WHITE PAPER

Software-Defined Wireless Networking:

Concepts, Principles and Motivations

April 27, 2015

Copyright © Anyfi Networks AB. All Rights Reserved.

Anyfi Networks

Carrier Wi-Fi That Just Works

Table of Contents

Executive Summary **3**

Why Software-Defined Wireless Networking? **4**

Breaking the Tie Between Radio and Service 5

Using Global Packet Delivery to Solve the Radio Access Problem 7

Connecting Radios and Services On Demand 9

Limitations of Previous Architectures **10**

Why not CAPWAP? 10

Why not OpenFlow? 12

Separating the IEEE 802.11 Networking Planes **13**

Management Plane 14

Control Plane 15

Data Plane 17

Motivation 18

Control Plane Abstractions for IEEE 802.11 .... **19**

Forwarding Abstractions 19

- Client 19
- Radio 19
- Service 19
- Service Termination Point 20
- Virtual Access Point 20
- Radio Policy 21

Network State Abstraction Layer 21

- Preference 22

Specification Abstraction Layer 22

- Binding 22

How it all Fits Together **23**

Seamless Community Wi-Fi 23

Traditional Hotspots and Homespots 25

Mobile Wi-Fi Offload with QoS integration 28

Security Model **28**

Problems with Previous Architectures 29

End-to-End IEEE 802.11i Security 30

Special Considerations 30

References **32**

Executive Summary

This white paper presents Software-Defined Wireless Networking (SDWN) to readers familiar with Software-Defined Networking (SDN).

In SDWN we essentially take the central tenet of SDN, the separation of control and data planes, and extend it with a second equally important principle: the **clear separation of radio access and service definition**. By combining these two principles we enable robust and scalable virtualization of the wireless network.

We also argue against shoehorning wireless into wireline SDN, and instead show that SDWN should be an overlay architecture with IP tunneling of raw radio frames as an integral part of its design. This lets radio access providers and service providers interconnect across the public Internet, enabling seamless user experiences across a truly heterogeneous but logically unified network.

Exemplifying with a concrete SDWN implementation for IEEE 802.11 we show that this architecture has strong security and flexibility benefits, enabling use-cases such as

- seamless and secure community Wi-Fi and remote access to Wi-Fi networks;
- very large scale hotspot and homespot deployments with end-to-end security and radio policy control;
- secure mobile Wi-Fi offload, with control plane integration between 3GPP and Wi-Fi networks for intelligent real-time traffic steering;
- sharing of infrastructure on a massive scale, where radio access providers sell raw radio access to multiple service providers, each in complete control of service definition and end-user experience.

Furthermore we show that SDWN for IEEE 802.11 can be implemented

- without any modifications whatsoever to client device software or hardware;
- with only minor software (and no hardware) modifications to Wi-Fi access points – deployable as a remote firmware upgrade to already installed equipment;

- without any new requirements on the wireline network (other than IP packet delivery); and
- with minimal dependencies on the constantly evolving IEEE 802.11 standard.

The forwarding data plane portion of our SDWN implementation is available to all vendors under a no-charge royalty-free license [1]. We have verified this software on Wi-Fi chipsets from all major vendors and make reference integrations available for several commercial Wi-Fi access point SDKs [2][3][4][5]. The first manufacturer to include the software in off-the-shelf Wi-Fi equipment is Inteno Broadband Technology, with a line of dual xDSL/Ethernet WAN residential gateways [6]. The functionality is also available on request from numerous other vendors in the residential gateway, public access point and mobile infrastructure markets.

In addition to the freely redistributable forwarding data plane implementation we also provide a free evaluation version of our Carrier Wi-Fi System [7], including the essential SDWN *controller*. This software is freely available and can be used for both commercial and non-commercial purposes, but is limited to a maximum of one hundred access points.

Why Software-Defined Wireless Networking?

Software-Defined Networking (SDN) is here to stay. Its key principle of separating the data plane from the control plane has benefits that are simply too important to ignore:

- hardware independence prevents vendor lock-in and enables unified service platforms,
- central service definition reduces complexity and improves business agility,
- real-time link level monitoring and control enables better optimization and personalization of services.

These benefits apply directly to SDWN as well. But a second principle is necessary before we can bring true network virtualization to wireless: the clear separation of radio access and service definition.

When we say radio access what we mean is delivering bits to a mobile device over an air interface, and what we propose is to keep this problem separate from everything that has to do with service definition: network identification, authentication, authorization, address assignment and higher level processing (OSI Layer 3-7).

"Modularity based on abstraction is the way things get done."

− Barbara Liskov

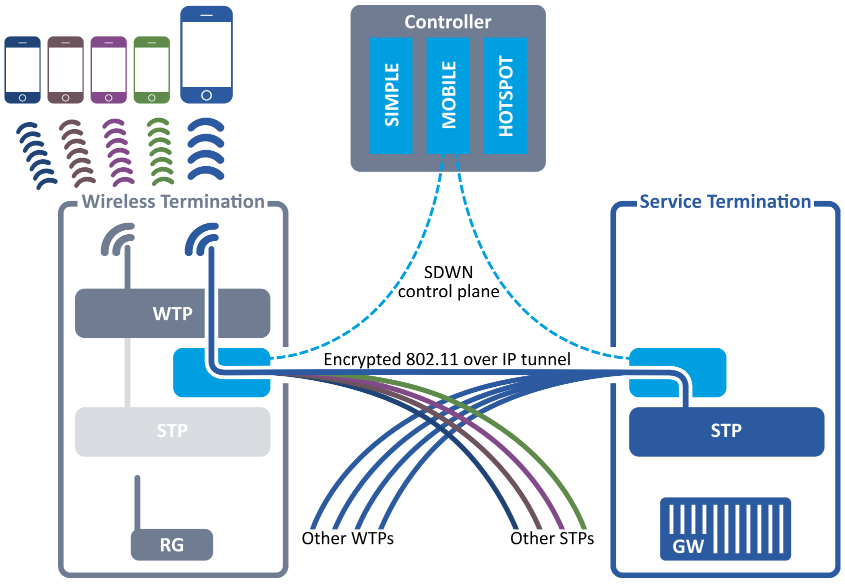
Why do we think this separation of concern is so important? Essentially because radio access and service definition are the two most difficult problems in wireless. Software engineers need separation of concern to get things done, but so do the rest of us. Meeting the future demand for mobile data will be a challenge for society at large, and we need an architecture that lets as many as possible contribute. The SDWN architecture is ideal for this, because it lets those who are well positioned to address the radio access problem (venues, landlords, municipalities and fixed-line ISPs) do so without getting into the complex business of defining, marketing and delivering a mobile data service.

Breaking the Tie Between Radio and Service

Packet delivery to and from mobile devices over an air interface is the defining problem in wireless communication. Transmitting large amounts of data long distances through the air is hard, and it is unlikely to become much easier. This is because wireless technologies, unlike wireline, are up against the hard limits of nature: the Shannon capacity [8]. A decade from now an optical fiber will likely be able to carry many orders of magnitude more data than today. The same is unlikely to be true for a mobile base station.

In many wireless architectures there is strong coupling between radio access and problems more associated with service definition, e.g. authentication and encryption. We believe that difficult problems are best addressed by separating and solving them individually. We therefore propose that the hallmark of an SDWN architecture should be that it separates the radio access problem from service related aspects such as authentication, encryption, address assignment, legal compliance and OSS/BSS integration.

FIGURE 1
Example of an SDWN architecture. A residential gateway is playing the role of *wireless termination point* (WTP) for a total of five *clients*. The blue *client* is connected to a Gateway acting as the *service termination point*. Note that radio access (left) is separated from service definition (right).



In practice this requires a repartitioning of wireless protocol stacks. In an SDWN architecture the *wireless termination point* (WTP) contains one or more *radios,* and relays raw radio frames between its wired network port and these air interfaces. The forwarding data plane consists of IP packets carrying the raw encrypted wireless frames. All higher level processing, e.g. authentication and encryption, are performed in a separate network element at the other end of the SDWN data plane tunnel: the *service termination point* (STP).

This partitioning has strong security benefits. Interestingly these benefits stem not from a separation of concern, but from purposefully keeping two things together: authentication and encryption. As any expert in cryptography will tell you encryption is meaningless without authentication, and both are pointless if an attacker can gain access to encryption keys or authentication credentials. The SDWN architecture preserves the strong mutual authentication property of underlying wireless protocols by extending both authentication and encryption across the wired network, ensuring that they both terminate in the *service termination point*. This provides strong cryptographic guarantees for user data plane integrity and confidentiality, even against an attacker in complete control of the *wireless termination point*. In fact, not even an attacker with complete control over both *wireless termination point* and *controller* can eavesdrop on or modify end-user communication; only the *client* and the *service termination point* have access to the encryption keys protecting this communication.

Using Global Packet Delivery to Solve the Radio Access Problem

But separating the radio access problem from service definition is only the first step. You also have to solve it, and when solving difficult problems you need powerful tools. We propose that SDWN should be an overlay architecture, layered on top of Internet Protocol (IP). This puts the Internet, the perhaps most powerful tool ever built, to bear on the radio access problem: a *wireless termination point* can be installed anywhere there is Internet access and it can be used to securely distribute any number of virtual networks, each securely terminated in a *service termination point* anywhere in the world (as long as it is reachable through the Internet). In the economically most interesting case a software component is installed in existing residential gateways or public access points through a routine remote firmware upgrade, adding the capability to act as *wireless termination point* within an SDWN architecture, with minimal impact on its *primary function* (e.g. providing home Wi-Fi and Internet access).

At first glance this type of overlay architecture may seem wasteful: there is of course a tunneling overhead from the IP and wireless frame headers, and payload data seldom travels the shortest path from the mobile device to the Internet services it is most likely destined for. But this is a misconception based on faulty premises:

- *Faulty premise 1: "Wireless and wireline capacity are both scarce resources, you cannot sacrifice one for the other."*

  This is an incorrect view of relative cost. Mobile data is several orders of magnitude more expensive to produce than wireline today [9], and the difference is likely to increase in the future. In fact once data is traveling over optic fiber the difference is so large that the tunneling overhead becomes negligible. There are for instance several radio over fiber technologies with well over 99% "tunneling" overhead in commercial use today [10].

  Note also that most wireless data is consumed either at home or at work [11], environments ideally suited for traditional Wi-Fi. The SDWN architecture is targeted only at those mobile use-cases where the end-user does not trust the access point operator, and wise versa. These use-cases represent just a fraction of total wireless data volume. The beauty of the SDWN architecture is that the same infrastructure can be efficiently and securely shared between a primary use (e.g. home and office Wi-Fi) and the secondary purpose of providing a mobile data service.

- *Faulty Premise 2:* *"Wireless and wireline should be integrated into the same SDN architecture."*

  We do not believe so. The concerns in wireless and wireline are quite different. Separating the control planes allows them each to evolve separately, focusing on the specifics of each domain. We find that this general line of reasoning has support in recent SDN research [12]. There are also great practical benefits in allowing SDN and SDWN to be rolled out independently. Also note that this separation of concern doesn't mean that wireless and wireline can't be integrated on the *control program* level.

- *Faulty Premise 3:* *"With Layer 3 security like HTTPS there is no need for link-level security, the wireless payload can go straight to the Internet."*

  At first glance this might seem like a reasonable objection, but for several reasons it is not. Consider for instance the role that link-level mutual authentication and data integrity serves in our legal system: Internet users are anonymous, yet responsible for their actions. The role of the service provider in this context is to hold the user's true identity in escrow, so that it can later be retrieved through a subpoena or court order should it become necessary. This role cannot be easily replaced with application-level authentication, since that would provide the end-user's true identity directly to each remote end-point (e.g. website).

- *Faulty Premise 4:* *"Wireless protocols where not designed to be transported over IP, they cannot be relied on for security in this new environment."*

  The same cryptographic principles obviously apply to both wired and wireless communication. We see few reasons to believe that the most widely used and studied wireless security protocols, e.g. IEEE 802.11i, should be any less secure when transported over a wireline network. The few reasons we do see are examined in detail in the Security Model chapter, leading us to conclude that the SDWN architecture is secure as long as the underlying wireless protocol is.

Yet another way to think about this is that an IP overlay architecture further decouples the radio access from service definition, making it a logical choice based on the core SDWN separation of concern principle.

Connecting Radios and Services On Demand

Software-Defined Networking (SDN) is about separating the control plane and the data plane, centralizing the control plane and finding reusable abstractions allowing it to be programmed. We believe that *client*, *radio* and *service* are such reusable abstractions, and that an SDWN *control program* (or *app*) should essentially control which *client* can connect to which *service* through which *radio*.

This places a fundamental requirement on the wireless protocol itself; it must be possible for an access point to detect and identify a client before it connects to any network on a logical level. This *early identification requirement* is luckily met by several widely used wireless protocols. For example, IEEE 802.11 *clients* will send out `Probe Request` frames when scanning for networks. These frames contain the MAC address of the *client*'s Wi-Fi radio, which is sufficient for early identification (but of course insufficient for authentication).

We are now ready to give the SDWN control plane its general form:

1. When a *client* comes within range of a *radio* it will be detected and identified by the *wireless termination point*. If the *wireless termination point* does not have local forwarding state for the client it will send a request to the *controller*.

2. The *controller* will reply with a set of *services*, determined by a *control program*, and the IP address(es) of each *service*'s *service termination point(s)*.

3. The *wireless termination point* then allocates a set of *virtual access points*, one for each *service*, and populates its forwarding state, tying the identity of the *client* to this set. The *wireless termination point* then presents the *virtual access points* to the *client* through its *radio*.

4. The *client* selects a *service* from the set by attempting to connect to the corresponding *virtual access point*. At that time the *wireless termination point* will connect to the relevant *service termination point*, setting up an SDWN data plane tunnel.

5. From then on the *wireless termination point* simply acts as a radio relay between the *client* and the *service termination point*, forwarding raw wireless frames between its wired and its wireless interfaces.

6. The *client* is authenticated to the *service termination point* using the security mechanism of the underlying wireless protocol. The *wireless termination point* is not part of the security equation.

The reason we believe that these are the right abstractions for the SDWN control plane are twofold. Firstly, as discussed, they allow for strong separation of concern between two of the most difficult problems in the wireless domain: radio access and service definition. Secondly, as we show in later chapters these abstractions enable complete virtualization of the wireless network.

Limitations of Previous Architectures

Traditional wireless network architectures with centralized control where designed before SDN, without a clear separation between data, control and management planes. New innovative SDN architectures on the other hand work on the IEEE 802.3 (Ethernet) level only, and therefore cannot address the unique challenges in wireless: network selection, authentication, encryption and radio resource management.

Why not CAPWAP?

The Control And Provisioning of Wireless Access Points (CAPWAP) [13] and its binding for IEEE 802.11 [14] were designed before SDN became the dominating force in network architecture. Unsurprisingly we find it lacking.

Firstly, as the name implies the protocol tries to solve both control and provisioning/management at once. We believe this conflation of control and management planes to be a mistake. From an academic perspective it is quite obvious that it goes against that most central principle of SDN: separation of concern. From a more practical perspective there are many use-cases where it makes sense to terminate the management plane and the control plane in different places and with different entities. As we will show in later chapters it is often safe to let an untrusted external party use the spare capacity of your access points (i.e. grant them SDWN control plane access), but obviously not safe to let them replace the firmware (i.e. grant them management plane access).

Secondly, while CAPWAP does provide data plane tunneling (in some implementations even on the IEEE 802.11 level) the control and data planes both terminate in the Access Controller (AC). A clear separation between data plane and control plane is the hallmark of a modern SDN architecture, and without it much of the flexibility is lost.

Recently some work has been initiated within the IETF to extend the CAPWAP protocol in order to support separate termination points for control and data plane tunnels [15]. This looks like a step in the SDN direction, towards greater scalability and flexibility. However, in our opinion CAPWAP incorrectly classifies the IEEE 802.11 authentication and encryption as belonging exclusively to the control plane, when in many use-cases it makes sense to see them as data plane abstractions (see Separating the IEEE 802.11 Networking Planes). This mistake had little consequence when both terminated in the same place, but has a severe and unnecessary negative impact on security once data and control planes are separated.

Lastly, the many optional parts of CAPWAP and ensuing diversity in implementation means that the protocol is less interoperable: it is relatively uncommon that access points from one vendor can be used with a controller from another vendor. Some recent work within the IETF attempts to address this issue by allowing the access point and controller to negotiate where IEEE 802.11 MAC functions such as fragmentation and encryption will reside [16]. This in our opinion drives protocol complexity without clear benefits. We also find it less than ideal that one of the most important security properties of a wireless network – which network elements have access to the user data plane encryption keys – should be decided by run-time negotiation.

FIGURE 2

Partitioning of functionality in CAPWAP and in our SDWN data plane tunnel protocol for IEEE 802.11 (ANY80211).

Functions		**CAPWAP**	**ANY80211**
Function	Distribution Service	WTP/AC	Service
	Integration Service	WTP/AC	Service
	Beacon Generation	WTP	Radio
	Probe Response Generation	WTP	Radio
	Power Mgmt/Packet Buffering	WTP	Radio
	Fragmentation/Defragmentation	WTP/AC	Service
	Assoc/Disassoc/Reassoc	AC	Service
IEEE 802.11 QoS	Classifying	WTP/AC	Service
	Scheduling	WTP/AC	Service
	Queuing	WTP	Radio
IEEE 802.11 RSN (WPA2	IEEE 802.1X/EAP	AC	Service
	RSNA Key Management	WPA/AC	Service
	IEEE 802.11 Encryption/Decryption	WTP/AC	Service

For the above reasons we have decided to build our SDWN implementation on a novel lightweight tunneling protocol for IEEE 802.11 that we call ANY80211. In contrast to CAPWAP it defines a single partitioning of the IEEE 802.11 MAC, making it possible to reason about its security properties (see Security Model). In many ways it can be thought of as the IEEE 802.11 equivalent of VXLAN [17] or Geneve [18].

As we outline in the next chapter (see Management Plane) we rely on existing management plane protocols like TR-069 [19] and NETCONF [20] to replace the provisioning functionality of CAPWAP. We believe that this approach minimizes functionality overlap and clarifies the role of network elements and interfaces.

Why not OpenFlow?

OpenFlow is an SDN control plane protocol standardized by the Open Networking Foundation [21]. Its focus is switching in physical and virtual IEEE 802.3 (Ethernet) networks.

OpenFlow itself does not provide the necessary mechanisms to control wireless networks, but numerous proposals for extending or augmenting the protocol with such capabilities have been made. For example OpenRoads attempts to control multiple wireless technologies with a combination of OpenFlow and SNMP [22]. Odin augments OpenFlow with a wireless-specific Odin protocol and IEEE 802.11 control plane abstractions [23]. CloudMAC takes a slightly different approach in that it embeds wireless control fields in a packet header, which can then be matched against and have actions performed on it by OpenFlow switches in the wired network [24].

We however believe that there is little to be gained from this conflation of wireless and wireline control planes. As we argue in the next chapter the concerns in wireless and wireline are quite different. Separating the control planes allows them each to evolve separately, focusing on the specifics of each domain.

But there are also other reasons to keep the control planes separate. One is that the physical network resources under control are often non-overlapping. Take for instance a service provider that uses SDN to control (at least) its edge network. If this operator were to leverage Wi-Fi equipped customer premises equipment (CPE) for carrier Wi-Fi services then that CPE would be outside the service provider edge. If this example is unconvincing consider an Over-The-Top (OTT) use-case where the domain under SDWN control may span the entire Internet.

One way to address this issue is to expand the edge to include the access point, e.g. using tunneling as in OpenRoads. This of course incurs an overhead, but may seem functionally equivalent to an overlay architecture such as SDWN. As we further explore in the Security Model discussion there are however significant differences between tunneling of plain text data on the one hand, and tunneling of raw encrypted radio frames on the other.

There is also another problem with extending the network edge onto customer premises: the service provider core is directly exposed to customers. This is of course less than ideal from a security and fraud prevention perspective. This seemingly difficult to solve problem is solved in the SDWN architecture by tunneling the raw radio traffic into the operator's network.

It should also be noted that tunneling overhead on the wired network is far less important in wireless than in wireline communications. In all realistic wireless use-cases the air interface is the bottleneck, at least in an economic sense: the opportunity cost associated with the tunneling overhead on the backhaul is negligible in comparison to further investment in radio access capacity. Importantly the SDWN overlay architecture has no tunneling overhead on the air interface.

This leads us to conclude that an overlay architecture on top of IP is the best choice for SDWN, and that wireline SDN can optionally be integrated on the *control program* application level.

Separating the IEEE 802.11 Networking Planes

The IEEE 802.11 protocol is more complex than IEEE 802.3, and separating the networking planes is therefore more challenging. In this section we outline the differences and apply the SDWN principles to IEEE 802.11.

To start with IEEE 802.11 is asymmetric, i.e. the client and the access point serve in two very different roles. Network name (SSID), security protocol (Open, WPA, RSN/WPA2 or FT), authentication method (PSK or EAP) and cipher suites (AES or TKIP) are all configurable on the infrastructure side.

Non-AP STAs discover this configuration through `Beacon` and `Probe Response` frames, and adapt accordingly.

IEEE 802.11 also supports strong mutual authentication and encryption. Pairwise encryption keys are derived individually for each client as part of the authentication process, along with a common group key used for broadcast communication.

As we have seen in the previous chapter this complexity leaves a lot more room for debate in how to separate the data, control and management planes. We here describe our proposal in more detail with reference to our concrete SDWN implementation for IEEE 802.11.

Management Plane

The role of the management plane is to configure network elements to a basic level; setting the boundaries within which the control plane can later adjust the forwarding data plane in real-time. The management plane also typically provides remote firmware upgrade functionality. For these reasons we believe that the managment plane should terminate with a trusted party, typically the owner of the network element under management.

SDWN requires only minimal changes to existing management plane interfaces. The reason is that the access point acting a *wireless termination point* often has a *primary function* that is already configurable through the management plane, e.g. TR-069 or a WEBUI. The *primary function* is thus remains responsible for the basic *radio* configuration, e.g. channel selection and supported rates.

The same is often true for the network element serving as the *service termination point*. Take for instance the case when the *service* terminates in a residental gateway. In this case there is usually a WEBUI through which the end-subscriber can configure Wi-Fi settings such as SSID, security protocol, cipher suite, and of course a WPA passphrase. These parameters are of course configurable through existing management plane interfaces, e.g. TR-069 or a WEBUI.

The management plane extensions needed for SDWN are thus very limited. For example, in our SDWN implementation for IEEE 802.11 parameters such as the IP address of the SDWN *controller*, the share of total capacity (spectrum as well as backhaul) that is always available to mobile devices (the so-called *floor*) and the maximum share of total bandwidth that can be allocated to mobile devices (the throttling *ceiling*) are under management plane control [25].

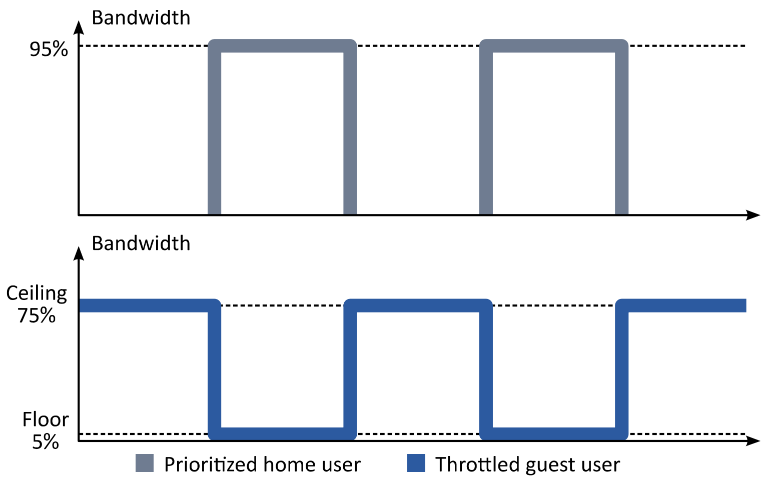


FIGURE 3
Throttling of guest users' traffic is built into our SDWN data plane, but the floor and ceiling parameters that set the boundaries within which the throttler operates are configured through the management plane (e.g. TR-069).

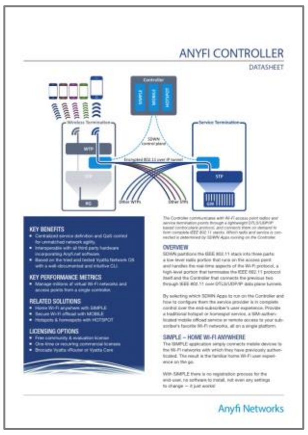
Control Plane

The role of the control plane is to compute the forwarding state and propagate that state in the network. In an overlay architecture that essentially means to “decide who should talk to whom”, and to introduce the parties. The core function of the control plane is thus to decide in real-time which *radios* should present which *services* to which *clients*.

In the first phase of the IEEE 802.11 association process the client device will scan for networks, sending out `Probe Request` frames. This is most often the first frame an access point will receive from a previously unknown *client*, i.e. a *client* for which it has no local forwarding state.

A classic access point with a local control plane will reply to this `Probe Request` frame with a locally generated `Probe Response`. This frame contains IEEE 802.11 `Information Elements` describing both the properties of the *radio* (e.g. which modulation rates it can support) and of the local Wi-Fi network (e.g. the SSID and the security protocol).

An SDWN access point in contrast will not generate a `Probe Response` frame locally. Instead it will send an ANYFI message to the *controller* containing the MAC address of the mobile device, essentially asking it to make the control plane decision of which `Probe Response` frames to transmit. The *controller* will reply with a set of *services* and their associated IEs, e.g. SSID, security protocol, authentication method and cipher suites. The access point combines these *service* `Information Elements` with


ANYFI CONTROLLER
DATASHEET
Anyfi Networks

ANYFI CONTROLLER

Our SDWN controller for IEEE 802.11 is based on the Vyatta Network OS and can be run as a virtual machine or on bare metal x86 hardware.

Related materials:

- Data Sheet
- Reference Guide
- Download

the set of `Information Elements` describing the local *radio*, e.g. its supported rates, and sends the resulting `Probe Response` frames to the waiting *client*.

In practice most access points in an SDWN network will operate in both these roles simultaneously; i.e. they will serve as classic access points for one or more local Wi-Fi networks, and simultaneously as an SDWN access point for an unlimited number of remote Wi-Fi networks (i.e. *services*).

When a mobile device has made its decision it will transmit an IEEE 802.11 `Authentication` frame indicating its choice. If the mobile device has selected a local Wi-Fi network then the selected access point will apply the normal MAC layer processing. If on the other hand the mobile device has selected a remote *service* then the access point will forward the `Authentication` frame through an IP tunnel to a *service termination point* associated with that *service*. If no such IP tunnel yet exists it will be set up dynamically, with the *controller* acting as an introducer for NAT traversal and load balancing purposes.

The control plane protocol also provides a feedback mechanism, keeping the *controller* informed of network state. Whenever a *client* connects to a Wi-Fi network, locally or remotely, the *service termination point* will generate and send a *preference* message to the *controller*, keeping it informed about *clients'* network preferences. *Radios* constantly monitor the quality of all radio links and the resource consumption of each *client*. This link level accounting information is aggregated by the *controller* and can be used to compute updated *radio policies*. In cases when the *radio* and *service* is operated by two different entities this link level accounting information can also be used to calculate roaming fees.

In summary we classify network and access point selection as SDWN control plane aspects. In the terms of IEEE Std 802.11 this means processing of Class 1 Frames [26], i.e. those frames that can be transmitted by a non-AP STA in State 1: `Beacon`, `Probe Request`, `Probe Response`, `Authentication` and `Deauthentication`.

As soon as an IEEE 802.11 non-AP STA transitions to State 2 [27] (by sending an Authentication frame to the AP) it has selected a network, and communication from then on is part of the data forwarding plane. The *controller*'s primary role at this stage is in assisting the *wireless termination point* and the *service termination point* in setting up an SDWN data plane tunnel between them.

Data Plane

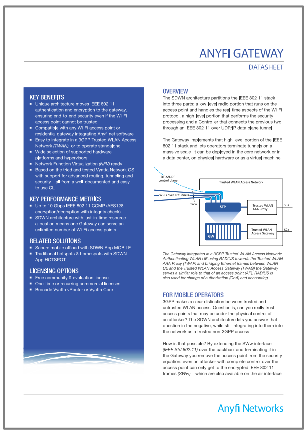
The role of the data plane is to “process packets with local forwarding state”. In our view essentially all frame processing after network selection falls into this category. In the terms of IEEE Std 802.11 this means all Class 2 and 3 Frame [28] processing: e.g. `Association Request/Response`, `Reassociation Request/Response`, `Data` and `Disassociation`.

But separation of data and control planes is only one key principle of SDWN. The other is the separation of radio access from service definition. Application of this principle demands a further partitioning of the Media Access Control (MAC) into a *radio* and a *service* related portion, with some practical constraints on what functionality can be implemented where.

As discussed we believe SDWN should be an overlay architecture on top of UDP/IP, and this means that data plane tunnels can span the globe, with latency ranging from a few up to hundreds of milliseconds. All real-time critical MAC functionality such as frame acknowledgement, power save buffering and similar must be implemented at the *radio*, in the *wireless termination point*. Also, to achieve a clear separation of concern all *service* related functionality must be implemented in the *service termination point*, including authentication and encryption.

In an IEEE 802.11 context this means that all Control frame processing, which is typically real-time critical, must be implemented in the *wireless termination point*. IEEE 802.11i security on the other hand must be implemented in the *service termination point*. But much of the partitioning is left up to the choice of the specific implementation. In our SDWN for IEEE 802.11 implementation we optimize for portability across as many Wi-Fi chipsets as possible, and ease of integration. That means adapting to standard interfaces and minimizing dependencies on the constantly evolving IEEE 802.11 PHY.

We have chosen to adopt the IEEE 802.11 MAC Protocol Data Unit (MPDU) as the SDWN data plane protocol; i.e. essentially raw (often encrypted) IEEE 802.11 frames are forwarded between the *wireless termination point* and the *service termination point*, encapsulated in UDP/IP datagrams. While the IEEE 802.11 PHY has evolved signficantly over time the basic MPDU format has remained largely unchanged, serving as a lowest common denominator for all PHYs (e.g. a, b, g, n and ac). This choice of tunneling protocol therefore acts as an effective isolation between the constantly evolving PHY and our SDWN implementation.


ANYFI GATEWAY
DATASHEET
Anyfi Networks

ANYFI GATEWAY

Our SDWN service termination gateway is based on the Vyatta Network OS and can be run as a virtual machine or on bare metal x86 hardware.

Related materials:

- Data Sheet
- Reference Guide
- Download

The other reason for choosing IEEE 802.11 MPDUs as the inner protocol of ANY80211 tunnels is ease of integration. Linux provides a framework for IEEE 802.11 drivers called `mac80211` [29]. This framework exposes a low-level API that allows a user space process to allocate BSSes ("extra SSIDs"), configure the template used by the driver to generate periodic Beacons and manage the set of associated STAs. This API also allows for sending and receiving of raw IEEE 802.11 frames in MPDU format on a so-called monitor interface. Our radio software is implemented as a user space daemon that uses these APIs to interface with the Wi-Fi chipset, and a regular UDP socket to communicate with the *controller* and *service termination points*.

However, not all Linux Wi-Fi drivers use the `mac80211` framework. The API of legacy drivers may need to be extended to support low-level control over the Wi-Fi chipset before integration of our radio software is possible. We have implemented such improvements in proprietary drivers from Broadcom, Qualcomm Atheros, Realtek as well as Mediatek, and make these driver improvements available to all OEMs free of charge.

Motivation

The skeptical reader may ask why this proposed partitioning of the IEEE 802.11 MAC is more correct than what can be more easily accomplished with OpenFlow or CAPWAP. The best short answer is that it allows for complete, as opposed to partial, virtualization of the wireless network: in an SDWN architecture literally millions of logical Wi-Fi networks can be distributed throughout a heterogeneous infrastructure, without sacrificing user experience or security.

“Make things as simple as possible, but not simpler!”

− *Albert Einstein*

This is ultimately the basis of our claim: our proposed separation of the IEEE 802.11 networking planes is the simplest possible that meets the complete network virtualization requirement.

Control Plane Abstractions for IEEE 802.11

It is not the mechanisms that make it SDN; it is the abstractions and the separation of concern they allow. And as we have seen above, in wireless networking there are different and more concerns than there are in wireline. In this chapter we take a closer look at those, through the lens of the familiar SDN abstraction layers [30].

Forwarding Abstractions

In order to control the forwarding data plane we need an abstraction for its function. As discussed the concerns are quite different in wireless and in wireline, and the abstractions too therefore need to be different.

Client

A *client* is an SDWN abstraction for a mobile device. We map it to the IEEE 802.11 concept of a non-AP STA.

A *client* is identified by its MAC address.

Radio

A *radio* is an SDWN abstraction for an IEEE 802.11 radio in an access point.

Note that a single access point may have multiple *radios*. We treat these as separate entities in the SDWN architecture, e.g. each *radio* is separately registered with the *controller*.

A *radio* has properties such as the channel it is operating on and the set of IEEE 802.11 PHYs it can support (e.g. a, b, g, n and ac).

A *radio* is identified by its hardware MAC address.

Service

A *service* is an SDWN abstraction for a logical network. We map it to the IEEE 802.11 concept of an Extended Service Set (ESS).

One important property of a *service* in our SDWN implementation is the IEEE 802.11 Extended Service Set Identifier (ESSID), also referred to as the SSID. Note that the SSID must be locally unique from the perspective of a *client*, but does not need to be globally unique from the perspective of the *controller*.

IEEE Std 802.11 places a number of requirements on an ESS, e.g. that all (virtual) access points providing access to the same ESS must bridge to the same IEEE 802.3 segment and use the same authentication mechanism. In the SDWN architecture these translate to requirements on the configuration of a *service termination points* (see below).

A *service* is identified by a Universally Unique Identifier (UUID) assigned automatically or by its operator.

Service Termination Point

A *service termination point* (STP) is an SDWN abstraction for the network element that terminates the wireless protocol. In our concrete implementation a *service termination point* acts as a gateway towards an IEEE 802.3 network, in much the same way as a classic Wi-Fi access point. The difference is that a single *service termination point* can be connected on demand to a large number of *radios*. We therefore map the *service termination point* abstraction to a set of IEEE 802.11 Basic Service Sets (BSSes).

As previously discussed the *service termination point* has essentially the same properties as an access point: SSID, security protocol (open, WPA or WPA2), authentication method (PSK or EAP) and cipher suites (TKIP or CCMP). Like with classic access points it is essential that all *service termination points* providing access to the same ESS be configured with the same SSID and authentication method. In the case where a number of *service termination points* provide access to the same SDWN *service* they should also be configured with the same *service* UUID.

Service termination points are grouped by their *service* UUID, and uniquely identified by their IP address and UDP port number.

Virtual Access Point

A *virtual access point* is an SDWN abstraction for a logical access point presented to a *client* by a *radio*.

Network engineers often refer to a *virtual access point* as an “extra SSID”. We try to avoid this nomenclature for many reasons but note its near equivalence. Our *virtual access point* concept differs from an “extra SSID” in that it is not statically and uniquely tied to a Basic Service Set; a single *radio* may have hundreds of *virtual access points* represented in its local forwarding state while transmitting only a single (generic) beacon stream. In layman terms this means that an SDWN access point can support an unlimited number of SSIDs, without incurring the timing and spectrum contention related issues of transmitting multiple beacon streams.

A *virtual access point* is identified in different ways in different parts of the forwarding data plane. In the access point *radio* it is primarily identified by its service UUID, whereas in the *service termination point* it is identified by its IEEE 802.11 BSSID. Note that these identifiers are unique only within their respective scopes.

Radio Policy

The SDWN architecture moves control plane decisions that affect network and access point selection to a central *controller*. At times it may however be beneficial to delegate low-level control decisions to individual *radios*. This reduces signaling overhead on the backhaul and improves scalability of the *controller*.

One way to do this is to let the *controller* compute a *radio policy*, which is then enforced at the *radio* in the access point. In our concrete implementation we support radio policies that set thresholds on quality parameters such as signal quality and attainable bandwidth (taking both spectrum and backhaul as well as prioritized users into account). The controller can also specify a time period during which these thresholds must be met, as well as the action to be performed if quality cannot be guaranteed (e.g. refuse association or force disassociation on the IEEE 802.11 MLME level).

A *radio policy* is uniquely identified by the pair of *client* and *virtual access point* to which it applies.

Network State Abstraction Layer

The forwarding data plane abstractions help us represent the state of the forwarding data plane, but they offer too much detail for the developer of the *control program*. Instead the *controller* provides an abstract view of the network, shielding the *control program* from some of the underlying complexity.

In our SDWN implementation for IEEE 802.11 the controller presents the *control program* with a reduced state consisting of *clients*, *radios* and *services*. The role of the *control program* then becomes essentially to decide which *client* should be presented with which *services* through which *radios*. To aid in that decision we introduce one additional abstraction on the network state level: *preference*.

Preference

A *preference* is an SDWN abstraction that represents a *client*'s preference for a particular *service*. The forwarding data plane provides the *controller* with hints that a certain *client* may prefer a certain virtual wireless network, and the *controller* presents these hints to the *control program* in the form of *preferences*.

In our SDWN implementation for IEEE 802.11 the service termination software can be configured to generate such hints when it detects that a *client* has connected to a local Wi-Fi network, e.g. in a residential gateway. The resulting *preference* will tell the *control program* e.g. that it may be meaningful to present the *client* with a virtual copy of this network when it shows up somewhere else in the network, within range of another *radio*.

The most important properties of a preference are its *client* and its *service*.

Specification Abstraction Layer

Specification abstractions give the control program the means to affect the outside world. As in many SDN architectures the SDWN *control program* output is a configuration for a virtual network with reachability constraints, a so-called *binding*. The task of superimposing such virtual networks and populating the data plane forwarding state accordingly is left to the *controller*.

Binding

The *binding* abstraction lets an SDWN *control program* tie a single client to a specific *service*, making that *service* available to the *client* through all *radios*.

Bindings are often created in response to a new *preference*. The logic behind this is quite simple: a *preference* indicates that a particular *client* may wish to connect to a particular *service* in the future, and the corresponding *binding* makes the same *service* available to the *client*. This is essentially all the control program logic necessary for a simple remote access SDWN application. But as we will see in in the next chapter there are other cases where a *control program* may wish to create a *binding*, in response to a new *preference*, the detection of a new *client* or even some entirely external event.

A *binding* can be thought of as a configuration for a simple virtual Wi-Fi network, consisting of a single *client*, a number of *radios* and a single *service*. In line with this thinking our SDWN implementation for IEEE 802.11 also allows the *control program* to specify a so-called *radio policy*, a configuration for the low-level radio management applied to this particular virtual Wi-Fi network. For example the *control program* can specify

1) a signal level threshold below which *radios* should not reply to IEEE 802.11 Probe Requests from *clients*;

2) an attainable bandwidth threshold below which a *client* should be prevented from associating and/or be forcefully disassociated on the IEEE 802.11 MLME level;

3) that the virtual Wi-Fi network should be suppressed within the coverage area of the corresponding local Wi-Fi network;

4) that the virtual Wi-Fi network should be suppressed within the coverage areas of other local Wi-Fi networks that the *client* has access to.

The *control program* retains a reference to the *binding* and can alter or delete it at any time.

How it all Fits Together

In this chapter we show how the different parts of the SDWN architecture come together in a few use-cases, exemplified with our own implementation for IEEE 802.11.

Seamless Community Wi-Fi

The SDWN architecture is ideally suited for remote access to familiar Wi-Fi networks, e.g. a fixed-line subscriber's home Wi-Fi network.

SIMPLE

Carrier Wi-Fi That Just Works

slide to sign on

Anyfi Networks

SDWN App: SIMPLE

SIMPLE is an SDWN App for seamless guest access and community Wi-Fi.

Related materials:

- Solution Brief
- Solution Presentation

When a mobile device is connected to a Wi-Fi network the first time the SSID and authentication credentials are typically stored by in the device's preferred network list. When the same SSID is presented to the device at a later time it will automatically connect and authenticate using those same credentials. Couple this device behavior with the functioning of a Software-Defined Wireless Network and the result is a completely seamless and secure community Wi-Fi user experience:

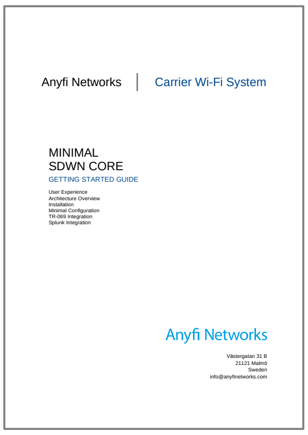
1. When a Wi-Fi *client* is connected to a Wi-Fi network for the first time the SDWN service termination software embedded in the home access point will generate and send to the *controller* a message indicating the *client*'s *preference* for the corresponding *service*.

2. The *preference* will be presented to the *control programs* running inside the *controller*, and they will be given the opportunity to generate a corresponding *binding*. A *control program* designed for seamless guest access will generate a *binding* matching the *preference*, i.e. tying the *client* to the *service* making this virtual Wi-Fi network available on the go.

3. When the same Wi-Fi device comes close to a visited access point the embedded SDWN radio software will detect and identify the *client* based on the MAC address in the IEEE 802.11 Probe Request frames, and will send a message to the *controller* requesting an update of the associated forwarding state.

4. The *controller* will generate a reply based on the *binding* created in step 2, instructing the visited access point to present a *virtual access point* with the SSID and other service-bound Information Elements (IEs) that the *client* has stored in its preferred network list and is now searching for. The radio software in the access point will start transmitting IEEE 802.11 Probe Response frames containing these IEs.

5. When the *client* receives a Probe Response containing the familiar SSID it will attempt to connect to the *virtual access point*. At this time the radio software in the visited access point and the service termination software in the home access point will set up a direct SDWN data plane tunnel between them, carrying the raw IEEE 802.11 frames. The *controller* acts as an introducer to assist in punching through NATs.


Anyfi Networks | Carrier Wi-Fi System

MINIMAL
SDWN CORE
GETTING STARTED GUIDE

User Experience
Architecture Overview
Installation
Minimal Configuration
TR-069 Integration
Splunk Integration

Anyfi Networks

Västergatan 31 B
21121 Malmö
Sweden
info@anyfinetworks.com

MINIMAL SDWN CORE

An example SDWN core capable of running our seamless guest access and community Wi-Fi solution (SIMPLE).

Related materials:

- Getting Started Guide
- Download

6. The device connects to and authenticates with the service termination software in the home access point, using the visited access point only as a "dumb" radio relay.

Note that the IEEE 802.11i security mechanism protects the connection end-to-end, all the way from the *client* to the *service termination point*. Even an attacker that is in control of the visited access point cannot eavesdrop on or modify the communication.

It should also be noted that an attacker in complete command of the control plane cannot do much damage. Authentication and key derivation is between the *client* and the *service termination point* (the home access point in this case). The *radio* only acts a "dumb" radio relay and the *controller* is not involved at all. Neither has access to derived encryption keys. The owner of the visited access point is also protected against such an attack since there are no SDWN control plane instructions that can provide access to local networks or in other ways compromise system security.

A skeptical reader may perhaps ask what happens if the *controller* connects the *radio* to a malicious *service termination point*. As long as the home Wi-Fi network is protected with an IEEE 802.11 security mechanism that provides mutual authentication this "evil twin" *service termination point* will be detected and avoided by the *client* on the radio link layer. The popular pre-shared key (PSK) authentication mechanism is sufficient for this purpose.

Our *controller* comes with a built-in *control program* for the seamless community Wi-Fi use-case: the SIMPLE SDWN App.

Traditional Hotspots and Homespots

The SDWN architecture also lends itself well to the traditional homespot and hotspot use-case and addresses many of the shortcomings of existing architectures.

Firstly, the same security principles as discussed above of course hold in this use-case as well: *client* and *service* are mutually authenticated and their communication is protected end-to-end by the IEEE 802.11 encryption (as opposed to piecewise with IEEE 802.11 encryption over the air and e.g. IPSec over the backhaul). This ensures user data plane integrity and confidentiality, even against an attacker that is in control of the access point.

Another typical problem in this use-case is ensuring a high quality user experience. Wi-Fi is a short-range radio technology and network coverage is typically spotty. Simple geometry dictates that a large portion of the coverage

HOTSPOT

Traditional Hotspots and Homespots

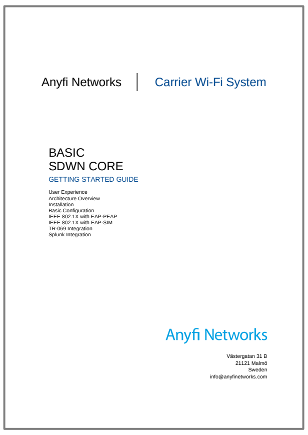
Anyfi Networks

SDWN App: HOTSPOT

HOTSPOT is an SDWN App for centrally terminated carrier Wi-Fi networks of the more traditional kind.

Related materials:

- Solution Brief
- Solution Presentation


Anyfi Networks | Carrier Wi-Fi System

BASIC
SDWN CORE
GETTING STARTED GUIDE
User Experience
Architecture Overview
Installation
Basic Configuration
IEEE 802.1X with EAP-PEAP
IEEE 802.1X with EAP-SIM
TR-069 Integration
Splunk Integration

Anyfi Networks

Västergatan 31 B
21121 Malmö
Sweden
info@anyfinetworks.com

BASIC SDWN CORE

An example SDWN core capable of running both our seamless community Wi-Fi solution SIMPLE and our traditional hotspot and homespot solution HOTSPOT.

Related materials:

- Getting Started Guide
- Download

area in such networks will be fringe coverage. Combine this with the typical behavior of mobile devices - they tend to connect to Wi-Fi as soon as a signal is detectable - and it is clear that the average session quality will suffer. The radio link level monitoring and control built into the SDWN architecture addresses this problem and greatly improves the average session quality of experience, by simply avoiding that devices connect too early.

The SDWN architecture also addresses some of the Distribution System (DS) challenges encountered in very large-scale carrier Wi-Fi deployments: per-device mobility anchoring and client isolation. These challenges stem from the fact that Wi-Fi was originally designed as a Wireless Local Area Network (WLAN) technology, where complete connectivity between all devices is a requirement. But this is usually not the desired behavior in a carrier Wi-Fi network where devices should instead typically be isolated on Layer 2, while remaining connected to the same Layer 2 segment as they roam from access point to access point. This can only be achieved with centralized Layer 2 aggregation and packet processing (Layer 2 switching and IP gateway functionality).

Traditionally each access point in such networks is assigned a tunnel aggregation point in the packet-processing core. But this can cause performance and scalability problems when devices roam from access point to access point, as their traffic will be tunneled to new entry points and will have to be switched within the core to reach the device's mobility anchor point.

With the flexibility of the SDWN architecture we can address this challenge in a different way, by slicing the problem along another axis. Each *client* will have its traffic tunneled to a *service termination point* in the Wi-Fi core, and as the *client* roams from access point to access point the *controller* will ensure that it remains connected to the same *service termination point*. This ensures that traffic from a particular *client* always enters the core in the same place, as close as possible to its mobility anchor point. It also makes it possible to implement packet processing in a distributed fashion with several completely separate Wi-Fi cores, each with its own DHCP and IP gateway functionality.

Also note how favorably this mobility anchoring interacts with encryption key caching on the IEEE 802.11 level. In the SDWN architecture encryption keys are derived between the *client* and the *service termination point*. Since the client remains connected to the same *service termination point* as it roams from access point to access point encryption keys can be cached and reused instead of being rederived as a part of IEEE 802.1X reauthentication. This property holds even when the visited access points are from different vendors, in fact even if they are owned and operated by different radio access providers.

We are now ready to outline the functioning of a system implementing the traditional hotspot and homespot user experience:

1. The operator implements a number of separate Wi-Fi cores, each capable of providing centralized packet processing for a number of *clients*. These separate cores will all process Wi-Fi traffic from the same SSID, but from an SDWN perspective they are still separate *services* (since they are separated on Layer 2 and thus cannot formally constitute a single IEEE 802.11 ESS) identified by separate UUIDs. Each Wi-Fi core will have a number of SDWN *service termination points*, configured with the same SSID and the same *service* UUID.

2. All *service termination points* will register with the *controller* as part of the bootstrap process. The *controller* will group *service termination points* by *service* UUID, and present these *services* to the *control program*. The *control program* is responsible for creating *bindings* that distribute *clients* across *services* (i.e. Wi-Fi cores). Load balancing and fail-over across *service termination points* within the same *service* is handled automatically by the *controller*.

3. When a new *client* is detected anywhere in the network the *controller* will inform the *control program*, which in turn creates a *binding* tying the *client* to one of the Wi-Fi cores. This can be done in a random fashion, or guided by the processing load indication periodically reported by *service termination points*.

4. From then on, whenever the *client* comes close to a new access point the embedded SDWN radio software will detect and identify the *client* based on the MAC address in the IEEE 802.11 Probe Request frames, and will send a message to the *controller* requesting an update of the associated forwarding state.

5. The *controller* will generate a reply based on the *binding* created in step 3, instructing the visited access point to present a *virtual access point* with the SSID of the Wi-Fi core.

6. When the *client* attempts to connect to the *virtual access point* the radio software in the visited access point and the *service termination point* in the Wi-Fi core will set up a direct SDWN data

plane tunnel between them, carrying the raw IEEE 802.11 frames. The *controller* acts as an introducer and ensures that each *client* remains connected to the same *service termination point*.

7. The device connects to and authenticates with the *service termination point*, using the visited access point only as a "dumb" radio relay.

A more advanced *control program* may also tie a *radio policy* to the *bindings* it creates, setting radio link quality thresholds ensuring a high quality of experience.

Our *controller* comes with a built-in *control program* for the hotspot and homespot use-case: the HOTSPOT SDWN App.

Mobile Wi-Fi Offload with QoS integration

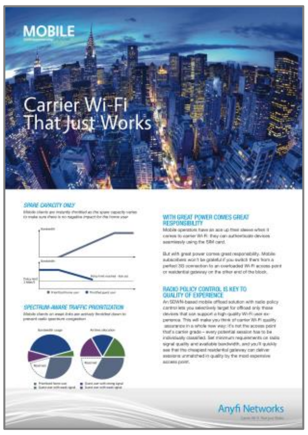
Many mobile operators are today looking for ways to offload mobile data onto fixed-line infrastructure. Residential gateways with remote firmware update capability are a prime target, with some operators reporting that they can offload up to 40% of all mobile data traffic from their 3GPP network at negligible marginal cost [31].

The homespot solution above can of course be used for secure mobile Wi-Fi offload with end-to-end security, but the SDWN architecture allows us to go one step beyond this data plane integration and also integrate the control plane towards the 3GPP network. This is accomplished by computing a dynamic *radio policy* for each mobile device based on the quality of service that the mobile network can deliver to that device, enabling intelligent traffic steering and an "always best connected" user experience.

The mobile network interfaces necessary for this control plane integration are currently in standardization within 3GPP. When such interfaces are generally available we will include a built-in *control program* for dynamic *radio policy* control based on 3GPP network state in our *controller*.

Security Model

Ensuring security in wireless networks is a challenge because there is no way to physically prevent eavesdropping, restrict access, or verify that your users connect to the right network. Instead we have to rely on cryptographic

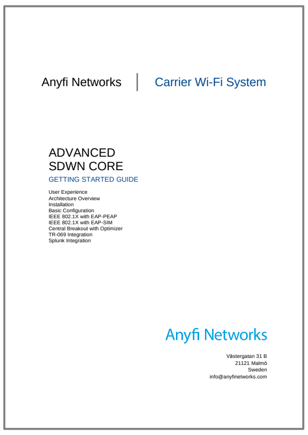


SDWN App: MOBILE

MOBILE is an SDWN App for secure mobile Wi-Fi offload with QoS integration towards the 3GPP network.

Related materials:

- Solution Brief
- Solution Presentation



Anyfi Networks | Carrier Wi-Fi System

ADVANCED
SDWN CORE
GETTING STARTED GUIDE

User Experience
Architecture Overview
Installation
Basic Configuration
IEEE 802.1X with EAP-PEAP
IEEE 802.1X with EAP-SIM
Central Breakout with Optimizer
TR-069 Integration
Splunk Integration

Anyfi Networks

Västergatan 31 B
21121 Malmö
Sweden
info@anyfinetworks.com

ADVANCED SDWN CORE

An example SDWN core capable of running both our seamless community Wi-Fi solution SIMPLE and HOTSPOT with EAP-PEAP and EAP-SIM authentication.

Related materials:

- Getting Started Guide
- Download

techniques to accomplish the same goals. But as anybody who has worked with cryptography will tell you the devil is in the details: mutual authentication is meaningless if an attacker has access to the network's authentication interface and encryption is only meaningful if the remote end-point is in a trusted location.

Problems with Previous Architectures

The most common approach to leveraging existing Wi-Fi assets for mobile data services is configuration of a so-called "extra SSID". The associated traffic is then bridged to a separate logical WAN, e.g. an ATM virtual circuit, a DOCSIS service flow or L2oGRE tunnel, and centrally aggregated. This approach is reasonable from a security perspective for open Wi-Fi with captive portal authentication, but can be less than ideal when combined with IEEE 802.1X authentication.

The perhaps most obvious problem is that guest users' clear-text communications are visible to the owners of visited access points. Note that this problem persists even if the backhaul is separately encrypted e.g. with IPSec or similar: clear-text data is still available inside the access point itself and there is little stopping a knowledgeable and determined attacker from gaining access to it [32].

But there is also a more serious problem with the local IEEE 802.11 termination approach: exposure of the authentication interface. In order to perform e.g. EAP-SIM authentication in residential gateways these must have access to a RADIUS interface towards the HLR/AuC in the mobile core. An attacker in physical control of a visited access point can thus operate an "evil twin" indistinguishable from a legitimate access point. Note that this form of attack is not geographically constrained or limited to certain devices; all mobile subscribers everywhere are put at risk of having their devices seamlessly switched over to an "evil twin" access point, operated by an attacker with (indirect) access to the mobile operator's HLR/AuC.

A skeptical reader may object that a centralized architecture like CAPWAP protects the authentication interface and eliminates this problem, but unfortunately this is not the case. Even if IEEE 802.1X authentication is moved into the network so that the RADIUS interface can be effectively protected the weakness still remains for as long as an attacker can gain access to the encryption keys protecting the user data plane. This is obvious if you think about it: an attacker may just as well tunnel EAPOL frames from his "evil twin" access point, through a compromised residential gateway and into the mobile core across a CAPWAP interface. The important question is if an attacker can gain access to the encryption keys protecting the user data plane or not. If said keys are received in the `MS-MPPE-Recv-Key` and `MS-MPPE-Send-Key` attributes of a RADIUS Access-Accept message, or in the CAPWAP `IEEE 802.11 Station Session Key` message element [33] is of less importance.

End-to-End IEEE 802.11i Security

An SDWN architecture in contrast will encrypt the user data plane end-to-end, all the way from the mobile device to the *service termination point*. In our implementation the encryption is the standard IEEE 802.11i CCMP or TKIP. The encryption keys are derived in the mobile device and the *service termination point*, and are as a rule only available in these two locations. The visited access point acts only as a "dumb" radio relay for encrypted IEEE 802.11 MAC Protocol Data Units (MPDUs), and is thus completely removed from the security equation.

This architecture provides strong and verifiable guarantees for user plane integrity and confidentiality: even an attacker in complete control of the access point cannot eavesdrop on or modify the communication. In fact, even if that attacker also has complete command over the SDWN control plane this principle still holds.

Special Considerations

The security of our SDWN implementation for IEEE 802.11 rests firmly on the tried and tested IEEE 802.11i security mechanisms. There are however some aspects that need careful consideration when employing these mechanisms in a different context.

The Internet is much larger than the coverage area of your average Wi-Fi network. Since our IEEE 802.11 stack in the *service termination point* is exposed to frames coming in over the network a security defect in this software is much more likely to be exploited and the potential consequences much more severe. We have therefore implemented some security measures intended to harden against such attacks.

Firstly, our service termination software does not provide remote access to networks protected with WEP.

We have also implemented protection against brute force attacks. A mobile device will not be allowed to authenticate until an ANYFI introduction message has been received from the *controller*. This prevents parallel "port scanning" type attacks. The service termination software will also only allow the mobile device a certain number of authentication attempts before disconnecting, and will report authentication failures to the *controller*. The *controller* can thus detect distributed brute force attacks and take protective action.

Lastly, Wi-Fi frames in transit through an SDWN data plane tunnel can, unlike radio transmissions, easily be intercepted and prevented from reaching their intended recipient. This may make man-in-the-middle (MITM) vulnerabilities in the IEEE 802.11i security mechanisms easier to exploit. There are however few known such vulnerabilities. The only one we are aware of at this time is Toshihiro Ohigashi and Masakatu Morii [34] and we do not consider that a significant threat in practice.

About Anyfi Networks
Anyfi Networks is the company behind the revolutionary Software Defined Wireless Networking (SDWN) architecture. Based on this unique technology we offer broadband operators, fixed as well as mobile, a range of carrier Wi-Fi software solutions: from traditional hotspots and homespots all the way to massively scalable secure mobile Wi-Fi offload. For more information please visit www.anyfinetworks.com or contact sales@anyfinetworks.com.
Copyright © 2015 Anyfi Networks AB

References

1 Anyfi.net - Free SDWN Data Plane for IEEE 802.11, http://anyfi.net.

2 Broadcom xDSL CPE solutions, http://www.broadcom.com/products/Broadband-Carrier-Access/xDSL-CPE-Solutions/.

3 Mediatek Connectivity xDSL products, http://www.mediatek.com/en/products/connectivity/xdsl/.

4 Realtek ADSL Router SoCs, http://www.realtek.com/products/productsView.aspx?Langid=1&PNid=8&PFid=9&Level=4&Conn=3.

5 Qualcomm Atheros SoCs, http://www.qca.qualcomm.com/resources/product-collateral/?product_collateral_category=wlan.

6 Inteno iopsys based CPEs, http://temp1.intenogroup.com/Products/CategoryDetail/tabid/112/categoryid/2/Default.aspx.

7 Anyfi Networks' Carrier Wi-Fi System, Community Edition, http://www.anyfinetworks.com/download.

8 Shannon's law of channel capacity, http://en.wikipedia.org/wiki/Channel_capacity.

9 Femto Forum Femtocell Business Case Whitepaper, April 2009, http://www.3gamericas.org/documents/Femto Forum Business Case Whitepaper Signals Research Apr09.pdf.

10 Radio over fiber, http://en.wikipedia.org/wiki/Radio_over_fiber.

11 Cisco Visual Networking Index (VNI), 2014, http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/.

12 Martin Casado, Teemu Koponen, Scott Shenker, Amin Tootoonchian. Fabric: A Retrospective on Evolving SDN, http://www.eecs.berkeley.edu/~sylvia/cs268-2013/papers/fabric.pdf.

13 Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification, Proposed Standard, IETF Proposed Standard, http://tools.ietf.org/html/rfc5415.

About Anyfi Networks

Anyfi Networks is the company behind the revolutionary Software Defined Wireless Networking (SDWN) architecture. Based on this unique technology we offer broadband operators, fixed as well as mobile, a range of carrier Wi-Fi software solutions: from traditional hotspots and homespots all the way to massively scalable secure mobile Wi-Fi offload. For more information please visit www.anyfinetworks.com or contact sales@anyfinetworks.com.

Copyright © 2015 Anyfi Networks AB

14 Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Binding for IEEE 802.11, IETF Proposed Standard, http://tools.ietf.org/html/rfc5416.

15 Alternate Tunnel Encapsulation for Data Frames in CAPWAP, IETF Internet Draft, http://tools.ietf.org/html/draft-zhang-opsawg-capwap-cds-03.

16 IEEE 802.11 MAC Profile for CAPWAP, IETF Internet Draft, http://tools.ietf.org/html/draft-ietf-opsawg-capwap-hybridmac-04.

17 VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks, IETF Internet Draft, http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-09.

18 Geneve: Generic Network Virtualization Encapsulation, IETF Internet Draft, http://tools.ietf.org/html/draft-gross-geneve-00.

19 TR-069, Broadband Forum Technical Report, http://www.broadband-forum.org/cwmp.php.

20 Network Configuration Protocol (NETCONF), IETF Proposed Standard, http://tools.ietf.org/html/rfc6241.

21 OpenFlow Switch Speficiation, Open Networking Foundation Specification, https://www.opennetworking.org/sdn-resources/onf-specifications/openflow.

22 Kok-Kiong Yap, Masayoshi Kobayashi, David Underhill, Srinivasan Seetharaman, Peyman Kazemian, Nick McKeown. The Stanford OpenRoads Deployment, http://yuba.stanford.edu/~peyman/docs/openroads.pdf.

23 Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, Teresa Vazao. Towards Programmable Enterprise WLANs with Odin, http://conferences.sigcomm.org/sigcomm/2012/paper/hotsdn/p115.pdf.

24 Jonathan Vestin, Peter Dely, Andreas Kassler, Nico Bayer, Hans Einsiedler, Christoph Peylo. CloudMAC - Towards Software Defined WLANs, http://teampal.mc2lab.com/attachments/682/p393-vestin.pdf.

25 Anyfi.net TR-069 vendor extensions for the TR-98 and TR-181 Issue 2 data models, https://anyfi.net/documentation#tr69.

26 IEEE Std 802.11-2012, Section 10.3.3 Frame filtering based on STA state, http://standards.ieee.org/getieee802/download/802.11-2012.pdf.

27 IEEE Std 802.11-2012, Section 10.3.2 State transition diagram for nonmesh STAs, http://standards.ieee.org/getieee802/download/802.11-2012.pdf.

28 IEEE Std 802.11-2012, Section 10.3.3 Frame filtering based on STA state, http://standards.ieee.org/getieee802/download/802.11-2012.pdf.

About Anyfi Networks
Anyfi Networks is the company behind the revolutionary Software Defined Wireless Networking (SDWN) architecture. Based on this unique technology we offer broadband operators, fixed as well as mobile, a range of carrier Wi-Fi software solutions: from traditional hotspots and homespots all the way to massively scalable secure mobile Wi-Fi offload. For more information please visit www.anyfinetworks.com or contact sales@anyfinetworks.com.
Copyright © 2015 Anyfi Networks AB

29 Linux Wireless mac80211 - a framework for SoftMAC wireless device drivers, http://wireless.kernel.org/en/developers/Documentation/mac80211.

30 Scott Shenker, Martin Cassado, Teemu Koponen, Nick Keown. The Future of Networking and the Past of Protocols, http://www.slideshare.net/martin_casado/sdn-abstractions.

31 Edited Transcript, Q3 2013 Telenet Group Holding NV Earnings Conference Call, http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9MjA3ODc0fENoaWxkSUQ9LTF8VHlwZT0z&t=1.

32 Insecure Vodafone femtocells allow eavesdropping and call fraud, http://arstechnica.com/security/2011/07/insecure-vodafone-femtocells-allow-eavesdropping-call-fraud/.

33 Control and Provisioning of Wireless Access Points (CAPWAP) Protocol Binding for IEEE 802.11, Section 6.5 IEEE 802.11 Station Session Key, http://tools.ietf.org/html/rfc5416#section-6.15.

34 Toshihiro Ohigashi, Masakatu Morii. A Practical Message Falcification Attack on WPA, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.158.1372.

About Anyfi Networks
Anyfi Networks is the company behind the revolutionary Software Defined Wireless Networking (SDWN) architecture. Based on this unique technology we offer broadband operators, fixed as well as mobile, a range of carrier Wi-Fi software solutions: from traditional hotspots and homespots all the way to massively scalable secure mobile Wi-Fi offload. For more information please visit www.anyfinetworks.com or contact sales@anyfinetworks.com.
Copyright © 2015 Anyfi Networks AB