Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 9359878, 10 pages
https://doi.org/10.1155/2018/9359878

Research Article

An SDN-Based Connectivity Control System for Wi-Fi Devices

Duc-Thang Nguyen and Taehong Kim

School of Information and Communication Engineering, Chungbuk National University, Cheongju, Republic of Korea

Correspondence should be addressed to Taehong Kim; taehongkim@cbnu.ac.kr

Received 27 March 2018; Revised 20 June 2018; Accepted 11 July 2018; Published 24 July 2018

Academic Editor: Kim-Kwang Raymond Choo

In recent years, the prevalence of Wi-Fi-enabled devices such as smartphones, smart appliances, and various sensors has increased. As most IoT devices lack a display or a keypad owing to their tiny size, it is difficult to set connectivity information such as service set identifier (SSID) and password without any help from external devices such as smartphones. Moreover, it is much more complex to apply advanced connectivity options such as SSID hiding, MAC ID filtering, and Wi-Fi Protected Access (WPA) to these devices. Thus, we need a new Wi-Fi network management system which not only facilitates client access operations but also provides a high-level authentication procedure. In this paper, we introduce a remote connectivity control system for Wi-Fi devices based on software-defined networking (SDN) in a wireless environment. The main contributions of the proposed system are twofold: (i) it enables network owner/administrator to manage and approve connection request from Wi-Fi devices through remote services, which is essential for easy connection management across diverse IoT devices; (ii) it also allows fine-grained access control at the device level through remote control. We describe the architecture of SDN-based remote connectivity control of Wi-Fi devices. While verifying the feasibility and performance of the proposed system, we discuss how the proposed system can benefit both service providers and users.

1. Introduction

The popularity of wireless devices is increasing rapidly, and an increasing number of end-devices connect to the Internet through Wi-Fi—anything from home appliances to factory facilities. It is not hard to find a Wi-Fi signal in places such as offices, cafes, or even across entire cities. In addition, most essential appliances such as smartphones, laptops, speakers, and wearable gadgets support the Wi-Fi protocol.

Basically, most home or office devices need a service set identifier (SSID), which is broadcasted by typical Wi-Fi routers. In order to enhance wireless network security, a number of authentication procedures such as SSID hiding, media access control identifier (MAC ID) filtering, or Wi-Fi Protected Access 2 (WPA2) are applied. For instance, in the case of the SSID hiding method, only users who know the SSID can access the Wi-Fi network. In the most popular authentication Wi-Fi mechanism—WPA2—the router works in a secure mode with an SSID and password pair. Anyone who provides the correct pair is granted network access. Besides, with MAC ID filtering, only devices with allowed MAC addresses can join the network or use the Internet. Moreover, with the appearance of advanced security mechanisms such as Extensible Authentication Protocol (EAP) and Lightweight Extensible Authentication Protocol (LEAP), users' data and information can be protected more efficiently while surfing the Internet.

Nevertheless, as a side effect of the many security advantages, the complex authentication procedure could affect user experience when connecting to the network. For instance, when a stranger visits the user's home and wants to use their Wi-Fi network, they must be told the SSID and the password. In the case of MAC-based authentication, an entry for the guest device's MAC address should be added to the authentication service in advance by the network administrator for setting up a connection. This means that the more complex the authentication setting, the more difficult it becomes to set up the authentication procedure for adding or replacing new devices. In particular, nontechnical elderly people who are not familiar with IT devices may find it tedious to use complicated authentication procedures.

Besides, with IoT trending, the number of smart appliances using home network protocols such as ZigBee [1], Z-WAVE [2], and Bluetooth [3] has been increasing explosively. Even though these kinds of protocols are suitable for lightweight or low-energy devices, IoT devices are still equipped with Wi-Fi functionality for easy management. Moreover, the "things" now being added to homes, offices, and industrial facilities are quite small and do not have a display or convenient data entry capability to allow users to configure them. As a result, the devices need to be configured by a smarter device that can transmit the necessary configuration information to join the network. For example, in order to make a smart switch join a home network, customers have to install a separate application on their smartphone or tablet and then the app can transmit Wi-Fi access information to the IoT devices via near-field communication (NFC) or Bluetooth. Then, the gadget can connect to the current network and perform related setup to use its features. As with the previous examples, instructions for such a configuration can sometimes be difficult for a novice user unfamiliar with IT.

Although a remote connectivity control scheme for Wi-Fi-enabled equipment is essential for both customers and network owner/administrator, it is hard to implement one through just minor enhancements to traditional network systems. Note that software-defined networking (SDN) provides an architecture enabling programmatic enhancements to network management and configuration [4–9], which form the base architecture for us to achieve our goals.

Thus, in this paper, we propose a new SDN-based mechanism that not only enhances the connection-establishment process for new wireless devices but also improves network management functionalities. More specifically, our system comes with interesting use cases. (1) It enables automatic network configuration and allows administrators or users to set up new Wi-Fi device joining their network process remotely. These events are announced to owner/administrator through remote notification services and displayed on a web-based user interface. (2) Using its associated station's information and per-client virtual access points, the connectivity control system supports flexible network functions (e.g., network access rules for individual devices). To demonstrate the feasibility and benefits of our approach, we evaluate it on a real prototype. We believe that our system can bring innovation to network features and additionally facilitate user-network interactions.

This paper is organized as follows. Section 2 reviews some related studies and Section 3 provides preliminaries for the proposed system. With the use cases of our system in Section 4, Section 5 describes the system architecture in detail. Section 6 provides performance evaluations from experimental testbed. And Section 7 discusses the security enhancements for the connectivity control system. Finally, we conclude this paper in Section 8.

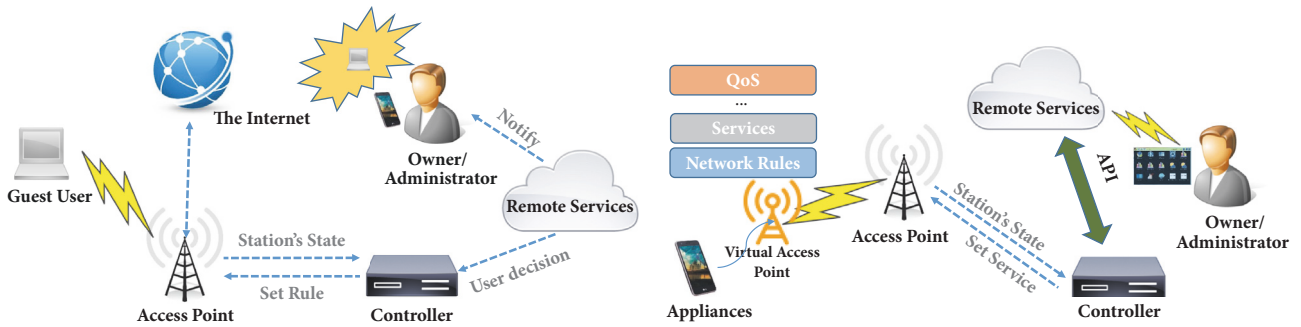
2. Related Works

Connectivity control of smart devices, especially wireless-enabled home or office devices, is one of the important issues in the IoT paradigm. Numerous manufacturers such as Samsung [10], Apple [11], and Philips [12] are developing their own smart device ecosystems, which users who buy their equipment can benefit from. For example, customers can configure and manage their devices through cloud services supported by the manufacturer. However, these kinds of services are specific to each manufacturer. This indicates that if customers own gadgets of different brands, they have to use different services for each device. Thus, this operation may require complex steps, inconveniencing customers.

In order to solve these connectivity control issues, M. Lee et al. in [13] proposed a new autoconfiguration method of home network with SDN controller support. Using Floodlight—SDN controller acting as a cloud-based home network controller—the system enables automatic recognition and management of home devices without requiring specific equipment. There are diverse home devices such as notebooks, smartphones, and light sensors and their MAC addresses as an identifier are stored in a database. Besides, SDN enables allocating bandwidth for quality of service (QoS) for each device. However, this system has not considered wireless environment. Thus, it lacks a number of functionalities for WLANs such as access point, association, and roaming. On the contrary, our paper targets WLANs and improves traditional access points with LVAP abstraction, which we will explain comprehensively in the next section, for connectivity control of home devices.

Yet another approach to IoT device management in smart homes was proposed by Vijay in [14]. In this study, the authors built and demonstrated the advantages of using SDN in IoT security and network services from an Internet service provider. Specifically, the implemented design showed the user interface, which contains the subscribers' household devices and allows update or deletion of the flow rules of access control. This also means that customers have the ability to remotely block/quarantine devices based on network activity via a web-based portal. Nevertheless, the author did not clarify the process of establishing a connection for each device and focused on security after the successful setup of all equipment. On the other hand, this paper specifies the detail procedure for connection setup as well as proving feasible through a testbed. In addition, the network owner/administrator has the right to allow/reject network access to clients without using MAC filtering or blocking device IPs.

The study in [15] suggests a simple mechanism that allows small network owner/administrator to provide Internet access to guests using OpenID or certificates issued by third-party services. It has been suggested that this system seems to be a useful approach since it reduces authentication operations. Similar to this work, authors in [16, 17] introduced a solution which enhances the wireless access services for enterprise WLAN by combining SDN and an authentication server. However, these mechanisms require display-enabled devices, which small IoT devices such as switches, plugs, and sensors are not. Hence, we extend this proposal in order to apply to any type of IoT devices regardless of their size, providing easy device configuration and management through the remote system. More technically, as mentioned before, our connectivity control system handles the client's primitive state and moves the network decision module to the



(a) Remote connectivity control of a guest's Wi-Fi device: owner allows/rejects a guest's device through rule set up to the Wi-Fi network

(b) Remote connectivity control and management of IoT devices: owner controls IoT devices through remote services provided by the controller

Figure 1: SDN-based connectivity system use cases.

remote controller. Thus, customers can install and manage their network automatically without complex steps.

3. Preliminaries

So far, a concept of SDN for wireless network was introduced in [18–20] and built upon Odin's architecture [21]. These systems also inherit Light Virtual Access Point (LVAP) abstraction, master-agent paired modules, and the traditional OpenFlow controller [22]. While the OpenFlow controller determines the best path for network traffic, the master module makes decisions about allocation of network resources. In addition, the client state is recorded and forwarded by the agent module, which is integrated into the wireless access points.

LVAPs are key components of the Odin framework. Basically, each client will be attached to the image of the access point they connect to. Specifically, in physical wireless access points, whenever the Wi-Fi card receives a valid 802.11 frame, it immediately spawns an LVAP associated with the client. Hence, the LVAP becomes a potential per-client virtual access point for the client to perform association. Besides, since this data is transmitted to the controller, the LVAP manager module built into the controller will hold the unique virtual access point information such as BSSID, MAC address, IP address, and SSID as a representation of the client. For instance, there are two particular access points which are both managed by the same controller. A client is previously associated with the system and attached an LVAP. When the client moves out of the coverage area of the first access point while entering the second areas, the handoff occurs without requiring a reassociation and exchanging additional layer 2 or 3 messages. This mechanism effectively enhances the handoff performance and distributed-client problems [23–25].

However, the above-mentioned architecture is implemented as an "open" system; any devices can join and access the Internet without permission. Thus, this system lacks a controllable connectivity module that deals with device authentication and management. In order to enable the connectivity control system, we propose an approach that involves using the SDN controller as a "decision center" that collaborates with other business services. Through this system, the controller holds all the information about associated clients and shares these resources with other network services. Thus, it can help the owners/administrators to manage and take control of their own network system.

Furthermore, LVAP can take advantage of the ability to support multiple services through each virtual access point. For example, the network administrator can control each client individually with respect to connection time and bandwidth limitation without affecting others. In our system, LVAP abstractions are used to mark the appearance and handle the connectivity process of clients. Particularly, since our device manager module holds the data of the attached LVAP in a physical access point, based on user actions, it can make decisions to allow/reject spawning LVAPs or provide network rules for each client. This also means that an administrator who is physically away from the network has full access to take over device connection establishment. We will describe some of these use cases in the next section.

4. Use Cases

Our solution is based on programmable network devices in the spirit of SDN. In this section, we discuss some use cases for the envisioned system.

4.1. Remote Connectivity Control of a Guest's Wi-Fi Devices. One of the use cases our system supports is sharing Internet connectivity with visitors in home, office, or enterprise. Basically, a visitor should be given the WPA passphrase or the administrator should manually add an entry of his/her MAC address for authentication. The disadvantages of a classic private shared key include the fact that it is impossible to revoke the secret key when he/she leaves as well as the fact that it is relatively easy to crack. The alternative solution, 802.1X, requires the installation of a software client and remote authentication server, thus making it difficult or impossible to use on devices that do not support such protocols. Using our system, we can innovate and simplify authentication. For example, in Figure 1(a), when a visitor wants to use the Internet, he/she will connect to the public SSID broadcasted by the access point. Then, the network owner will be notified of the identity of the newcomer's device

via the remote services working on top of the controller. Finally, he/she will accept or deny the connection and the corresponding rules on the access point will be set by the controller. Thus, the guest does not have to input any password and the network's owner could accept the request remotely. This can, of course, be done at a large-scale offices and enterprises with a supervisor or a network administrator.

4.2. Remote Connectivity Control and Management of IoT Devices. Our connectivity control system also facilitates the connection and management of IoT devices. Note that most IoT devices lack a display or a keypad, requiring the use of other smart devices to establish a connection. As Figure 1(b) illustrates, with our system, the client state will be forwarded immediately to the controller when it interacts with the system. Thus, this information can be used through multiple remote or cloud services to provide network services to customers. For example, a customer buys a smart plug that contains default SSID information from the manufacturer. In addition, an access point in the owner's home also broadcasts the same SSID. This is supported by our concept. As a result, when the plug is powered on, it automatically associates with our system. In addition, owners/administrators will be informed about new devices and can control it through several web or mobile services. Moreover, since each associated client is attached to a virtual access point, it allows fine-grained control of network resources according to user demand and expected QoS level. Note that SSID information can be updated after the IoT device successfully joins the network. Detail procedure will be discussed in the next section.

5. SDN-Based Connectivity Control System

In this section, we describe an SDN-based connectivity control system to achieve the above-mentioned use cases.

5.1. Overview of Connectivity Control System. In this subsection, we describe the procedure of the connectivity control system as shown in Figure 2, together with its software architecture in Figure 3. Figure 2 describes the procedure of granting permission and establishing connection for newcomer. Figure 3 shows that the proposed architecture is composed of diverse software components such as *ClientStateResource* for each access point and *ClientStateHandler*, *DeviceManager*, and *ConnectivityControlApplication* for the controller and remote services. Specifically, when an 802.11 management frame is received by a *ClientStateResource* module which is implemented in the access point, it checks whether the client is assigned an LVAP. Immediately, this process also takes place in the *ClientStateHandler* module in the controller, which assigns an LVAP to each client if they have not been assigned LVAPs or have been previously removed. After that, the client information is checked against the allowed MAC addresses—whether one exists for the client that generated the frame. Registered devices will appear in the *AllowedList* and managed by the *DeviceManager* module, used for device identification and authentication. If so, via client-state information saved in the LVAP, the controller sets up the connection with the station. If there is a newcomer that has not interacted with the system before, the agent informs the controller, and the *ConnectivityControlApplication* on top of the controller framework will start the authentication. Moreover, with the notification system, the network owner/administrator decides to allow/deny the joining requests from devices. Subsequently, the controller attempts to respond to the clients accordingly. The process ends when permission is granted to the new station or the frame is dropped and the joining process fails if the administrator denies the connection. Since all new device-joining events come to the remote owner/administrator, it is possible to achieve the first use case. In other words, it eliminates the complex procedure of modifying configuration such as SSID hiding and MAC filtering at the access point level. It is also possible to make IoT devices without a display joining the access point and then the remote controller just accepts them. Moreover, for the second use case, since each device is attached to an LVAP, the controller has the ability to monitor all Wi-Fi devices status and apply individual device-level control in the form of access control and bandwidth control.

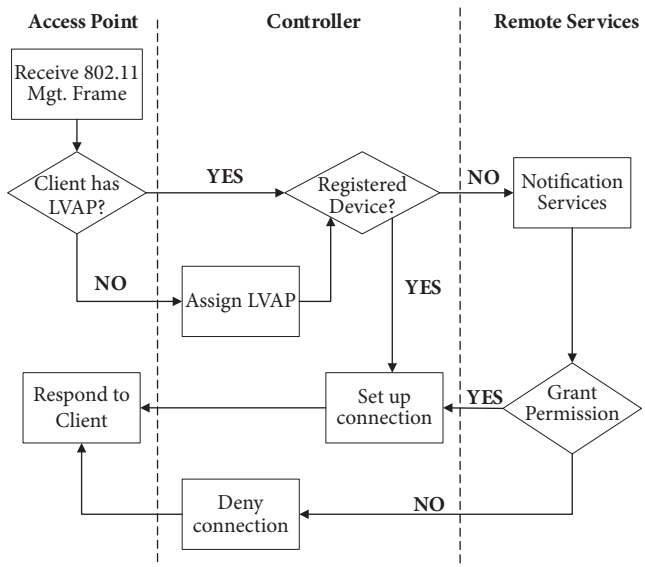


Figure 2: Procedure of SDN-based connectivity control system.

5.2. Software Architecture

5.2.1. Access Point. In our system, an access point contains both OpenFlow-enabled switch instances managing the communication over wired data paths and a *ClientStateResources* holding station states as virtual access points and 802.11 radio signals. Specifically, access points will be equipped with the *ClientStateResources* module, which communicates with the *ClientStateHandler* module in the controller via a persistent TCP connection. Therefore, not only can we perform the normal features of an access point, they are also involved in gathering client states before forwarding it to the controller. In technical detail, when a client is associated with the access point, the access point will hold the SSID, client MAC address, and client IP address information (if granted). Based on this information, a unique

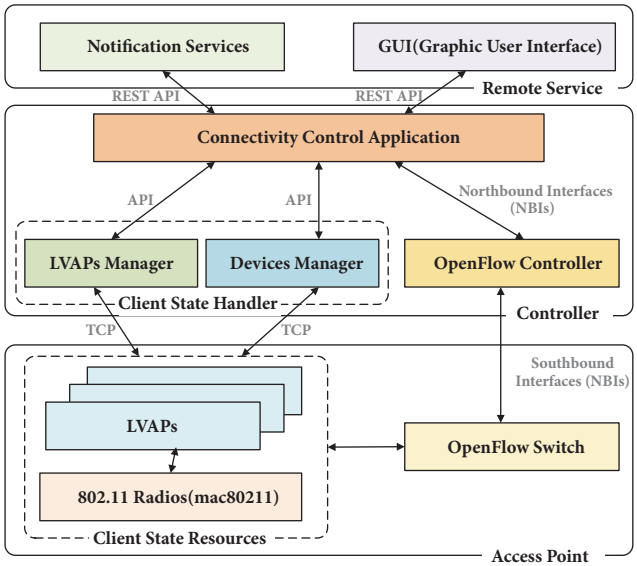


Figure 3: The software architecture of an SDN-based connectivity system.

virtual access point will be attached to the client, which operates a number of services provided by the controller.

In a conventional wireless network, there are numerous access points acting as a central gateway for clients. However, due to restricted programmability levels, current access points are limited in both enhancing network services and simplifying setup operations for users. For example, creating a new SSID or configuring a wireless access point is still complex to nontechnical users. SDN brings an innovation to traditional networks by moving the decision center to the remote controller. Despite the potential of the OpenFlow protocol in SDN controllers, it is still not possible to apply it to a wireless environment effectively. Thus, besides the OpenFlow protocol securing the connection between the switch and controller, a TCP connection is used to transfer the information of client states to the higher layer. Moreover, this connection secures the wireless data path and control path between the controller and the access point.

5.2.2. Controller. The subcomponent of our controller in Figure 3 builds upon the Empower [26] and consists of the following components: (1) *ClientStateHandler* stores and manages LVAP through the LVAP manager and the device manager, (2) *OpenFlowController* is responsible for deciding how a specific packet is handled via a wired control path, and (3) the *ConnectivityControlApplication* on top of the framework. Technically, when a device joins through an access point, its primitive states are immediately transmitted to the *Controller* through the communication protocol between access point and controller. Based on this information, the *LVAPsManager* module can build a logical representation of the client in the controller—an LVAP for handling another process. Furthermore, the *DeviceManager* also recognizes whether devices are registered with the system in advance. If it is registered, it decides which services or rules can be applied to these devices. Finally, *ConnectivityControlApplication*, which is associated with the remote services via an API, will decide whether to keep the connection alive or terminate it based on user demand. After the network application completes its operations, the response state will be sent to the access point and client, and flow rules will be applied to the *OpenFlowController*.

5.2.3. Remote Services. Remote services lay on top of the application layer, which provides a wide range of services to users such as web and cloud services. By passing data through network applications in the controller, a number of associated services can be delivered through the Internet to users. For example, in an SDN-based connectivity control system, a remote server collaborating with the *ConnectivityControlApplication* is responsible for authentication services. Thus, we set up automatic notification services that announce to users which devices are trying to join the network. Additionally, through a RESTful web service, owner/administrator can set up devices connection remotely as well as monitoring the status of active devices.

6. Performance Evaluations

To demonstrate the feasibility and benefits of our approach, we conducted experiments in a test bed. Our controller was evaluated on a nonvirtualized machine with 8 CPU cores supporting hyperthreading and 16 GB of RAM. The host's operating system was Ubuntu 16.04 and it ran modified software named Empower Wi-Fi, introduced by R. Riggio et al. in [19]. We chose this particular apparatus due to the efficiency of developing and maintaining network applications on top of the framework with the support of the Empower software development kit. In addition, a web service contains a graphic user interface (GUI) providing the user with an interactive interface via their personal computer or mobile phone.

All access points run OpenWRT [27] release Chaos Calmer with the ath9k Linux driver, user-level Click modular router [28], and OpenvSwitch (OvS) [29] version 2.39 supporting OpenFlow (OF) [30] version 1.3 and conntrack table management. Besides, each access point also contains the *ClientStateResources* module, collecting client state and communicating with the *ClientStateHandler* module in the higher layer. While OpenvSwitch instances manage the communication via a wired network topology, Click modular router instances implement IEEE 802.11 data paths. Technically, Click is a framework for handling multipurpose packet processing and is used to implement LVAP frame exchange, while all the decision logic is built into the controller. Other gadgets supporting our experiments are Raspberry Pi Model 3B [31] running a FreeRADIUS [32] server and one TP-LINK1043ND router for providing DHCP and Internet access.

In order to support our evaluation, we defined some APIs based on the features of notification systems. Table 1 shows samples of the RESTful API that was used in the network controller. Specifically, when the authentication process takes place in devices control application at the network controller, through this API, the system can capture the decision of

Table 1: REST APIs in the connectivity control module.

Action	API	Description
Allow	/api/device/allow/id	Allow device {id} to join the network
Reject	/api/device/reject/id	Deny device {id} from joining the network
Notify	/api/services/device/id/message	Notify user's device {id} trying to join the network with {message}

Figure 4: Web interface showing information of devices such as stations, SSID, LVAP, access point, station names, and status.

owner/administrator to allow/reject a connection as returned by the notification service. In addition, to control a smart device using remote services, developers can create any kind of web, mobile, or PC application and all of them can easily work with our system.

6.1. Experiments on the SDN-Based Connectivity Control System. Experiments were conducted to verify connectivity control including device recognition and granting of Internet access remotely as in Figure 1(a). Several Wi-Fi devices such as smartphones and notebooks used in our environmental test bed will be identified by their MAC addresses stored in our remote service. The results of the experiment for providing network access to guests are presented in Figure 4. Figure 4 demonstrates the results of the real-time recognition and configuration in our test bed using a Web UI. The process of granting network access is done by attaching LVAP to new devices, checking its information through remote services, and finally notifying the network's owner. When the owner/administrator accepts the guest device's joining request, an IP address is provided by the DHCP service and it successfully connects to the Internet.

In contrast, if the owner/administrator rejects the request, the device will be kept in a "blocked list" and ignored by the system. In addition, the device list is updated in real time with the associated state of each device. The above results demonstrate that our system is possible to use for remote network control based on SDN abstraction, which handles clients' primitive states and programmable access points. Besides, since our access points support multiple SSIDs, we have the ability to create a virtual SSID and multiple networks from the same box. Therefore, each network can work independently for our purpose but still notifies the controller about interactive devices status.

6.2. Evaluation of Round Trip Time for Authentication. This subsection focuses on establishing a connection process for new devices in our system. The round trip time for user operations is measured from when he/she decided to choose an SSID from the device's interface until the connection between the access point and device is established. This procedure consists of two main phases: authentication and DHCP lease. In traditional IEEE 802.11, when an SSID is broadcasted by an access point, there may be a security method such as WPA2-PSK applied to protect the network. This process makes the network more secure but also requires more user operations such as typing passwords. More specifically, in the case of public Wi-Fi, users just choose an SSID and join without typing a secret phrase and the association process starting. But in the case of WPA2-PSK, we measure the time taken for users to provide the password. In our system's case, a virtual SSID without encryption was broadcasted by our access point and a similar process took place as in public Wi-Fi. However, in this case, we built a remote module in the Raspberry Pi acting as a third-party web service containing our test device information such as MAC address and device name. When the authentication happens via the above-mentioned REST API in Table 1, the controller can compare our device's identifier and grant network permission specifically. We performed this evaluation with a Wi-Fi-enabled notebook computer running Ubuntu 16.4 and repeated 10 times for each case.

The result is shown in Figure 5. In the case of the public Wi-Fi, the round trip time from the moment the terminal command for connecting to the network was executed to the time it returned successfully is between 0.79 s to 1.9 s. Besides, the round trip time for WPA2-PSK is from 0.83 s to 2.85 s. In our system, it takes a longer time from 2.15 s to 3.67 s for devices access. Thus, we clearly see that using the LVAP concept in our connectivity control system gives a slightly slower than traditional methods. We can analyze the reason based on Algorithm 1. Instead of immediately responding to the station with an assocResponse frame as usual, our system uses the station state to identify which BSSID belongs to the physical access point and LVAP. After that, the *sendNotificationToRemoteService* containing stations' MAC address calls to ask the system administrator or owner whether to permit the new associated stations or not. The wait time for owner/administrator to reply and execute this function contributes most to latency in authentication time. Besides, the time for OpenFlow rule-matching process and setting up of the Internet connection by *addOpenFlowRule* added to the round trip time. Even though it has a relatively long latency compared to traditional methods, our scheme facilitates users operations. It also means that the guest does not have to ask about the wireless network password or retype it since it can be changed easily. In addition, owners/administrators have the ability to manage

```
1 Function handleAssociationRequest (STA);
  Input: Station Information, STA
  Output: Handle association request frame
2 if APConnectionExists then
3   sta⟵getSTA(currentRequest);
4   bssid⟵getBSSID(STA);
5   lvap⟵getLVAP(STA, BSSID);
6   if lvap exists then
7     physicalAP⟵getPhysicalAP(lvap);
8     switchAP⟵getOFSwitch(physicalAP);
9     if STA exists in allowedList then
10      sendAssoResponse(STA);
11      addOpenFlowRule(switchAP, match, action);
12    else
13      if STA exists in rejectedList then
14        addOpenFlowRule(switchAP,match,action: drop);
15      else
16        //Send Authentication Request to Remote Server;
17        sendNotificationToRemoteService(STA);
18      end
19    end
20    //Updating allowed/rejected list with results returned from remote services.
21  end
22 end
```

Algorithm 1: Pseudo-code of handling association frame from associated station.

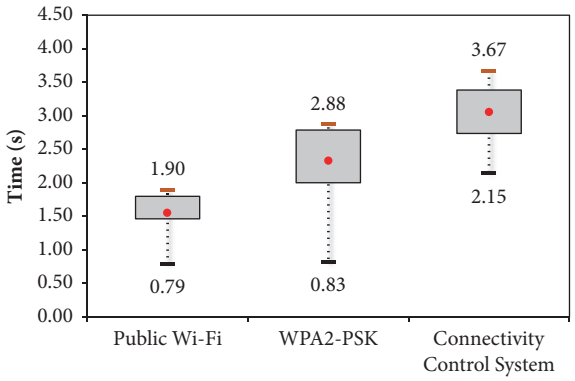


Figure 5: Actual round trip time in authentication phase with notebook.

and configure his/her network in real time remotely with different services supported by the network provider.

Note that, for the measurements in Figure 5, LVAP needs to traverse to the SDN controller, whereas for others it does not need to visit beyond the access points. For a more fair comparison, we compare our evaluation system with IEEE 802.1x Enterprise WPA, which requires traversal to the remote service for authentication and authorization. In order to conduct this experiment, we built a small authentication server running the FreeRADIUS operating system and point to this server through WPA Enterprise security mode on our access points. FreeRADIUS is an open source RADIUS server used by many organizations. It performs authentication, authorization, and accounting (AAA) functions, based on many modularized authentication protocols. Basically, in enterprise WLAN, RADIUS authentication servers are usually used to provide a username/password for each client to join the network. After providing this kind of information, clients will be issued a certificate and use it for authentication. Because only our mobile phone supports 802.1x so we compared two methods on the same device. In this situation, we wrote a simple code that contains device's authentication information and stored it on the mobile. Thus, we do not have to input this information manually on our mobile's screen, which would have taken more time. The result is shown in Figure 6. In this evaluation, while connection establishment in our system takes between 3.82 s and 6.01 s, the IEEE 802.1X authentication time is from 5.25 s to 6.72 s. The results indicate that the 802.1X authentication method takes a slightly longer time than our mechanism. While our method focuses on facilitating devices' joining the network process, 802.1X strengthens the security of the traditional WPA2-PSK authentication mechanism. Therefore, we can conclude that SDN-based connectivity control does not require large latency until it allows new Wi-Fi devices, even though it needs to traverse to the SDN controller through the access point.

Since our system follows the Odin abstract architecture and also inherits its advantages such as client mobility support, multiple logical networks on top of the physical infrastructure, and per-client virtual access points, our connectivity control platform can support these functions including the use cases in Section 4. For example, owner/administrator can apply each device policy or time access rule independently via our provided web GUI. Since devices connect to physical *AccessPoints* which are controlled by the *Controller*, these network services affect the clients. In addition, extensive researches have been conducted on efficient QoS using

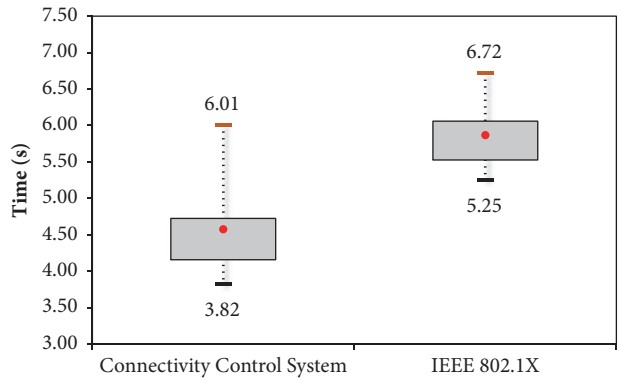


Figure 6: Actual round trip time in authentication phase with smartphone: IEEE 802.1X versus our system.

OpenFlow management in wireless networking environment such as video streaming and VoIP [33, 34]. Other advantages of SDN-based system are building an intelligent WLANs based solution which help handling massive data for future IoT network as [35, 36]. Therefore, our connectivity control system has the potential to provide QoS for applications in easier and more flexible ways compared to traditional wireless network architectures.

7. Security Enhancements

The main goal of this paper is to enable quick and easy connectivity control of Wi-Fi devices including guest devices and IoT devices as discussed in previous sections. In other words, security is not main concern of this paper, but it is an essential requirement since the target system, Wi-Fi, of the proposed solution is commercial. Therefore, this section discusses how the proposed connectivity control system in Section 5 provides security functions such as authentication of devices as well as encryption of packets.

First of all, it is possible to doubt that the connectivity control system is vulnerable to the MAC cloning and spoofing attack, since the MAC address of a device is a unique identifier. Indeed, the device is registered with only MAC address in the proposed system. However, it is interesting to note that the connectivity control system requires intervention from the network owner or administrator, who can check the identity of device through offline directly or indirectly. In detail, the main use cases are guest users and IoT devices in home and office, as discussed in Section 4. In most cases when accepting the new connection from guests or installing new IoT devices in home, the network owner is located in the same place with them. Even though the network owner/administrator delegates its management role to service provider due to any reasons such as lack of knowledge, the owner needs to be connected to the service provider to confirm the identity of a newly connecting device. Therefore, we can prevent spoofing attack from malicious device at the initial connection phase through the identity check in offline. In addition, the proposed system allows us to keep monitoring the status of client through LVAP. Note that our system assigns a unique LVAP to individual client. Moreover, as introduced in Figure 2, the *DeviceManager* module not only verifies whether the connecting device's MAC address is contained in the *AllowedList*, but also checks whether a new spawn LVAP is bound to the current access point or not. This indicates that when a client moves to other locations and disconnects/reconnects to other access points, it immediately triggers the notification system and announces to the system owner. Therefore, we can conclude that the proposed connectivity control system has many options to prevent diverse attacks based on device identification.

The next security issue is about whether the proposed system can establish a secure connection from Wi-Fi devices or not. Note that Section 5 describes the connectivity control in an open network environment, which may result in diverse vulnerabilities from nonencryption of packets. Therefore, we need to discuss the feasibility to apply the existing security protocols such as WPA and WPA2 to our system. Before describing the security enhancement, it is important to remind that the proposed system is designed and targeted to IoT devices which are not equipped with any input and output interfaces. These security protocols such as WPA and WPA2 require a connecting Wi-Fi device to deliver its id and password information, but it is impossible to input this information manually. There exist contradictions between the assumption of the proposed system and the requirement of security enhancement. The only way to fill the gap is to assume that the IoT devices including guest Wi-Fi devices are equipped with preinstalled application, which is used for the connection phase, with help from manufacturers. Since the quick and easy connection management of increasing number of IoT devices is one of the important issues, this is not infeasible assumption.

The proposed security enhancement utilizes that access point can create multiple SSIDs. As shown in Figure 7, the client connects to the open SSID, and it reconnects to the secure SSID after successfully being identified and granted permission by the system. In detail, the client device connects to the access point the same as the procedure in Figure 2. Whereas it finishes the connection establishment once the device is accepted by system administrator in Section 5, the security enhancement requires additional procedures. First, the client device, especially the preinstalled application in the client devices, has to transfer a public key to the access point. Second, the access point replies with information about secure SSID which is encrypted with public key received. Finally, the preinstalled application in client device decrypts the information using its private key, and it changes the connection to the secure SSID. Note that it is highly required to apply lightweight public key cryptography mechanism for IoT devices due to their limited resource, but we can conclude that the proposed security enhancement mechanism is feasible to apply to the proposed connectivity control system and provide the similar level of security with existing solutions.

8. Conclusion

With the explosion of tiny IoT devices and diversification of complicated security options, it becomes hard to configure the connection information of Wi-Fi devices. In this paper,

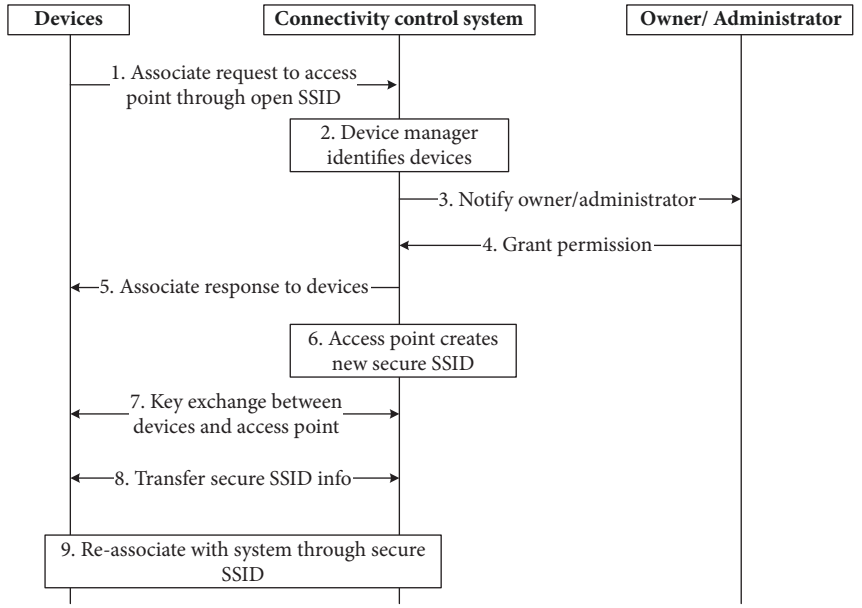


Figure 7: Procedure of secure connection establishment in the connectivity control system.

we have proposed an SDN-based connectivity control system to facilitate connectivity control of Wi-Fi devices as well as providing a high-level authentication. The proposed system allows the network owner or administrator to manage Wi-Fi devices through remote service, and it does not require any configuration information setup to the devices. We have proved the feasibility of the proposed system through the real testbed, and we expect the proposed connectivity control system to be widely utilized to enhance the experience of both users and manufacturers.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1D1A1B03933007).

References

[1] ZigBee, ZigBee Alliance, https://www.zigbee.org/.

[2] Z-Wave, https://www.z-wave.com/.

[3] Bluetooth, The global standard for connection, https://www.bluetooth.com/.

[4] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[5] D. B. Rawat and S. Reddy, "Recent advances on Software Defined Wireless Networking," in *Proceedings of the Southeast-Con 2016*, Norfolk, VA, USA, April 2016.

[6] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "Softran: software defined radio access network," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 25–30, August 2013.

[7] C. Xu, W. Jin, G. Zhao, H. Tianfield, S. Yu, and Y. Qu, "A Novel Multipath-Transmission Supported Software Defined Wireless Network Architecture," *IEEE Access*, vol. 5, pp. 2111–2125, 2017.

[8] M. Casado, N. Foster, and A. Guha, "Abstractions for software-defined networks," *Communications of the ACM*, vol. 57, no. 10, pp. 86–95, 2014.

[9] P. Ruckebusch, S. Giannoulis, D. Garlisi et al., "WiSHFUL: Enabling Coordination Solutions for Managing Heterogeneous Wireless Networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 118–125, 2017.

[10] Samsung Smart Home, https://www.samsung.com/smart-home/smartthings/.

[11] Apple Home Kit, https://www.apple.com/shop/accessories/all-accessories/homekit.

[12] Philips, https://www2.meethue.com/.

[13] M. Lee, Y. Kim, and Y. Lee, "A home cloud-based home network auto-configuration using SDN," in *Proceedings of the 12th IEEE International Conference on Networking, Sensing and Control, ICNSC 2015*, pp. 444–449, Taipei, Taiwan, April 2015.

[14] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, "Network-level security and privacy control for smart-home IoT devices," in *Proceedings of the 11th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2015*, pp. 163–167, October 2015.

[15] K.-K. Yap et al., *Separating Authentication, Access and Accounting: A Case Study with OpenWifi*, Stanford University NEC OPENFLOW-TR2011-1, 2011.

[16] Z. Cao, J. Fitschen, and P. Papadimitriou, "FreeSurf: application-centric wireless access with SDN," in *Proceedings of the 17th*

IEEE International Conference on High Performance Switching and Routing, HPSR 2016, pp. 206–212, June 2016.

[17] T. Enghardt, *Authentication, Authorization and Mobility in Openflow-enabled Enterprise Wireless Networks[Master thesis]*, Technische Universitat, Berlin, Germany, 2014.

[18] J. Schulz-Zander et al., "OpenSWDN: Programmatic Control over Home and Enterprise WiF," in *Proceedings of the ACM HotSDN*, pp. 25–30, 2013.

[19] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, "Programming abstractions for software-defined wireless networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 146–162, 2015.

[20] H. Gacanin and A. Ligata, "Wi-Fi self-organizing networks: Challenges and use cases," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 158–164, 2017.

[21] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," in *Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*, pp. 115–120, Helsinki, Finland, August 2012.

[22] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[23] E. Coronado, J. Villalon, and A. Garrido, "Wi-balance: SDN-based load-balancing in enterprise WLANs," in *Proceedings of the 2017 IEEE Conference on Network Softwarization, NetSoft 2017*, Bologna, Italy, July 2017.

[24] Z. Yang, J. Zhang, K. Tan, Q. Zhang, and Y. Zhang, "An adaptive mobility manager for Software-Defined Enterprise WLANs," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '15)*, pp. 1436–1444, Kowloon, Hong Kong, 2016.

[25] F. De Turck, P. Chemouil, W. Kellerer et al., "Guest editors' introduction: special issue on advances in management of softwarized networks," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 786–791, 2016.

[26] R. Riggio, T. Rasheed, and F. Granelli, "EmPOWER: a testbed for network function virtualization research and experimentation," in *Proceedings of the Workshop on Software Defined Networks for Future Networks and Services (SDN4FNS '13)*, pp. 1–5, IEEE, Trento, Italy, November 2013.

[27] OpenWrt, http://openwrt.org/.

[28] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000.

[29] B. Pfaff, "Extending networking into the virtualization layer," in *Proceedings of the ACM HotNets*, pp. 1–17, 2009.

[30] OpenFlow, OpenFlow Specifications, https://opennetworking.org/.

[31] Raspberrypi, Teach, learn and make with Raspberry Pi, https://www.raspberrypi.org/.

[32] FreeRADIUS, The FreeRADIUS Server Project, https://freeradius.org/.

[33] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, 2017.

[34] D. Das, J. Bapat, and D. Das, "A Dynamic QoS Negotiation Mechanism between Wired and Wireless SDN Domains," *IEEE Transactions on Network and Service Management*, 2017.

[35] D. Tu, Z. Zhao, and H. Zhang, "ISD-WiFi: An intelligent SDN based solution for enterprise WLANs," in *Proceedings of the 8th International Conference on Wireless Communications and Signal Processing, WCSP 2016*, Yangzhou, China, October 2016.

[36] M. S. Carmo, S. Jardim, T. De Souza, A. V. Neto, R. Aguiar, and D. Corujo, "Towards enhanced connectivity through WLAN slicing," in *Proceedings of the 16th Annual Wireless Telecommunications Symposium, WTS 2017*, Chicago, IL, USA, April 2017.