SAMF: An SDN-Based Framework for Access Point Management in Large-scale Wi-Fi Networks

Ying-Dar Lin, *Fellow, IEEE,* Chun-Hung Hsu, Minh-Tuan Thai, Chien-Ting Wang, Yi-Jen Lu, and Yi-Ta Chiang

***Abstract*—Large-scale Wi-Fi networks have encountered several critical issues of access point (AP) management such as manual configuration, channel interference, and unbalanced loads, which should be carefully addressed to ensure efficient system performance. Since most of the commercial Wi-Fi products are proprietary and hardware-dependent, some recent studies have aimed at introducing open and programmable solutions. Unfortunately, the studies demand additional protocols and software agents but cannot provide complete solutions. To this end, this experience paper presents the design, prototype implementation, and evaluation of SAMF, which is an open, programmable, and generic framework for access point management in large-scale Wi-Fi networks. By adopting the concept of SDN technology and OpenFlow protocol, SAMF can readily be deployed on low-cost commodity access point hardware and a cloud-based controller, while enabling new network services to be integrated rapidly. Furthermore, experimental results confirm that the framework can significantly reduce operational costs since it accelerates the AP configuration process by approximately 15 times. Besides, SAMF can increase system throughput up to 26.5% and improve the balanced degree of the system by about 40%.**

***Index Terms*—Channel assignment, IEEE 802.11 WLAN, OpenFlow, Wi-Fi AP management, Wi-Fi load balancing.**

I. Introduction

WI-FI has become the most dominating last hop networking technology providing more convenient access to the Internet for users compared to its expensive wired counterparts. However, a large number of Wi-Fi APs are required to accommodate complete connectivity coverage for large-scale networks, such as university campus networks or enterprise networks, which results in several AP management issues. The first is *AP manual configuration* where there are many deployed APs with diverse setting requirements, and where one-by-one manual AP configuration can become a challenging and time-consuming task. The second issue, *AP channel interference*, arises from the fact that there is a small number of non-overlapping channels for neighboring APs. The APs, which operate on the same or adjacent channels, may suffer from spectrum interference problems. The problems cause network congestions, which severely downgrade system throughput. Finally, most of Wi-Fi users likely tend to connect to an AP whose Received Signal Strength Indicator (RSSI) is the strongest. This may cause *AP unbalanced load* issues between APs in the same coverage area, i.e., single extended service set (ESS), which severely degrade the performance of the entire network.

Manuscript received August 4, 2017; revised October 20, 2017. Date of publication: December 29, 2017.

Ying-Dar Lin, Chien-Ting Wang, and Yi-Jen Lu are with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.

Minh-Tuan Thai is with the EECS International Graduate Program, National Chiao Tung University, Hsinchu, Taiwan.

Chun-Hung Hsu and Yi-Ta Chiang are with the Network Benchmarking Lab (NBL), National Chiao Tung University, Hsinchu, Taiwan.

Minh-Tuan Thai is the corresponding author (e-mail: tmtuan.eed03g@nctu.edu.tw).

Digital Object Identifier (DOI): 10.24138/jcomss.v13i4.398

A. Wi-Fi APs: Thin vs. Fat

To address the above AP management issues, conventional Wi-Fi networks utilize Fat APs, i.e., Autonomous APs, which can perform all network processing and functionalities in each unit. These APs have built-in advanced intelligence for enhancing performance and security beyond what the IEEE 802.11 standard provides. This solution is relatively costly since Fat APs need to be built of powerful hardware and require sophisticated software. Furthermore, it just supports some limited management capabilities such as initial and on-going configurations. Besides, network modification and general administration are challenging tasks for the Fat AP solution. With such critical drawbacks, Fat APs are not suitable for large-scale Wi-Fi networks, which require a large number of APs and have continuously changing network states.

In contrast to Fat AP solutions, Thin APs rely on a centralized controller for advanced and complicated management, performance enhancement, and security assurance. In other words, these functions are offloaded to the controller instead of being left in the APs. With this approach, Thin APs generally cost less since they are not equipped for much more than network traffic transmission functions. This can significantly reduce the deployment cost of a Wi-Fi network, especially large-scale ones. Furthermore, the Thin AP solution can provide a centralized management interface, which performs network modification and general administration efficiently. It also allows cost-effective upgrades and migration to future technologies. To sum up, the solution of Thin APs with a centralized controller is promising to address the requirements of large-scale Wi-Fi networks.

The idea of Thin APs with a centralized controller has recently been introduced into academia as well as industry [1]–[9]. However, current studies [1]–[5] only solve AP management issues individually, which cannot offer a complete solution for large-scale Wi-Fi networks. Furthermore, additional protocols and software agents are needed in such solutions. Although industrial products [6]–[9] can provide complete solutions for Wi-Fi AP management issues, specialized equip-

ment and hardware-dependent software must be applied to the products.

B. Our solution

In this experience paper, we aim to provide an open, programmable, and generic solution for large-scale Wi-Fi AP management by presenting a framework, called *SDN-based AP Management Framework (SAMF)*, which is based on the concept of Software-defined Networking (SDN) [10] and OpenFlow protocol [11]. We decouple the system into three tiers namely management, control, and data plane so that the framework not only manages APs efficiently but also enables new network services to be rapidly integrated. Moreover, OpenFlow protocol is extended to transfer our proposed APIs among system components, hence eliminating the need for additional protocols and software agents. By exploiting the benefits of SAMF, we further develop three service functions, *Per-subnet automatic configuration (PAC)*, *Dynamic channel assignment (DCA)*, and *Dynamic AP load balancing (ALB)*, to address the issues of AP manual configuration, AP channel interference, and AP unbalanced load, respectively.

We combine OpenWrt [12] and Open vSwitch [13] to build OpenFlow-enabled APs while RYU SDN framework [14] is used to implement control and management functions. By doing so, SAMF can be deployed using low-cost commodity AP hardware and a cloud-based controller, which considerably reduces deployment cost. Our design and implementation are evaluated using a real testbed, which assesses how well SAMF handles the AP management issues. Experimental results show that the framework significantly reduces operational costs since it speeds up AP configuration task up to 15.0 times. Also, SAMF can improve system throughput by 26.5%, while performing AP load balancing efficiently under different AP load status scenarios.

We like to emphasize that the contributions of this paper focus on introducing a proof-of-concept framework for Wi-Fi AP management in large-scale networks, rather than developing novel algorithms. Fortunately, since the framework provides open and programmable interfaces; it would be a useful platform for future work whose objectives are to study algorithms for addressing AP management issues.

The rest of this paper is organized as follows. Section 2 briefly presents industrial products and existing studies of Wi-Fi AP management. We then discuss the design of SAMF and its implementation in Section 3. Next, evaluation study and experiment results are presented in Section 4. Finally, Section 5 offers the conclusions of this work.

II. Related Work

This section first gives an overview of industrial products and existing studies of Wi-Fi AP management issues. We then provide a survey on Wi-Fi extensions for OpenFlow protocol.

A. Wi-Fi AP Management products and studies

Table I compares SAMF with several popular commercial products of Wi-Fi AP management. These products also use a centralized controller to manage APs via CAPWAP [15] or OpenFlow protocols. Unfortunately, they are proprietary and hardware-dependent, which can neither be deployed on commodity network equipment nor support third-party service integration. In contrast to these products, our SAMF design provides a generic framework for Wi-Fi AP management using low-cost hardware and bare metal appliances. Furthermore, SAMF allows new services to be integrated rapidly by introducing open and programmable interfaces.

There have been several existing studies which aim at addressing Wi-Fi AP management issues. In [1], an open-source platform, called OpenRoads, is proposed for the incorporation of different wireless technologies, i.e., Wi-Fi and WiMAX. The platform exploits OpenFlow and SNMP protocol for data path control and device configuration, respectively. A cloud-based solution for reducing channel contention among APs in a residential environment is introduced in a COAP framework [2]. In other words, COAP enables the APs to share their information via a controller which provides different centralized channel assignments. In contrast to COAP, Odin [3] focuses on enterprise and provider networks by introducing programmability for APs. The study introduces the concept of Light Virtual Access Point (LVAP) which is a programming abstraction for addressing the specific requirements of Wi-Fi networks. In addition to OpenFlow, Odin protocol and its software agents have to be deployed in APs and the controller for handling the demands of Wi-Fi networks in such a solution. OpenSDWN [4] and EmPower [5] also exploit the LVAP concept and extend the Odin design. To be more specific, OpenSDWN leverages virtualized network functions [16] to support seamless user mobility and flexible function allocation, while EmPower presents a software development kit to enable developers to create new service functions. To sum up, Wi-Fi AP management issues are have been individually solved in these recent studies, but they require additional protocols and software agents.

B. Wi-Fi extensions for OpenFlow protocol

OpenFlow protocol currently focuses on providing programmability for Ethernet switches while it does not support wireless technologies well. For example, the protocol cannot accommodate statistic information exchanges or configuration updates from the controller to APs. There have been some efforts made to address such limitations, and which can be classified into two groups. Odin [3] belongs to the first group, which develops a separate protocol for the wireless networks while leveraging OpenFlow for wired parts. In the second group, COAP [2] introduces Wi-Fi extensions for OpenFlow, which uses switch-related features to implement communications between the controller and APs.

Our SAMF design follows the second technique since we aim to propose a pure solution, in which only OpenFlow protocol is involved in the framework. The fundamental difference between our solution and COAP is that we leverage the Experimenter messages (EXPs) of OpenFlow for the communications between the control and data plane, instead of using current switch related features like COAP. Since

TABLE I: Comparison between SAMF and commercial WI-FI AP management products

Product	Protocol	Hardware	Software	Feature
Aruba [8]	CAPWAP	Proprietary, chip-driven	Hardware dependent	Full cost
Meraki [9]	CAPWAP	Proprietary, chip-driven	Hardware dependent	Full cost
Tallac [6]	OpenFlow	Proprietary, chip-driven	Cloud	Full cost
Meru [7]	OpenFlow	Proprietary, chip-driven	Hardware dependent	Full cost
SAMF	OpenFlow	Open, bare metal appliance	Cloud, generic platform	On-demand purchase

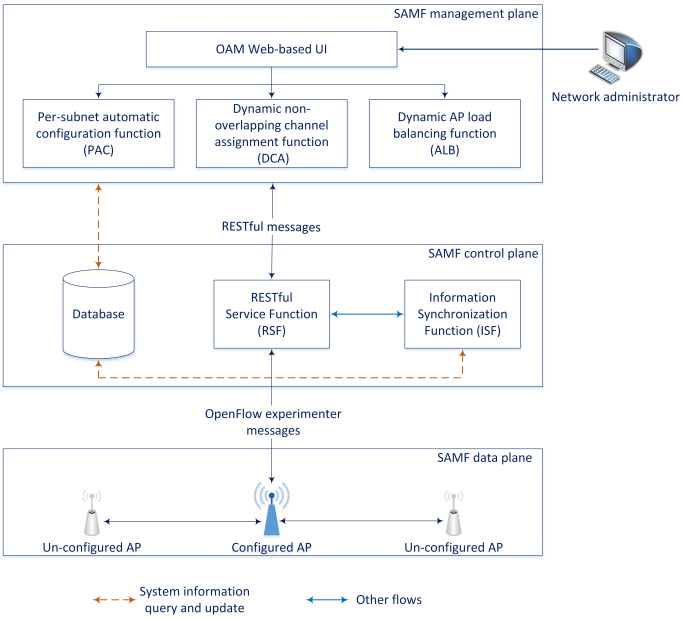


Fig. 1: SAMF software architecture

OpenFlow specification defines the EXPs for developers to implement additional functionalities or future revisions, there are no compatibility problems in our Wi-Fi extensions for OpenFlow protocol.

III. WI-FI ACCESS POINT MANAGEMENT THROUGH SAMF

This section first presents the software architecture of SAMF. Second, the designs of three proposed functions solving Wi-Fi AP management issues are discussed.

A. Software architecture

Fig. 1 illustrates the software architecture of SAMF in which management, control, and data planes are decoupled to achieve centralized administration and support multi-controllers. By separating control and data path, this approach also enables new service functions to be readily integrated.

1) SAMF management plane: In our design, the management plane is responsible for ensuring that the whole network runs optimally and meets Quality of Service (QoS) requirements. In other words, the service functions that configure and manage the system, such as PAC, DCA, and ALB, should be deployed in this tier. These functions communicate with the components in the control plane via RESTful messages by transmitting HTTP requests and responses. Our design also provides a centralized management point for network administrators by introducing a web-based user interface in this plane.

2) SAMF control plane: RESTful messages delivered from the management plane are processed by RESTful Service function (RSF), which is implemented in this plane. To be more specific, RSF is responsible for analyzing the messages and triggers corresponding operations. If the operations are for AP management, RESTful messages will be translated to EXP messages, including the necessary parameters for executing the operations. After that, the EXP messages are sent to the relevant APs in the data plane. In the opposite direction, the EXP response messages from the data plane are also converted to RESTful messages by RSF; and then redirected to the management plane.

This plane is also responsible for maintaining the global view of the entire network via the Information Synchronization function (ISF) which periodically sends queries to the network devices in the data plane by using the service of RSF. It should be noted that multiple controllers can be deployed in the network to eliminate the problem of controller bottlenecks resulting from a large number of APs in large-scale networks.

3) SAMF data plane: EXP messages from the control plane are mostly forwarded to OpenFlow-enabled APs in this tier. Because the APs support OpenFlow protocol, they can analyze the content of EXP messages, and then carry out the corresponding actions themselves. When APs report their status or trigger a function in the upper tiers, EXP messages are also generated and get back to the control plane.

Fig. 2 shows an example of RESTful and EXP messages of our SAMF framework. As can be seen, RESTful messages, which are HTTP requests and responses, are used for the communications between PAC and RSF functions. Their data is format in JSON syntax, which contains the necessary parameters and returning results of the execution of PAC such as controller's IP, APs' MAC and IP. The function RSF communicates with APs by transmitting EXP messages. In other words, RSF acts as an interpreter between PAC and the components in data plane. The format of EXP messages is defined by our SAMF APIs, which are described more details in the implementation section.

B. AP management functions

We here exploit the benefits of SAMF by further developing three functions, which address the fundamental issues of AP management in large-scale Wi-Fi networks.

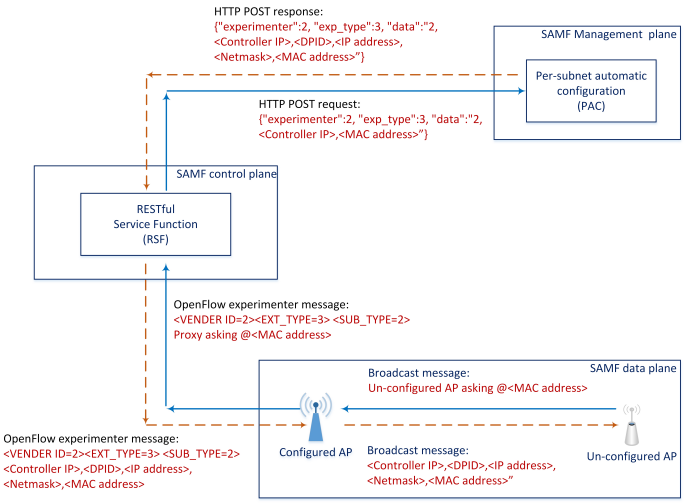


Fig. 2: Example of RESTful and EXP messages.

1) Per-subnet automatic configuration (PAC): This function addresses the issue of AP manual deployment, which can automatically configure new APs without manual settings. As a result, it not only significantly reduces the need for human resources, but also ensures the correct and valid configurations for the APs.

As can be seen in Fig. 1, there should be at least an AP configured in advance by network administrators in a subnet. The AP acts as a proxy, which forwards configuration requests and responses from un-configured APs to a controller. To be more specific, whenever a new AP joins to a subnet, it will carry out broadcasting to find any configured APs which can help forward its configuration requests to the PAC function. After PAC receives the requests, it will respond with the correct configuration loaded from the system database. When the configuration responses arrive at a new AP, it will carry out self-settings. By doing so, the AP can successfully connect to the system network.

2) Dynamic channel assignment (DCA): The issue of channel interference is solved by this function. The function dynamically adjusts the wireless channels used by APs to avoid interference problems as much as possible. By doing so, DCA can improve transmission quality, which considerably increases system performance.

To provide proper channel assignment schemes, DCA maintains the database of AP channel in the network by periodically sending queries to the SAMF data plane, using the services of ISF. The function then determines the APs, which should change their channels by formulating a channel assignment problem. In this paper, we adopt a greedy algorithm as developed in [17] to solve the problem. Noted that more complicated and optimal AP channel assignment algorithms such as [18], [19] can also be implemented by DCA. Eventually, DCA sends channel modification requests to the APs, which will change their channel by themselves. Although the function is periodically triggered to avoid channel interference, network administrators can trigger DCA manually when it is required.

3) Dynamic AP load balancing (ALB): This function handles the issue of AP unbalanced load by deriving a two-tier load balancing algorithm developed in [20]. Whenever there is an overloaded AP in the same coverage area, i.e., in the same ESS, ALB is automatically invoked to reallocate user connections of the APs in the ESS, hence balancing the load status of APs. We define the *load status* of an AP is the maximum value of its CPU utilization, memory utilization, and average packet error rate. Since the APs in an ESS are likely to have different load levels, we further define the *average load level* of an ESS as the mean value of load status of all its APs. The level is used to classify the APs into over-loaded and under-loaded groups so that ALB can transfer some client connections from the overloaded APs to under-loaded APs to maintain the load balancing status for the ESS.

When an AP has just successfully connected the system network, it reports its capacity information to ALB, after that it will receive a number of suggested associated clients from the function. Whenever the number of currently associated clients of the AP reaches the threshold, it will send a notification message to trigger ALB. Then the function will collect the necessary information of other APs in the same ESS. Next, a proper client connection distribution is determined for the ESS. Eventually, the APs will re-associate their clients based on the re-association list received from the ALB function.

Instead of burdening ALB with too many client re-allocation requests, we define a parameter named *level control*, which is the ratio between the suggested number of association clients for an AP and the maximum number of association clients that the AP can support. The value of the level is from 0.0, i.e., without load balancing control, to 1.0, i.e., the most centralized load balancing control. In other words, the higher the value of the level, the less frequently ALB will be triggered, but the more client stations an over-loaded AP would transfer to under-loaded APs to maintain the load balancing status of an ESS.

C. Implementation details

1) OpenFlow-enable AP firmware: To accomplish the communication between SAMF's data plane and control plane, we combine OpenWrt and Open vSwitch to develop OpenFlow-enable firmware for APs. Furthermore, to manage the APs effectively, we define a set of SAMF APIs, which are summarized in Table II. We add some shell scripts into Open vSwitch to enable APs support our SAMF APIs.

We implement SAMF APIs by leveraging OpenFlow EXP messages. The messages consist of two major fields, $ofp\_experimenter\_header$ and $ofp\_experimenter\_data$. There are two standard 4-byte sub-fields, $experimenter\_ID$ and exp_type, in the first field. The former contains our experimenter ID whereas the latter denotes the first-tier category of our Wi-Fi APIs. In our implementation, the second field $ofp\_experimenter\_data$ of EXP messages, which is uninterpreted by standard OpenFlow processing, is further divided into $exp\_subtype$ and $exp\_payload$ subfields. We define the 1-byte $exp\_subtype$ to denote the second-tier category of the APIs for a better arrangement. The latter sub-field

TABLE II: SAMF APIs

APIs Category	Exp_type	Exp_subtype
System management	0	0: System reboot 1: OpenFlow daemon patch upgrade 2: Firmware upgrade 3: Firmware initialization 4: Configuration backup 5: Configuration restore
Generic wireless control	1	0: Wireless on/off 1: Channel setting 2: Transmit power setting 3: Encryption setting 4: SSID setting
Device information retrieval	2	0: System description 1: Machine description 2: OpenWrt version description 3: Wireless configuration information 4: SSID configuration information 5: Wireless host information 6: OpenFlow daemon patch information
Local service management	3	0: LUCI setting 1: TFTP setting
Diagnostics	4	0: Ping testing 1: Traceroute testing
Device status Retrieval	5	0: CPU utilization information request 1: Memory usage information request
Automatic configuration	6	0: Proxy role trigger 1: Proxy role close 2: Configuration request

$exp\_payload$ is a string, which consists of the corresponding SAMF APIs parameters.

Note that the value of the $experimenter\_ID$ is allocated by the Open Network Foundation (ONF) [21], so OpenFlow developers need to contact the organization to obtain their IDs. Our experimenter ID, which is "0xFF000008", can be seen in ONF's registry.

2) SAMF controller: RYU SDN framework is used to develop all service functions such as PAC, DCA, ALB, RSF, and ISF. The functions are implemented as RYU modules using Python programming language which can support SAMF APIs. A database is also placed in our controller to store the necessary information of the whole system enabling service functions can make decisions accurately.

3) SAMF web-based user interface: We introduce a web-based user interface to enable administrators efficiently manage APs over the network. The interface provides the overall/individual status of the APs such as hardware/software description, CPU/memory utilization, network traffic, configuration, and flow table. Further, the setting of the APs such as firmware/patch upgrade, service enable/disable, and device reboot can also be centrally accomplished via the interface.

IV. Performance Evaluation

In this section, the design and implementation of SAMF are evaluated. We first describe the experimental testbed and then analyze the results.

A. Experimental testbed

Our experiments aim to evaluate how the proposed AP management functions, i.e., how PAC, DCA, and ALB will perform on a real Wi-Fi network. For this purpose, we set up a local Wi-Fi network which consists of four APs of AP222 model from EstiNet [22] and one computer running control and management functions. The APs are equipped with our SAMF OpenFlow-enable firmware and are located in a classroom whose size is shown in Fig. 3. To evaluate the performance of SAMF's functions, 20 handheld devices were used to establish connections to the APs. Figs. 3b to 3d show three cases of the locations of the devices in our experiments. In case 1, the devices are situated around one AP. while in case 2 and case 3, they are placed around two and four APs, respectively.

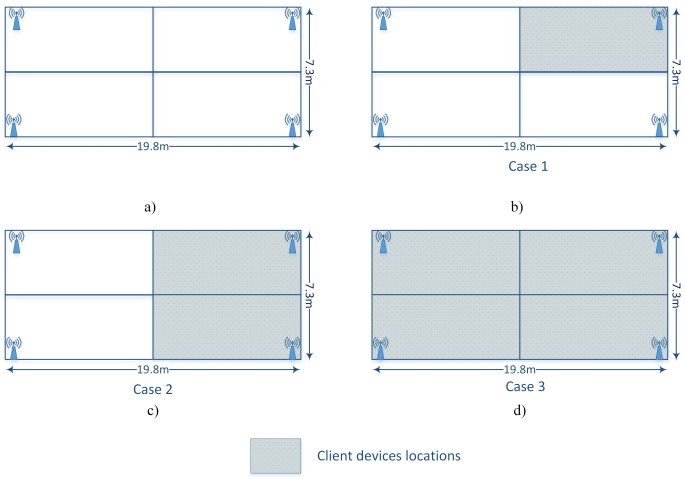


Fig. 3: Experimental testbed.

We developed a testing tool derived from iPerf [23] to measure the downlink and uplink throughput of the devices. This tool was installed in the testing devices and was controlled remotely by a centralized program. By doing so, we could run and control our experiments accurately.

B. Analysis of results

1) Automatic Deployment (AD) vs. Manual Deployment (MD): In this experiment, we observed the benefit of the PAC function, by configuring APs by two methods namely Automatic deployment (AD) and Manual deployment (MD). In the former, there was an AP which was manually configured in advance; and the others were automatically configured by PAC. On the other hand, all APs were manually configured in the later method.

Table III presents the amount of time consumed by two methods. The second and third columns of the table show the AP booting time and manual setting time of 19.7 and 90.3 seconds, respectively. In the AD method, a un-configured AP sends configuration requests to PAC and performs self-setting after receiving responses. The amounts of time of these two processes are about 5.6 and 8.4 seconds, respectively. The seventh column presents the total time spent by the two methods where the numbers of APs are 4 and 20. For the AD method, the amount of time in both cases is about 144.0 seconds that is $(19.7+90.3+(19.7+5.6+8.4))$, since new APs can be configured concurrently. That of MD is 440.0 seconds,

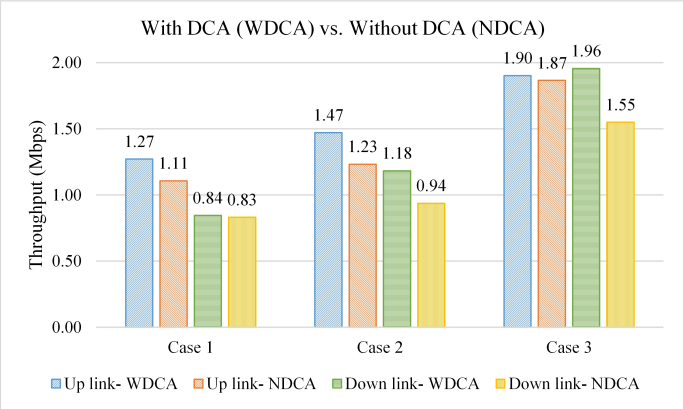


Fig. 4: With vs. Without Dynamic channel assignment function.

i.e., $(19.7+90.3)*4$, and 2200.0 seconds, i.e., $(19.7+90.3)*20$, because the APs are configured one by one.

In summary, PAC function can speed up the AP deployment process by approximately 3.0 and 15.0 times, compared to manual methods, when the numbers of AP are 4 and 20, respectively. Furthermore, it ensures correct and valid configurations for APs in the entire network, which is a challenging task for the manual method.

2) With vs. Without Dynamic channel assignment function: We carried out a further experiment in which the performance of DCA function was evaluated. In this case, the average throughput of the testing devices was compared under two situations, With DCA (WDCA) and Without DCA (NDCA). In NDCA, APs operated on the same Wi-Fi channel, while their channels were automatically adjusted to avoid interference problems in WDCA.

As expected, the results demonstrate that the DCA function can increase the throughput of devices in all test cases, except the downlink in case 1, where results are roughly the same in both WDCA and NDCA scenarios. As can be seen from Fig. 4, throughput is improved about 14.5% in the up-link of case 1 and 26.5% for the down-link of case 3. Fig. 4 also clearly shows that when client devices are located around all APs, as in case 3, their throughput is increased significantly, compared to test cases 1 and 2. To sum up, the experiment results confirm that dynamic channel assignment is crucial for the performance of Wi-Fi networks.

3) AP Load Balancing: This experiment investigated how the proposed AP load balancing function ALB performed in two scenarios. The first scenario, in which all APs were initialized with the same load level, was done to determine how the number of client connections maintained by the APs affects the load balancing results. In the second scenario where APs with different load levels showed how the re-association time was changed as the load diversity increased. We adopted an evaluation formula from [24] to define a metric named *balanced degree*, which represents the differences in APs load status. The metric is approximately 0.0, for the worst level of AP load balancing, and should reach 1.0 if all APs are equally loaded.

Fig. 5 gives the balanced degree and re-association times for the control level for the two scenarios. The experimental results show that the balanced degree increases with the control level and converges to 1.0 before the control level reaches 1.0. The re-association time also increased with the control level, as a result of an increasing number of client connections that need to be handed over from one AP to another. A large re-association list also results in a long data transfer time and a long computation process at the ALB function. Fig. 5 also shows that by automatically determining the minimum value of the control level of 0.4 and 0.7 in the first and second scenarios, respectively, ALB improves the system balanced degree by $34 - 40\%$, compared to a without load balancing control scheme, i.e., the value control level being at 0.0. Furthermore, our design reduced the AP re-association time by about $28 - 36\%$, compared to the most centralized load balancing scheme with a control level of 1.0.

V. Conclusion

In this paper, we first present the design and implementation of SAMF, an open, programmable, and generic framework for AP management in large-scale Wi-Fi networks. Based on the concept of SDN technology and standardized OpenFlow protocol, SAMF can readily be deployed on low-cost commodity AP hardware and a cloud-based controller, while enabling new service functions to be rapidly integrated. Further, by exploiting the benefits of SAMF, three service functions are developed to address common AP management issues, such as manual configuration, channel interference, and unbalanced loads.

The proposed functions were experimentally evaluated under different scenarios using a real testbed. The experimental results confirm that the functions can efficiently manage APs and significantly reduce operational cost and increase system performance. Our results show that the application can reduce the AP configuration time up to about 3.0 and 15.0 times, compared to manual methods where the numbers of APs are 4 and 20, respectively. Furthermore, SAMF can improve system throughput up to 26.5% with its dynamic channel assignment. Finally, our method can carry out AP load balancing efficiently under different AP load status scenarios, and improve the degree of system load balance by 40%.

In the future, we like to extend SAMF for supporting different service classes, user priorities, and QoS constraints, which are critical requirements of large-scale networks. Another possible future work is to evaluate SAMF in broader scenarios such as highly crowded Wi-Fi testbeds which consist of with more APs and client devices

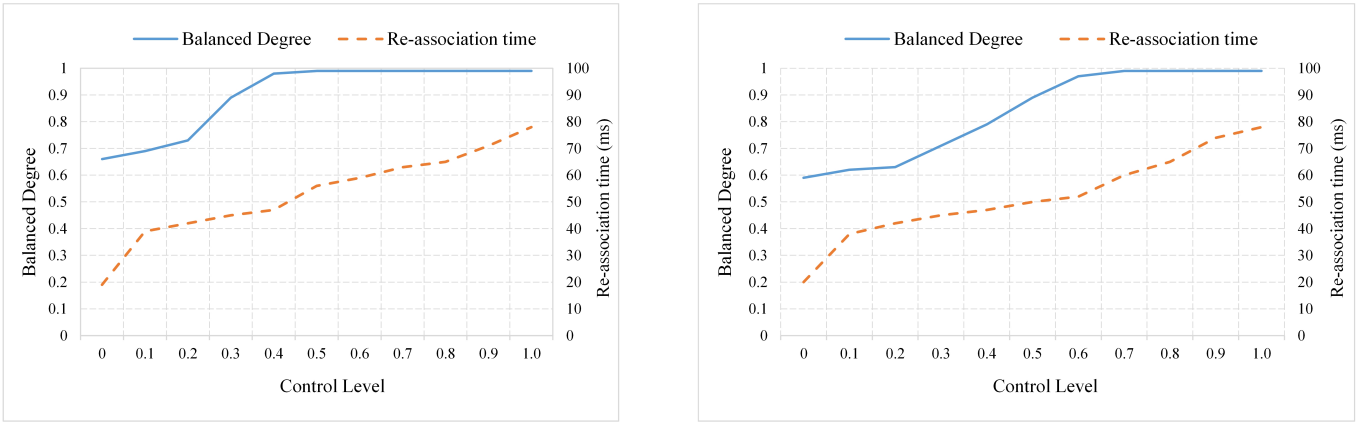
References

[1] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "OpenRoads: Empowering research in mobile networks," vol. 40, no. 1, pp. 125–126. [Online]. Available: http://doi.acm.org/10.1145/1672308.1672331

[2] A. Patro and S. Banerjee, "Outsourcing coordination and management of home wireless access points through an open API," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1454–1462.

TABLE III: Automatic Deployment (AD) vs. Manual Deployment (MD)

Deploy method	AP booting time (sec)	AP manual setting time (sec)	Number of APs (#)	Configuration request time (sec)	AP self-setting time (sec)	Total spending time (sec)	Speedup to MD
AD	19.7	90.3	4	5.6	8.4	144.0	∼3.0
AD			20			144.0	∼15.0
MD			4	N/A	N/A	144.0	N/A
MD			20			2200.0	



(a) APs are initialized with the same load levels

(b) APs are initialized with different load levels

Fig. 5: AP Load Balancing

[3] J. Schulz-Zander, L. Suresh, N. Sarrar, A. Feldmann, T. Hhn, and R. Merz, "Programmatic orchestration of WiFi networks," in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX ATC'14. USENIX Association, pp. 347–358. [Online]. Available: http://dl.acm.org/citation.cfm?id=2643634.2643670

[4] J. Schulz-Zander, C. Mayer, B. Ciobotaru, S. Schmid, and A. Feldmann, "OpenSDWN: Programmatic control over home and enterprise WiFi," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15. ACM, pp. 16:1–16:12. [Online]. Available: http://doi.acm.org/10.1145/2774993.2775002

[5] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, "Programming abstractions for software-defined wireless networks," vol. 12, no. 2, pp. 146–162.

[6] Tallac networks SD-LAN. [Online]. Available: http://www.tallac.com/

[7] Meru fortinet. [Online]. Available: https://www.fortinet.com

[8] Enterprise wireless LAN solutions | aruba, a hewlett packard enterprise company. [Online]. Available: http://www.arubanetworks.com/

[9] M. Inc. Cisco meraki. [Online]. Available: https://meraki.cisco.com/

[10] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," vol. 103, no. 1, pp. 14–76.

[11] "OpenFlow switch specification version 1.5.1." [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[12] OpenWrt. [Online]. Available: https://openwrt.org/

[13] "Open vSwitch." [Online]. Available: http://openvswitch.org/

[14] Ryu SDN framework. [Online]. Available: https://osrg.github.io/ryu/

[15] D. Stanley, P. Calhoun, and M. Montemurro. Control and provisioning of wireless access points (CAPWAP) protocol specification. [Online]. Available: https://tools.ietf.org/html/rfc5415

[16] M. T. Thai, Y. D. Lin, and Y. C. Lai, "A joint network and server load balancing algorithm for chaining virtualized network functions," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[17] A. Hills and J. Schlegel, "Rollabout: a wireless design tool," vol. 42, no. 2, pp. 132–138.

[18] R. Akl and A. Arepally, "Dynamic channel assignment in IEEE 802.11 networks," in *2007 IEEE International Conference on Portable Information Devices*, pp. 1–5.

[19] M. Seyedebrahimi, F. Bouhafs, A. Raschell, M. Mackay, and Q. Shi, "SDN-based channel assignment algorithm for interference management in dense wi-fi networks," in *2016 European Conference on Networks and Communications (EuCNC)*, pp. 128–132.

[20] Y.-D. Lin, C. C. Wang, Y.-J. Lu, Y.-C. Lai, and H.-C. Yang, "Two-tier dynamic load balancing in SDN-enabled wi-fi networks," pp. 1–13. [Online]. Available: https://link.springer.com/article/10.1007/s11276-017-1504-3

[21] Open networking foundation. [Online]. Available: https://www.opennetworking.org/

[22] Estinet. [Online]. Available: http://www.estinet.com/es/

[23] "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool." [Online]. Available: https://iperf.fr/

[24] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," vol. 17, no. 1, pp. 1–14. [Online]. Available: http://dx.doi.org/10.1016/0169-7552(89)90019-6

Ying-Dar Lin is a Distinguished Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose, California, during 2007-2008, and the CEO at Telecom Technology Center, Taipei, Taiwan, during 2010-2011. From August 2017, he has been jointly appointed as the Vice President of National Applied Research Labs (NARLabs). Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL, www.nbl.org.tw), which reviews network products with real traffic and has been an approved test lab of the Open Networking Foundation (ONF) since July 2014. He also cofounded L7 Networks Inc. in 2002, which was later acquired by D-Link Corp. His research interests include network security, wireless communications, and network cloudification. His work on multihop cellular was the first along this line, and has been cited over 850 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014-2017), and ONF Research Associate. He currently serves on the editorial boards of several IEEE journals and magazines, and is the Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST). He published a textbook, Computer Networks: An Open Source Approach (www.mhhe.com/lin), with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).

Chun-Hung Hsu is a manager at Network Benchmarking Lab, National Chiao Tung University, Taiwan. He received his M.Sc. degree from the Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, Taiwan. His current job focuses on developing test methodologies with real-world network traffic to test Ethernet switches, routers, and Wi-Fi interconnected devices.

Minh-Tuan Thai received his M.Sc. degree in computer science from National Chiao Tung University (NCTU), Taiwan in 2013. He is currently pursuing his Ph.D. in computer science at NCTU. His research interests include software-defined networking, network function virtualization, and 5G mobile edge computing.

Chein-Ting Wang received his M.Sc. degree in communications engineering from National Chung Cheng University, Taiwan in 2013. He is currently pursuing his Ph.D. in computer science at National Chiao Tung University, Taiwan. His research interests include software-defined networking and network function virtualization.

Yi-Jen Lu received her M.Sc. degree in computer science from National Chiao Tung University, Taiwan in 2015. She currently works as a senior software engineer for ZYXEL, a network equipment supplier.

Yi-Ta Chiang received his M.Sc. degree from Institute of Network Engineering, National Chiao Tung University, Taiwan in 2010. He is currently a senior engineer at Network Benchmarking Lab, Taiwan. His work focuses on the design of network equipment and automatic testing framework.