Software-Defined Wi-Fi Networks with Wireless Isolation

This page contains the information how to enable SDN on off-the-shelf Access Points, either based on OpenWRT or others without full customization capabilities. The instructions are somewhat outdated and waiting for a controller (under development). New version of the page is available at: https://version.helsinki.fi/swift/swift-controller

Introduction

Wi-Fi networks were one of the first use-cases for Software-Defined Networking (SDN). However, to deploy a software-defined Wi-Fi network today, one has to rely on research prototypes with availability, documentation, hardware requirements, and scalability issues.
To alleviate this situation, we demonstrate two simple techniques to bring SDN functionality to existing Wi-Fi networks and discuss the benefits and short-comings of each technique.
Researchers can use our techniques to convert their existing Wi-Fi testbeds into software defined Wi-Fi testbeds.
Our two techniques therefore significantly lowering the barrier-to-entry to deploy and experiment on Software-Defined Wi-Fi Networks.

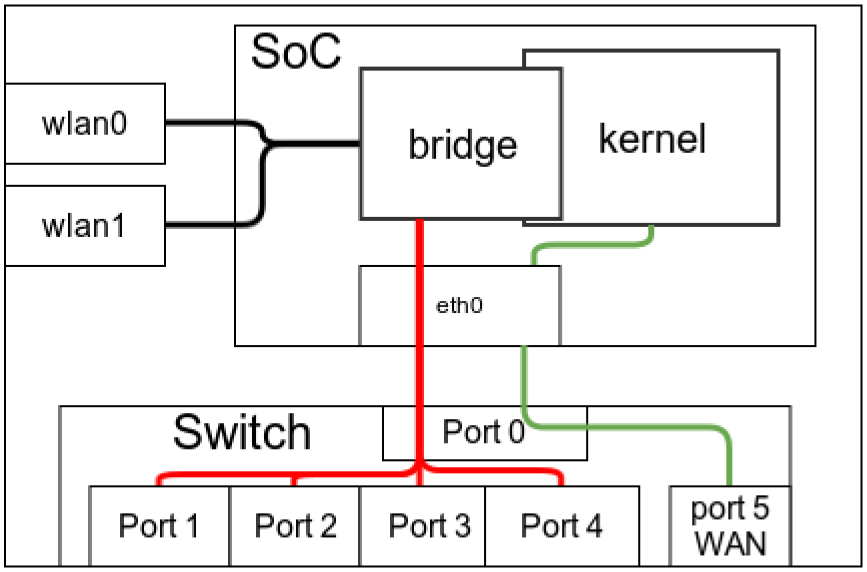
We categorize our two approaches as Intelligent Edge and Thin Edge. The Intelligent Edge is an approach where as much as possible is done at the AP, i.e. Open vSwitch (OVS), or any other software OpenFlow switch, is located at the AP. The other approach, Thin Edge, relays all traffic to an another device running OF switch. The Intelligent Edge in general is more useful for research testbeds and offers more options on device management, whereas the Thin Edge allows the usage of older or non-customizable equipment.

In our work, we leverage on OpenWrt, Open vSwitch and hostapd to create off-the-self OpenFlow Access Points. The instructions provided here detail how to implement our approaches on an OpenWrt AP, an AP created with a Linux laptop, and how an existing device without OpenWrt or custom firmware support can be attached to create an SDN controlled Wireless network. How to install or compile LInux or OpenWrt are out of scope for these instructions as they are better served by distribution providers.

Background

In modern WLAN networks the APs, and clients to some degree, are managed by wireless controllers. While this allows better resource usage and makes building of large networks possible, the network still has restrictions on how to manage and control the actual clients and where they can communicate. In normal APs, the wireless interface is actually a bridge. If a frame from a wireless client to an another client associated to the same AP arrives to the bridge, the frame is sent straight to the other client without going to the actual network stack of the AP. To combat this, a solution called wireless isolation (or peer-to-peer blocking or Public Secure Packet Forwarding (port security) in equipment Cisco) is implemented in some APs. With isolation enabled, depending on the isolation, the bridge sends all frames to the upper network stack. This in theory alleviates some security problems in wireless networks, but does not prevent two clients associated to different APs to communicate with each others unless the isolation goes deeper. For example, the Port Security on Cisco devices is also supported on Cisco switches.

SDN is a networking paradigm where the control and forwarding planes are separated. The control plane of a network device (switch, AP) is programmed using a SDN controller and appropriate protocol, typically OpenFlow (OF). SDN is currently used in many places, in data centers and in backbone Internet routes. Google for example is using their SDN solution in their network. Where SDN is lacking is wireless networks. The SDN is based on ports and wires, which are missing in wireless networks. Bringing SDN to wireless networks is a hot research topic, and there are already several different deployments, such as BeHop[4] in Stanford and Aetherflow[5]. Both approaches are viable and use ingenious methods, but both have limitations on their deployment and resource usage. Mainly both require OpenWrt based hardware and have scalability issues in how many virtual wireless APs they can support and how the flows between wireless clients are managed.



The diagram above shows how a typical access point is wired internally. The AP consists of a System-on-a-Chip with one or more Ethernet interfaces (eth0), one or more WiFi interfaces (wlanX) and a Virtual LAN (VLAN) capable switch. The eth0 of the SoC is connected to one of the switch ports and rest of the ports are typically divided into two VLANs, one for switched ports (ports 1-4), and one for the WAN port of the AP (port 5).

Wireless Isolation

Wireless Isolation is a technology available on various Access Points. The Wireless Isolation denies wireless clients from communicating between each others by not forwarding wireless frames between the clients. The main benefit is for large public WiFi networks to have an additional layer of security. Since no two clients can communicate, the attack surface against clients is smaller. The wireless isolation is available in Linux MAC80211 drivers, and for example in Cisco APs (called Port Security, PSPF or Peer-to-Peer blocking depending on if the AP is autonomous or connected to a controller). In Linux, the isolation is implemented by sending the packets in wireless frames to the upper network stack instead of just using the wireless network interface as a bridge between wireless clients. If the frames are just bridged between clients, the network stack of the AP does not see the packets and cannot affect them. The Cisco on the other hand sends only ARP to upper network stack.

OpenFlow Rules to Work Around Wireless Isolation

The default case when a packet (ARP or others) arrives, it is send out from an another port and never back to the port where it came from. In OVS, the key action is actions=output:in_port,local i.e. send the packet also back to the interface where it came from. Normal output:<port> does not send the packet back if it came from <port>.

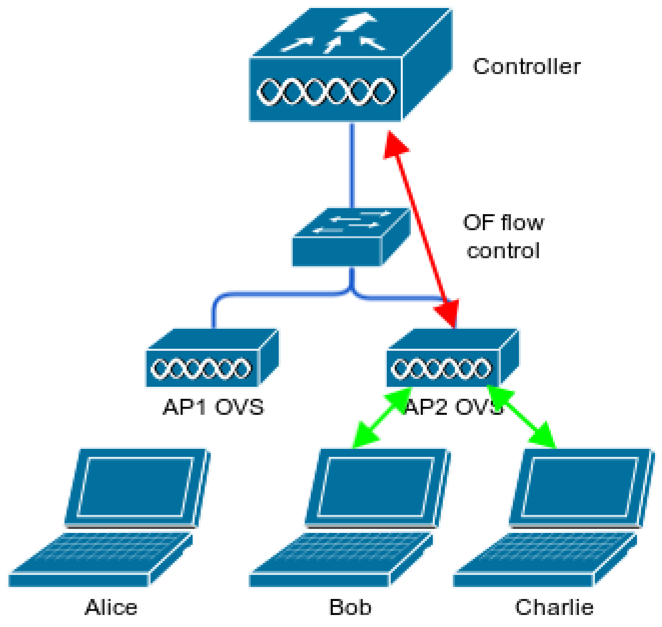
VLAN Aggregation for Efficient IP Address Allocation (PVLAN)

Another way to provide ARP support is to use VLAN Aggregation for Efficient IP Address Allocation, or Proxy ARP for Private VLANs, detailed in RFC 3069. It was mainly developed to allow a better allocation of IPv4 addresses in cases where the same IPv4 subnet is divided among multiple Virtual LANs. For our approaches, the main usage is to allow a host, the AP in our case, to impersonate other wireless clients by using its own MAC address as the response for ARP queries. With wireless isolation, the ARP queries sent by the wireless clients are visible only for the AP and not to the other wireless clients. With PVLAN, the AP sees the ARP queries arriving from the wireless interface and if the AP knows that a client in the wireless network has the MAC address of the query, the AP replies with its own MAC address. Subsequently, the requesting wireless client will now use the AP's MAC address as the destination MAC for the packet to the other wireless client. When the AP receives such packets, it replaces the source and destination MAC addresses with its own MAC and the destination's MAC address, and sends the packet back to the wireless interface. With this, the wireless clients can now communicate using the AP as a relay. In combination with the wireless isolation, the PVLAN allow the OVS in the AP to see and handle all flows between wireless clients with minimal changes to any of the hosts. The main drawback for PVLAN is that some of the MAC based flow rules may not work. Also, as the AP acts as a relay server for the wireless clients, the Time-To-Live value is decremented by one. This causes some device discovery protocols to break, such as the protocols used to find a Chromecast in the network. On the other hand, the OpenFlow rule set in general is slightly smaller.

Our Approaches

We have devised two easily deployed methods to bring SDN to the wireless world. These approaches enable the SDN research on testbeds using off-the-shelf equipment and can even be applied to existing networks. While our approaches are relatively simple, they solve both scalability issues and the flow management between wireless clients. Both approaches require an SDN switch in the network. The SDN switch can be installed on the AP (Intelligent Edge) or at some other host in the network (Thin Edge). We leverage on the wireless isolation to get the wireless bridge to send all traffic to the upper network stack of the AP, including ARP requests. Depending on the Wireless Isolation implementation, i.e. are all packets or just ARP send to the upper network stack, we also enable PVLAN on the SDN switch host in latter case, which allows ARP requests to be send back to their originating interface. With these two, wireless isolation and PVLAN, many wireless networks can be made to support SDN.

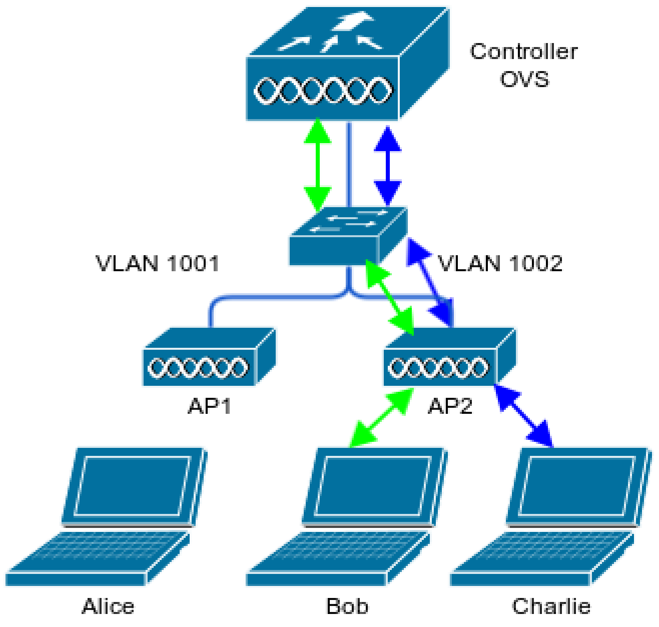
Intelligent Edge



The Intelligent Edge approach runs an SDN switch on the APs, i.e. at the network edge. This approach is deployable on the APs which support SDN, and wireless isolation. The easiest way to combine these is to use an AP with OpenWrt support. The wireless isolation is enabled to force the wireless interfaces to send all packets from wireless clients to the upper network stack of the AP, which makes all network traffic visible to the AP, including traffic between wireless clients (Bob and Charlie in the figure). The wireless interfaces are moved to the OVS, which in turn allows the fine grained control of the network traffic. This makes applications such as access control at the AP level possible. We can allow the clients to reach certain services such as Internet access and email, and deny other such as access to file servers or printing services.

The OVS also allows the control and management of the APs through a controller. While OF does not have any commands to actually control the hardware side of the controller, the controller can be used with other AP management solutions to bring the benefits of a managed WiFi network to the SDN world.

Thin Edge



The Thin Edge approach is used when the APs support wireless isolation as detailed before, but do not support custom firmware or users are not allowed to install custom software on the APs. This is typically the case with enterprise grade APs or with some of the consumer equipment. The APs are directly connected to an host running OVS (or other SDN switches), where each AP is presented as a network interface to the OVS.

Currently, this approach several requirements in addition to the APs and the controller. First, a host with OVS (or other SDN switch) is needed. Depending on the host and the wireless isolation implementation, either the host or the Controller has to use PVLAN to allow the wireless clients to use ARP. The OVS, controller and PVLAN can be co-located or run in separate hosts. The second requirement is for the network infrastructure to support VLANs. This requirement is due to fact that the OVS host has to be able to distinguish each AP and that in normal switched environment, the switches send packets to their destination instead of having them to flow through certain devices. The VLANs are the easiest way to accomplish this (also using separate physical network interfaces and running wires from APs to the OVS would work but it does not scale nor is physically possible in buildings).

When the requirements are met, the Thin Edge allows non-SDN capable equipment to be used in SDN testbeds.

Limitations

So far, our approaches have a few limitations. First, since all traffic has to go to an SDN switch, we incur some performance penalties. In OpenWrt and OVS, the traffic has to traverse the CPU of the device. These devices typically are not very powerful, although our brief tests do not show large drop in performance for single flows.

The second limitation comes from the PVLAN. With pvlan, the AP shows as the destination MAC address or source MAC address depending if the wireless clients are sending or receiving frames. Due to this, each packet is seen twice in the network with different MAC addresses.

Third, the PVLAN causes the Time-To-Live (TTL) value of the packets to be decreased by one. This is due to fact that the OVS acts as a router instead of a switch. Main issue with this that some LAN discovery protocols such as DIAL will stop working as they use TTL value of one to probe the local subnet and not beyond.

Implementation

The Intelligent Edge approach is deployable on APs that allow custom firmware and software to be run on the AP, such as APs that are supported by OpenWrt. We install OpenWrt and OVS on the AP. The default configuration of OpenWrt uses a Linux bridge to bridge WiFi interfaces (2.4GHz and 5GHz) and Ethernet interfaces into a single bridge interface. In Intelligent Edge, the interfaces (both WiFi and Ethernet) are removed from the bridge, and are added separately to the OVS bridge.

Wireless Isolation with hostapd on a Linux Laptop

It is possible to create an AP using a Linux laptop. This requires a WiFi card / USB dongle which supports Master mode. An example hostapd configuration with WIreless Isolation and WPA2 enabled is attached below. To create a full selfcontained AP a DHCP server and routing needs to be provided. For example. dnsmasq provides both DHCP and DNS for clients. In this example, the AP uses wlan2 interface for the AP.

hostapd.conf

```
#http://wiki.stocksy.co.uk/wiki/Multiple_SSIDs_with_hostapd
#change wlan0 to your wireless device
interface=wlan2
driver=nl80211
hw_mode=g

# MAC address of the interface comes here, change first value 02 for local configuration
# and last to 30for the first bssid (next will be 31 and so on)
bssid=02:3a:35:c1:e1:30
ssid=MY-SSID
channel=6
ap_isolate=1

# comment out auth, wpa, rsn configurations if you do not need encryption

auth_algs=1           # 1=wpa, 2=wep, 3=both
wpa=2                 # WPA2 only
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=SecretPassword
bss=wlan2_0
ssid=MY-SECOND-SSID
ap_isolate=1
auth_algs=1           # 1=wpa, 2=wep, 3=both
wpa=2                 # WPA2 only
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=SecretPassword
```

Wireless Isolation with OpenWrt

Enabling isolation on OpenWrt AP requires logging into the AP via telnet or SSH as there is no GUI available to enable the isolation. After logging to the AP, edit /etc/config/wireless and add "option isolate '1'" on the wifi-iface where isolation is needed. Afterwards, the networking needs to be restarted for the isolation to be activated.

Wireless isolation

```
vim /etc/config/wireless

# locate wifi-iface and add isolation
config wifi-iface
    option isolate '1'
```

Wireless Isolation with Cisco 1131 Autonomous AP

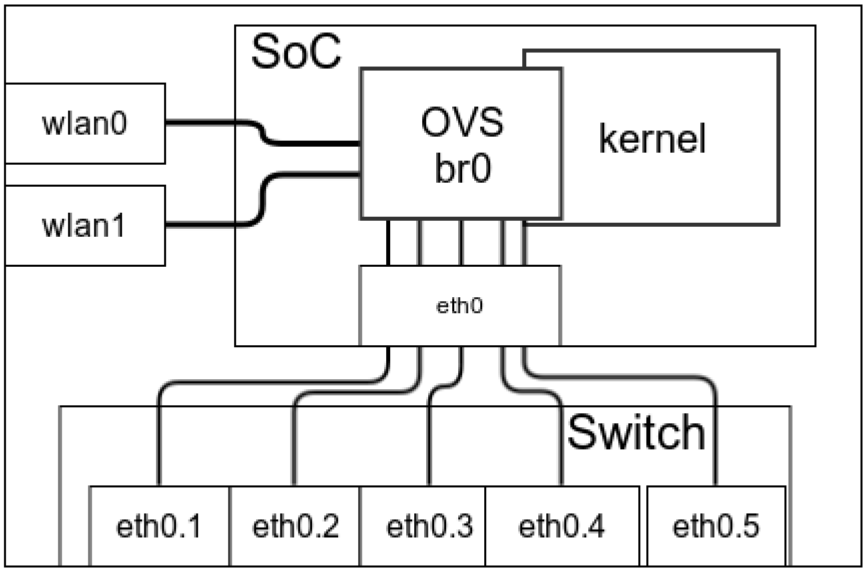
With Cisco devices, wireless isolation is called Port security. If the whole network is created using Cisco equipment, the Port Security can be enabled also in the switches to provide full isolation of clients across APs. In the code below, the Port Security is enabled both on 2.4GHz and 5GHz radio interfaces. Currently PVLAN is required on the SDN switch.

Cisco Thin Edge

```
configure terminal
        interface dot11radio 0
                bridge-group 1 port-security
                exit
        interface dot11radio 1
                bridge-group 1 port-security
                exit
        exit
```

OVS and OpenWrt

Running OVS on OpenWrt is relatively straightforward. At best, the OVS can be installed from the package manager but otherwise OVS needs to be compiled and then installed.



After the OVS is up and running, the network interfaces need to be added to the OVS. Depending on the AP, the ethernet ports of the AP are all in single VLAN and the WAN port is in an another VLAN. These VLANs are then used to map the interfaces to the OpenWrt. WLAN interfaces are separate interfaces on the SoC. To create a full OpenFlow managed AP, each of the Ethernet interfaces need to be in its own VLAN. First remove the Ethernet interfaces from any bridges running in the OpenWrt and then add these to the OVS separately. Same is done for the WLAN interfaces. Removing interfaces from default bridges can be done through the LuCI GUI or by hand.

Remove interfaces from Linux bridges

```
# Find the bridge where interfaces are by default
brctl show
# Remove interfaces
brctl delif <bridge> <interface>
```

Create OVS bridge:

Create OVS bridge

```
ovs-vsctl add-br br0
```

Add wireless and Ethernet ports to OVS bridge:

Add interfaces to the OVS bridge

```
ovs-vsctl add-port br0 wlan0
ovs-vsctl add-port br0 wlan1
ovs-vsctl add-port br0 eth0.1
ovs-vsctl add-port br0 eth0.2
ovs-vsctl add-port br0 eth0.3
ovs-vsctl add-port br0 eth0.4
ovs-vsctl add-port br0 eth0.5
```

If PVLAN is used to allow ARP to work in the wireless network, PVLAN needs to be enabled on the bridge. This allows the AP to impersonate other wireless clients (one could even say NAT the MAC addresses).

proxy_arp_pvlan

```
echo 1 > /proc/sys/net/ipv4/conf/br0/proxy_arp_pvlan
```

OVS and Laptop

Running OVS on a Linux Laptop is relatively straightforward. At best, the OVS can be installed from the package manager but otherwise OVS needs to be compiled and then installed. Hostapd is also available from the package manager, but might be an old version. Also, a patch is needed for the hostapd to detect OVS so compiling the hostapd by hand is recommended. Details are below this chapter.

After the OVS is up and running, the WiFi network interfaces need to be added to the OVS.

Create OVS bridge

```
ovs-vsctl add-br br0
```

Add wireless ports to bridge.

Add interfaces to the OVS bridge

```
# wlan2 is the interface we use for the main SSID, wlan2_0 is created for the second SSID
ovs-vsctl add-port br0 wlan2
ovs-vsctl add-port br0 wlan2_0
```

For the laptop to act as a home router, a DHCP server such as dnsmasq is needed. Also, IP forwarding and NAT between primary Ethernet (typically eth0) and the OVS bridge (br0 here) has to be enabled using for example iptables. For this, there are multiple howtos available in the Internet.

If PVLAN is used to allow ARP to work in the wireless network, PVLAN needs to be enabled on the bridge. This allows the AP to impersonate other wireless clients (one could even say NAT the MAC addresses).

proxy_arp_pvlan

```
echo 1 > /proc/sys/net/ipv4/conf/br0/proxy_arp_pvlan
```

OpenFlow Rules

To enable packets to be send back to the interface where they came from, specific OpenFlow rules need to be set. The main enabler is the "actions=in_port", which sends the packet back to the interface it came from. actions=output:<port where packet came> is not enough.

Thin Edge

Basic rule that allows the AP to act as a router, but wireless clients are isolated. These rules assume that an AP running the Thin Edge is located in OVS port 1.

```
" Allow normal operations but keep wireless clients isolated (no ARP between wireless clients)
ovs-ofctl add-flow br0 priority=0,actions=NORMAL
```

To remove isolation in OVS for Thin Edge:

```
# Wlan is in port 1, no Ethernet ports in OVS except the LOCAL bridge port
ovs-ofctl add-flow br0 "priority=2,arp,in_port=1,actions=output:in_port,normal"
ovs-ofctl add-flow br0 "priority=2,icmp,in_port=1,actions=output:in_port,normal"
ovs-ofctl add-flow br0 "priority=2,udp,in_port=1,actions=output:in_port,normal"
ovs-ofctl add-flow br0 "priority=2,tcp,in_port=1,actions=output:in_port,normal"
```

To remove isolation in OVS for Thin Edge and PVLAN:

```
# Wlan is in port 1, no Ethernet ports in OVS except the LOCAL bridge port
ovs-ofctl add-flow br0 "priority=2,arp,in_port=1,actions=local"
ovs-ofctl add-flow br0 "priority=2,icmp,in_port=1,actions=output:in_port,normal"
ovs-ofctl add-flow br0 "priority=2,udp,in_port=1,actions=output:in_port,normal"
ovs-ofctl add-flow br0 "priority=2,tcp,in_port=1,actions=output:in_port,normal"
```

The above rules are very simple rules. As a side effect, they cause all packets to be flooded back to the wireless networks. To remedy this and provide more restricting connectivity, MAC and IP addess based rules can and should be used. Also if there are multiple SSID's, they are visible as different ports in the OVS. ARP queries are hard to isolate as they are broadcasts and other clients may learn other devices by just listening the ARP queries.

Intelligent Edge

Basic rule that allows the AP to act as a router, but wireless clients are isolated since ARP requests are not sent back to the Wi-Fi.

```
ovs-ofctl add-flow br0 priority=0,actions=NORMAL
```

To remove isolation in OVS for Thin Edge:

```
# Wlan is in port 1, no Ethernet ports in OVS except the LOCAL bridge port
ovs-ofctl add-flow br0 priority=1,actions=normal,output:in_port
```

The above rules are very simple rules. As a side effect, they cause all packets to be flooded back to the wireless networks. To remedy this and provide more restricting connectivity, MAC and IP addess based rules can and should be used. Also if there are multiple SSID's, they are visible as different ports in the OVS. ARP queries are hard to isolate as they are broadcasts and other clients may learn other devices by just listening the ARP queries.

OVS and WPA2

Currently hostapd does not work well with the OVS. While normal access point associations etc. work with the default hostapd of OpenWrt, WPA encryption does not. Reason for this is that frames used for association are control frames and the frames used for setting up encryption (EAP over LAN, or EAPOL, 802.1x) are data frames. In general, the hostapd can detect Linux bridges, i.e. the normal bridges where wireless interfaces are part of. When wireless interface is added to the OVS, the interface is no longer a normal Linux bridge, so hostapd can no longer see the 802.1x frames. The modification added by Helmut Jacob "hschaa" allows hostapd to tap into OVS bridges. With the patch, the encryption can be set up, but the hostapd has to be compiled separately. The patch is not needed if encryption is not required.

https://github.com/hschaa/hostapd/commit/c89daaeca4ee90c8bc158e37acb1b679c823d7ab#diff-165dd5a1681d9394993972f6923fddf8R153

Troubleshooting Association Failures

If the authentication handshake does not work, there are two probable reasons. First, the OVS might have stale information about the WLAN interface (for example the hostapd was restarted or the interface was not up when OVS was started). This can be remedied by removing and adding the interface in OVS.

```
ovs-ofctl del-port br0 wlan2
ovs-ofctl del-port br0 wlan2_0

ovs-ofctl add-port br0 wlan2
ovs-ofctl add-port br0 wlan2_0
```

Second, especially if the OVS is controlled by a controller, the rules pushed by the controller may prevent the hostapd from seeing the 802.1x authentication frames. This can be remedied by adding flows with Ethertype (dl_type) 0x888E to and from LOCAL port of the AP.

EAP

```
# Assume wlan0 is in port 1, wlan1 is in port 2. The ports may change if the interfaces are removed and
added again

ovs-ofctl add-flow br0 "priority=10000,dl_type=0x888e,in_port=1,actions=LOCAL"
ovs-ofctl add-flow br0 "priority=10000,dl_type=0x888e,in_port=2,actions=LOCAL"
ovs-ofctl add-flow br0 "priority=10000,dl_type=0x888e,in_port=LOCAL,actions=output:1,output:2"
```

Acknowledgements and Contact Information

This work has been supported by the Tekes 5GTrek research project, Digile Cyber Trust project, Academy of Finland Cloud Security Services project, and the Nokia Center for Advanced Research (NCAR).

http://ncar.cs.helsinki.fi/

Seppo Hätönen, Petri Savolainen, Ashwin Rao, Hannu Flinck, and Sasu Tarkoma. 2016. Off-the-Shelf Software-defined Wi-Fi Networks. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference* (SIGCOMM '16). ACM, New York, NY, USA, 609-610. DOI: http://dx.doi.org/10.1145/2934872.2959071

Primary contact:

Seppo Hätönen (seppo.hatonen (at) cs.helsinki.fi)

Ashwin Rao (ashwin.rao (at) cs.helsinki.fi)

References

1. OpenWrt
2. hostapd
3. http://openvswitch.org/
4. Stanford University - BeHop Project
5. {\AE}therFlow: Principled Wireless Support in SDN
6. OVS support for hostapd