Programmatic Orchestration of WiFi Networks

Julius Schulz-Zander[†] , Lalith Suresh[†] , Nadi Sarrar[†] , Anja Feldmann[†] , Thomas Hühn[‡†] , and Ruben Merz[1]

TU Berlin, Germany[†] DAI-Labor, Germany[‡] Swisscom, Switzerland[1]

Abstract

With wireless technologies becoming prevalent at the last hop, today's network operators need to manage WiFi access networks in unison with their wired counterparts. However, the non-uniformity of feature sets in existing solutions and the lack of programmability makes this a challenging task. This paper proposes Odin, an SDN-based solution to bridge this gap. With Odin, we make the following contributions: *(i)* Light Virtual Access Points (LVAPs), a novel programming abstraction for addressing the IEEE 802.11 protocol stack complexity, *(ii)* a design and implementation for a software-defined WiFi network architecture based on LVAPs, and *(iii)* a prototype implementation on top of commodity access point hardware without modifications to the IEEE 802.11 client, making it practical for today's deployments. To highlight the effectiveness of the approach we demonstrate six WiFi network services on top of Odin including load-balancing, mobility management, jammer detection, automatic channel-selection, energy management, and guest policy enforcement. To further foster the development of our framework, the Odin prototype is made publicly available.

1 Introduction

Today's access networks are increasingly dominated by wireless technology at the last hop. Indeed, the WiFi Alliance, the certification authority for WiFi devices, reports almost 1.1 billion WiFi devices were shipped in 2011 [1], and predicts that this number will double by 2015. However, supporting this ever increasing number of wireless capable devices across residential, public, and enterprise networks is non-trivial and raises new challenges for network management, in particular for integrating wired, cellular, and wireless network management. To highlight this need, we point to the fact that large operators such as Deutsche Telekom (DT) [3] and Swisscom [6] are offloading data from their cellular networks to WiFi networks to reduce the stress on the former. Indeed, DT aims to deploy 2.5 million WiFi hotspots by 2016. Thus, these operators face the challenge of managing their different networks in unison and all the way to the users' premises.

Furthermore, modern enterprise WiFi networks typically consist of few dozens to thousands of Access Points (APs) serving a multitude of client devices including smart-phones, laptops, and tablets. For performance and scalability reasons, these networks require services which include mobility management, load-balancing, interference management, and channel reconfigurations. These services have to be realized as applications on top of the basic management functionality of the individual access points. However, different devices from different vendors typically offer different interfaces and do not offer native support for the needed applications. Additionally, today's enterprises and provider networks are Bring-Your-Own-Device (BYOD) networks, implying that the network has to accommodate an even more diverse set of user device types of different generations.

To manage this growing complexity, network operators need novel abstractions as well as new tools to uniformly manage the wired and wireless parts of their network, e.g., to verify network configurations, perform troubleshooting, or systematic debugging. In wired networks, recent advances in Software-Defined Networking (SDN) have enabled such features through programmatic control of networks. In an SDN, the control plane and data plane are decoupled, allowing network intelligence and state to be logically centralized. Using this centrally-available global view of the network, SDN allows operators to perform principled control and management of networks through the use of abstractions [26]. The best known SDN interface is OpenFlow, which specifies a protocol for a logically centralized controller to remotely manage forwarding tables within switches.

However, OpenFlow does not address the complexities of WiFi protocols and WiFi networks which include interference mitigation, mobility management, and channel selection techniques. This is unfortunate, because point-solutions exist for these WiFi-specific network problems but are often provided only by enterprise vendors through vertically integrated solutions. However, most cheap, off-the-shelf commodity hardware as deployed in today's access networks is outside the purview of such enterprise solutions.

Yet, proposals exist for extensible and programmable WiFi networks [20, 39]. However, these depend on client-side modifications which we argue is impractical to deploy. This is an obstacle not only for provider networks, but also for enterprise deployments given the trend towards BYOD.

We, in this paper, present Odin, an SDN-based solution which presents a programming abstraction which can provide the features enterprise and provider networks need. It bridges the gap between the range of features required by network operators and the lack of programmability in today's WiFi networks. In the process of designing Odin, we address the following research questions:

1. What programming abstractions are needed to address the complexities of the IEEE 802.11 protocol stack?
2. How can these abstractions be fit into an SDN architecture?
3. Can the SDN architecture already be realized on top of today's commodity access point hardware and without client modifications?

We find that the above questions can be answered affirmatively through the following contributions:

- The proposed Light Virtual Access Point (LVAP) abstraction captures the complexities of the IEEE 802.11 protocol stack.
- We present a prototype implementation of the LVAP approach which we have made publicly available[1].
- We evaluate the framework by presenting six typical WiFi network applications.

Odin is extensible in accordance with the features required in today's WiFi networks, whilst being deployable on top of low-cost commodity access point hardware. While we introduced the basic concept of LVAPs in our HotSDN workshop paper [30] and showed the system's capabilities in multiple demos [17, 24], this paper includes the detailed architecture for software-defined WiFi networks, a prototype implementation, as well as a system evaluation using multiple WiFi applications.

2 Use cases

Odin has been designed for the following use cases:

[1]Odin source: `http://sdn.inet.tu-berlin.de`.

Traffic Offloading and Client Mobility: Offloading user's devices to WiFi allows operators to reduce stress on their cellular infrastructure. To this end, it is beneficial to provide users with consistent authentication credentials across their home networks, hotspots, and cellular connections, whilst managing client mobility. This will prevent the user from having to maintain multiple authentication credentials, whilst allowing operators to offload a user's traffic onto a hotspot when available. This is similar to what is proposed by the Hotspot 2.0 initiative, which however requires clients to support IEEE 802.11u. Furthermore, mobility management is an important feature within enterprise WiFi deployments, typically offered by today's vendors [5] and also explored by the research community [14, 18, 19, 20].

Network Performance Management: Channel selection, load balancing and wireless troubleshooting are crucial for the performance of WiFi networks, particularly within dense deployments like large enterprises or residential networks. Channel selection [7, 15, 35] involves continuously monitoring and then reacting to changes in the wireless environment. Load-balancing [9, 21] typically requires control of clients' attachment points to the network or the ability to hand off clients between WiFi access points. Lastly, there is a need for the ability to measure, detect, and localize interferers. This is because interference caused by non-WiFi devices can severely impact the achievable throughput of WiFi devices within the same vicinity [23], since both kinds devices share the same wireless spectrum.

3 The Odin System

In this section, we describe the components of Odin and the Light Virtual Access Point (LVAP) abstraction.

3.1 Odin System Components

Figure 1 illustrates the components of the proposed design and their interactions. In line with the SDN concept, the design decouples the control from the data plane. This is done by having a logically centralized controller that leverages OpenFlow for the wired network, and a separate control plane protocol for the wireless part (elaborated upon in § 8). We chose to have separate protocols for programming the wired and wireless parts. This is because in its current state, OpenFlow does not extend well into the realm of the IEEE 802.11 MAC, as its scope is restricted to programming flow table rules on Ethernet- based switches. For instance, it cannot perform matching on wireless frames, cannot accommodate measurements of the wireless medium, report per-frame receiver side statistics, or be used for setting per-frame or -flow transmission settings for the WiFi datapath. We now describe the individual components in Odin:

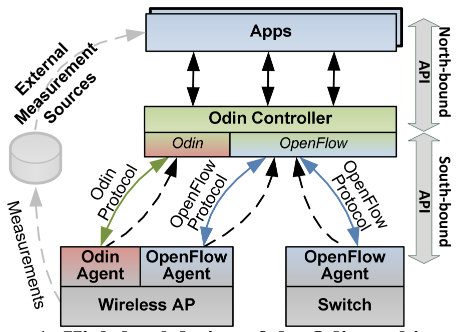


Figure 1: High-level design of the Odin architecture.

Odin controller: The controller enables network applications to programmatically orchestrate the underlying physical network. It exposes a set of interfaces to the applications (the northbound API) and then translates these calls into a set of commands on the network devices (the southbound API). The controller also maintains a view of the network including clients, APs, and OpenFlow switches, which the Odin applications can then control.

Odin agents: Agents run on the wireless APs and expose the necessary hooks for the controller (and thus applications) to orchestrate the WiFi network and report measurements. Time critical aspects of the WiFi MAC protocol (such as IEEE 802.11 acknowledgments) continue to be performed by the WiFi NIC's hardware. On the other hand, non time-critical functionality including management of client associations is implemented in software on the controller and the agents. This realizes a distributed WiFi split-MAC architecture. In addition, they perform matching on incoming frames to support a publish-subscribe system wherein network applications can subscribe to per-frame events.

Applications: For wireless network applications to take effective control decisions, they need access to statistics not only at a per- frame granularity, but also measurements of the medium itself (for instance, to infer interference from non-WiFi devices operating in the same spectrum). Thus, applications in Odin work either reactively or proactively by accessing measurements from multiple layers. This includes *(i)* measurements collected by the agents, *(ii)* OpenFlow statistics and *(iii)* measurements collected by external tools (e.g. `snmpd`). Odin applications can program the network through the northbound API offered by the controller.

3.2 Light Virtual Access Points

The Light Virtual Access Point (LVAP) is the abstraction in our system that allows us to address the specific requirements of WiFi networks, whilst allowing for unified management of the wired and wireless portions of the network. The LVAP is a per-client AP which simplifies the handling of client associations, authentication, handovers, and unified slicing of both the wired and wireless portions of the network. It enables a port-per-source view of WiFi networks akin to that of wired networks. In doing so, it remains orthogonal, but complementary, to trends in physical layer virtualization and RF spectrum slicing [29]. LVAPs are hosted on the agent, and their assignment to agents is controlled by the controller.

3.2.1 LVAPs as per-client APs

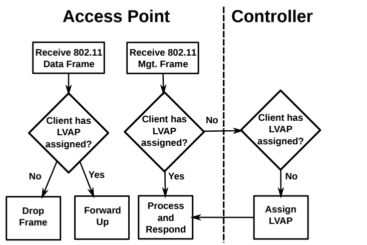
In regular IEEE 802.11 networks, clients need to associate with a physical AP before sending data frames. The association process begins with the discovery phase, where a client either actively scans for APs by generating probe requests, or passively learns about APs through beacon frames generated by the latter. During an active scan, APs that respond with probe response messages become candidates for the client to associate with. The client then decides which AP to associate with via a locally made choice. At this point, the association is defined between the client's MAC address and the BSSID of the AP. The BSSID of an AP is a MAC address of the AP's wireless interface and is different from the SSID, which is a network name.

This design of the WiFi protocol is inconvenient; there is no mechanism for centralized control over the client's association because the client makes the association decision entirely on its own. Furthermore, the infrastructure cannot instruct the client to re-associate without introducing additional signaling techniques such as [20].

The approach of LVAPs overcomes these difficulties without introducing additional signaling mechanisms between clients and the infrastructure, and thus conforms to our objective of not introducing client-side modifications. With LVAPs, every client receives a unique BSSID to connect to, essentially making them client-specific APs. Figure 2 indicates the decision flow in handling a client's association using LVAPs.

When a client probe scans, a new LVAP is spawned within the Odin agent on the physical AP. This LVAP then responds to the client with a probe response as instructed by the controller, following which, the clients completes the association handshake with its LVAP. As a result, a physical AP hosts a unique LVAP for each connected client. Every LVAP periodically unicasts beacon frames to its corresponding client. This ensures that a client never processes a beacon frame from another client's LVAP. The overhead of per-client beacon generation can be reduced by increasing the beacon interval, by setting the `NO_ACK` bit on the beacon frame, and also leveraging higher data-rates because of the unicast transmission. Note, beacons are typically broadcasted but are identical to probe response frames which are unicasted. Unicasting beacons does not confuse client devices (*cf.* 6.4).

As long as the client receives ACKs for the data frames it generates and receives beacons from the AP it is asso-


Access Point
Controller
Receive 802.11 Data Frame
Receive 802.11 Mgt. Frame
Client has LVAP assigned?
Client has LVAP assigned?
Client has LVAP assigned?
No
Yes
No
Yes
No
Drop Frame
Forward Up
Process and Respond
Assign LVAP
Figure 2: Processing path for WiFi frames: Agents invoke a controller for handling management frames.

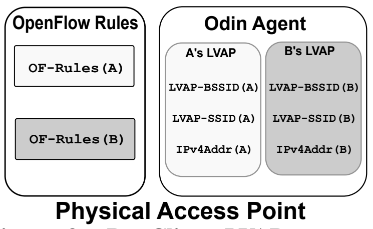


Figure 3: Per-Client LVAP state: a unique BSSID, set of SSIDs, client's IPv4 address, and OpenFlow rules (∼80 bytes of state excluding OF rules).

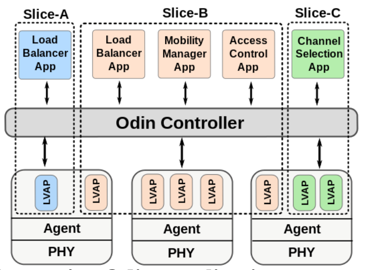


Figure 4: Odin applications operate upon a view of LVAPs and physical APs in their respective slices.

ciated to (in this case, an LVAP), the client stays associated. If the state corresponding to the client's LVAP is migrated to and instantiated at another Odin agent fast enough, the client does not attempt to re-scan (since from the client's point of view, its AP is still available). Thus, by migrating a client's LVAP between physical APs, the infrastructure can now control the client's attachment point to the network, without triggering a re-association at the client. The LVAP is thus an abstraction for the client's association state, and simplifies the expression of any handoff-based service like mobility managers and client load-balancers in the form of network applications. Since it does not introduce any additional signaling mechanism between the infrastructure and the client, it is legacy client compatible. In addition, it brings a port-per-source view of WiFi networks akin to that of wired networks, which simplifies fine-grained policy enforcement. Note, if a client experiences significant signal strength reduction as a result of an LVAP being migrated to a distant AP, the client will perform a regular re-scan.

While the notion of per-client BSSIDs is employed commercially to handle mobility [5], the concept of an LVAP is new. The LVAP as a programming abstraction solves problems that extend beyond mobility management, as we will demonstrate in this paper.

3.2.2 State Encapsulated by LVAPs

Figure 3 represents the state that is bound to each LVAP. For every associated client (identified by the client's WiFi MAC address), there is a corresponding LVAP which comprises the following information: a unique virtual BSSID, one or more SSIDs, the IP address of the client, and a set of OpenFlow rules. With encryption, the session key will be part of the LVAP state. When an LVAP is migrated from one physical AP to another, all corresponding state (the BSSID, SSIDs, IP address of the client, and OpenFlow rules) is migrated as well. Since the LVAP's BSSID is always consistent, the client does not perform a re-association. By binding a set of OpenFlow rules to the LVAP and allowing applications to program the wireless and wired side of the AP, we integrate our framework with OpenFlow.

3.2.3 Slicing and Control Logic Isolation with LVAPs

Accommodating multiple logical networks on top of the same physical infrastructure with different policies and control applications is called *network slicing*. A network slice is a virtual network with a specific set of SSIDs, where for example, the traffic may be VLAN tagged or directed to a specific destination port. Figure 4 indicates how slicing can be layered on top of LVAPs. A slice is defined as a set of physical APs (or agents), clients (and thus LVAPs), network applications, and one or more unique SSIDs. When clients attempt to associate to a particular SSID, they are automatically assigned to the slice to which the SSID belongs. Thus, the client and its LVAP are now assigned to the same slice. Applications operating on this slice can now manage the client (*e.g.*, perform migrations, or add/remove/update OpenFlow rules on the client's LVAP (*cf.* § 5)). The controller ensures that an application is only presented a view of the network corresponding to its slice. Since LVAPs are the primitive type upon which applications make control decisions, and applications do not have visibility of LVAPs from outside their slice, we thus achieve control logic isolation between slices.

3.2.4 Supporting Authentication Through LVAPs

Our architecture is compatible with the two most commonly deployed approaches for authentication.

WPA2 is the de-facto standard for authentication in today's WiFi networks (defined by IEEE 802.11i). In WPA2 Enterprise, a client authenticates against an authentication server with the AP acting as an authentication proxy to negotiate a session key. This session key is added to the client's LVAP state (*cf.* 3.2.2) and then used to encrypt the connection.

Guest WiFi: In this mode, a client's first HTTP request is redirected through OpenFlow rules associated with the LVAP to a login page. The authentication server returns a security token for the client to the controller after a successful authentication.

3.2.5 Multi-Channel Operation

Odin benefits from operating physical APs' wireless interfaces on the same channel for performing seamless client migration. However, when performing LVAP migrations between physical APs of different channels, the operation is similar to regular WiFi handovers where clients needs to perform a re-association. For multi-channel operation, Odin can leverage IEEE 802.11h (restricted to 5GHz band) to instruct clients to switch to a different channel while keeping association state intact. Additionally, Odin's port-per-source approach to managing clients with LVAPs is complementary to upcoming trends in RF spectrum slicing such as [29]. This will enable multiple LVAPs on the same AP to operate on different channels using a single antenna.

3.3 Reactive and Proactive Applications

Network applications written on top of Odin can function both reactively and/or proactively. Proactive applications are timer-driven whereas reactive applications use triggers and callbacks to handle events. The latter mode of operation is important particularly within WiFi networks due to the dynamic nature of the channel, and the system needs to react based on inputs from different measurement sources. To this end, in our current implementation, an application can utilize multiple measurement sources.

Measurements from the agent: Reactive applications make use of a publish-subscribe system of the Odin agent in order to have a handler invoked at the application whenever a per-frame event of interest occurs at the agents. In our current implementation, applications register thresholds for link-based (PHY and MAC layer) rx-statistics like receiver signal strength indicator (RSSI), bit-rate, and timestamp of the last received packet. For instance, an application can ask to be notified whenever a frame is received at an agent at an RSSI greater than -70dBm. In addition, applications can make use of measurements such as spectral scans that can be collected by the agents.

OpenFlow statistics: OpenFlow provides flow and port-based statistics of entries in switches' flow tables. Applications can query these statistics through the controller to make traffic-aware routing decisions.

External measurement sources: In addition to the usual per-link and per-flow statistics, applications can access data from multiple measurement sources outside the Odin framework, too, including the CPU and memory utilization and the channel active/busy times collected by tools such as `collectd`. We demonstrate this in § 5.

4 Odin on Commodity Hardware

In this section, we describe implementation details of the Odin prototype.

4.1 Controller

The controller is implemented as an extension to Floodlight OpenFlow controller. This allows us to use OpenFlow for Odin specific functionality such as tracking client IP addresses to be attached to their respective LVAPs by tapping into DHCP messages (*cf.* § 4.4). The initial assignment of agents to slices, the initial set of SSIDs per slice, and the network applications to run on each slice are defined via a configuration file. The controller uses a TCP-based control channel to invoke the Odin protocol commands on the agents (*cf.* 7). The controller organizes state on a per-slice basis, allowing it to present applications only a view of their respective slice in terms of associated clients, their LVAPs, and physical APs. Applications are expressed as Java code and run on top of the controller as threads. The programming API includes hooks for applications to view and control mappings of clients to APs, add/remove SSIDs to slices, and to register/unregister subscriptions for the pub-sub mechanism. As a result of using Floodlight, the controller is not distributed and runs on a single machine.

4.2 Agent

Odin agents run on physical APs, and are implemented in the Click Modular Router [16]. The agents implement the WiFi split-MAC together with the controller, host LVAPs, and collect statistics on a per-frame and host basis. They notify the controller whenever a frame is received that matches a per-frame event subscription registered by a particular application (*cf.* § 3.3). Alongside the agents, we run Open vSwitch on the APs to host OpenFlow rules carried by LVAPs as well as those expressed explicitly by network applications and the controller (for instance, to handle DHCP acknowledgments as described in § 3.2.2). Excluding the OpenFlow rules, the state associated with each LVAP hosted by an agent is approximately 48 bytes in size, and up to 32 bytes per-SSID in the slice (slices can announce multiple SSIDs).

4.3 ACK Generation

As mentioned in Section 3.2, the agent needs to ensure the IEEE 802.11 requirement of generating ACKs for each data frame that the client sends to its LVAP. ACK frame generation is handled in hardware by the WiFi cards due to their strict timing constraint. On Atheros WiFi cards, this is implemented using a BSSID mask register which indicates the common bits of all the BSSIDs being hosted on that card.

Whenever the card receives a valid frame, it verifies whether the destination address of the frame matches one of the BSSIDs it is hosting as per the bits set in the BSSID mask. If yes, an ACK frame is generated. However, a practical limitation exists with this mechanism. Consider the following two BSSIDs *02:00:00:00:00:02* and *02:00:00:00:00:01*. In this case, the last two bits are uncommon between the two BSSIDs, causing the mask to be *ff:ff:ff:ff:ff:fc*. This leads to the hardware ignoring the last two bits of the destination address of an incoming frame to decide whether to generate an ACK frame. In this case, a frame destined to *02:00:00:00:00:03* will also cause the hardware to generate an ACK, even though it is not hosting a BSSID with that value: a false positive.

In Odin, since we use one BSSID per client, this needs to be handled carefully. One way to overcome this issue is to assign BSSIDs to client LVAPs such that the mask on the AP where the LVAP is being assigned retains as many set bits as possible and remains orthogonal to the masks of neighboring APs. This can be achieved in software by the controller. Spreading LVAPs over multiple NICs and APs will also alleviate the problem. Another approach is to suppress spurious ACKs by modifying the check that the hardware performs upon receiving a frame. Today's low-end Broadcom WiFi cards support custom firmware such as OpenFWWF (our Atheros hardware does not support this). However, we conjecture that a programmable content-addressable memory for matching incoming frames in hardware enables possibilities beyond just selective ACK generation, with little increase in cost [2] and performance impact. This is particularly important as 802.11ac adoption is increasing, which supports throughputs on the order of 6.77 Gb/s. Recent work on software radios such as OpenRadio [8] will also aid in this direction.

4.4 LVAP Assignment

We now explain how Odin assigns LVAPs to clients.

Discovery: As per IEEE 802.11, clients perform active scans on all possible channels by broadcasting probe request messages. An agent that receives such a probe request forwards it to the controller. The controller then generates a BSSID unique to the client, and retrieves the list of SSIDs to announce (the union of SSIDs across all slices that the agent belongs to). It then instructs the agent to generate a probe response for each of these SSIDs, through the client-specific BSSID. This is how clients discover SSIDs being hosted via Odin.

Association: When a client tries to associate to a specific SSID, it generates probe requests that specify the corresponding SSID. An agent that receives such a probe request forwards the message to the controller. If the controller has not already created an LVAP for the client, it spawns an LVAP for the client on the agent from which this probe request was first received. The client is mapped to the slice that the SSID belongs to (an SSID can only be part of one slice). Once the LVAP is spawned for the client at an agent, the association is performed between the client and the LVAP. If a client does not associate to its LVAP within a configurable amount of time, it is removed from the agent. The agent process maintains a lookup table with the mappings of the client's MAC address to the LVAPs state (*cf.* 3.2.2). It then makes use of this per client state to prepare the right 802.11 frames and ARP packets when communicating with clients.

DHCP and ARP: The IP address of the client is required for the agent to correctly handle ARP requests that concern the client. The IP address of each client is obtained dynamically by the controller which sets up OpenFlow rules in order to receive an OpenFlow `PACKET_IN` event whenever a DHCP-ACK packet is received at an AP. This is done when an agent first registers with the controller. After a client associates and begins to obtain an IP address over DHCP, the controller receives the DHCP-ACK via OpenFlow, obtains the IP address, updates the client's LVAP, and then forwards the DHCP packet to the client via an OpenFlow `PACKET_OUT`.

5 Network Services on top of Odin

On top of our framework, we realized six different Odin applications which are correlated to the use cases described in § 2. For the evaluations, we use ten APs from our indoor testbed distributed across the 16^{th} floor (roughly 750 m^2) of the TEL building at the TU Berlin campus. The WiFi APs are based on embedded hardware (PC Engines Alix 3D2) equipped with Atheros IEEE 802.11abgn cards. All APs are running `OpenWrt` with the `ath9k` Linux driver, user-level Click, and Open vSwitch supporting OpenFlow version 1.0. The Odin controller runs on a x86-based server equipped with 2 CPUs at 2.1 GHz and 4 GB of RAM. We did not hit CPU or memory limitations in any of our experiments.

Application I: Mobility Manager

Supporting client mobility is a crucial feature in enterprise WiFi deployments. We have implemented a purely reactive mobility manager (89 source lines of code (SLOC)) on top of Odin, that leverages LVAP migrations. The application registers a subscription at the agents to be notified whenever an agent receives a frame at a receiver signal strength indicator (RSSI) above a specified value. Using context information passed through the corresponding callback (such as the exact value of the RSSI value and source that triggered the event), the application maintains a map of each client's RSSI value from the point of view of different agents. It then assigns the clients to the agents where they can get the best RSSI value, whilst subjecting its decisions to a hysteresis to prevent spurious oscillations of a client between APs. With legacy switches in the core, a packet is sent out by the new AP to trigger the "backwards learning" mechanism (ARP flushing) to setup new flow entries. With OpenFlow in the core, this can be achieved by updating flow entries along the new path.

We evaluate the architectural consequence of our reactive mobility manager's design, *i.e.*, the number of noti-

Table 1: Notifications generated between a handoff for two RSSI thresholds (T_{rss}) signal strength difference (Δ).

Frame Reception Rate (frames/sec)	$T_{rss} = -96\,dBm$		$T_{rss} = -76\,dBm$	
	$\Delta = 5$	$\Delta = 20$	$\Delta = 5$	$\Delta = 20$
1	13.2	15.8	13.0	16.2
1000	731.66	910.4	670.2	927.0
5000	3373.4	4609.2	3223.8	4515

fications required before performing a client handoff under a given mobility scenario. We show in § 6.2 that LVAP migrations have a negligible effect on the client's throughput. We note that this is only one example of a mobility manager that can be built atop Odin. As demonstrated in 5, Odin applications can utilize different metrics from multiple sources to base mobility decisions on.

Experiment scenario: We use two APs and a single x86-based client. The client associates to the network and initiates a UDP flow. We vary three parameters for the evaluation: (1) the threshold T_{rss} the application sets for subscription notifications, (2) the threshold Δ, *i.e.*, the minimum required difference of the client's RSSI observed at its current AP and potential new AP for the mobility manager to perform a handoff, and (3) the client's transmission rate. We artificially add a fixed offset to the client's RSSI value being recorded by the APs. Using this, we initially set the client's RSSI at the first AP to be 20dB more than at the other, and then reduce it by 0.1 unit every 100ms whilst increasing it at the other AP by the same amount. After 10s, the client's RSSI is higher at the second AP. When the difference is above Δ, the client is LVAP-migrated to the new AP. Thus, only the relative RSSI values of the client at the two APs affects the results (not the absolute values), which enables testing the application using a stationary client. We conduct 5 runs for each combination of parameters and average the results.

Results: Table 1 shows the results of our experiments for different combinations of T_{rss}, Δ, and the client's transmission rate. A decreasing T_{rss} leads to an increased number of notifications generated. A smaller Δ leads to the handoff being performed faster, and reduces the number of notifications in between handoffs. However, the dominant factor here is the transmission rate of the client itself. This shows that it is beneficial to introduce a rate-limiter for generating notifications by the agents. After all, for the same mobility scenario and during the handoff, there is a large number of notifications generated that do not further improve the mobility manager's decisions. Note, the framework cannot track clients that do not transmit any frames at all. One workaround is to use Odin's beacons as a mechanism to track idle clients at different physical APs. In regular WiFi, ACK frames do not contain the source address, but only the recipient address. Since beacons in Odin are unicast, they cause the client to generate an ACK frame addressed to their unique BSSID (which identifies the client). In order to reduce overhead, the system can set the `NO_ACK` bit on the beacons to avoid ACKs from active clients.

Application II: Load Balancer

The benefit of using a load-balancer in a WiFi setting is to increase the throughput for clients due to increased airtime fairness. To illustrate this, consider a scenario where there are multiple clients and one AP: each device gets almost the same share of channel access when operating at the same physical data rate. If only one of the clients generates upload traffic whereas the other stations only download data via the AP, the total upload throughput almost equals the combined throughput of the downloaders (since all download traffic is transmitted by the AP and it has to share channel access with a single uploader). This leads to airtime unfairness among the clients. With more APs and proper load balancing, this unfairness can be alleviated. Furthermore, load-balancing can lead to better resource utilization due to spacial reuse and the capture effect when the collision probability is high. The 802.11k amendment also attempts to address load-balancing, but requires modifications to the client.

Since LVAP-migrations are cheap, fast, and infrastructure-controlled (§ 6.2), client-migration based load-balancing is a good fit for an Odin application. We implemented a load-balancer (76 SLOC) to demonstrate the feasibility of such an application on top of Odin. This application queries the framework once per minute to obtain the list of clients that can be seen by different APs and their corresponding RSSI values. It uses this information to build a map of clients to lists of agents that are candidates for hosting their respective LVAPs. The application then evenly re-distributes LVAPs (clients) across physical APs, constrained by the hearing map.

Experiment scenario: We use up to ten APs. 32 clients automatically associate to the network and request files from a server. We use Harpoon [28] for flow-level traffic generation using a heavy-tailed flow size distribution, similar to traffic on the Internet. After the standard WiFi association, each client sends web requests to the Harpoon server. We conduct experiments with and without load-balancing enabled. Without load-balancing, the client is assigned to the first AP that receives the association request. With load-balancing, each LVAP is placed on a physical AP that has the highest RSSI and does not violate the client load on the AP. Because of the fixed PHY rate, no rate anomaly [31] can arise. We set the rate for management and data frames to the basic rate (6 Mbps). This ensures that all associated clients can exchange data with the APs.

Results: As expected, the overall TCP throughput increases when load-balancing is enabled (see Figure 5).

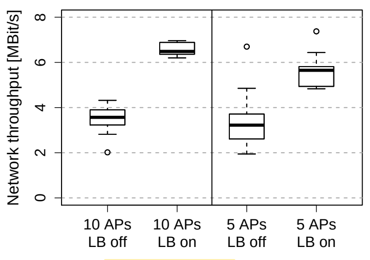


Figure 5: Throughput comparison with and without load-balancing.

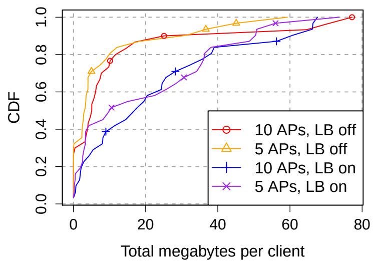


Figure 6: Total throughput per client with and without load-balancing.

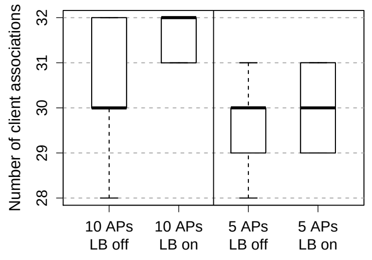


Figure 7: Number of clients contributing TCP traffic.

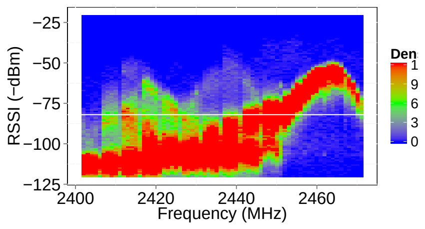


Figure 8: Troubleshooting detects jamming on frequency 2462MHz.

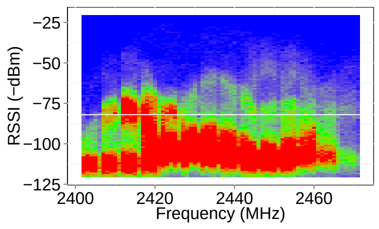


Figure 9: Spectral scan results during normal office working hours.

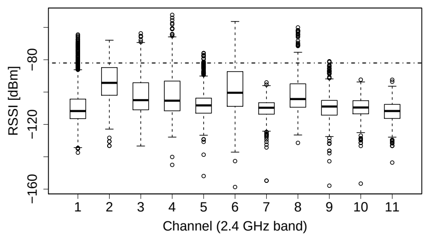


Figure 10: Snapshot of the 2.4 GHz band as seen by the ACS application.

Furthermore, the total throughput is increased when increasing the number of APs. The gain in throughput is attributed to spacial reuse and the capture effect when collisions occur. We observe that TCP connections were established by at least 28 clients across all runs with a median of 30 clients requesting data from the server (see Figure 7). Figure 6 shows the CDF of the per-client throughput of a single run. We observe an increase in fairness among clients with load balancing enabled: *i.e.*, roughly 50% of the clients were able to transmit around 20 MB of data with load balancing enabled compared to 15% without load-balancing. The gain of per-client throughput can be attributed to the previously mentioned spacial reuse, capture effect, and medium access probability of the APs, where each client gets roughly an equal share of airtime at the AP.

Application III: Wireless troubleshooting

Interference from non-WiFi devices such as microwave ovens, cordless phones, wireless security systems, and RF jammers can significantly impede the achievable throughput of nearby WiFi devices. To address this, interference identification systems (*e.g.*, Cisco CleanAir) are starting to become a part of today's enterprise deployments. These systems detect, localize, and quantify the interference impact caused by non-WiFi sources.

To this end, Odin leverages functionality of modern WiFi cards like Atheros AR9280 that provide coarse-grained energy samples per sub-carrier (frequency spacing of 312.5 KHz) of a WiFi channel. This provides the necessary interface for the development of systems like WiFiNet [23] on top of Odin for detection, localization and quantification of interference from a variety of non-WiFi interference sources.

Our troubleshooting application (102 SLOC) periodically (roughly every 5*s*) collects channel snapshots. Figure 8 shows the effect of a jammer (continuous stream of garbage frames) on channel 11 at 2462Mhz over a period of 5 minutes. This data can be used by a jammer detection application, e.g., to localize a jammer via triangulation.

Application IV: Automatic Channel Selection

Automatic Channel Selection (ACS) algorithms aim at automatically determining the best available channel for a WiFi interface. However, identifying combinations of channels for different APs while minimizing interference is challenging. Due the increasing amount of different channel bandwidths within the 2.4 and 5 GHz band. On top of Odin, an ACS application can query different channel properties from the agent (or external sources) for data that characterizes the channel properties. This includes, but is not limited to, spectral samples from the sub-carriers or the active- and busy-time in order to estimate the amount of interference on the channel.

We implemented a simple ACS (97 SLOC) application on top of Odin that is based on a per-AP channel selection scheme. It scans across all available channels and computes the average and the max RSSI for each channel center frequency. Based on multiple subsequent spectral scans, the ACS application picks the channel with the smallest maximum and average RSSI. This example Odin application can be extended to also utilize additional channel properties provided by the Odin agent or external data sources in order to estimate the channel load, e.g., channel active- and busy-time. This information can then be used to implement functionality akin to [25].

Figure 10 shows a snapshot of channel load of all center frequencies within the 2.4 GHz band during the day in our office environment.These snapshots are aggregated by our ACS application over time in order to get to a view similar to the one in Figure 9. Based on this aggregated view, the application then performs channel selection according to the heuristic described above. As indicated within the snapshot and the aggregated view, it can be seen that channel 11 is less utilized than channel 1, which we confirmed to be correlated with the number of APs operating on each channel.

Application V: Energy Efficient WiFi Networks
The problem of energy consumption in telecommunications infrastructure and mechanisms to address it have been studied in detail [33, 22, 12]. In earlier work [24] we demonstrated a system that uses Odin and leverages an integrated energy and mobility management system. The APs are organized into clusters, with each cluster having a master AP and multiple slave APs. The master APs always remain online and provide full coverage. Using a combination of observed network demand and an energy saving policy, the system activates or deactivates slave APs, and offloads clients between the master and the slaves accordingly. This is expressed as an energy manager written as an Odin application, which collects energy measurements via energy meters in order to make informed handover decisions.

Application VI: Guest policy enforcement
Centralized policy enforcement is an important requirement in enterprise WiFi deployments. This is one avenue where LVAPs complement OpenFlow-based access control particularly well. A guest network application uses the framework's API in order to instantiate a guest network on top of a slice of physical APs. It then attaches OpenFlow rules to all LVAPs of that slice which restricts the corresponding clients to be able to access only a certain set of subnets and ports. Since the OpenFlow entries follow the LVAP, other applications such as a mobility manager or load-balancer can operate on the same slice and perform LVAP migrations as well.

6 System Evaluation

In this section, we evaluate the CPU and memory utilization of the Odin controller as well as the latency involved in handling probe requests.

6.1 Controller load due to Pub-Sub

We evaluate the controller's CPU and memory utilization when running the mobility manager (*cf.* Section 5) under synthetically generated load. The aim is to understand the load involved in running a realistic application that makes use of the publish-subscribe subsystem.

We use nine APs of our testbed. The mobility manager is notified whenever a frame is received by any of the APs above a given signal strength threshold. Based on these notifications, the mobility manager decides on whether or not to trigger a client handover. A load generator running on a dedicated server invokes RPCs on the agents in order to mock client associations from a fixed list of clients. It then creates 1000 mock frame receptions per client per second at the APs at varying signal strengths to simulate the reception of arbitrary 802.11 frames. Depending on the signal strength of each frame, the agents notify the controller. Across different runs of the experiments we vary the number of clients as well as the number of APs that can overhear a single frame transmission by a client (*density factor*). The density factor determines how many APs generate a notification for a single frame transmission by a client. Each run of our load generator for a particular parameter takes 250 seconds. We repeat the experiment 10 times for each combination of the parameters and observe the steady state CPU and memory utilization.

We find that an increase in the number of clients for a fixed density factor leads to an increase in the controller's CPU utilization (see Figure 11). Furthermore, for a fixed number of clients, an increase in the density factor leads to an increased number of the mobility manager's subscriptions being triggered, leading to more control messages to the controller. For 500 clients with density factors of 5 and 7, our APs were CPU bottlenecked before being able to saturate the controller. However, we note that 500 is already a very large number of clients to support with only 9 APs. The memory utilization at the controller is 180 ± 7MB across all runs.

6.2 LVAP Handoff Micro-Benchmark

Since LVAPs are a central primitive of Odin, we perform experiments to gauge their effectiveness. The goal is to understand what performance related assumptions Odin applications can make. To this end, we compare LVAP-handoffs against standard WiFi handoffs. We also demonstrate that frequent LVAP-based handoffs do not affect the throughput of a TCP connection.

We use a single client and two APs of our testbed. An HTTP server in the same network acts as a traffic end-point. Since DHCP and authentication related delays only appear in the first connection to the network, the client is provided a static IP and no authentication is performed. Note that an LVAP handoff is not susceptible to the authentication delay. We conduct this experiment on a 5 GHz channel during the night to limit interference.

Comparison of Handoffs
For comparing the impact of handoffs, a client associates to an AP and begins an `HTTP` download of a large file. After 13 seconds, the client is made to handoff to another AP. When using Odin, the handoff uses an LVAP migration, whereas with regular WiFi, the client is explicitly told to perform a handoff using the `iw` command.

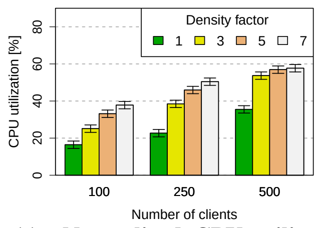


Figure 11: Normalized CPU utilization (2-cores) on the controller per density factor.

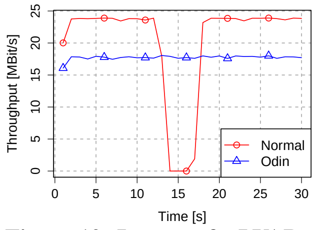


Figure 12: Impact of a LVAP migration and WiFi handoff on TCP throughput.

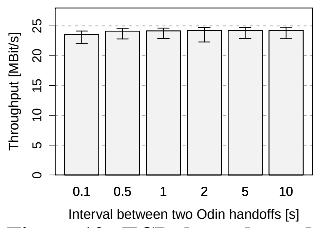


Figure 13: TCP throughput during Odin LVAP-handoffs.

Table 2: Latency for serving probe requests (excluding transmission time on the channel) across 9 APs

Scans per AP/s	Avg. Latency [ms]	Std-deviations [ms]
10	1.791	1.078
20	1.633	0.911
100	1.442	3.266
200	7.373	28.881

Figure 12 shows the TCP throughput over time with standard WiFi compared to Odin. For regular WiFi, the throughput drops to zero for several seconds before recovering. With Odin's LVAP handoff, the TCP throughput is unaffected. As Figure 12 indicates, there is an overall reduction of throughput (close to 5 Mbit/sec) with Odin as opposed to regular WiFi. This is, because we currently use userspace Click to run the Odin agents, resulting in slower and jittery forwarding performance on our APs which makes TCP to throttle down. However, this is orthogonal to continuously maintaining L2 and L3 connectivity, which Odin successfully achieves through LVAP migrations.

LVAP-Handoff frequency benchmark

To understand how often an LVAP-handoff can be executed against a client without affecting its performance, a single `iperf`-based TCP flow is executed with the client as the source over a period of 30 seconds. Between the 5th and 25th seconds of the measurement, LVAP-handoffs are repeatedly triggered between the two APs at fixed rates. Figure 13 shows that LVAP-based handoffs are leading to no significant throughput degradation of the TCP flow. Specifically, even when repeatedly performing LVAP-handoffs every 100 ms the throughput degradation is negligible. This illustrates the inexpensive nature of this operation. Furthermore, in the event of LVAP oscillations due to poorly written control-logic, client performance will not be impacted significantly.

6.3 Probe request serving latency

Since Odin invokes the controller for handling active-scans by clients, we evaluate whether our system can deliver probe responses to clients within the stipulated 30ms constraint.

For the experiment, a load-generator uses a hook on the agent that triggers the effect of a probe request reception. Nine APs of our testbed are used. We increase the rate of probe requests received at each agent. Each agent measures the time it takes in between receiving the probe request, informing the controller, having the controller respond with a BSSID, and then for the agent to construct a probe response message.

Table 2 shows, that the delays introduced due to our split-MAC design are well within the 30ms bound described above. We note that the latency is dominated by the network round-trip delay. Running the load-generator at 1,800 scans per-second (200 scans per-second-per-AP) lead to excessive queuing in the 100 Mbit/s Ethernet switch that our APs were connected to, which lead to the larger delays.

6.4 Compatibility with clients

We have tested our framework with common WiFi client devices, such as Windows, Linux, Mac OS X, iOS and Android devices. Compatibility with a multitude of client devices was demonstrated at [24, 17].

7 Related Work

We next position our work with respect to existing approaches that introduce programmability and/or perform centralized management of wireless networks.

Why not OpenFlow?: There have been efforts to bring OpenFlow to wireless APs (*e.g.*, using OpenFlow together with SNMP [38]). However, we argue that OpenFlow in its current state is ill-suited to orchestrate WiFi networks for many reasons. It cannot perform matching on wireless frames, cannot accommodate measurements of the wireless medium, report per-frame receiver side statistics, or be used for setting per-frame or -flow transmission settings for the WiFi datapath. Yet, extending OpenFlow to accommodate these requirements does not yield any specific benefits. By implementing a custom protocol for handling Odin agents, we thus achieve a cleaner separation of concerns.

Vendor solutions: A plethora of commercial enterprise WiFi solutions exist. These solutions typically manage APs centrally via a controller which is hosted either in the local network [5], or remotely in the cloud [4]. Unfortunately, these solutions do not extend into the purview of cheap low-cost commodity AP hardware that is used by provider networks, nor do they support common, open and programmable interfaces.

Virtual APs: Virtualization of APs have been studied in different contexts. [5] uses a one-BSSID-per-client approach to provide seamless mobility. SplitAP [10] pools together multiple APs in order to regulate air-time fairness. On the other hand, we demonstrate multiple use-cases for the LVAP abstraction as well as its utility as an API for building an SDN for WiFi networks.

Programmable wireless networks and centralized scheduling: Dyson [20] addresses the problem of extensibility in wireless LANs, by defining a set of APIs for clients and APs to be managed by a controller. The controller can query these nodes for channel information, form a global view of the network, and then control the network's behavior to enforce a set of policies. Flashback [11] proposes a control channel technique for WiFi networks, by allowing stations to send short control messages concurrently with data transmissions, without affecting throughput. This ensures a low overhead control plane for WiFi networks that is decoupled from the data plane. DIRAC [39] proposes a split-architecture wherein link-layer information is relayed by agents running on the APs to a central controller to improve network management decisions. However, these systems require special software or hardware on the client, which raises questions of practicality, and goes against the design requirements for our framework. There are systems that do not modify the client in order to deliver services. In DenseAP [19], channel assignment and association related decisions are made centrally by taking advantage of a global view of the network. However, it does not offer slicing of the WiFi, and provides a limited form of client association management because explicitly forces clients to disconnect, and then perform a re-scan in order to change the client's attachment point to the network. Thus, they do not perform client handoffs seamlessly.

CENTAUR [34] improves the data path in enterprise WiFi networks by using centralization to mitigate hidden terminals and to exploit exposed terminals. It is a natural fit for an application on top of Odin. FlowVisor [27] slices the network resources at the flow level and delegates control of different slices to controllers for wired networks. It achieves this by acting as a transparent proxy between OpenFlow switches and multiple OpenFlow controllers. This results in isolation of slices by ensuring that a controller operating on one slice cannot control traffic of another slice. With our framework, we have brought these concepts of isolation into WiFi networks. [37] supports multiple concurrently running experiments using slicing by SSIDs. However, as we show in this paper, slicing by BSSIDs as is done in Odin offers more powerful client isolation and management abilities.

8 Discussion and Further Steps

In designing Odin, we were careful to keep in mind upcoming trends in physical layer virtualization techniques, datapath programmability, hardware-based packet matching and operator requirements.

Virtualization of the PHY layer: Although we have addressed isolation at the IEEE 802.11 MAC layer, our system does not handle virtualization of the PHY layer, which is a logical next step. The IEEE 802.11 standard defines a Point Coordination Function (PCF), for centrally scheduled channel access. However, the PCF is rarely implemented in today's WiFi hardware/drivers. Picasso [29] enables virtualization across the MAC/PHY. It proposes a technique to perform spectrum slicing and allows a single radio to receive and transmit on different frequencies simultaneously. MAClets [13] allows multiple MAC/PHY protocols to share a single RF frontend. These advances can be used by Odin to operate multiple LVAPs with different characteristics on different channels on top of the same AP. Alternative approaches, such as [32] and [36], are incompatible with today's WiFi MAC/PHY and thus do not fit our design requirements.

Programmability of the WiFi data path: Odin's current implementation does not yet provide programmability of per-flow WiFi PHY settings. This is well within the scope of our design because the per-flow and -client transmission settings can be added as LVAP state. Enabling per-flow transmission settings will allow applications to centrally implement rate and power control. With OpenRadio [8], our system could also benefit from a clean-slate programmable network dataplane. This would allow Odin to work around hardware limitations such as that with the BSSID registers used for ACK generation (*cf.* § 4.3). We see OpenRadio, combined with Odin, as a steps towards WiFi networks that are fully programmable down to the PHY.

Performance isolation between slices: Odin in its current form achieves control logic isolation between slices. As of now, it is difficult to enforce FlowVisor-like bandwidth and CPU isolation (on an AP) between slices. First, per-flow bandwidth isolation can be performed on the agents using a token-bucket approach, but this only provides weak isolation on the physical layer, due to the dynamic characteristic of the wireless medium. Although modern WiFi cards are equipped with multiple queues to provide QoS, the assigned priorities and scheduling are hard to adjust. Hence, the FlowVisor approach of per-port queues does not suffice, and WiFi-specific QoS mechanisms need to be incorporated. Second, for agent CPU isolation, throttling control messages between the controller and agent does not suffice. This is because the performance of the pub-sub mechanism has a direct bearing on the effectiveness of a reactive application. If we throttle notifications being sent from an agent to the controller, it may negatively affect the decision-making at the application. We are currently exploring what the right design points are.

9 Conclusion

In this paper, we introduced Odin, an SDN framework for WiFi networks. Through the LVAP abstraction, Odin is well suited to address the complexities of the IEEE 802.11 protocol as demonstrated via the six common network services we have realized with it. Odin runs on top of *today's* commodity access point hardware without requiring client modifications, whilst being well-suited by design to take advantage of upcoming trends in physical layer virtualization and hardware extensions. Thus, with our publicly available prototype, we present one promising way to uniformly manage both wired and WiFi networks given the requirements of today's network operators. We are exploring this further by focusing on joint allocation of both wired and wireless resources.

Acknowledgments: We would like to thank our shepherd, Anthony D. Joseph, and the anonymous reviewers for their valuable feedback.

References

[1] Wi-Fi.org, http://shar.es/Qmnez, May 9th, 2014.

[2] Pica8, http://pica8.org/blogs/?p=201, May 9th, 2014.

[3] Deutsche Telekom. http://tinyurl.com/dtoffload.

[4] Meraki. https://meraki.cisco.com.

[5] Meru Networks. http://www.merunetworks.com.

[6] Swisscom Selects Aptilo. http://tinyurl.com/swisscomoffload.

[7] A. Mishra et al. A client-driven approach for channel management in wireless lans. In *IEEE INFOCOM 2006*.

[8] M. Bansal, J. Mehlman, S. Katti, and P. Levis. OpenRadio: a Programmable Wireless Dataplane. In *ACM HotSDN '12*.

[9] Y. Bejerano and S. Han. Cell breathing techniques for load balancing in wireless LANs. *IEEE ToMC*, June 2009.

[10] G. Bhanage, D. Vete, I. Seskar, and D. Raychaudhuri. Splitap: Leveraging wireless network virtualization for flexible sharing of wlans. IEEE GLOBECOM 2010.

[11] A. Cidon, K. Nagaraj, S. Katti, and P. Viswanath. Flashback: decoupled lightweight wireless control. In *ACM SIGCOMM '12*.

[12] Eduard Goma et al. Insomnia in the Access or How to Curb Access Network Related Energy Consumption. In *ACM SIGCOMM'11*.

[13] G. Bianchi et al. Maclets: active mac protocols over hard-coded devices. CoNEXT '12.

[14] Y. Grunenberger and F. Rousseau. Virtual access points for transparent mobility in wireless lans. In *IEEE WCNC 2010*.

[15] J. Herzen, R. Merz, and P. Thiran. Distributed spectrum assignment for home wlans. In *IEEE INFOCOM 2013*.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ToCS 2000*.

[17] L. Suresh et al. Demo: programming enterprise WLANs with Odin. In *Proc. SIGCOMM '12 (Demo)*.

[18] M.E. Berezin et al. Multichannel virtual access points for seamless handoffs in ieee 802.11 wireless networks. In *IEEE VTC Spring 2011*.

[19] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill. Designing high performance enterprise Wi-Fi networks. In *Proc. NSDI '08*, 2008.

[20] R. Murty, J. Padhye, A. Wolman, and M. Welsh. Dyson: an architecture for extensible wireless LANs. In *Proc. USENIX ATC '10*.

[21] I. Papanikos and M. Logothetis. A study on dynamic load balance for IEEE 802.11b wireless LAN. In *COMCON '01*.

[22] R. Bolla et al. The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization? *IEEE Communications Magazine 2011*.

[23] S. Rayanchu, A. Patro, and S. Banerjee. Catching whales and minnows using wifinet: deconstructing non-wifi interference using wifi hardware. NSDI'12.

[24] R. Riggio, C. Sengul, L. Suresh, J. Schulz-Zander, and A. Feldmann. Thor: Energy Programmable WiFi Networks. In *IEEE INFOCOM '13 (Demo)*.

[25] E. Rozner, Y. Mehta, A. Akella, and L. Qiu. Traffic-aware channel assignment in enterprise wireless lans. In *ICNP 2007*, pages 133–143, Oct 2007.

[26] S. Shenker. An attempt to motivate and clarify sdn. http://www.youtube.com/watch?v=WVs7Pc99S7.

[27] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the production network be the testbed? In *OSDI '10*.

[28] J. Sommers, H. Kim, and P. Barford. Harpoon: a flow-level traffic generator for router and network tests. In *ACM SIGMETRICS '04/Performance '04*.

[29] Steven S. Hong et al. Picasso: Flexible RF and Spectrum Slicing. In *ACM SIGCOMM 2012*.

[30] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao. Towards programmable enterprise WLANS with Odin. In *ACM HotSDN '12*.

[31] G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate wlans. In *Proc. of USENIX'04*, Boston, MA, 2004.

[32] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang. Fine-grained channel access in wireless LAN. In *ACM SIGCOMM 2010*.

[33] The Climate Group. SMART 2020: Enabling the low carbon economy in the information age. Technical report, 2008.

[34] V. Shrivastava et al. CENTAUR: realizing the full potential of centralized wlans through a hybrid data path. In *Proc. MobiCom '09*.

[35] V. Shrivastava et al. Pie in the sky: online passive interference estimation for enterprise wlans. NSDI'11.

[36] L. Yang, W. Hou, L. Cao, B. Y. Zhao, and H. Zheng. Supporting demanding wireless applications with frequency-agile radios. In *USENIX NSDI '10*.

[37] K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown. The Stanford OpenRoads deployment. In *WiNTECH 09*.

[38] K. Yap, R. Sherwood, M. Kobayashi, T. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar. Blueprint for introducing innovation into wireless mobile networks. In *ACM VISA 2010*.

[39] P. Zerfos, G. Zhong, J. Cheng, H. Luo, S. Lu, and J. J. Li. DIRAC: a software-based wireless router system. In *MobiCom*, 2003.