OpenNet: A Simulator for Software-Defined Wireless Local Area Network

Min-Cheng Chan[1], Chien Chen[1], Jun-Xian Huang[1], Ted Kuo[2], Li-Hsing Yen[3] and Chien-Chao Tseng[1]

Dept. Computer Science
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.[1]

Global Strategy Office
D-Link Systems, Inc.
Santa Clara, California, U.S.A.[2]

Dept. Computer Science and Information Engineering
National University of Kaohsiung
Kaohsiung, Taiwan, R.O.C.[3]

Abstract—**This study is motivated by a plan to install a software-defined wireless local area network (SDWLAN) on campus, which possesses a desired property that both data flow and device behaviors can be software-definable. Because the installation involves hundreds of access points, we must conduct simulations beforehand to verify the design and scalability of the target system. However, existing SDN simulator like Mininet does not support modeling of wireless channel and mobility. On the other hand, common network simulator like ns-3 only has limited support for software-defined controllers and does not fully implement handover process. We thus develop OpenNet, which connects Mininet to ns3 to enjoy both Mininet's advantage of controller compatibility and ns-3's ability in the wireless/mobility modeling. OpenNet also complements ns-3 by adding probe mechanism, which is missing in the current ns-3 implementation. Our simulation result demonstrates the effectiveness of OpenNet.**

Index Terms—**Software-defined network, wireless local area network and simulator.**

I. Introduction

Wi-Fi access points (APs) deployment and management on campus networks present constant challenges to the IT staffs. Unplanned or uncoordinated Wi-Fi AP installations, which are common on campus, cause mutual interference and throughput degradation. Besides, traffic from APs installed for experimental purpose is often mixed up with regular traffic, resulting in an unstable wireless environment.

Concerning the problems caused by legacy APs, we envision software-defined wireless local area network (SDWLAN) that applies the idea of software-defined network (SDN) to wireless local area network (WLAN). SDN provides a centralized and flexible network by applying different rules to different data flows. In the envisioned SDWLAN, not only rules for data flow but also behaviors (operating channel, access control, etc.) of devices (more specifically, APs) can be software-definable. Several advantages of SDWLAN have been envisioned:

- *Low cost*. To manage numerous APs in a traditional WLAN, commercial wireless-management products (e.g. [1]) demand specialized APs to interwork with, which incurs high cost when numerous APs are involved. In SDWLAN, OpenFlow-enabled APs follow the OpenFlow standard to provide a unified interface. The price of OpenFlow-enabled APs is much reasonable than specialized APs and thus we are able to manage the SDWLAN with lower cost.
- *Flexible management*. Several studies have shown significant handover performance improvement using a centralize controller that knows the entire network topology [2]. SDN controller can obtain information such as the neighbor list of each OpenFlow-enabled AP. With that information, it is possible to reduce the interference of adjacent APs by power control.
- *Traffic isolation*. SDN provides wireless resource virtualization and slicing that can be used to isolate certain type of traffic from others for better network stability. Deep packet inspectors can also cooperate with SDN to identify, isolate and redirect malicious traffic flow for further inspection in real-time.
- *Link layer mobility management*. Mobile devices in a traditional WLAN usually need to change their IP address when moving to a different sub-net. SDN controller is able to track the location of each mobile device and thereby redirect packets to the current location of mobile device without changes of IP addresses. This ability avoids performance bottleneck incurred by conventional network-layer handover procedure [3]. SDN can also provide seamless handover by n-casting [9]. Moreover, SDN can make multiple APs act as single big virtual AP to avoid unnecessary link-layer disconnections [4].

An increasing number of universities and enterprises plan to deploy SDN as an evolution of their network environment. We believe that SDWLAN will be soon applied to campus networks. However, since the SDN deployment involves hundreds of APs, we must conduct simulations beforehand to verify the design and scalability of the target system. The simulator under consideration must meet the following requirements:

Wireless functionality support. The simulator must be able to simulate the nature of mobile devices such as AP scanning, modeling of wireless channel and moving trajectory.

Controller compatibility. The simulator must be able to connect to and receive command from any type of OpenFlow controller. With this property, we need not modify existing controllers to be used in the simulation environment.

Extendibility. Some experimental actions such as change of vlan tag are not defined in the OpenFlow specification.

This work was sponsored in part by D-Link Systems, Inc.

Fortunately, the specification allows us to implement vendor-specific actions on OpenFlow-enabled APs. The simulator must be easily modified and deployed at the experiment SDWLAN.

We have investigated three most popular simulators used in SDN: Mininet, ns-3 and EstiNet. Mininet does not support modeling of wireless channel and mobility. Ns-3 has limited support for software-defined controllers and does not fully implement handover process. EstiNet is a commercial simulator and thus does not provide the flexibility of source-code level modification and extension. The result of investigation leads to a conclusion that existing simulators are not suitable for SDWLAN study in terms of wireless functionality support, controller compatibility and extensibility. We therefore propose an open-source [10] simulator OpenNet based on Mininet and ns-3. It connects Mininet to ns3 to enjoy both Mininet's advantage of controller compatibility and ns-3's ability in the wireless/mobility modeling. OpenNet also complements ns-3 by adding channel-scanning mechanism which is missing in the current ns-3 implementation.

The rest of this paper is arranged as follows. The next section briefs three existing simulators that support OpenFlow and exhibit the need of a new simulator. We highlight the contribution of proposed system, OpenNet, in Section III and detail the implementation in Section IV. In Section V, we demonstrate the usage of OpenNet and illustrate a simple evaluation. Finally, Section VI concludes this paper with a discussion of future works.

II. Existing Simulators

Mininet [5], a BSD licensed open-source project, has been widely used as a simulator for SDN. It is based on Linux container, a lightweight virtualization feature supported by Linux kernel version 2.2.26 and above. Linux container provides each individual processes with separate network interfaces, routing tables and ARP tables. Mininet provides a simple and inexpensive network testbed for OpenFlow applications, which supports up to 4096 hosts and switches. Mininet is also compatible with real network and various controllers. However, Mininet does not support the modeling of either wireless channel or mobility. Consequently, simulating entire handover procedure in SDWLAN, which is important in our study, is simply impossible.

Ns-3 [6] is a GPLv2 licensed open-source discrete-event network simulator for Internet systems. It is well documented and highly modularized and thus can be easily modified or extended. Ns-3 includes simulation tools for wireless channel and mobility modeling. However, the latest version of ns-3 does not yet implement scan mechanism which is necessary for layer-2 handover. This limitation makes the simulation of a complete layer-2 handover procedure in ns-3 impossible. Moreover, simulating OpenFlow behavior in ns-3 demands an implementation of OpenFlow controller inside ns-3 (as an extension of ns-3 object *ofi::Controller*). This requirement makes a testing of existing controller difficult.

EstiNet [7] are commercial simulation and emulation tools for OpenFlow networks. Unlike above-mentioned simulators, EstiNet allows not only observation but also configuration through a GUI. User can easily setup network entities and topology using drag-and-drop. It also supports wireless channel modeling and the simulation of OpenFlow switches. The main benefit of EstiNet is that it modifies some time-related functions in Linux kernel and provides more performance fidelity for users who want to evaluate system performance by simulation. However, since EstiNet is not a complete open-sourced solution, it cannot be easily extended even if we can obtain part of the source codes after purchasing it.

The comparison among these simulators is summarized in Table 1. We conclude that these simulators are not suitable for SDWLAN in terms of compatibility and flexibility. We therefore propose a new simulation system, OpenNet, which takes Mininet's advantage of controller compatibility and the wireless/mobility modeling ability of ns-3.

III. Contribution

OpenNet is a simulator for SDWLAN that is built on top of Mininet and ns-3. These two simulators by itself have high flexibility to develop and deploy new experimental protocols. Additionally, OpenNet inherits the backward compatibility from ns-3 and controller compatibility from Mininet.

OpenNet supports all simulation types supported by Mininet and ns-3, including the simulation of OpenFlow-enabled APs. In addition to putting Mininet and ns-3 together, OpenNet also augments ns-3 by implementing Wi-Fi scan mechanism, which enables the simulation of layer-2 handover of mobile nodes between two OpenFlow-enabled APs that operate on two different channels.

TABLE I. Comparison between Existing Simulators and OpenNet

Simulator	Mininet	Ns-3	EstiNet	OpenNet
OpenFlow Version	1.3.1	0.8.9	1.3.1	1.3.1
Simulation/Emulation	E	S/E	S/E	S/E
Wireless Functionality		No Scan	V	V
Controller Compatibility	V		V	V
Extendibility	V	V		V

IV. Implementation

A. Connecting Mininet to ns-3

To use ns-3 to model link layer behavior of Mininet, OpenNet needs to bridge Mininet with ns-3. Figure 1 shows how OpenNet uses a technique named "TAP device" to achieve the goal. First, for each node in Mininet, we create a corresponding *WifiNetDevice*, which is a simulated wireless network device in ns-3. This simulated device could be an AP or station (mobile node) depends on what kind of link layer module it uses. Here we use *ApWifiMac* and *StaWifiMac* to simulate the link layer behaviors of AP and station, respectively. *WifiPhy* is used to simulate the physical layer behavior of both AP and station. (*WifiNetDevice*, *ApWifiMac*, *StaWifiMac* and *WifiPhy* are all provided by the original ns-3.) All *WifiNetDevices* connect with each other by an emulated ns-3 Wi-Fi channel. In a traditional ns-3 simulation, the upper layer of *WifiNetDevice* is directly connected to a network layer module (e.g. IP). However, in OpenNet, it is connected to the IP layer of the Mininet node.

Second, we create a TAP device for each node at the Mininet side. TAP device is a virtual network interface supported by Linux kernel that simulates a link layer device. It is widely used to bridge one user-space program with another. Ns-3 provides a module named *TapBridge* that can connect the simulation environment with a TAP device, which is used by OpenNet to connect *WifiNetDevice* in ns-3 to a proper TAP device in Mininet.

An alternative open-source project in [8] uses a similar approach. That project modifies *TapBridge* in ns-3 for the connection with Mininet. At the Mininet side, a new module is written to generate corresponding *TapDevice* to connect with ns-3 through *TapBridge*.

B. Implementing scan mechanism in ns-3

Wi-Fi stations in ns-3 do not perform automatic AP scanning among different channels, which is fine if all APs operate on the same channel. However, in real-world wireless environment, adjacent APs usually operate on different channels to avoid co-channel interference. To simulate a handover procedure between two APs that operate on different channels, we have made the following modifications on ns-3 which conform to IEEE 802.11 specification.

1) Adding scan-related attributes

a) In *wifi/bindings,* we have added the enumeration of *MacScanType: NOTSUPPORT, ACTIVE and PASSIVE.*

- *NOTSUPPORT*
  Wi-Fi station does not support scan and acts exactly the same as in the original ns-3.
- *ACTIVE*
  Wi-Fi station supports active scan. When a station no longer associates with any AP, it broadcasts *Probe Request* proactively on each channel.
- *PASSIVE*
  Wi-Fi station supports passive scan. When a station no longer associates with any AP, it passively listens to beacons on each channel.

b) In *wifi/model/sta-wifi-mac,* we have added three scan-related attributes.

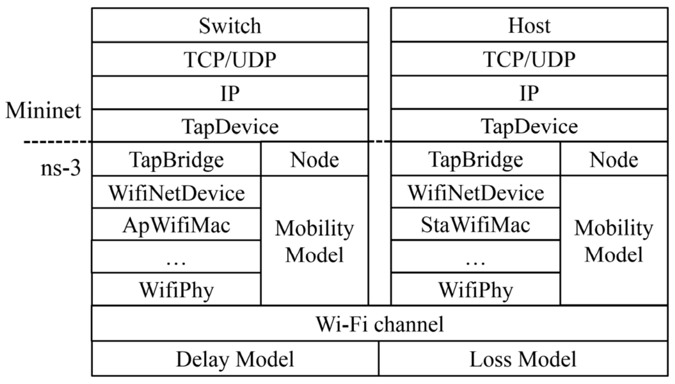


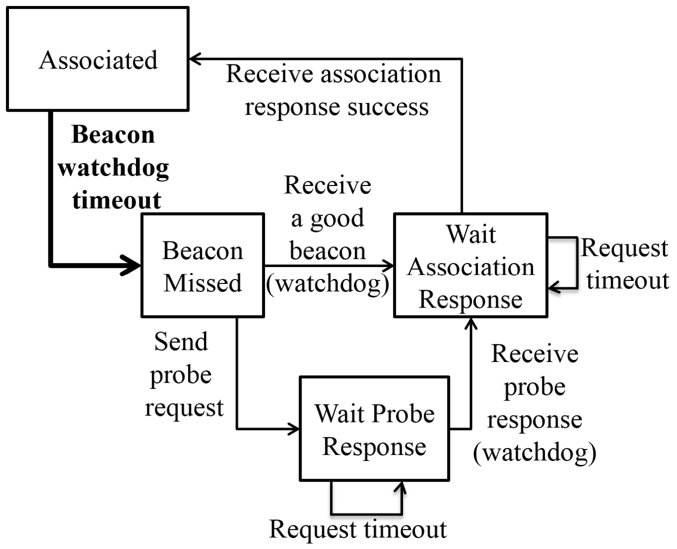
Fig. 1. How OpenNet connects Mininet with ns-3

- *MaxScanningChannelNumber*
  The maximum number of channels to be scanned.
- *MaxChannelTime*
  The maximum scanning time spent in one channel.
- *MinChannelTime*
  The minimum scanning time spent in one channel.

2) Adding scan-related states into state machine

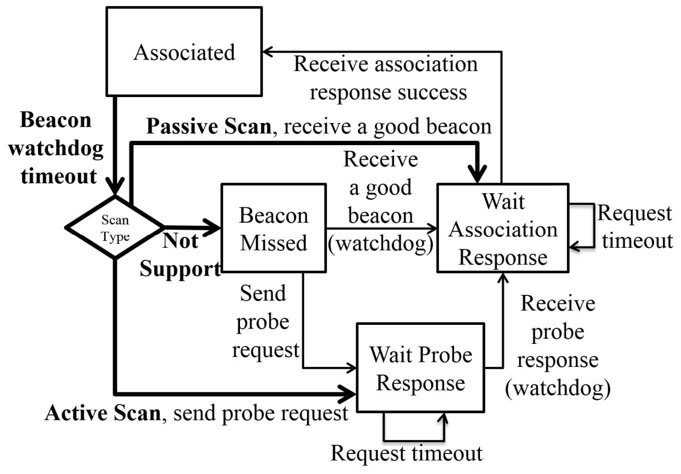
Figure 2 shows the original state machine of *StaWifiMac* module in ns-3. When a specific number of beacons get lost and the beacon watchdog timer eventually expires, station changes state from *Associated* to *Beacon Missed*. In *Beacon Missed* state, station sends probe requests with original SSID or listens to beacons only on the original channel.

Figure 3 shows the state machine modified by OpenNet. When beacon watchdog expires, station first checks the *MacScanType* attribute. For *NOTSUPPORT* type, the state transits to original *Beacon Missed* state followed by the original ns-3 reconnection behavior without performing a cross-channel scan. For the *ACTIVE* type, station first switches to a specific channel and attempts accessing the medium using distributed coordination function (DCF) before sending a probe



Associated
Receive association response success
Beacon watchdog timeout
Receive a good beacon (watchdog)
Beacon Missed
Wait Association Response
Request timeout
Send probe request
Receive probe response (watchdog)
Wait Probe Response
Request timeout

Fig. 2. Original State Machine of StaWifiMac in ns-3



Associated
Receive association response success
Beacon watchdog timeout
Passive Scan, receive a good beacon
Receive a good beacon (watchdog)
Scan Type
Not Support
Beacon Missed
Wait Association Response
Request timeout
Send probe request
Receive probe response (watchdog)
Wait Probe Response
Active Scan, send probe request
Request timeout

Fig. 3. Modified State Machine of StaWifiMac in OpenNet

request. If the medium is never busy during *MinChannelTime* (which indicates that it is likely no one communicates on this channel), station continues probing the next channel. Otherwise it waits for probe response until *MaxChannelTime.* For the *PASSIVE* type, station switches from channels to channels to listen to beacons.

3) *Obtaining SNR information*

After station discovers an available AP (by either active or passive scan), it decides which AP to connect based on Signal to Noise Ratio (SNR) of the beacon. We use SNR instead of RSSI since SNR can reflect the noise ratio and thus react to collision and retransmission. In the real world, SNR can be obtained from driver if supported. In ns-3, a slight modification is required to obtain SNR. Figure 4 shows the block diagram of station's MAC layer in ns-3. In the original implementation, SNR information is only stored in *MacLow*. Since the state machine as well as the scan procedure is located in *StaWifiMac*, we have modified both *MacLow* and *MacRxMiddle* to let SNR information be forwarded to *StaWifiMac*.

For handover policy, OpenNet currently chooses the candidate AP with the highest SNR. We can modify the state machine to implement other AP selection rules or even pre-scan mechanism that selects new AP before disconnection of current link to further shorten handover latency.

V. Demonstration and Evaluation

Running simulations on OpenNet demands a simulation script that specifies location/connection of network entities, and the mobility model of stations. The script can also set delay and loss models in the simulation. OpenNet users can load topology files in JSON format that can be bulk generated outside OpenNet. OpenNet is able to bridge a real network with a simulated network and thus switches can connect to controllers either inside or outside OpenNet through normal TCP/IP connections.

We ran OpenNet to confirm the effectiveness of our design. Our demonstration script deploys seven OpenFlow-enabled APs s1-s7 as shown in Figure 5. Each AP has several Ethernet interfaces and one Wi-Fi interface and is connected to a real OpenFlow controller that is omitted in the figure. In this demonstration, APs act like layer-2 learning switches. One mobile node h1 roams from switch s1 straightly toward s3 while another station h2 stays stationary at s3 as a corresponding node. H1 keeps sending ICMP echo requests to h2 during the whole procedure. Latency is set to 2 ms for all wired link and 200 ms for the wireless links.

OpenNet logs simulation events and generates an XML file as a result. A GUI provided by ns-3 named NetAnim can use the XML file to render an animation for the whole simulation. Table 2 shows a simplified simulation log.

A. Before handover (Packets 1-3)

Before the handover, the first ICMP packet went through h1, s1, and s3 to h2. Table 2 shows that the latency between packet 1 and packet 2 is 350 ms. Except for 200 ms wireless transmission latency, there is an extra 150 ms latency since s1 did not know where to forward the very first packet and asked the controller for direction.

B. During handover (Packets 6-11)

During the handover, only s2, s3, and s4 received requests sent by h1 since other APs were out of h1's communication range. Packets 6 and 7 were actually sent at the same time since s2 and s4 used the same channel. H1 finally associated with s3 because s3 had the best SNR at the location of h1.

C. After handover (Packets 12-13)

We will discuss the location update procedure in two cases:

a) Mobile node proactively sends data after handover

In this case, the new AP does not know how to process the packet and thus forward it to the controller. The controller soon realizes that this packet is sent from mobile node (by analyzing its source MAC address) and then updates forwarding table of all switches to redirect packets destined for the mobile node.

b) Mobile node reactively receives data after handover

In this case, the new AP must inform the controller of h1's new location when the mobile node attaches to the new AP.

Mobility management in case b requires some modifications to the behavior of APs, which remains to be our future work.

Our example verifies that OpenNet can simulate the roaming and scanning behaviors of mobile nodes and can interact with OpenFlow controller.

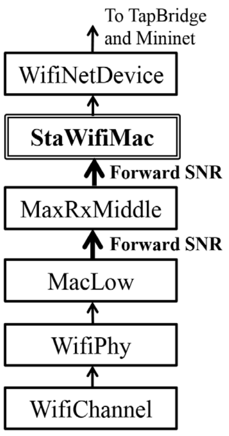


Fig. 4. Block diagram of StaWifiMac in ns-3

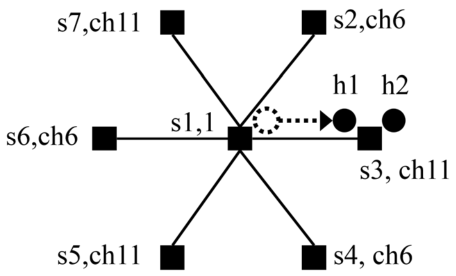


Fig. 5. Simulation Topology

TABLE II. Simplified Simulation Log

Packet No.	Tx Time (s)	From	To	Type
Before Handover				
1	0.2659	h1	s1	Wi-Fi data
2	0.6158	s1	s3	Ethernet data
3	0.8150	s3	h2	Wi-Fi data
During Handover				
4	21.2702	h1	s3	Probe request
5	21.2706	s3	h1	Probe response
6	21.3715	h1	s2	Probe request
7	21.3715	h1	s4	Probe request
8	21.3721	s2	h1	Probe response
9	21.3749	s4	h1	Probe response
10	21.4243	h1	s3	Association request
11	21.4253	s3	h1	Association response
After Handover				
12	21.546	h1	s3	Wi-Fi data
13	21.747	s3	h2	Wi-Fi data

VI. Conclusion and Future Works

In this paper, we first discuss the benefits of SDWLAN and the need for a SDWLAN simulator. We point out that existing network simulators do not fulfill our needs. Specifically, Mininet does not support modeling of wireless channel and mobility; ns-3 has limited support for OpenFlow controllers and does not fully implement handover process; and EstiNet does not enable source-code level modification or extension. We propose OpenNet, which is an open-source simulator based on Mininet and ns-3. OpenNet connects Mininet to ns3 to enjoy both Mininet's advantage of controller compatibility and ns-3's ability in the wireless/mobility modeling. OpenNet also complements ns-3 by adding channel scan mechanism. Our preliminary experiment shows that 1) OpenNet can correctly simulate an SDWLAN in which controllers can communicate with OpenFlow-enabled APs and 2) wireless station in OpenNet can successfully handover to another AP by scanning and switching to another channel.

In the future, we shall focus on the performance evaluation and enhancement of large-scale simulation in OpenNet. In addition, we shall study possible solutions for network-assisted mobility management by implementing wireless access controller (WAC) over SDWLAN controller. The WAC will manage resource allocation, association, authentication and even pre-authentication, providing comprehensive information for Wi-Fi stations to make handover decisions.

References

[1] Aruba Mobility Controller, available at http://www.arubanetworks.com/products/mobility-controllers/

[2] Chien-Chao Tseng *et al.* "Topology-aided cross-layer fast handoff designs for IEEE 802.11/Mobile IP environments," *IEEE Communications*, Vol. 43, No. 12, pp. 156-163, 2005.

[3] Ki-Sik Kong *et al.* "Handover latency analysis of a network-based localized mobility management protocol", *IEEE ICC*, 2008.

[4] Lalith Suresh *et al.* "Towards programmable enterprise WLANs with Odin." *Hot Topics in Software Defined Networking*, 2012.

[5] Bob Lantz, Brandon Heller, and Nick McKeown. "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks." *9th ACM Workshop on Hot Topics in Networks*, 2010.

[6] ns-3 OpenFlow switch support in version 3.18, available at http://www.nsnam.org/docs/release/3.18/models/html/openflow-switch.html

[7] Shie-Yuan Wang, Chih-Liang Chou, and Chun-Ming Yang. "EstiNet OpenFlow network simulator and emulator." *IEEE Communication Magazine*, Vol. 51, Issue 9, 2013.

[8] Link modeling using ns-3, available at https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3

[9] Kok-Kiong Yap *et al.* "OpenRoads: Empowering Research in Mobile Networks." ACM SIGCOMM Computer Communication Review archive, Vol. 40, Issue 1, pp. 125-126, 2010.

[10] Source code of OpenNet, available at *http://github.com/dlinknctu/OpenNet*.