# An Adaptive Mobility Manager for Software-Defined Enterprise WLANs

Yunong Han, Kun Yang
School of Computer Science and Electronic Engineering
University of Essex
Colchester CO4 3SQ, UK
Email: {hyunon,kunyang}@essex.ac.uk

***Abstract*—In current Enterprise WLANs (EWLANs), mobility management remains a challenging issue due to the rapid growth of mobile user number and varying traffic load. With conventional mobility managers, client handoff are based on signal strength of APs which often leads to imbalanced traffic load among APs. Software Defined Networking (SDN) has gained increasing popularity as a novel network architecture. Leveraging SDN to existing EWLANs, mobility management can be improved efficiently. In this paper, we present an adaptive mobility manager which utilises dynamic hysteresis of AP load to make handoff decisions. With proposed solution, network throughput is improved significantly compared to the conventional handoff algorithm.**

***Keywords*—Enterprise WLAN, SDN, AP Load, Mobility Management.**

## I. Introduction

Over the past decades, IEEE 802.11-based WLANs [1] have become the global trend in wireless communication technologies due to their characteristics, such as wide coverage and high data throughput. Companies have deployed WLANs as wired network extensions to provide mobility support, which enables clients to access network resources and roam freely anywhere within the coverage. Enterprise WLAN (EWLAN) is different from home WLAN in that it introduces a special entity called the wireless access controller (AC). The AC can not only control various parameters of AP (e.g. operating channel, transmission power and etc.) but also monitor the whole network by collecting wireless medium information and each APs status information. Due to the increasing demand for network performance, modern EWLANs require network services which include mobility management, load-balancing, power management, and interference management. In order to manage this growing complexity, novel architecture for EWLANs is needed.

Software-Defined Networking (SDN) has gained growing popularity as a novel technology for network configuration and management [2]. With the great potential to simplify the existing complex and inflexible network infrastructure, SDN can be a promising solution of modern EWLANs deployment. In the SDN architecture, the control plane is separated from the forwarding plane, which enables network intelligence and state to be logically centralized. Leveraging this global view of the network, network operators are able to achieve flexible programmatic management of the network. OpenFlow [3] is the best known communication interface defined between the central controller and the network forwarding devices in an SDN network. OpenFlow-based SDN architectures adapt the network to rapid growing business demands and significantly simplify network management complexity.

Mobility management is a key issue in EWLANs. With the trend of fast adaption to implement real-time multimedia services, for instance, video streaming over EWLANs, seamless handoff has become one of the crucial criteria to guarantee the Quality of Service (QoS). The conventional handoff algorithm only use the fixed RSSI value as the hysteresis margin when making handoff decisions. This mechanism is not able to provide seamless mobility over multiple wireless channels and often leads to imbalanced traffic load of APs, which affects user experience. To solve these WLAN-specific mobility problems, Enterprise vendors provide commercial solutions through vendor proprietary software and hardware. Unfortunately, these existing solutions [4], [5] do not provide open programmable interface and are not able to be achieved with the low-cost commodity AP hardware that is used by provider networks.

滞后
滞后

In recent years, some research works have been focusing on investigating the possibility of implementing SDN into wireless networks and the mobility management for software-defined EWLANs. For instance, Authors in [6] presented and prototyped an innovative open source platform called OpenRoads. It is the first work that implements OpenFlow into mobile wireless networks. In [7], authors present CloudMAC, an OpenFlow-based novel architecture for EWLANs where the MAC layer processing is partially offloaded to virtual machines provided by cloud services. In CloudMAC, APs only perform MAC frames forwarding between virtual APs and mobile stations. All the processing of MAC layer frames and the generation of management frame is done by virtual APs. CloudMAC achieves similar performance as normal WLANs but a higher level of flexibility and programmability which allows novel services to be easily added. Authors in [8] proposed an SDN based WLANs framework for handoff management with real-time video streaming. The system allows clients to associate with multiple APs simultaneously and to perform fast handoff between them, which improves the quality of real-time video streaming over WLANs significantly. Odin [9] is a novel OpenFlow-based SDN framework for EWLANs.

Based on the SDN concept, the system separates the control plane from the data plane by having a logically centralized SDN controller, which uses OpenFlow and a custom control plane protocol to manage the wired and wireless networks respectively. It provides a virtual AP association for each mobile client by introducing a novel programming abstraction called Light Virtual Access Points (LVAP). With LVAP abstraction, clients no longer need to re-associate with AP during a handoff process. Odin support seamless mobility but only when APs are operated on the same wireless channel. The handoff decision in Odin is triggered based on the fixed RSSI value hysteresis.

One common feature of these work is that none of them has considered the effect of AP traffic load on client handoff performance. Handoff decisions based on the RSSI value ensure that the client is always handed off to the AP with best signal strength. However, this does not guarantee the best user experience as the AP with best signal strength may be heavily loaded with network traffic. Therefore, it is vital to consider the AP traffic load when making client handoff decisions.

Authors in [10] proposed a load aware handoff algorithm and implemented it based on Odin [9] framework. The handoff parameter called Traffic Intensity is considered together with the signal strength value when making handoff decisions. Test results show that network throughput is improved with proposed algorithm compared to the legacy handoff mechanism used in WLANs. In addition, by adding Channel Switch Announcement (CSA) messages in beacon frames, seamless handoff between different WLAN channels on 2.4GHz band is achieved. However, the algorithm uses fixed hysteresis margin of Traffic Intensity, which is not efficient under the certain channel load conditions.

In this paper, we present an adaptive mobility manager for software-defined EWLANs which mitigates the imbalanced traffic load of APs and provides seamless mobility over multiple wireless channels. By introducing dynamic hysteresis margins of the AP load level in addition to the RSSI value, the proposed adaptive mobility manager running on top of the centralized SDN controller utilizes the AP load level, and the RSSI value provided by each connected AP to find the best candidate AP where clients can achieve improved network throughput after seamless handoff over multiple wireless channels. We demonstrate the efficiency of the proposed handoff algorithm by evaluating the effect of handoff on the network throughput of the TCP connection test.

The remainder of this paper is organized as following. Section II further describes the architecture of the selected software-defined EWLANs and the major extension work. Section III presents the dynamic hysteresis margins of the handoff metrics and the proposed adaptive handoff algorithm. The performance evaluation and analysis of the proposed mobility manager are demonstrated in Section IV before the paper concludes at Section V.

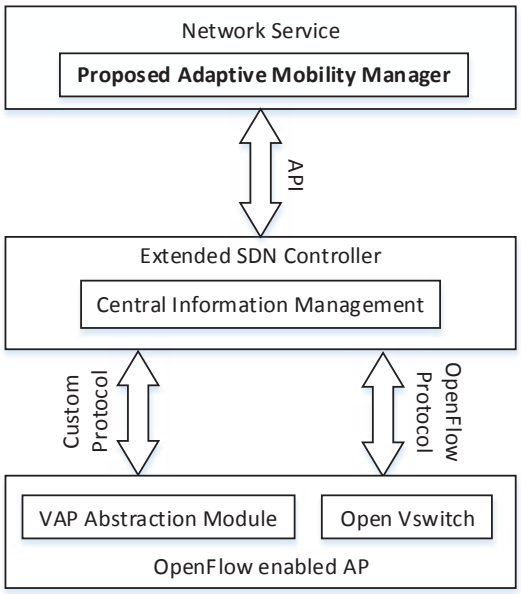


Figure 1: Arichitecture of software-defined EWLANs

## II. System Architecture

This paper adopts Odin architecture as the base of its mobility manager due to Odins light-weight and open source nature. Thanks to the authors in [9] who made the Odin Prototype publicly available [11], this paper further extend it to implement the adaptive mobility manager. Major extension works are as following:

- To achieve the load level of each AP, the ath9k based Wi-Fi driver is modified to export the content of related registers by using debugfs filesystem.
- To make the central controller aware of updated load level of each AP, a new custom protocol message called UPDATE-AP-LOAD is introduced.
- To achieve seamless handoff between multiple wireless channels, the CSA element is added into the modified beacon frame.
- To improve user experience during handoff over software-defined EWLANs, an adaptive mobility algorithm which is implemented on top of the SDN controller is proposed.

Fig. 1 shows the architecture of software-defined EWLANs. The three-layer architecture separates the control plane from the data plane by having a logically centralized Floodlight [12] SDN controller, which uses OpenFlow protocol and a set of custom protocol messages to manage the wired and wireless network respectively. The proposed adaptive mobility manager is implemented on top of the controller as an add-on module. The central controller is extended with a central information management module to have a global view of the network information, which includes data flows, associated APs and clients. APs in the architecture are integrated with Open vSwitch [13] and Click Modular Router [14]. The former one turns the normal AP into the OpenFlow enabled AP to support OpenFlow protocol. The latter one enables the VAP abstraction which significantly simplifies client association management

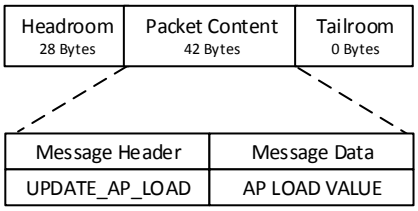


Figure 2: Format of protocol message UPDATE-AP-LOAD

by assigning a unique basic service set identification (BSSID) for each client that forms an association with a VAP.

In order to implement the proposed adaptive mobility manager, the controller requires to have the knowledge of the load level of each connected AP. To do so, a new protocol message called UPDATE-AP-LOAD is introduced to the system. Leveraging this new message, the load level of each connected AP is calculated in the VAP abstraction module and is encapsulated into a data packet with a proper message header before it is finally sent to the central controller via a separate control channel. The format of this protocol message is illustrated in Fig. 2. The headroom size is 28 bytes by default which contains extra information about the packet like the destination address to be used for routing. The message header and the AP load value are stored as a message string in the packet content and the total size is 30 bytes. The Tailroom is configured to be 0 bytes as it is not used in this case.

The central controller stores and updates the load value within a hash map matched by the IP address of the pysical APs. Mobility manager is implemented reactively based on an event-focused publish-subscribe solution. To enable the handoff management, the mobility manager first registers a subscription of the event that the RSSI value of the client from all physical APs is greater than a pre-defined threshold value. If the subscribed event is triggered at one of the physical APs, the PUBLISH message is sent to the controller with relevant context information including the clients media access control (MAC) address, the IP address of the physical AP where the event is triggered, and the corresponding up-to-date RSSI value that matches the subscription. This information is then passed to the mobility manager as a handoff metric together with AP load value to make handoff decisions. Fig. 3 shows the detailed handoff process and relevant protocol messages that are transmitted in the software-defined EWLANs system. In order to achieve seamless handoff over multiple wireless channels, the CSA element is added into the modified beacon frame. The CSA element contains three fields which are *channel switch mode*, *new channel number*, and *channel switch count*. The first field is either 0 or 1 which indicates any restrictions on transmission until a channel switch happens. The second field stores the new channel number while the last field contains the value of remaining beacons that will be sent before the channel switch takes place. In our testbed, whenever the client is triggered to handoff to an AP on a different wireless channel, the modified beacon frame with CSA element is sent from the clients current associated AP before the clients VAP is removed.

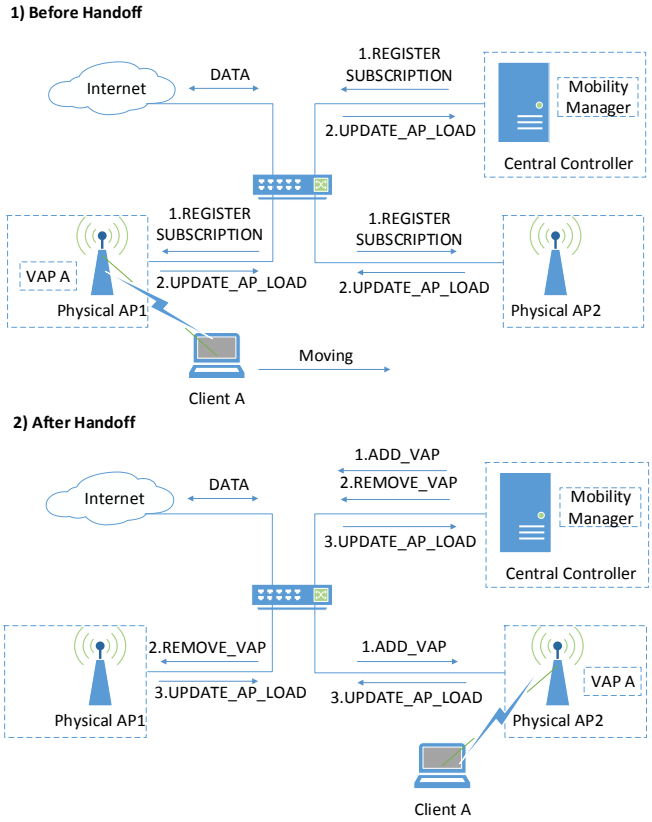


Figure 3: VAP Handoff in software-defined EWLANs

## III. Proposed Adaptive Handoff Algorithm

### A. *Handoff Metrics*

In the traditional handoff algorithm, signal strength is the only metric to depend on the handoff decision. When a client needs to perform a handoff, it selects the AP with best signal strength to send an association request. However, the selected AP with the highest RSSI does not always provide the best user experience due to the issue of unbalanced traffic load of APs. In order to solve this issue during handoff, a new handoff metric called AP load level is introduced in this paper. The AP load level is defined with two parameters, which are the channel load and the number of associated clients Nac. According to IEEE 802.11k [15] standard, the channel load is defined as the percentage of time in which the wireless channel is sensed busy by either the physical or virtual carrier sense (CS) mechanism. This channel busy fraction is calculated using the equation in (1) and the computation of AP load level is achieved with equation in (2). 80% weightage is given to the channel load parameter as it is the major factor that affects the user experience and 20% weightage is given to the number of associated clients.

$$Load_{ch} = \frac{ChannelBusyTime}{MeasurementDuration} \tag{1}$$

$$Load_{ap} = \begin{cases} Load_{ch} & (N_{ac} = 0) \\ Load_{ch} * 0.8 + N_{ac} * 0.2 & (N_{ac} > 0) \end{cases} \tag{2}$$

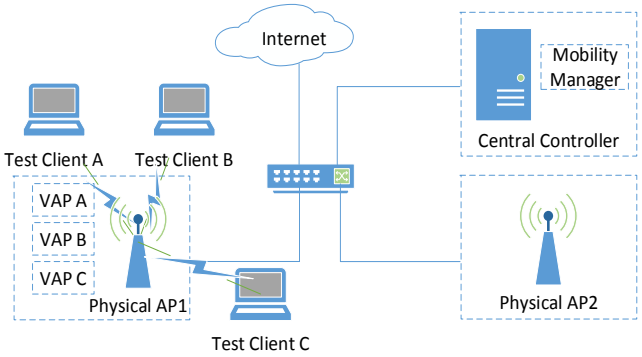


Figure 4: Architecture of the testbed

The channel load value is exposed in the channel load report of the Wi-Fi cards that implements IEEE 802.11k standard. The Wi-Fi card we use in our testbed are based on Atheros QCA9558 802.11 b/g/n chipset and it does not support IEEE 802.11k standard. However, the ath9k driver of this chipset contains two registers called AR_RCCNT (0x80f4) and AR_CCCNT (0x80f8) which store information about the number of time cycles that are sensed busy by the clear channel assessment (CCA) carrier sense mechanism and the total number of time cycles that have elapsed. To achieve the AP load value, the ath9k driver is mofdified to expose two debugfs based userspace interfaces to read the value of the two registers and the channel load value is calculated as AR_RCCNT/AR_CCCNT.

In order to set the hysteresis margin of AP load level, three simple tests with three levels of AP load (i.e. low, medium, and high) respectively are conducted on the testbed shown in Fig. 4. In this work, the low level of channel load is defined as 0 to 0.49, the medium level is defined as 0.5 to 0.79, and the high level is defined as 0.8 to 1.0. For each test, client C is associated with AP1 to perform a TCP test with the length of 30 seconds by using Iperf. Each test is conducted five times and results are averaged. By increasing the number of associated client and the network traffic, the effect of AP load level on network throughput is investigated. Based on the result shown in Fig.5, the hysteresis margins of AP load level are defined as 0.6 ($LH_h$), 0.3 ($LH_m$), and 0.15 ($LH_l$). As any margin of AP load level is greater than 0.6 can make significant effect on network throughput and this effect decrease gradually when the margin turns to be 0.3 and 0.15. To properly define the hysteresis margins of RSSI value, a similar test is conducted with one client associated with AP1 and perform an Iperf TCP test. By moving the clients location, the effect of RSSI value on network throughput is investigated. The high signal strength level is defined with any RSSI value greater than 205, the medium signal strength level is defined with RSSI value 190 to 204, and the low signal strength level is defined with any RSSI value lower than 189. Based on the result shown in Fig.6, the hysteresis margins of RSSI value are defined as 15 ($RH_l$), 30 ($RH_m$) and 40 ($RH_h$) because of its different extent of effect on network throughput. In addition, a dynamic time hysteresis with value of 3 ($TH_s$) and 6 ($TH_l$) seconds are used to avoid the influence of sudden variation

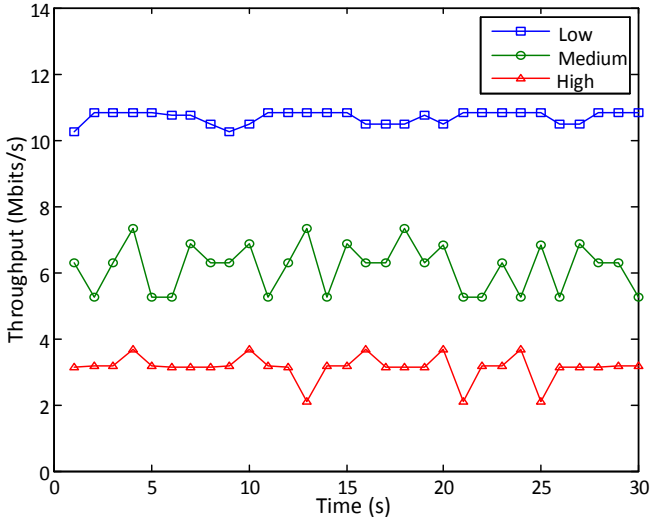


Figure 5: Network throughput with different level of AP load value

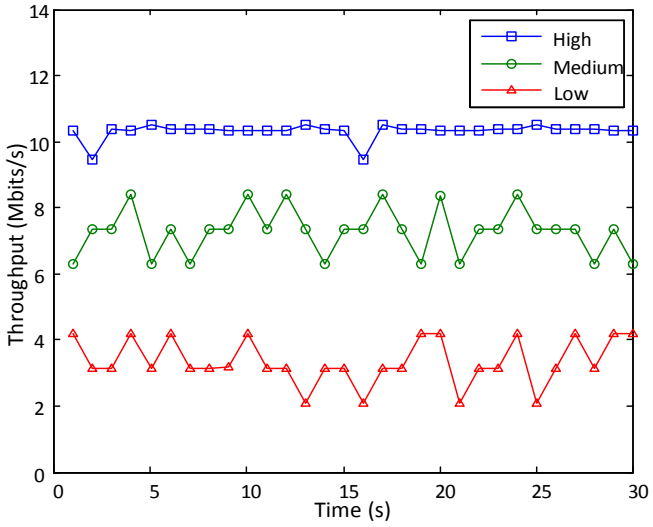


Figure 6: Network throughput with different RSSI value

of two handoff metrics and the ping-pong handoff effect and the initial threshold of RSSI value ($RT_i$) for the subscribed handoff event to be triggered is defined as 180 where the network throughput starts to get affected significantlly.

### B. Adaptive Handoff Algorithm

Based on the above handoff metrics, an adaptive handoff algorithm is proposed, which pseudo code is shown in Algorithm 1. The mobility manager first register the subscription of handoff event with each AP through the central controller, and if the RSSI value of the client on any AP is greater than the pre-defined signal strength threshold $RT_i$ which is 180, then the registered handoff event is triggered. The main loop of the algorithm starts from Line 2, the mobility stats which contain the RSSI value of the client, the AP load value, the current time and the initial time when the mobility stats is assigned. The loop first checks the IP address of the AP where the event is triggered so as to see if it is the same of the AP that the client is currently associated, the mobility stats of the client are updated with current system time and the latest value of RSSI

**Algorithm 1** Adaptive Handoff Algorithm

1: Mobility Manager subscribe handoff event with APs
2: **while** RSSI $> RT_i$ **do**
3: Assign initial mobility stats of client
4: **if** $IP_c$ = $IP_a$ **then**
5: Update mobility stats of client
6: **else**
7: **if** ($RSSI_c$ - $RSSI_a > RH_h$) or ($Load_a$ - $Load_c > LH_h$) **then**
8: $TH = TH_s$
9: **else**
10: $TH = TH_l$
11: **if** ($Time_{cur}$ - $Time_{ass} < TH$) **then**
12: back off
13: **else**
14: **if** ($RSSI_c > RSSI_a + RH_l$) and ($Load_c + LH_l < Load_a$) **then**
15: Handoff client and update mobility stats
16: **else if** ($RSSI_c > RSSI_a + RH_m$) and ($Load_c < Load_a + LH_l$) **then**
17: Handoff client and update mobility stats
18: **else if** ($RSSI_c + RH_l > RSSI_a$) and ($Load_c + LH_m < Load_a$) **then**
19: Handoff client and update mobility stats
20: **else if** ($RSSI_c + RH_m > RSSI_a$) and ($Load_c + LH_h < Load_a$) **then**
21: Handoff client and update mobility stats
22: **else if** ($RSSI_c > RSSI_a + RH_h$) **then**
23: Handoff client and update mobility stats
24: **end if**
25: **end if**
26: **end if**
27: **end if**
28: **end while**

and AP load. However, the initial assigned time is not updated at this stage. If the event is triggered from a competing AP, the dynamic back off time hysteresis margin is then decided based on the difference of the RSSI value and the AP load level. If the handoff event is triggered from a competing AP, the back off time is then checked to prevent ping-pong handoff effect with Line 11. If the notification is received outside the time hysteresis period, the handoff decision is then made based on the handoff metrics. Line 14 ensures the client is handed off to less loaded AP with better signal strength. Line 16 hands off client to the competing AP with much better signal stength even it is slightly heavier loaded than the current associated AP. Line 18 guarantees that the client is handed off to much less loaded AP with relative lower signal strength but well enough to provide good QoS. Similar decision is made in Line 20 but with larger hysteresis margins of RSSI value and AP load value. Line 22 ensures that client is handed off if it is going to disassociate from current associated AP no matter how crowd the competing AP is. Once the handoff completes, the mobility stats including the initial assigned time are updated.

Table I: Initial Setup for the Experiment

| | |
|---|---|
| AP Model | TP-Link WR1043ND Ver 2.1 |
| AP Firmware | Openwrt Barrier Breaker 14.07 |
| Number of APs | 2 |
| Client Operating System | Linux Ubuntu 14.04 |
| Number of Clients | 2 |
| Client Activity | Iperf TCP test |
| AP Channels | 5 and 10 |
| SDN Controller | Floodlight |
| Integrated Software | Open vSwitch 2.3.0, Click Modular Router 2.0.1 |

## IV. Performance Evaluation

### A. Experiment Setup

In this Section, two experiments are conducted to investigate the performance of the proposed adaptive handoff algorithm. All of the experiments are performed based on the testbed shown in Fig. 4 with the initial setup listed on Table I. In both experiments, test clients are assigned with the static IP to eliminate delays caused by DHCP. Authentication is disabled to exclude related delays as well. In order to evaluate the performance of the adaptive handoff algorithm, results are compared to the fixed RSSI handoff algorithm.

### B. Performance Evaluation

The purpose of the first experiment is to show that the proposed handoff algorithm can perform efficiently when the difference of both APs RSSI value are quite small. Initially, both test client $A$ and $B$ are associated with AP1 to perform the Iperf TCP test. Once the TCP session begins, test client $A$ starts to move slowly towards AP2 until it reaches the overlapping area of two APs where the difference of the signal strength between those two APs is quite small. The whole session runs 30 seconds and the TCP test result of test client $A$ is collected over one second interval. The experiment is conducted 5 times and results are averaged and presented in Fig. 7. With fixed RSSI algorithm, test client $A$ remains associated to AP1 during the whole session as the difference of RSSI value between AP1 and AP2 is not big enough to trigger the handoff. This leads to the result of low network throughput as test client $A$ is sharing bandwidth with test client $B$. However, with adaptive handoff algorithm, test client $B$ is handed off to AP2 after 15 seconds as the load level of AP2 is much lower than AP1. It can be observed that the network throughput of test client $A$ is improved by more than 5 Mbps compared to the results achieved with handoff algorithm with fixed RSSI hysteresis.

The purpose of the second experiment is to show that how efficiently the proposed handoff algorithm can perform when the AP with better signal strength is heavily loaded. At the beginning, test client $A$ is associated with AP1 to run the Iperf TCP test while test client $B$ is associated with AP2 to

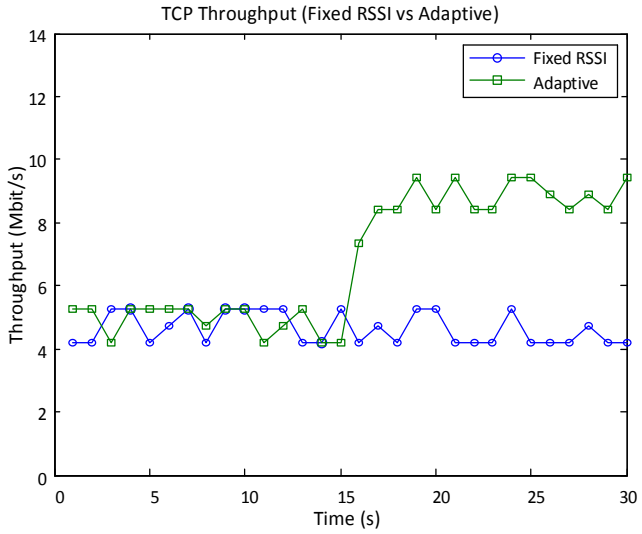


Figure 7: Network throughput comparison for Experiment 1

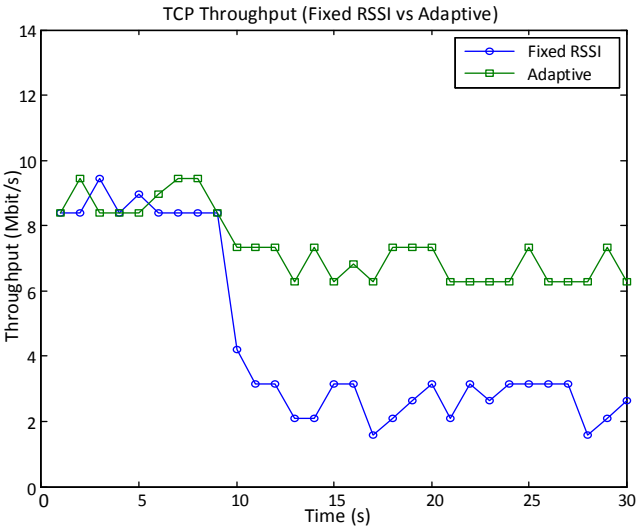


Figure 8: Network throughput comparison for Experiment 2

download a large file by using Wget. Test client $A$ starts to move to AP2 after the TCP test is triggered. The whole test lasts 30 seconds and is repeated 5 times to average the results. It is shown that the network throughput of test client $A$ is dropped by more than 6 Mbps after 10 seconds compared to the result based on proposed algorithm (as shown in Fig. 8). This significant decrease is because test client $A$ is handed off to heavily loaded AP2 with the fixed RSSI algorithm due to the signal strength difference between two APs. As the result of handoff, test client $A$ has to share the bandwidth with test client $B$ which makes both of them suffer low throughput. In contrast to the fixed RSSI algorithm, the proposed adaptive handoff algorithm works more efficiently as it detects the heavy traffic load of AP2, and instead of performing a handoff, the proposed mobility manager remains the current association of test client $A$ with less loaded AP1, which is able to provide better network throughput.

## V. Conclusion and Future Work

This paper has proposed an adaptive handoff algorithm to improve the mobility management in a software-defined EWLANs system. The proposed adaptive handoff algorithm takes the AP load value into account in addition to RSSI value to make handoff decisions which address the issue of imbalanced traffic load on APs. The experiment results demonstarte the effectiveness and the efficiency of the proposed adaptive handoff algorithms in terms of network throughput when compared to handoff algorithm based on fixe RSSI hysteresis. The future work includes further extending the architecture to support a wide variety of EWLAN management functions and to implement a more diverse set of EWLAN-specific services.

## VI. Acknowledgement

The work was partially funded by UK EPSRC Project DANCER (EP/K002643/1), and EU FP7 Projects CLIMBER (GA-2012-318939) and CROWN (GA-2013-610524).

## References

[1] "IEEE Standard, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

[2] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks,," Apr 2012, White Paper. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[4] "Meru Networks," http://www.fortinet.com/meru.

[5] "Cisco Meraki," https//meraki.cisco.com.

[6] K-K. Yap, M. Kobayashi, R. Sherwood, T-Y. Huang, M. Chan, N. Handigol and N. McKeown, "OpenRoads: Empowering Research in Mobile Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 125–126, Jan. 2010.

[7] J. Vestin, P. Dely, A. Kassler, N. Bayer, H. Einsiedler and C. Peylo, "CloudMAC: Torwards Software Defined WLANs," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 393–396.

[8] P. Dely, A. Kassler, L. Chow, N. Bambos, N. Bayer, H. Einsiedler, C. Peylo, D. Mellado and M. Sanchez, "A software-defined networking approach for handover management with real-time video in WLANs," *Journal of Modern Transportation*, vol. 21, no. 1, pp. 58–65, 2013.

[9] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann and T. Vazao, "Towards Programmable Enterprise WLANS with Odin," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 115–120.

[10] A.K. Rangisetti, H.B. Baldaniya, B.P. Kumar and B.R. Tamma, "Load-aware hand-offs in software defined wireless LANs," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, Oct 2014, pp. 685–690.

[11] "Odin git source," https://github.com/lalithsuresh/odin.

[12] "Floodlight OpenFlow Controller," http://www.projectfloodlight.org.

[13] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon and M. Casado, "The Design and Implementation of Open vSwitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 117–130.

[14] E. Kohler, R. Morris, B. Chen and J. Jannotti and M.F. Kaashoek, "The Click Modular Router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

[15] "IEEE Standard, Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs," *IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007)*, pp. 1–244, June 2008.