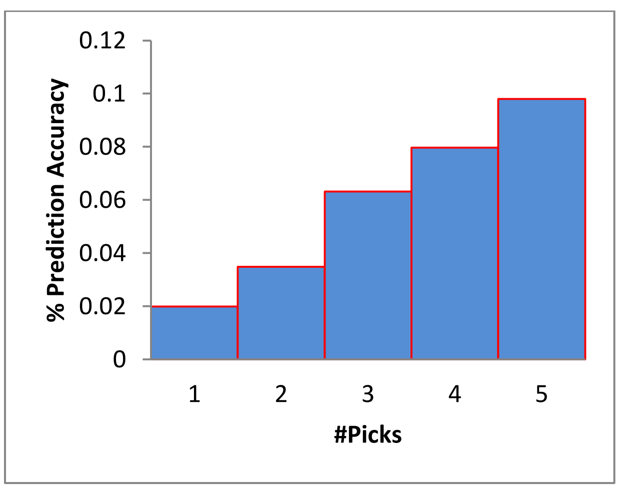

Figure 22: Prediction accuracy increases when increasing the number of users selected from an Average load AP with First In First Out approach.

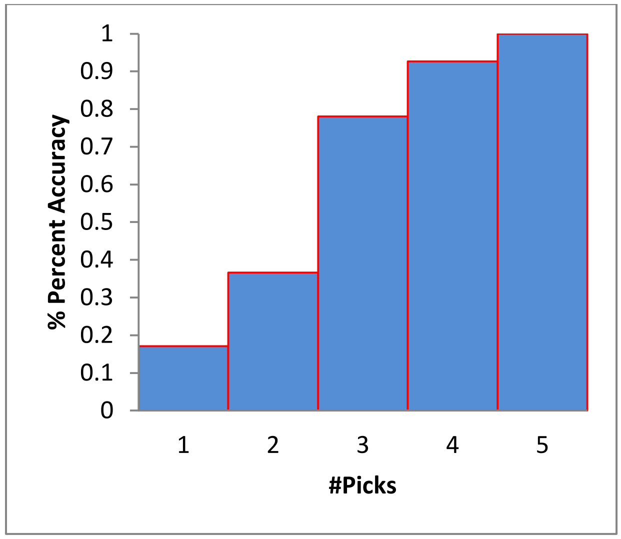

Figure 23: Prediction accuracy increases when increasing the number of users selected from a low load AP with First In First Out approach.

The other prediction scheme we tested was Last in First out, where the user joined last are the one most likely to leave, this method showed an improved results. As with the previous method, we tested it on three APs with different loads, and compared picking different number of users as in Figure 24, 25, 26.

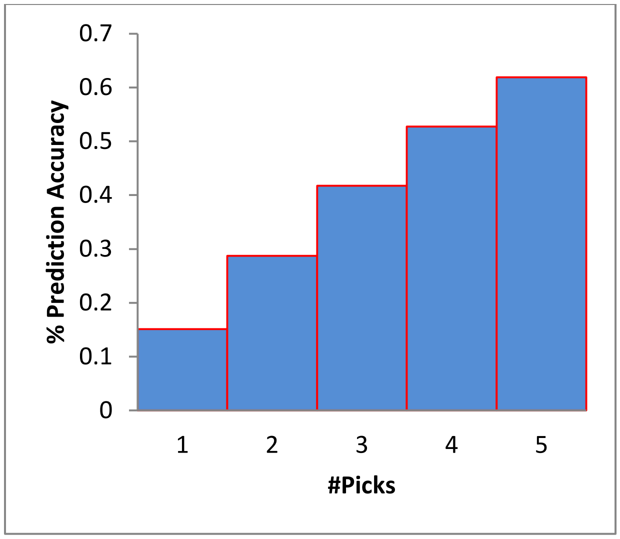

Figure 24: Prediction accuracy increases when using a Last In First Out approach and upon increasing the number of users selected from a high load AP.

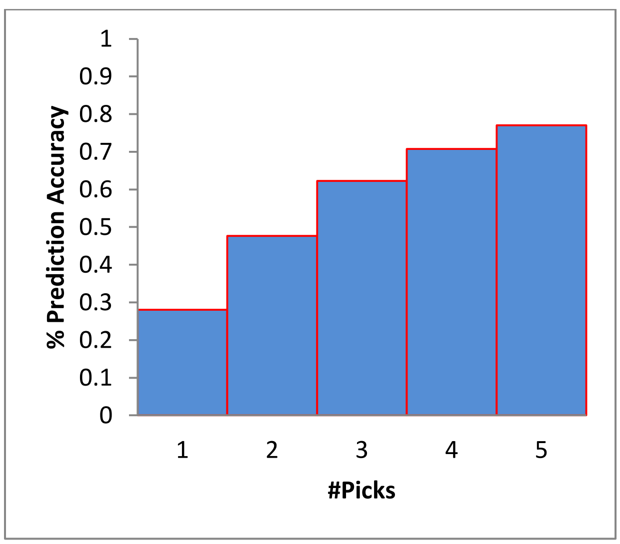

Figure 25: Prediction accuracy increases when using a Last In First Out approach and upon increasing the number of users selected from an average load AP.

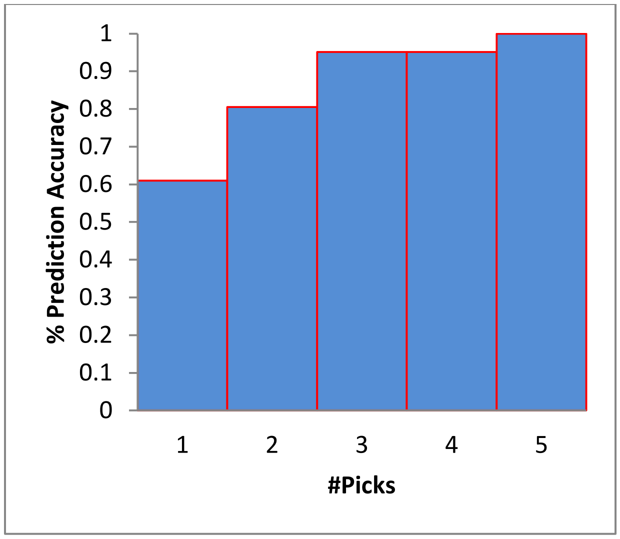

Figure 26: Prediction accuracy increases when using a Last In First Out approach and upon increasing the number of users selected from a low load AP.

4. IMPLEMENTATION AND RESULTS

For implementing OpenFlow in wireless access points, we installed an OpenWRT open platform for embedded device that allows extensions with OpenFlow extensions on TP-LINK TL-WR1043ND V2 with 64MB of RAM. For the controller we chose Ryu because of its support for OpenFlow version 1.3.1. We tested packet loss and latency with the following topology shown in Figure 27.

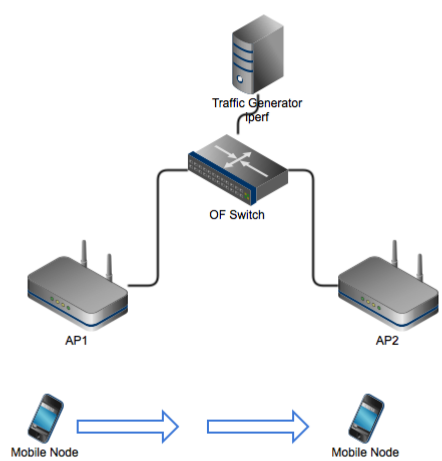

Figure 27: Test topology

Iperf was installed on virtual machine, used to generate traffic, and to test packet loss and throughput with UDP and TCP packets. Traffic was directed from the iperf traffic generator to the mobile node (IPhone 5S with iperf client installed), while walking between two APs. We run the test for 120 seconds; results were recorded every two seconds as shown in Figure 28 for UDP, where using OpenFlow resulted in an improved packet loss rate from user's perspective, compared to traditional wifi network without OpenFlow, as shown below.

	With OF	Without OF
Packet Loss	0.20% - 0.33%	1.5% - 3%

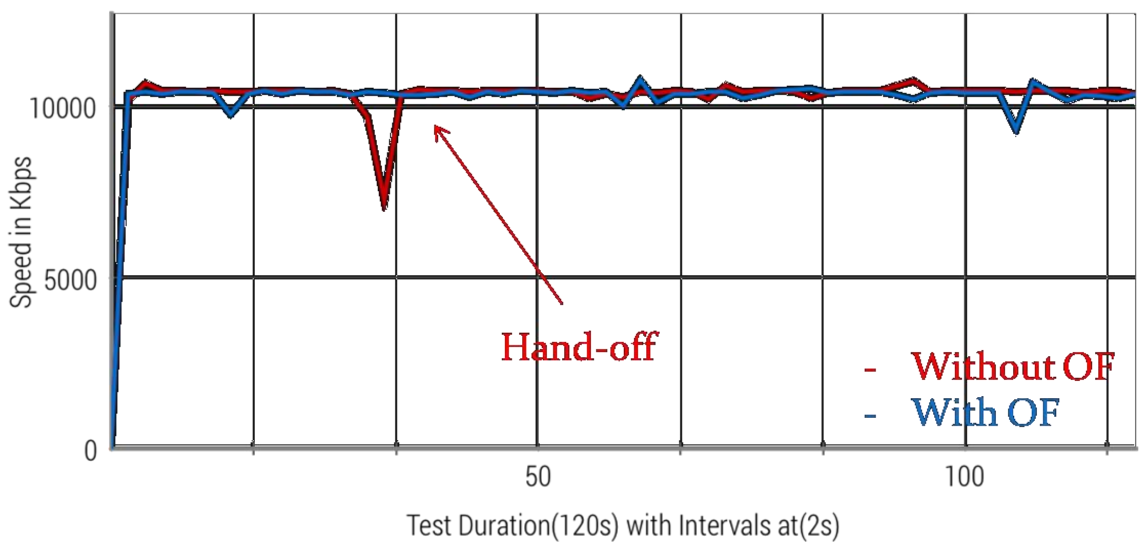

Figure 28: Test result using UDP packets

Results were also positive when tested with TCP packets. The average throughput when OpenFlow were used was 30820.33 Kbps, and 28450.53 Kbps when tested without OpenFlow, as shown in Figure 29.

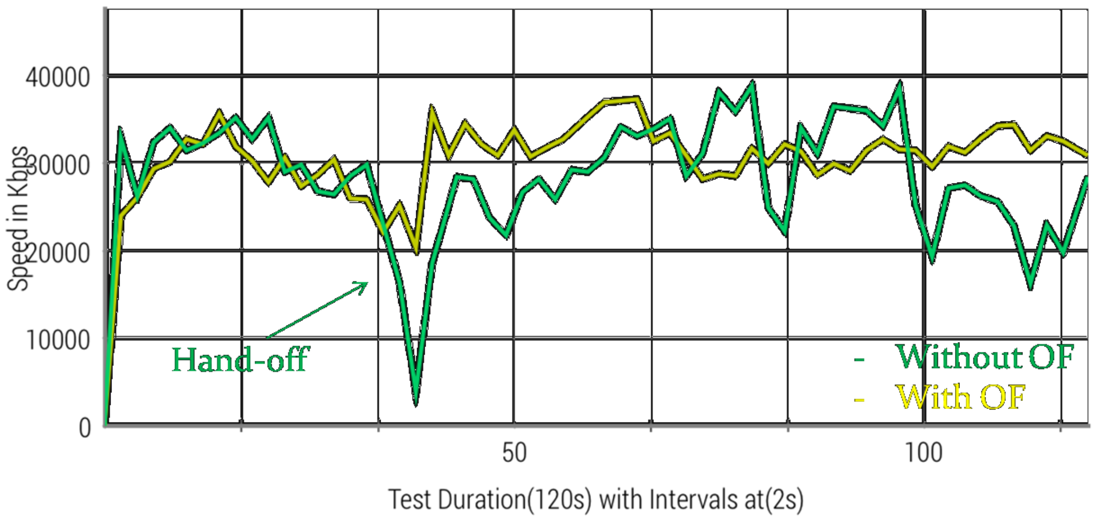

Figure 29: Test result using TCP packets

5. CONCLUSION

According to the obtained results, using OpenFlow can have a positive impact on network performance in regards to packet loss rate from the user's perspective. In addition, the use of prediction when duplicating flows helps to improve packet loss rate in the network in general by duplicating the packets to a targeted AP instead of sending the packets to all APs in range.

To improve the obtained results, more work has to be done on extensions in order to be able to report more AP related events to the controller. Furthermore, refining the used prediction model can help obtain more accurate prediction results, which will improve network performance and help enhance users' experience in the network. As for the testing part, using more APs will help to improve the accuracy of the obtained results, and with more APs, prediction algorithm can be tested in a practical setting with real time data, instead of depending on previously collected data to measure accuracy.

REFERENCES

[1]P. Zave, J. Rexford, "The Design Space of Network Mobility", in H. Haddadi, O. Bonaventure (Eds.), Recent Advances in Networking, (2013), pp. 379-412.

[2]Hashim, Aisha HA, Farhat Anwar, Shaffiah Mohd, and Hatina Liyakthalikh. "Mobility Issues in Hierarchical Mobile IP." In IEEE 3rd International Conference SETIT. 2005.

[3]Zhang, Xiaowei, Javier Gomez Castellanos, and Andrew T. Campbell. "P-MIP: paging in mobile IP." In Proceedings of the 4th ACM international workshop on Wireless mobile multimedia, pp. 44-54. ACM, 2001.

[4] Tiwari , Vivek. "Chapter 2 - What is SDN." In SDN and OpenFlow for beginners with hands on labs. Northville: M.M. D.D. Multimedia LLC, 2013.

[5]Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peter-son, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2):69-74, 2008.

[6] Open Flow protocol description. Retrieved June 2, 2014 from http://flowgrammable.org/sdn/openflow

[7] OpenFlow Switch Specification – Version 1.3.1 (Wire Protocol 0x04). Retrieved May 5, 2014 from https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

[8] Gast, Matthew. "Chapter 8 - Management Operations." In 802.11 wireless networks: the definitive guide. 2nd ed. Sebastopol, CA: O'Reilly, 2005.

[9] Yap, Kok-Kiong, Rob Sherwood, Masayoshi Kobayashi, Te-Yuan Huang, Michael Chan, Nikhil Handigol, Nick McKeown, and Guru Parulkar. "Blueprint for introducing innovation into wireless mobile networks." In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, pp. 25-32. ACM, 2010

[10] Yap, Kok-Kiong, Masayoshi Kobayashi, Rob Sherwood, Te-Yuan Huang, Michael Chan, Nikhil Handigol, and Nick McKeown. "OpenRoads: Empowering research in mobile networks." ACM SIGCOMM Computer Communication Review 40, no. 1 (2010): 125-126.

[11] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown, "The stanford openroads deployment," in Proc. ACM WiNTECH 2009, Beijing, China, 2009, pp. 59-66

[12] Yap, Kok-Kiong, Te-Yuan Huang, Masayoshi Kobayashi, Michael Chan, Rob Sherwood, Guru Parulkar, and Nick McKeown. "Lossless Handover with n-casting between WiFi-WiMAX on OpenRoads." ACM Mobicom (Demo) 12, no. 3 (2009): 40-52.

[13] Handigol, Nikhil, and Wei Wei. "Hoolock: Seamless Mobility in OpenFlow-enabled Wireless Access Networks." Class report, from Stanford University, Stanford, California , December 12, 2008.

[14] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. 2012. 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.

[15] Dely, P.; Vestin, J.; Kassler, A.; Bayer, N.; Einsiedler, H.; Peylo, C., "CloudMAC — An OpenFlow based architecture for 802.11 MAC layer processing in the cloud," Globecom Workshops (GC Wkshps), 2012 IEEE , vol., no., pp.186,191, 3-7 Dec. 2012

[16] Tuduce, Cristian, and Thomas Gross. "A mobility model based on wlan traces and its validation." In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 1, pp. 664-674. IEEE, 2005.

[17] Göndör, S.; Uzun, A.; Rohrmann, T.; Tan, J. and Henniges, R. Predicting User Mobility in Mobile Radio Networks to Proactively Anticipate Traffic Hotspots. In Proceedings of the 6th International Conference on Mobile Wireless Middleware, Operating Systems, and Applications, MOBILWARE 2013, IEEE, Bologna, Italy, 11/2013.

[18] Yavaş, Gökhan, Dimitrios Katsaros, Özgür Ulusoy, and Yannis Manolopoulos. "A data mining approach for location prediction in mobile environments." Data & Knowledge Engineering 54, no. 2 (2005): 121-146.

[19] Gondor, S., Abdulbaki Uzun, and A. Kupper. "Towards a dynamic adaption of capacity in mobile telephony networks using context information." In ITS

Telecommunications (ITST), 2011 11th International Conference on, pp. 606-612. IEEE, 2011.

[20] Kim, Minkyong, and David Kotz. "Modeling users' mobility among WiFi access points." In Papers presented at the 2005 workshop on Wireless traffic measurements and modeling, pp. 19-24. USENIX Association, 2005.