Int. J. Nonlinear Anal. Appl. 10 (2019) No. 1, 177-183
ISSN: 2008-6822 (electronic)
https://dx.doi.org/10.22075/IJNAA.2019.4062

A new SDN-based framework for wireless local area networks

Ahmad Jalili*

Faculty of Computer Engineering, Gonbad Kavous University, Gonbad Kavous, Iran

(Communicated by J. Vahidi)

Abstract

Nowadays wireless networks are becoming important in personal and public communication and growing very rapidly. Similarly, Software Defined Network (SDN) is an emerging approach to overcome challenges of traditional networks. In this paper, a new SDN-based framework is proposed to fine-grained control of 802.11 Wireless LANs. This work describes the benefits of programmable Access Points. The proposed framework is evaluated on a prototype. We show that this architecture allows us to react to application needs, adjust network configuration for a priority application, detect network problems, and improve the user experience of the network. Our architecture relies on the IEEE 802.11 standard, and exploits it to collect information from the stations.

Keywords: Software Defined Networks, Wireless Networks, Control Plane, Wireless LANs, Data Plane.
2010 MSC: 90B18

1. Introduction

Wireless networks have become ubiquitous and dense. Emerging high-speed network standards (e.g. 802.11ad) are migrating to higher frequencies, which are absorbed by walls, thus requiring the densification of APs. Thus, a more refined control of all the network devices is needed to increase future wireless networks scalability in these environments. Current management architectures for Wireless LANs (WLANs) employ proprietary controllers. These controllers perform network-wide optimizations such as adjustment of transmission power at each AP, selection of best operational channels, faster client mobility, and enhanced traceability, as well as Quality of Service (QoS) policy

*Corresponding author
Email address: jalili@gonbad.ac.ir (Ahmad Jalili*)

Received: December 2018 *Revised:* November 2019

(rate limiting) enforcement and security. However, those controllers only manage compatible devices, usually from a single manufacturer, since they rely on proprietary interfaces and Management Information Bases (MIBs) [1]. Similarly, SDN is a promising architecture that can overcome the challenges facing traditional wireless networks. Unlike traditional networks that both control and data planes are tightly coupled on the same boxes, it decouples control and data planes [2]. Such separation architecture enables administrator/operator to build a simpler, customizable, programmable, and manageable network. In SDN, network owners can dynamically and efficiently configure their network by the external intelligent elements called controllers [3].

This paper presents a Software Defined Networking (SDN) framework for IEEE 802.11 WLANs. The proposed framework refactors the control plane functionalities between the APs and the controller, creating hooks in the AP implementation that trigger events to be treated by the controller. Also, the controller can use getter/setter methods to change the AP behavior, controlling features such as client mobility, association and disassociation, QoS, linklevel parameters and current state, and virtual APs. The contributions of this work is presenting an SDN-based framework for Wi-Fi networks, that only depend on the Linux AP implementation, so it can be used on any device that supports that implementation. It focuses only on IEEE 802.11/2016 which includes previous amendments, and because of that it provides more fine-grained control of the APs.

The remaining of this paper is organized as follows. The related work is discussed in Section 2. The proposed framework and its components are described in Section 3. As a proof of concept, we develop a prototype of our framework. Section 4 show the performance evaluation and results obtained on a testbed. Finally, Section 5 presents our conclusions.

2. Related Works

In this section, we provide an overview of the existing control methods for SDNs and for wireless networks. SDNs are characterized by the existence of a control system (software) that controls the switching fabric of routers and switches (the data plane hardware) through a well defined programming interface. One programming interface that adheres to the SDN philosophy is OpenFlow [4], which is tailored to control routers and switches. OpenFlow is currently the de facto standard in SDN programmability.

Wireless networks have their particularities with regards to wired networks, mostly related to the highly variable characteristics of the links and the mobility of the users. Thus, the abstractions and primitives proposed by wired SDN approaches (e.g. OpenFlow) are not sufficient to perform the proper control of a wireless network.

Many advanced wireless transmission technologies have been developed to maximize spectrum utilization in wireless networks. Among them, Software-Defined Radio (SDR) permits control of wireless transmission strategy via software [5]. Given its similar nature, the SDR technology can be easily integrated with SDN. For example, Bansal et al. point out that many computationally intensive processing blocks are common at the physical layer of all modern wireless systems, differing only in configurations [6]. One instance is that almost all wireless systems use Fast Fourier Transform (FFT) with probably different FFT-lengths. This observation motivates them to propose OpenRadio to decouple wireless protocol definition from the hardware and use a declarative interface to program wireless protocols. In another study, Murty et al. present Dyson where a wireless NIC driver is modified to support statistic collection and Dyson command API [7]. Clients and Access Points (APs) passively report measurement information, including total number of packets, total packet size, and total airtime utilization, to a central controller. The central controller can manage link

association, channel selection, transmission rate and traffic shaping for both clients and APs through the API based on current and historical measurement information.

Systems like CENTAUR [8] and Dyson [7] propose architectures that assume heavy modifications to the IEEE 802.11 client side logic. Systems like DenseAP [9] and DIRAC [10] assume legacy 802.11 clients. However, these systems have certain performance limitations. For instance, they cannot work around the delays involved with a handoff by a client. With regards to improving performance in the network, Rozner et al. describe traffic aware channel assignment in enterprise WLAN deployments [11], and Bahl et al. [12] propose the use of desktop computers as monitoring stations in order to collect statistics about the network, which can be used for making network management decisions. For Odin, these works represent typical applications that we expect to eventually support. Odin is built on top of an OpenFlow controller [13].

3. The Proposed Method

The proposed framework is an SDN-based architecture for IEEE 802.11 WLANs with many APs and clients. This architecture has three types of devices: the controller, the OpenFlow-enabled switch and modified APs. The controller runs on a computer connected to the infrastructure, i.e, in the wired network or virtualized in the cloud. The modified AP are Aps that are modified to run our framework code. A modified AP has two components: wired and wireless. The wired component is a configurable switching element that supports the OpenFlow protocol. Since OpenFlow does not provide a control interface for QoS and for wireless components, the proposed framework adds this functionality using the OVSDB management protocol defined in RFC 7047. The wireless components receive commands from the framework Controller via a secure channel using a separate connection that handles the protocol. This approach makes our solution OpenFlow independent, i.e., we can divide the architecture into wireless and wired control protocols, and act independently or in a coordinated fashion on both networks. Thus we can control the wireless components of an AP that does not support OpenFlow. Our framework uses a communication protocol based on two mechanisms: (i) Get-Set model: A controller sends a get message to the AP. The requested information is read, modified or deleted. This feature supports a proactive behavior from the controller. (ii) Publish-Subscribe model: The AP offers a set of events to which the controller can register. This supports a reactive, event-based operation. The framework API is designed upon an object oriented approach that works with entities having properties and methods, and handling events. Our entities are physical or virtual objects that can be configured or observed. The properties are accessed by the controller using get/set methods or publish/subscribe methods. Finally, entities can have events. One such event could be a wireless client requesting an association. The controller registers with the AP which event it wishes to receive a message, and if applicable the threshold for that information. Figure 1 shows the entities, properties and events of Ethanol. To improve readability we have omitted all getter and setter methods. All methods with the "ev" prefix correspond to an event that can be managed by the controller.

4. Results and Performance Evaluation

This section describes the experiments performed on the prototype and shows the results of the experiments. One way to improve the throughput of wireless networks is to control which client is associated to each AP. Suppose that a network has two APs, both covering the same location, in order to accommodate a larger number of clients. Those APs are allocated to non-interfering channels, and the objective is to assign the same amount of clients to both APs for load-balancing.

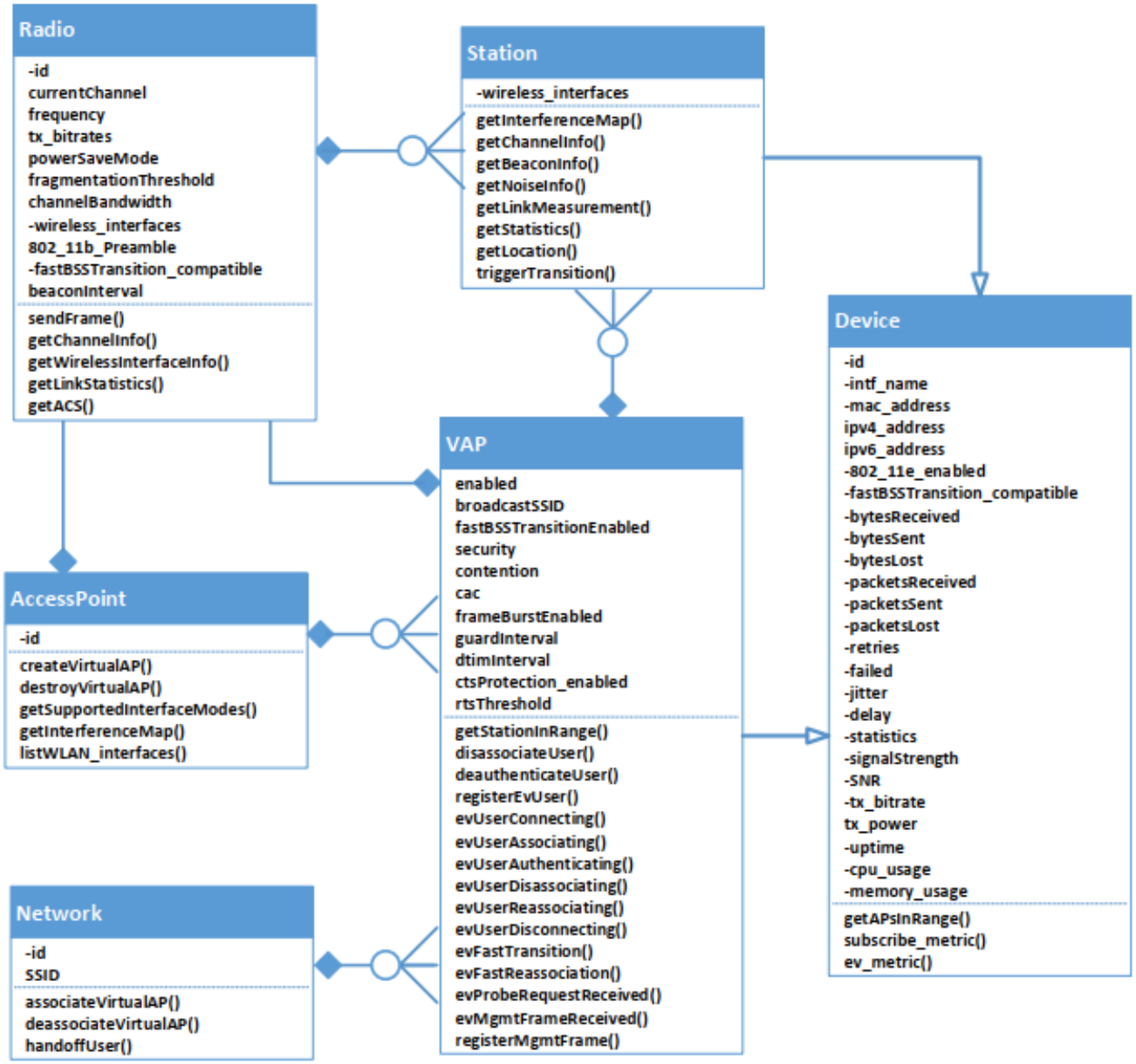

Figure 1: API model in the proposed framework

Nowadays, clients associate based on signal strength. So, if an AP has a slightly better signal, most clients will attempt to connect to the AP with the strongest signal, generating a load imbalance. Commercial wireless controllers use proprietary algorithms that rely on generic network behavior to control the association process. The network administrator can only set a few parameters, but cannot define the behavior she really wants. In our proposed framework, the controller is notified whenever new association attempts occur, as shown in Algorithm 1. Therefore, the controller is able to deny the association or redirect it to other APs in order to perform load balancing (using IEEE 802.11r), or using application-specific rules. In this experiment we implement a very simple load-balancing scheme where the controller divides the number of clients equally among APs using only the ResultCode in the MLME-ASSOCIATE message. Each client association request is forwarded to the controller by the AP. The controller decides if the client should be allowed to associate to the AP based on the actual number of stations already associated to the wireless network. Observe that the framework does not need to modify the clients software to control the distribution of clients in each AP.

Algorithm 1: \\Client Cardinality-aware association control

1 : `Function:`$evUserConnecting(userMAC, ap)$
2 : $apsInRange \leftarrow Station : getStationByMAC(userMAC) : getAPsInRange()$,
3 : $minClients \leftarrow \underset{(ap \in apsInRange}{(argmin)} \{ap.numClients()\}$;
4 : `Return:`$ap.numClients() = minClients \rightarrow Allowconnectionifequals$

In the experiment, when a wireless client C wants to connect to an AP, first it has to go through the association process: C sends an MLMEASSOCIATE.request message to the AP, which responds with an MLME-ASSOCIATE.response message. We can control the clients distribution in the APs by changing the confirmation response, forcing the C to look for other APs. Refusing re-association requests and forcing client disconnection can make the overall handover slower, but there is no other way to control the association without changes in the clients. Fast BSS transition, as proposed by IEEE 802.11r, relies on a client decision to change access points. BSS transition management enables an AP to request a client to transition to a specific AP, using MLME-BTM.request message, but this feature is not present on our devices. The association control algorithm shown in this section is extremely simple compared to the stateof-the-art load-aware methods. The implementation could be aware of network traffic, but to keep it simple we assume that all stations are transmitting at full speed so there is a strong correlation between connected stations and overall throughput. Note that the current algorithm even lead to performance problems by repeatedly denying customer access. However, our architecture can be used to implement more effective approaches such as a handover from an overloaded AP to a lightly loaded as in [14]. In the following, we show that the framework can handover a user. Here, we have two modified APs with the same ESSID, and covering the same area. The wireless clients connect to them. All clients are Samsumg Galaxy Tab with Android 4 and an 802.11bgn interface. Figure 2 shows the number of associated clients on two modified APs on the y-axis. Clients are manually connected to the ESSID. For this reason, the x-axis of Figure 2 shows the request connection sequence number generated by the controller.

The first stage of the experiment, represented by the interval $0 \leq x \leq 13$, consists in activating our wireless stations. During this period, only one modified AP is running (AP2), shown as a dashed blue line. The behavior shown is the expected behavior for a IEEE 802.11 association process. Each request is allowed, shown in the graph as a monotonically increase of the blue curve. There is only one AP, so no load balancing is performed. Between steps $13 \leq x \leq 15$, the second AP is activated (AP1), shown as a solid line in red. Within $15 \leq x \leq 20$, we manually force some stations to disconnect,

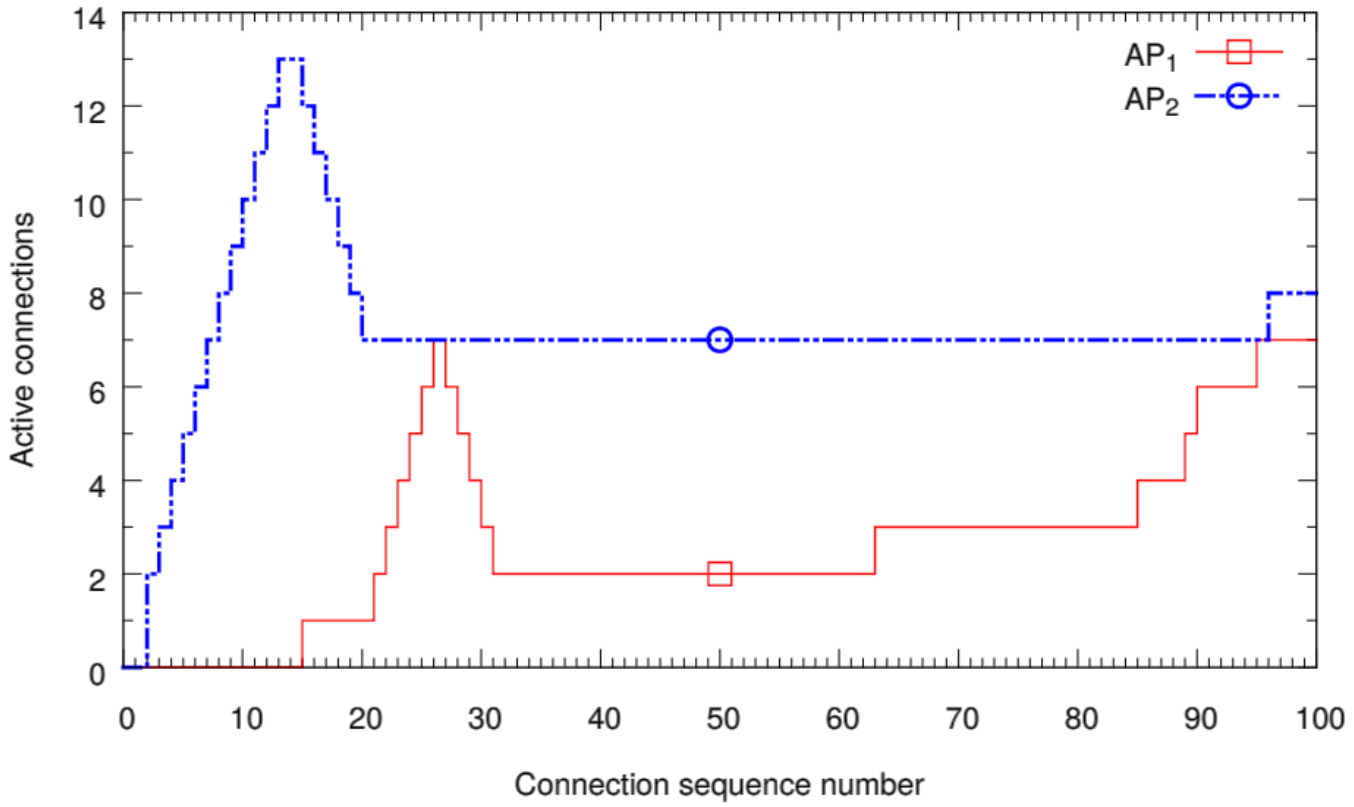

Figure 2: Controlling client association in an 802.11 network

shown as a decrease in AP2 line. In $21 \leq x \leq 27$, the disconnected stations attempt to reconnect. The controller only allows clients to associate to AP1, which has less connected clients. Until the number of stations is equal on both APs, the controller does not allow associations to AP2. In this way, in $32 \leq x \leq 62$, several requests are denied. During this period, both lines did not undergo transition, remaining parallel to the x-axis, thus indicating no change in the active connections. In an IEEE 802.11 network, the association decision is made by the wireless station. So in this period, stations seek connection to AP2 (better signal), and the association request is denied by the controller. As the number of clients connected to AP2 is higher than AP1, the controller performs load balancing. We observe that there are five stations trying to make a new connection, which implies that in this period we have on average 6 failed attempts per station. Although the whole period lasts less than one second, this may imply in an undesirable interruption for the applications that run on the stations. A better alternative would be not to perform a forced disconnect, but instead force the station to handover to a less loaded AP. But as in this experiment we intend to only to show the capabilities of Ethanol to respond to IEEE 802.11 association and reassociation messages, we kept the procedure simple and disregarded this problem. In $x = 63$, a client requests an association to AP1, which is allowed, since AP1 has the lowest number of active connections. This scenario shows that the proposed architecture is able to control the number of clients associated to the APs, making a simple load balancing. Besides deciding based the number of clients, the controller could steer clients based on other metrics, such as link quality, communication speed (e.g. 6 Mbps or 54 Mbps), or the type and amount of traffic of each client.

5. Conclusion

This paper describes an SDN approach for the control and management of wireless networks. This framework provides an API for mobility of wireless networks. We argue that SDN-enabled APs will also be used for the creation of context and location aware services. We present the architecture of a SDN-enabled WLAN, as well as the methods, properties, and events of the control API. The proposed framework is evaluated on a prototype developed with Linux computer acting as APs. The experiments show that the network performance can be enhanced by programmable APs, controlling user association and handover.

Acknowledgements

The author has been supported by Gonbad Kavous University within the project with title "proposing a framework for improving the functionality of control plane in Software Defined Networks" with grant no. 6/717.

References

[1] A. Jalili, M. Keshtgari, and R. Akbari, Optimal controller placement in large scale software defined networks based on modified NSGA-II, Applied Intelligence, 48(9), 2809-2823, 2018.

[2] A. Jalili, H. Nazari,S. Namvarasl, and M. Keshtgari, A comprehensive analysis on control plane deployment in SDN: In-band versus out-of-band solutions. IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI) (pp. 1025-1031), IEEE, 2017, December.

[3] A. Jalili,M. Keshtgari,R. Akbari, and R. Javidan, Multi criteria analysis of controller placement problem in software defined networks. Computer Communications, 133, 115-128, 2019.

[4] D. Erickson, The beacon openflow controller. Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 13-18). ACM, 2013, August.

[5] T. Ulversoy, Software defined radio: Challenges and opportunities , IEEE Commun. Surveys Tuts., 12(4), 531550, 2010 , May.

[6] M. Bansal, J. Mehlman, S. Katti, and P. Levis, OpenRadio: A programmable wireless dataplane, Proc. 1st Workshop HotSDN, 109114 , 2012.

[7] R. Murty, J. Padhye, A. Wolman, and M. Welsh, Dyson: An architecture for extensible wireless LANs, Proc. USENIXATC, p. 15 , 2010.

[8] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, CENTAUR: realizing the full potential of centralized wlans through a hybrid data path. Proceedings of the 15th annual international conference on Mobile computing and networking, 2009.

[9] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill, Designing high performance enterprise Wi-Fi networks. Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, 2008.

[10] P. Zerfos, G. Zhong, J. Cheng, H. Luo, S. Lu, and J. J. R. Li, DIRAC: a software-based wireless router system. Proceedings of the 9th annual international conference on Mobile computing and networking, 2003.

[11] E. Rozner, Y. Mehta, A. Akella, and L. Qiu, Traffic-Aware channel assignment in enterprise wireless LANs, IEEE ICNP , 2007.

[12] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill, Enhancing the security of corporate Wi-Fi networks using DAIR. Proceedings of the 4th international conference on Mobile systems, applications and services, 2006.

[13] 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., 38,6974, 2008, March.

[14] K. Nahida, C. Yin, Y. Hu, Z. A. Arain, C. Pan, I. Khan, Y. Zhang, and G. M. S. Rahman, Handover based on ap load in software defined wi-fi systems, Journal of Communications and Networks, 19(6), 596-604, 2017,December.