Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 9246824, 11 pages
https://doi.org/10.1155/2018/9246824

Research Article

A Software Defined Wireless Networking Approach for Managing Handoff in IEEE 802.11 Networks

José Quaresma Filho,[1,2] Nailson Cunha,[2] Robertson Lima,[2] Eudisley Anjos,[2] and Fernando Matos[2]

[1]*Academic Unit of Informatics, IFPB, João Pessoa, PB 58015-430, Brazil*
[2]*Centre of Informatics, Universidade Federal da Paraíba, João Pessoa, PB 58051-085, Brazil*

Correspondence should be addressed to Fernando Matos; fernando@ci.ufpb.br

Received 16 April 2018; Revised 8 August 2018; Accepted 10 September 2018; Published 21 November 2018

Academic Editor: Vicente Casares-Giner

In Wireless Local Area Networks (WLAN) with more than one access point (AP), the handoff process plays a crucial role to guarantee the user service continuity. Usually initiated by the client's equipment, it occurs smoothly on the order of seconds. However, despite being functional and well-established, this process can be inadequate in scenarios where users are executing multimedia applications, such as real-time video streaming or VoIP. For these applications, those few seconds may cause loss of packets, resulting in loss of essential information. Because of that, this study proposes a Software Defined Wireless Networking (SDWN) approach, in which a controller decides when to initiate the handoff process and chooses the AP the client's device must connect. This approach was implemented in a testbed scenario and the results have shown its efficiency by decreasing the handoff delay and providing more stability to the process.

1. Introduction

In IEEE 802.11 networks, handoff is a process where a client device disassociates from one access point (AP) and associates to another. It is a functional and well-established process that takes few seconds to complete in usual conditions. For simple applications, such as web browsing and email, it works smoothly and as expected. However, in delay sensitive applications (e.g., real time video streaming, VoIP), those few seconds it takes to execute may compromise the fidelity and/or the comprehension of the communication.

There are basically two main reasons for this delay. The first one is that the client station (STA) is in charge of deciding when to initiate the handoff, which occurs during the detection phase. However, this phase is not standardized, which leads to the situation where each equipment vendor is in charge to implement it [1]. Usually, the devices trigger the handoff when one or more thresholds are reached, such as a high number of lost frames or poor signal strength [2]. The problem is that each vendor uses their own threshold values, thus causing a variation in the detection phase delay. The second reason lies on the discovery phase that is when the STA searches for a new AP to connect. First, the client station scans all channels looking for possible APs. Afterwards, it has to wait for the AP responses, which can take a considerable time depending on the traffic volume on that AP.

The aforementioned reasons heavily contribute to the delay in the handoff. Besides, this vendor-dependent situation hampers flexibility and innovation, since it is necessary to wait for updates or patches to use a new feature. Because of that, this work proposes a solution called Detection and dIScovery Control in Handoff (*DISpatCH*), which aims to improve the handoff process by triggering and controlling the detection and discovery phases. *DISpatCH* is based on Software Defined Wireless Networking (SDWN) principles, where a controller decides when to trigger the handoff process (detection phase) and selects the AP the client device should connect to, thus removing from the STA the responsibility to take care of these issues. A prototype was implemented and experimental tests were carried out to verify *DISpatCH* efficiency. The results showed that by using *DISpatCH* it was possible to decrease the handoff process

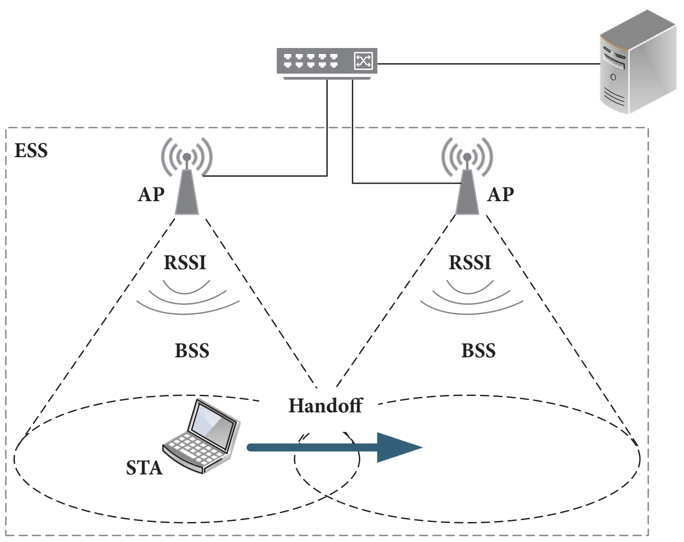


Figure 1: Handoff.

time, thus improving the quality of the communication. Moreover, the controller opens up the possibility to have a more fine-grained control on the process, by using other decision parameters than the signal strength and establishing more appropriate thresholds according to the network status. Finally, the *DISpatCH* approach provides flexibility and agility in the handoff process since it is not necessary to wait for updates from vendors.

The rest of this paper is organized as follows. In Section 2 concepts about the handoff process and SDWN are presented, while in Section 3 some related works are discussed. The proposed solution is outlined in Section 4. In Section 5, the experimental tests are presented and the results are discussed. Finally, in Section 6 the conclusions are drawn along with some future work.

2. Background Information

Some main concepts about handoff and SDWN are used in *DISpatCH*; therefore, this section presents these concepts in order to clarify the proposed solution.

2.1. Handoff. Mobility is one of the most important features of a wireless network. In these networks, two or more access points (AP) create an Extended Service Set (ESS). An ESS is composed of two or more interconnected Basic Service Sets (BSS), which appear to the client as a single BSS. The client station (STA) constantly measures the signal strength received from the APs, the so-called Received Signal Strength Indicator (RSSI). Based on the RSSI or on the number of retransmissions, the STA may decide to move within the same ESS and associate to another AP. This process is called handoff and is illustrated in Figure 1.

According to the IEEE 802.11 standard [3], the handoff operates in three phases: discovery, authentication, and reassociation. However, at first, the STA must detect the necessity to disconnect from its current AP and to connect to another.

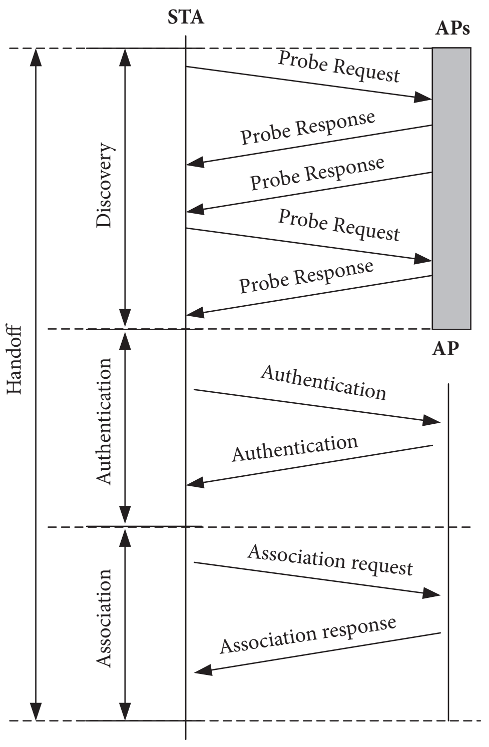


Figure 2: IEEE 802.11 standardized handoff phases [3].

This initial action is called detection phase and it is not standardized, thus leaving up to the vendors the implementation of this decision. Figure 2 shows the standardized handoff phases along with the messages exchanged in each phase.

The nonstandardized detection phase is responsible for initiating the handoff. Actions during this step may vary depending on the entity initiating the process. When initiated by the network, the detection starts with a *Disassociation*

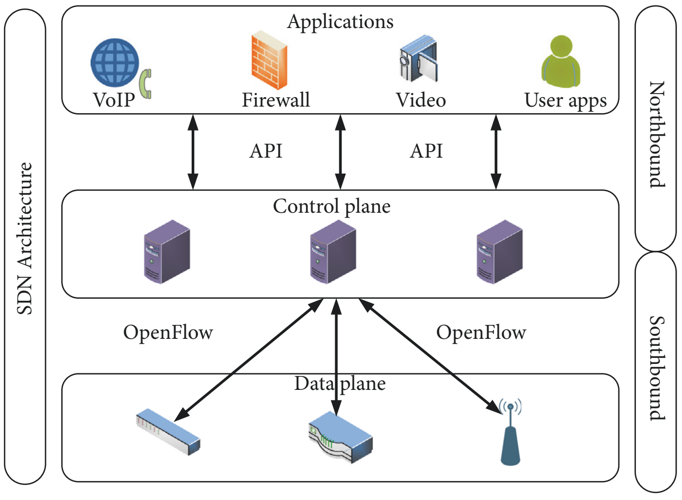


Figure 3: SDN architecture, adapted from [4].

frame sent by the AP to the STA. However, it is usually initiated by the STA when it detects failures in frame transmissions or poor RSSI values. Since it is not standardized, each vendor uses the threshold it considers more appropriated, which can generate a large variation in the delay imposed by this phase.

Along with the detection, the discovery phase is one of the main reasons responsible for the handoff delay. The discovery may be active or passive. In passive discovery, the STA waits for *Beacon* frames sent by the APs to choose one of them to connect. In the active discovery, the STA sends *Probe Request* frames in all available channels. The 802.11 standard specifies two parameters (*MinChannelTime* and *MaxChannelTime*) that determine how long a STA must wait for a *Probe Response* after sending a *Probe Request*. *MinChannelTime* and the *MaxChannelTime* are the minimum and maximum times the STA must wait on a channel, respectively. These values are configurable, but most APs use 20 ms for the *MinChannelTime* and 40 ms for *MaxChannelTime*. By using this strategy, the time to scan 14 channels would fall between 280 ms and 560 ms, which is considerable for multimedia applications.

Finally, the times spent in the authentication and reassociation phases were optimized by IEEE in the 802.11f, 802.11r, 802.11k, and 802.11v amendments and usually take 50 ms or less to complete [5].

2.2. Software Defined Wireless Networking. Software Defined Networking (SDN) is a new network paradigm that removes the control functions from the equipment, thus creating a control plane and a data plane, as shown in Figure 3. The control plane is responsible for deciding what to do with each flow in the network and the decisions taken by the control plane are enforced in the equipment of the data plane. This separation of functions allows making the data plane programmable, thus removing the need to manually configure each equipment [6]. Software Defined Wireless Networking (SDWN) is the use of SDN concepts in wireless networks. By using a controller in the control plane, SDWN facilitates the creating of new adaptive mechanisms according to different applications and user demands, such as mobility (handoff), security, and quality of service (QoS) [7].

OpenFlow is a southbound protocol used between the controller and the network devices (switches, routers, and access-points) [8]. According to [9], the goal of OpenFlow was to provide the means to test experimental protocols in production networks without harming its operation. OpenFlow operates by adding entries in flow tables located in each network device. These entries define the actions to be taken according to each flow, for instance, dropping packets, forwarding packets to a specific port, performing broadcast, among others.

When a SDN network starts operating, all flow tables are empty. When the network entry equipment (e.g., switch) receives a packet, it does not know what to do. Therefore, it forwards the packet to the controller, which inspects the packet header. By matching the information on the packet header with predefined rules, the controller decides what to do with the packet and informs this action, through OpenFlow, to the switch. A new entry is then installed in the flow table. From now on, every time a packet with the same characteristics arrives, the switch knows the action it must take.

3. Related Work

Improving handoff performance and adapting it to fulfill quality of service (QoS) requirements of multimedia applications have been a major challenge for industry and academia. Several solutions have been proposed over the years, some of which are described below.

Traditionally, when the handoff process initiates there is an interruption in the data transmission. With this in mind, the authors in [10–13] added a second WiFi interface in the APs. By using two interfaces, the APs maintain data transmission while performing the handoff at the same time. However, the APs on the market are usually equipped with one interface only, thus limiting the use of such approaches. In [14, 15] the authors propose to reduce handoff time by using mechanisms to determine STA location and the nearby APs. Therefore, the STA can search for APs based on its location (selective scan). However, these solutions still maintain the STA in charge to initiate the handoff process. In the solution proposed in [10], the AP in which the STA is currently connected decides the destination AP. This decision is taken by using messages exchanged between the current and nearby APs. Despite the advantages of this method, the authors performed only simulation tests. Experimental or real test scenarios would be welcome.

In [16] BIGAP is presented, which is a highly scalable and efficient architecture for enterprise WiFi networks. Seamless handoff is achieved by switching through the channels of the AP, which supports Dynamic Frequency Selection (DFS). The proposed solution also requires two WiFi interfaces, one for traffic network and the other for passive monitoring of wireless statistics. Data collection is performed by the interface in promiscuous monitor mode (hopping over all channels); then the data is sent to BIGAP controller, which is responsible for taking decisions regarding the handoff. One limitation is that the required hardware is very specific. It also requires a large number of available channels to work without collision issues.

Regarding SDN-based solutions, the authors in [17] presented a framework for carrier grade Wi-Fi networks that allows performing load balancing, seamless handover, unified authentication, and traffic offloading with the aid of a controller. It uses Software Access Points (SAP) in each AP to abstract the connection between the user equipment and the AP. However, despite the proposal showed good throughput results during SAP-based handoff, it lacks experimental tests regarding APs' traffic. In [18], the authors presented M-SDN: a management scheme that reduces traffic pause time caused by STA-initiated handoff. This is accomplished by providing a handoff preparation step, which consists of calculating the routes from the current connected AP to each candidate AP, and then duplicating the packets to each one of them (N-casting). Once the handoff is complete, unused created flows are cleared.

In [19], the authors propose a solution based on fuzzy logic that uses the RSSI values to predict the STA is moving; i.e., a handoff may occur. After that, an algorithm in the controller uses the RSSI value and bandwidth to decide if the handoff must really occur and to select the AP the STA must connect. A drawback of this work relies on the fact that the STA is responsible for testing the RSSI values against a predefined threshold to infer if it is moving. This approach resembles the vendor-dependent problem, since it is necessary to update the threshold values in each STA every time adjustment is needed, thus hampering flexibility and agility. In [20], the authors demonstrate the development and tests of a SDN enterprise solution based on virtual APs. In this work, the SDN controller manages handoff by exchange messages with the APs, which resembles our proposal. However, this solution also relies on the necessity of an extra Wi-Fi interface in each AP to monitor traffic in other channels.

In [21], a SDN architecture is proposed to obtain statistical data of the network usage, as well as information about the traffic load in each AP. This information is then used to choose the destination of AP during the handoff, aiming to improve the load balancing among nearby APs. However, this work does not decide when the handoff should take place. The solution proposed in [22] uses RSSI levels perceived by the STA to predict which AP the STA should connect during handoff. Afterwards, the controller modifies the APs' flow tables and sends multicast packets addressed to the STA to both the source and destination AP. This strategy guarantees the client receives the packets as soon as it associates to the destination AP, thus minimizing packet losses. Although this work predicts the destination AP, it does not take into consideration the detection phase.

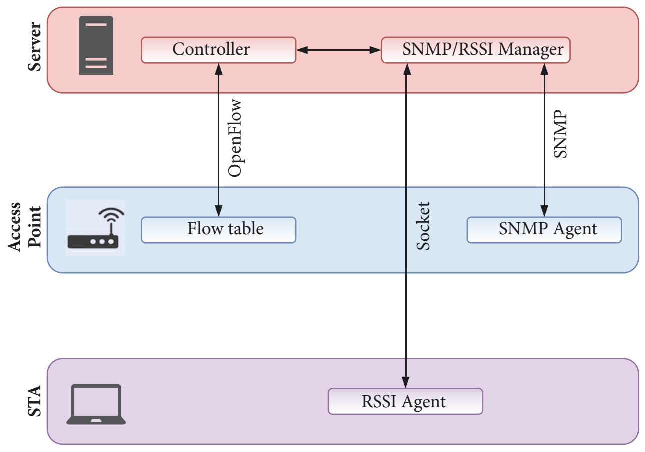


Figure 4: *DISpatCH* architecture.

The proposal of [23] presents a design scheme based on AP traffic load using the SDN paradigm. Each AP will transmit to the controller information about the current traffic load. When one STA is located at an overlapping coverage area, the controller will evaluate if the handoff is necessary, i.e., when the STA is moving or one AP is overloaded. The handoff is initiated by the controller by sending a disconnection message to the current AP. It also sends a message forcing the destination AP to accept the STA connection. This solution showed significant improvement in throughput and jitters aspects. One limitation is that the authors performed only simulation tests and there is no clear information on how the overlapping coverage area is discovered.

4. DISpatCH

As mentioned earlier, the detection and discovery are the phases that most influence the handoff process delay. Therefore, this work focused on these two phases by proposing Detection and dIScovery Control in Handoff (*DISpatCH*), which is a solution based on the SDWN approach. In *DISpatCH*, a controller manages the flow tables in the APs and constantly monitors the network in order to take two important decisions: (i) when to initiate the handoff process (detection phase) and (ii) which AP the STA must connect to discovery phase. Figure 4 presents the *DISpatCH* architecture.

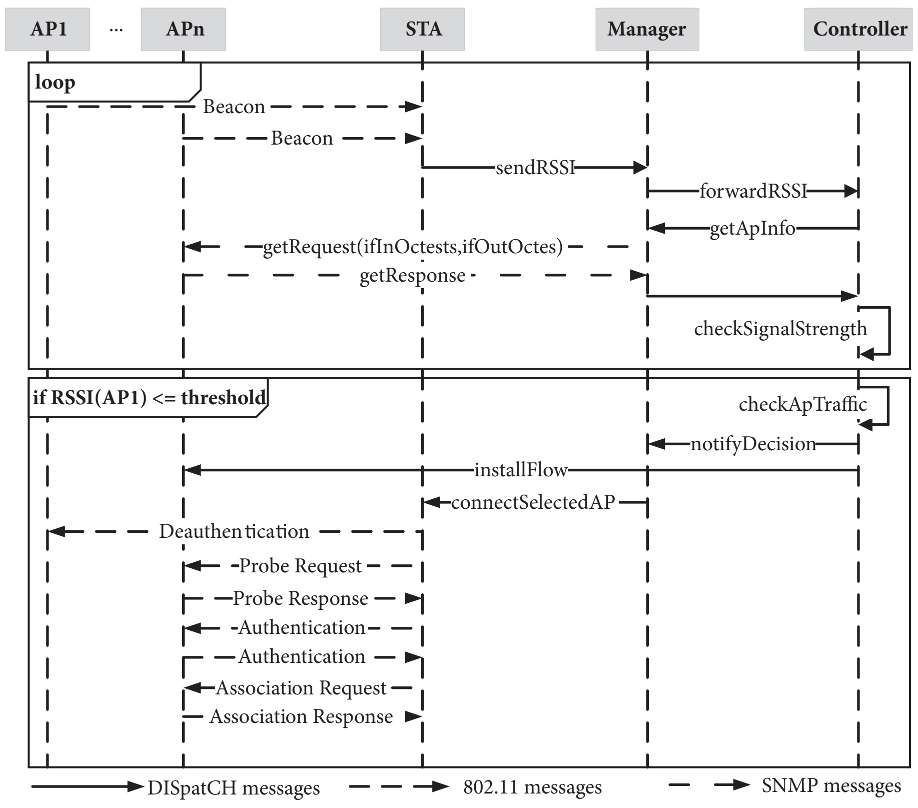


Figure 5: Sequence diagram of the *DISpatCH* operation.

DISpatCH is composed of four modules: the *SNMP Agent*, the *RSSI Agent*, the *SNMP/RSSI Manager*, and the *Controller* itself. The *RSSI Agent* resides on the STA and has two functions. First, it periodically sends to the *SNMP/RSSI Manager* the RSSIs perceived by the STA from all APs within its range. Second, it forces the STA to connect to the AP chosen by the *Controller*. The *SNMP Agent*, which is located in every AP, is responsible for measuring the amount of traffic at the Wi-Fi interfaces and sending this information to the *SNMP/RSSI Manager*. That information is taken into account when deciding which AP the STA must connect to.

In the server the *SNMP/RSSI Manager* resides, which communicates with both the *RSSI Agent* and *SNMP Agent* modules. It forwards the RSSI and the amount of traffic values received from those modules to the *Controller*. It also notifies the *RSSI Agent* about the *Controller* decision. Finally, the *Controller* is the module responsible for selecting, among all possible APs, the one which the STA will connect, as well as triggering the handoff process.

Although the proposed approach depends on executing a piece of software on the STA, it does not compromise its applicability. There are several scenarios in which customized software and/or execute configuration scripts is necessary to install, such as in a campus of a university to use a VPN service, or when contracting a premium service. In such cases, it is feasible to ask the users to execute a configuration script so their user experience can be improved through a faster handoff.

4.1. Operation. In order to clarify how *DISpatCH* works, Figure 5 presents a sequence diagram that illustrates all main activities involved in the decision-making process performed by the *Controller*. In this work, it is assumed as source AP and destination AP, the AP in which the STA is currently connected and the AP the STA will connect to, respectively.

As usual, the APs continuously broadcast *Beacon* frames that are captured by the STA. In the STA, the *RSSI Agent* extracts the RSSI values and sends them to the *SNMP/RSSI Manager* (*sendRSSI*). In its turn, the *SNMP/RSSI Manager* forwards these values to the *Controller* (*forwardRSSI*), which compares the RSSI of the source AP to a defined threshold (*checkSignalStrength*). The *Controller* also periodically asks the SNMP/RSSI Manager (*getApInfo*) for information about the traffic in each AP. The *SNMP/RSSI Manager* then requests this information to the *SNMP Agents* in all APs and forwards the responses to the *Controller*. This traffic information is necessary for the destination AP selection. All these activities are continuously performed until this RSSI threshold is reached.

Let R be the set of APs and T be the defined RSSI threshold. Given a source AP $i \in R$, the *Controller* checks the following conditions:

(i) $RSSI(i) \geq T$: in this case, the *Controller* takes no action whatsoever, since the RSSI from the source AP i is strong enough to maintain the quality of the connection

(ii) $[RSSI(i) < T]$ **AND** $[RSSI(i) \geq RSSI(j), \forall j \in R, i \neq j]$: in this case, the RSSI of the source AP i reached the threshold; however, there is no other AP j with a RSSI value higher than the RSSI from i. Thus, the *Controller* takes no action as well

(iii) $[RSSI(i) < T]$ **AND** $[RSSI(i) < RSSI(j)$ for at least one $j \in R, i \neq j]$: in this case, the RSSI of the source AP i reached the threshold and there is a RSSI value from another AP j that is higher than the RSSI from the source AP. Therefore, the *Controller* decides that the STA must perform the handoff process, which corresponds to the detection phase.

After the *Controller* decides the handoff process will initiate, it must choose the destination AP. If there is only one possible destination AP, then the choice is obvious; otherwise, the *Controller* uses some parameters to decide. The simplest approach is to compare the RSSI values themselves and select the AP with the highest RSSI value. However, in this work, the amount of traffic as a decision parameter for the selection process is also used. Once in possession of such information, the *Controller* selects the destination AP based on the amount of traffic (*checkApTraffic*). Considering TR as the traffic measured in a time interval, TR_{max} as the maximum allowed traffic, $\overline{R}$ as the set of destination APs, and $j \in \overline{R}$ and $k \in \overline{R}$ as two possible destination APs, the *Controller* may face two situations:

(i) $[RSSI(j) \geq RSSI(t), \forall t \in \overline{R}, j \neq t]$ **AND** $[TR(j) \leq TR_{max}]$: in this case, the RSSI of j is higher than the RSSI of any other AP and the amount of traffic in j is less than or equal to the maximum allowed traffic. Thus, j is selected as the destination AP.

(ii) $[RSSI(j) > RSSI(k) > RSSI(t), \forall t \in \overline{R}, j \neq k \neq t]$ **AND** $[TR(j) > TR_{max} \geq TR(k)]$: in this case, the RSSI of j is also higher than the RSSI of any other AP t; however, its amount of traffic is higher than the maximum allowed traffic. Because of that, the *Controller* selects as the destination AP the one with the next highest RSSI value that does not exceed the maximum allowed traffic.

Once the *Controller* selects the destination AP, it must inform its decision to the *SNMP/RSSI Manager* (*notifyDecision*) and install the flow at the destination AP (*installFlow*). Thus, when the STA migrates to the destination AP, the appropriate flow is already installed on it. At the same time, the *SNMP/RSSI Manager* initiates the handoff process by sending to the *RSSI Agent* a message (*connectSelectedAP*) containing the SSID of the destination AP.

To migrate to the destination AP, the *RSSI Agent* executes the command "*iwconfig wlan0 essid SSIDname*", where *SSIDName* is the SSID of the destination AP. Basically, this command results in two actions. First, it forces the STA to send a *Deauthentication* frame indicating that it is disconnecting from the source AP. Second, the command forces the STA to deliver a *Probe Request* frame specifying the SSID of the destination AP. Therefore, only this AP replies with the *Probe Response* frame. Next, the remaining handoff frames are exchanged, thus completing the STA association to the destination AP.

By using *DISpatCH*, the responsibility in deciding when to initiate the handoff process (detection phase) and which AP to connect to (discovery phase) is transferred to the *Controller*. It is important to stress that both the RSSI threshold and the maximum allowed traffic are configurable values. This approach gives two main advantages. First, it increases the flexibility in managing the network. Second, it makes the network less vendor-dependent.

It is worth mentioning that, despite being a very important concern, security is out-of-scope of this work. Nonetheless, when considering the proposed architecture, the *SNMP/RSSI Manager* intercepts the messages from the STAs before they reach the *Controller*. Thus, it is possible to employ security strategies in order to protect the *Controller* against, for example, Distributed Denial of Service (DDoS) attacks from the STAs.

4.2. Estimated Traffic at APs. One of the parameters *DISpatCH* takes into account the amount of traffic at each AP to select the destination AP. To calculate these values, two objects from the Management Information Base (MIB) located at each network equipment are used, which are (i) *ifInOctets*, which specifies the total number of bytes that arrives at the interface, and (ii) *ifOutOctets*, which specifies the total number of bytes that are transmitted from the interface. The values of these two objects are cumulative from the time the *SNMP Agent* is initiated and they are updated at every 15 seconds. Because of that, at every 15 seconds, the octets information are requested and the amount of traffic is calculated. Thus, it is assumed the current traffic at the AP is the current octet values informed minus the last octet values informed, according to (1), where

(i) TR: total amount of traffic at the interface (B/s)

(ii) Ct: current octets requisition

(iii) Lt: last octets requisition

(iv) $TRin$: value of the *ifInOctets* object

(v) $TRout$: value of the *ifOutOctets* object

(vi) T: time of the requisition

$$TR = \frac{TRin(Ct) + TRout(Ct) - TRin(Lt) + TRout(Lt)}{T(Ct) - T(Lt)} \quad (1)$$

5. Experimental Tests and Evaluation

DISpatCH performance was evaluated by comparing its operation to the traditional handoff method, in which the STA makes the decision regarding detection and discovery phases. An initial set of experiments was carried out to verify the efficiency regarding *DISpatCH* decision about the instant the STA must migrate from one AP to another, in other words, the detection phase. In this case, the RSSI was used as the unique decision parameter to trigger the STA migration from the source AP to the destination AP. In a

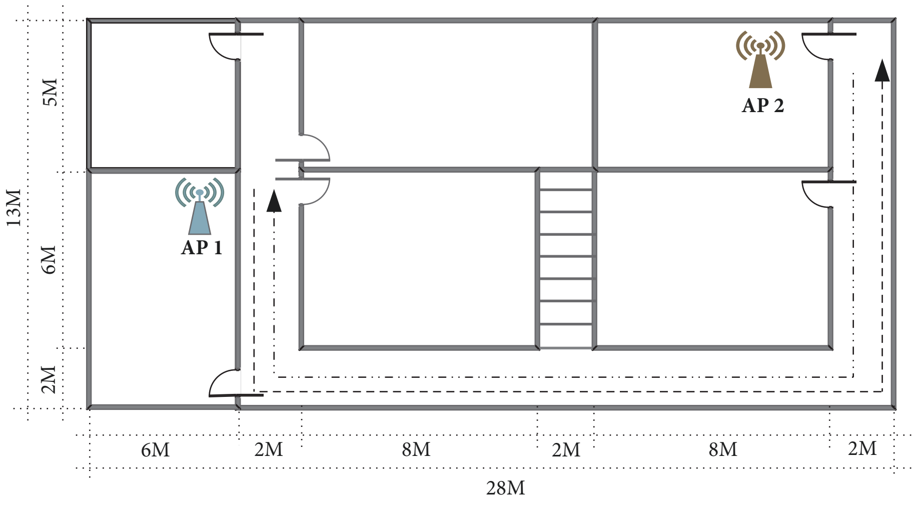


Figure 6: Floor plan of the detection phase experiment.

second set of experiments another AP was added to verify *DISpatCH* efficiency in choosing an appropriate destination AP (discovery phase). The amount of traffic was also used as decision parameter in order to improve the AP selection process. A final experiment was conducted to verify the impact caused by *DISpatCH* and the traditional approaches in a real VoIP application.

5.1. Setup. To perform the tests, the following equipment was used: (i) TP-LINK Wi-Fi routers (WR1043ND model) with a modified OpenWrt firmware in order to support OpenFlow v1.0.0; these routers act as APs; (ii) a Core i5 desktop running the POX controller and the manager software; (iii) a Quad Core desktop running the iPerf software in the server mode, which is responsible for receiving the traffic generated by the STA; (iv) a Core i5 notebook with a Centrino N1030 Wi-Fi adapter running the iPerf software in client mode, which generates UDP traffic at 1 Mbps, thus simulating a multimedia transmission; this machine acts as the STA; (v) a linux virtual machine running the airodump-ng software to capture the traffic. This virtual machine runs on the STA.

The results were analyzed by using Wireshark. Regarding the total handoff process time, it is assumed that the initial time corresponds to the time of the last UDP package transmitted by the STA to the server when the STA still is connected to the source AP, and the final time as the time of the first UDP package transmitted by the STA to server when the STA is already connected to the destination AP. Furthermore, the AP signal strengths were manually attenuated to 10 dBm (10mW) in order to adjust them according to the size of the testbed scenario. This calibration was necessary because when the traditional handoff process took place, the distance between the APs was not far enough to initiate it.

It was defined as threshold (T) the value of -70 dBm; i.e., when the RSSI reaches this value, the *Controller* decides to start the handoff process. Moreover, it was also defined that maximum allowed traffic (*MAX_TR*) is the rate of 40 Mbps. In other words, if the AP traffic exceeds this rate, this AP is not selected as destination AP.

It is worth noting that handoff times are greatly influenced by the hardware components, such as Wi-Fi adapter and AP chipset. Moreover, the low-cost APs used in this work have limited capabilities and their firmware was replaced with OpenWrt, so they could support OpenFlow. These issues can also contribute to the handoff delay times. Nonetheless, the aforementioned equipment was used in both tests (with and without DISpatCH), which guarantees the uniformity of the test scenarios.

5.2. Detection Phase. The objective of the following tests was to verify the *DISpatCH* efficiency regarding the detection phase. Figure 6 shows the testbed floor plan along with the APs' disposition. Two tests were carried out. The first one employed the traditional handoff process, where STA decides when to initiate the process. The second test employed the *DISpatCH* approach, which takes into account the RSSI as decision parameter only. Both tests began with the STA connected to *AP1* (source AP). The STA then was moved at a steady pace towards *AP2* (destination AP), thus resulting, at a certain point, in the STA connection to *AP2* and disconnection from *AP1*. After that, the reverse movement was performed, resulting in the connection to *AP1* and disconnection from *AP2*. Each step was performed 5 times, with a total of 10 connections.

Figure 7 show the total handoff time when comparing *DISpatCH* to the traditional approach. It is easy to observe how fast and more similar the handoff times are in *DISpatCH* when comparing to the traditional approach. In some cases, the handoff time in the traditional process is 4 times higher. This gain is significant when dealing with delay-sensitive applications, such as VoIP or video streaming.

Figure 8 shows the results of the RSSI values collected at the initial time of the handoff process. In the traditional approach, the RSSI considerably degrades. Several times, during the tests using the traditional approach, the RSSI

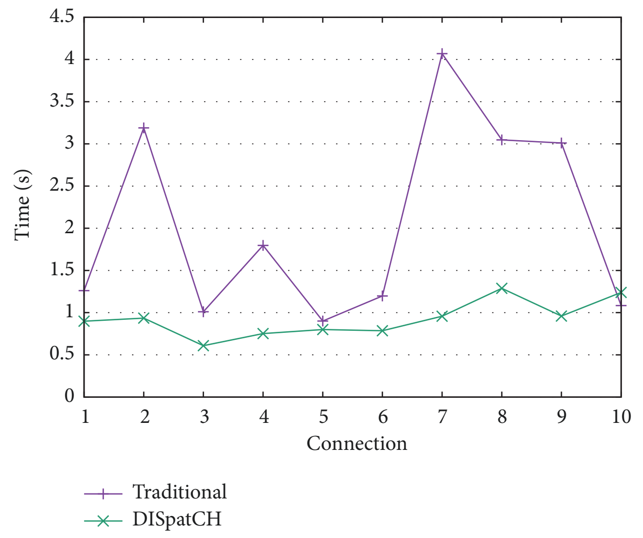


Figure 7: Handoff time.

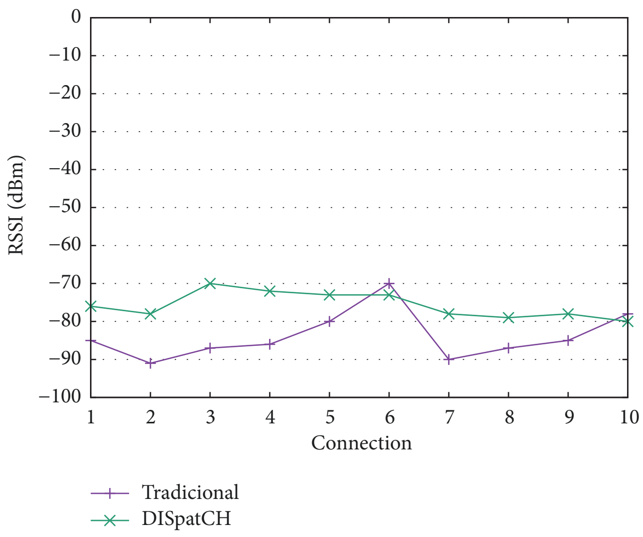


Figure 8: RSSI at the initial handoff time.

received from the destination AP was considerably stronger than the RSSI received from the source AP and yet the STA did not initiate the handoff. Because of this delay in initiating the handoff process, the communication deteriorated, thus resulting in several packet retransmissions. On the other hand, by controlling when to initiate the handoff process, it was possible to maintain the signal strength at an acceptable level during all tests.

5.3. Discovery Phase. The objective of the next tests was to verify the *DISpatCH* efficiency regarding the discovery phase. As in the previous subsection, the tests compared the results between the traditional approach and *DISpatCH*. However, in this case, the *DISpatCH* approach uses the amount of traffic at the APs as decision parameter as well, thus refining the *Controller* decision about the destination AP selection. Figure 9 shows the same testbed floor plan with one extra AP. *AP3* was configured with its maximum signal strength of 27 dBm (501mW), while the other APs remained with the signal strength of 10dBm (10mW). Despite being in the same room, this calibration was necessary to guarantee different RSSI values between *AP2* and *AP3*.

Ten tests were performed and each test began with the STA connected to the *AP1* (source AP). The STA then was moved at a steady pace towards *AP2* and *AP3* (the two possible destination APs). At this point, the *Controller* had to decide which AP the STA should be associated with.

Figures 10 and 11 show the results when the traffic at *AP3* is less than or equal to 40 Mbps and higher than 40 Mbps, respectively. It is worth stressing that *DISpatCH* also considers the amount of traffic to select the destination AP while the traditional approach uses the RSSI values only. Thus, in the first case, since *AP3*'s RSSI is higher than the *AP2*'s RSSI, *DISpatCH* chooses *AP3* as destination AP. In the second case, *DISpatCH* chooses *AP2* as destination AP because *AP3*'s traffic exceeded 40 Mbps. On the other hand, in the traditional approach, *AP3* is always chosen as destination AP, regardless its measured traffic.

Even when considering the amount of traffic along with the RSSI to select the destination AP, the times of the handoff process remain in a narrow range (between 1s and 1,5s) when employing *DISpatCH* approach. They are also lower than the handoff process times when employing the traditional method. Moreover, it is interesting to note that in some cases the handoff time in the traditional method exceeded 6s when the destination AP was congested. When considering multimedia application scenarios, 6s may represent loss of considerable amount of data.

Another test was carried out to verify how much the decision parameter influences the handoff time. In this test (Figure 12), it is compared when the *Controller* uses the RSSI only as decision parameter and when the *Controller* also employs the amount of traffic. In the last case, the AP with the highest RSSI is congested (>40 Mbps), so the *Controller* chooses an AP that is not congested (≤40 Mbps), however, with a lower RSSI. By analyzing the results in Figure 12, it is possible to observe that when using the traffic to select the destination AP, the handoff times are better when compared to those when using RSSI only. Thus, these results show that RSSI is not the best parameter to use when selecting a destination AP.

5.4. Real VoIP Application. All previous experiments were performed by using traffic generator software, thus simulating a real-time application. In the current test scenario, a real VoIP application is executed and a comparison between *DISpatCH* and the traditional handoff approach is performed. The Ekiga software was used to perform a voice call and Wireshark to capture the traffic and codec G711U with a sample rate of 8000 Hz. To carry out the test, the STA was initially associated with *AP1*. The voice call was initiated and at the moment the connection was established, the STA was moved towards the destination AP.

Figures 13 and 14 show the audio waves when using the traditional handoff process and the *DISpatCH*, respectively. It is easy to see the interruption that occurs when

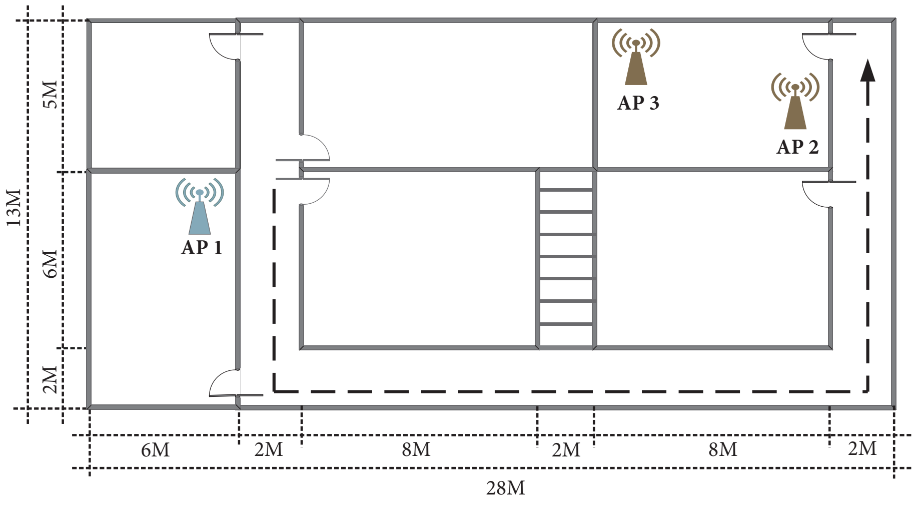


Figure 9: Floor plan of the discovery phase experiment.

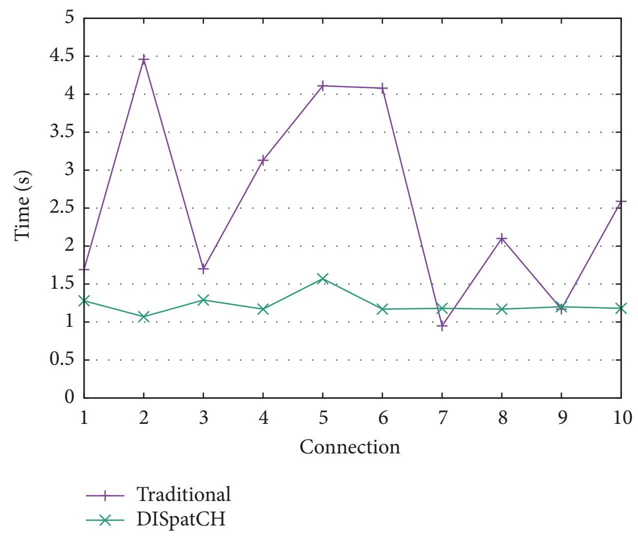


Figure 10: Handoff time with TR(AP3) ≤ 40 Mbps.

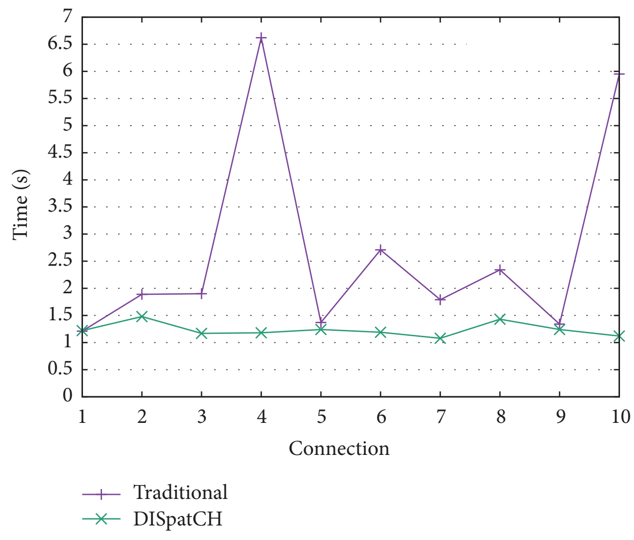


Figure 11: Handoff time with TR(AP3) > 40 Mbps.

the STA is responsible for initiating the handoff. Although the connection was not lost, the communication was clearly affected. On the other hand, when employing *DISpatCH*, the gap resulting from the handoff is considerably smaller, thus showing the effectiveness of the proposed solution.

6. Conclusion

This work proposed a SWDN approach, called *DISpatCH*, to manage the handoff process in IEEE 802.11 Networks. This process is usually initiated by the STA, thus causing handoff to be highly vendor-dependent. Because of that, *DISpatCH* aimed to control the detection and discovery phases, which are the phases that most influence the handoff delay. By developing a controller that decides when to initiate the handoff and which AP the STA must connect to, it was possible to improve the process, which is essential when using real-time multimedia applications.

The performed tests demonstrated the *DISpatCH* efficiency by decreasing the handoff times in about 50% and maintaining these times in a narrow range (1s-1,5s), which improves the communication stability. Moreover, it was also possible to observe that, when the STA initiates the handoff, the RSSI may considerably degrade, thus causing packet retransmissions. On the other hand, by using *DISpatCH*, it is possible to maintain the signal strength at an acceptable level. The tests also showed the benefits in using the amount of traffic in the AP along with the RSSI as decision parameters. Finally, it was tested the two approaches (*DISpatCH* and traditional) during the execution of a real VoIP application. This test showed that, by using *DISpatCH*, the communication was considerably less affected, thus confirming the efficiency of the proposed solution.

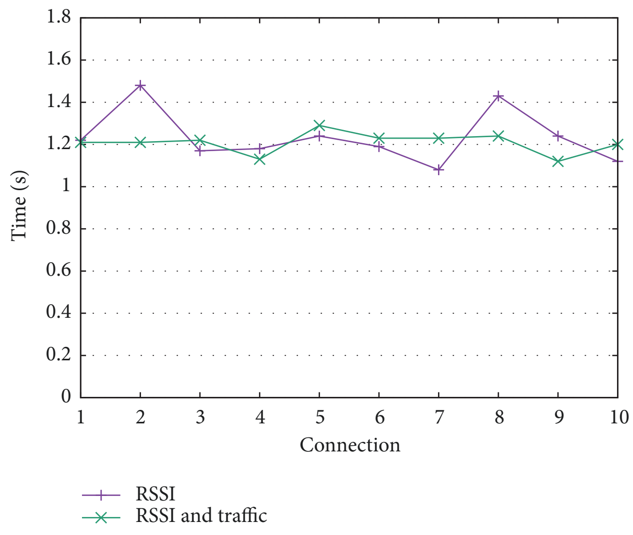
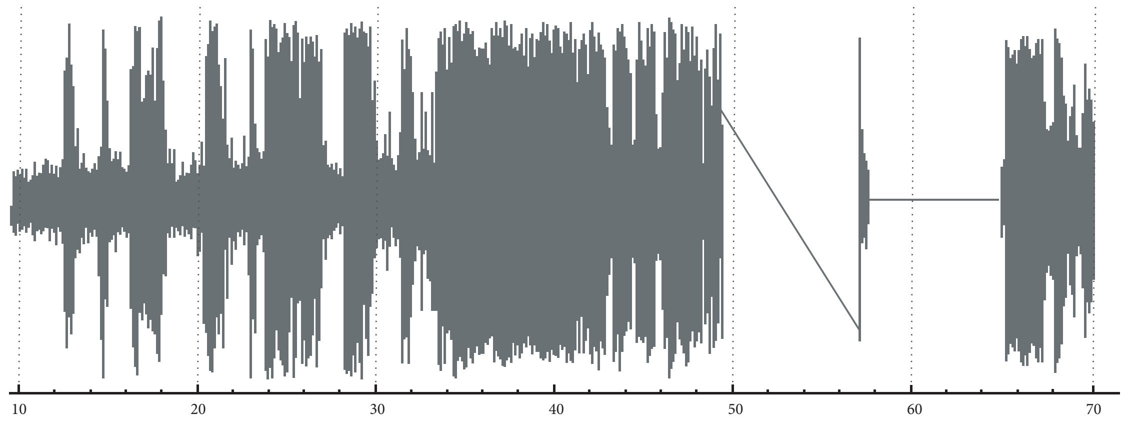
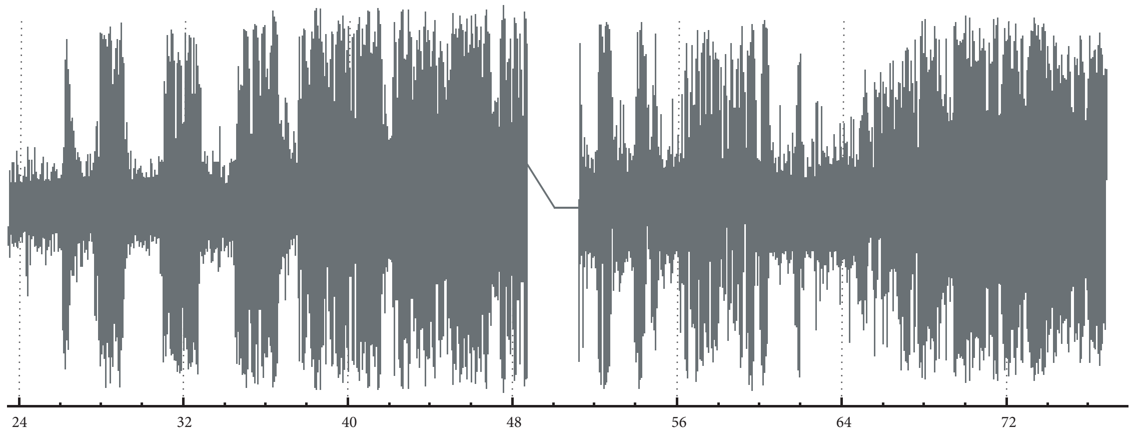


FIGURE 12: Handoff time (RSSI x Traffic).



Source Address	Source Port	Destination Address	Destination Port	SSRC	Setup Frame	Packets	Time Span (s)	Sample Rate (Hz)	Payloads
192.168.0.100	5062	192.168.0.200	5086	0xe36a4d74	77	2669	9.28 - 70 (60.8)	8000	g711U

FIGURE 13: Traditional handoff process during a voice call.



Source Address	Source Port	Destination Address	Destination Port	SSRC	Setup Frame	Packets	Time Span (s)	Sample Rate (Hz)	Payloads
192.168.0.100	5094	192.168.0.200	5094	0xdb921a51	898	2632	23 - 76.8 (53.9)	8000	g711U

FIGURE 14: *DISpatCH* handoff process during a voice call.

As future work, we plan to use other decision parameters in the detection and discovery phases, which may result in better quality of service for real-time applications.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

A preliminary version of this work was presented at the VI Brazilian Symposium on Computing Systems Engineering.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the Universidade Federal da Paraíba. We thank the Instituto Federal de Educação Ciência e Tecnologia da Paraíba-IFPB, for the access to its facility and the use of its equipment to perform the experimental tests. We would also like to thank Dr. Anand Subramanian for assistance with the math formalism.

References

[1] G. Yao, J.-N. Cao, J. Siebert, and X. Liu, "OppoScan: Enabling Fast Handoff in Dense 802.11 WMNs via Opportunistic Probing with Virtual Radio," in *Proceedings of the 14th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2017*, pp. 198–205, USA, October 2017.

[2] B. Zhang, X. Wen, Z. Lu, T. Lei, and X. Zhao, "A fast handoff scheme for ieee 802.11 networks using software defined networking," in *Proceedings of the In 2016 19th International Symposium on Wireless Personal Multimedia Communications (WPMC*, pp. 476–481, November 2016.

[3] Institute of Electrical and Electronics Engineers (IEEE). IEEE 802.11 Parte 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY), March 2012.

[4] V. Tiwari, *SDN and Openflow for beginners with hands on labs*, Amazon Digital Services, 1st edition, 2013.

[5] Q. M. Salih, U. Mueen, and M. Nikos, "A review of handoff latency reducing techniques in ieee 802.11 wlan networks," *Mathematics and Computers in Science and Engineering Series*, pp. 175–184, 2015.

[6] T. Bakhshi, "State of the art and recent research advances in software defined networking," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[7] H. Hu, H.-H. Chen, P. Mueller, R. Q. Hu, and Y. Rui, "Software defined wireless networks (SDWN): Part 1 [Guest Editorial]," *IEEE Communications Magazine*, vol. 53, no. 11, pp. 108-109, 2015.

[8] B. Agborubere and E. Sanchez-Velazquez, "OpenFlow Communications and TLS Security in Software-Defined Networks," in *Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 560–566, Exeter, June 2017.

[9] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using open flow: a survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493–512, 2014.

[10] N. Sanghavi and R. S. Bansode, "Improving IEEE 802.11 Wlan Handoff Latency by Access Pointbased Modification," *Global Journal of Computer Science and Technology: E Network, Web & Security*, vol. 15, no. 5, pp. 37–40, 2015.

[11] Y. Chan, "The Design of an AP-Based Handoff Scheme for IEEE 802.11 WLANs," *International Journal of e-Education, e-Business, e-Management and e-Learning*, 2014.

[12] S. Jin and S. Choi, "A seamless handoff with multiple radios in IEEE 802.11 WLANs," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 3, pp. 1408–1418, 2014.

[13] V. Brik, A. Mishra, and S. Banerjee, "Eliminating handoff latencies in 802.11 WLANs using multiple radios: Applications, experience, and evaluation," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, IMC 2005*, pp. 299–304, USA, October 2005.

[14] A Misal and S. Santosh, "Reduction of Handover Latency by Horizontal Distance Measurement using GPS," *International Journal of Computer Applications*, vol. 105, no. 11, 2014.

[15] A. Mishra, M. Shin, and W. Arbaush, "Context caching using neighbor graphs for fast handoffs in a wireless network," in *Proceedings of the IEEE INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. 351–361, Hong Kong, PR China, 2004.

[16] S. Zehl, A. Zubow, and A. Wolisz, "BIGAP - A seamless handover scheme for high performance enterprise IEEE 802.11 networks," in *Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium, NOMS 2016*, pp. 1015-1016, Turkey, April 2016.

[17] T. Lei, X. Wen, Z. Lu, X. Zhao, Y. Li, and B. Zhang, "SWN: An SDN based framework for carrier grade Wi-Fi networks," *China Communications*, vol. 13, no. 3, pp. 12–26, 2016.

[18] C. Chen, Y. Lin, L. Yen, M. Chan, and C. Tseng, "Mobility management for low-latency handover in SDN-based enterprise networks," in *Proceedings of the 2016 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, Doha, Qatar, April 2016.

[19] M. Sun and H. Qian, "Handover management scheme in SDN-based wireless LAN," *Journal of Communications*, vol. 11, no. 3, pp. 282–289, 2016.

[20] L. Sequeira, J. L. De La Cruz, J. Ruiz-Mas, J. Saldana, J. Fernandez-Navajas, and J. Almodovar, "Building an SDN enterprise WLAN based on virtual APs," *IEEE Communications Letters*, vol. 21, no. 2, pp. 374–377, 2017.

[21] H. Moura, G. V. C. Bessa, M. A. M. Vieira, and D. F. Macedo, "Ethanol: Software defined networking for 802.11 Wireless Networks," in *Proceedings of the 14th IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, pp. 388–396, Canada, May 2015.

[22] M. Yan, J. Casey, P. Shome, A. Sprintson, and A. Sutton, "ÆtherFlow: Principled wireless support in SDN," in *Proceedings of the 23rd IEEE International Conference on Network Protocols, ICNP 2015*, pp. 432–437, USA, November 2015.

[23] K. Nahida, C. Yin, Y. Hu et al., "Handover based on AP load in software defined Wi-Fi systems," *Journal of Communications and Networks*, vol. 19, no. 6, pp. 596–604, 2017.