LegoFi the WiFi Building Blocks!

The Case for a Modular WiFi Architecture

Julius Schulz-Zander[†] Stefan Schmid[⋆] James Kempf[*] Roberto Riggio[‡] Anja Feldmann[†]

[†] TU Berlin, Germany [⋆] Aalborg University, Denmark [*] Ericsson Research, USA [‡] CREATE-NET, Italy

ABSTRACT

The increasing demand for flexibility in WiFi network deployments along with more stringent requirements on performance and security stand in stark contrast to today's ossified and expensive WiFi architecture. In particular, today's WiFi networks consists of a large number of control and data plane network functions that are either bundled into a single access controller running on proprietary hardware or are distributed across the network to run on WiFi access points. This approach does not properly reflect the broad and evolving diversity of scenarios in which WiFi is deployed.

This paper makes the case for a functional decomposition of the WiFi: we want to support the allocation and composition of individual (virtualized and programmable) WiFi function blocks, where and when they are most useful. This allocation may also be adjusted dynamically, *e.g.*, during a failover or a scale-out. We present our vision and a rough functional decomposition, describe our proposed *LegoFi architecture*, and explore how LegoFi can benefitial in four different deployment scenarios.

1. INTRODUCTION

Practically all portable end-devices today are WiFi enabled, and with the advent of the Internet-of-Things networks, WiFi is likely to extend to even more objects in the near future. Accordingly, mobile operators are increasingly offloading media traffic from cellular networks to WiFi [17], and several mobile operators plan massive WiFi hotspot deployments as part of the HotSpot 2.0 initiative. Along with this trend, new requirements are being imposed on WiFi networks. Stronger and more easily managed security is one example while simple and fast deployment (and chaining) of innovative new network services like seamless mobility is another. Moreover, the wireless hop is critical for network performance, and can contribute non-negligible delay and jitter especially for high definition media [7, 16].

Over the last years, many specialized solutions, tailored towards specific use cases (*e.g.*, city-wide WiFi, WiFi for the football stadium, etc.) have emerged, to deal with these new requirements. However, what is missing today is a general architecture which makes WiFi an *integral part* of the broadband network. In this paper, we propose a novel *modular* and *open* architecture which decomposes WiFi into network functions. Our modular architecture takes into account the diverse requirements of the network functions in different deployment scenarios, and also facilitates faster innovation, a more flexible service deployment, and cheaper evolution.

In summary, in the LegoFi architecture introduced in this paper, the WiFi control and data planes are decomposed into network functions (the "lego bricks") which can be virtualized and flexibly deployed according to the needs of the deployment scenario. To this end, we also apply the notion of *Lightweight Virtual Network Function*, or *LVNF*, a programming abstraction allowing network programmers to implement complex services by composing several elementary packet processing blocks, *i.e.*, the *LVNF*s, into a more complex packet processing sequence. This concept is similar to the microservices architecture used in three tier enterprise applications. We illustrate the advantages of LegoFi over the existing WiFi architecture for supporting four typical WiFi network deployments: residential cable or DSL, operator hotspots, fibre to the building/curb, and centralized or controller-less enterprise deployments.

2. LEGOFICATION & USE CASES

WiFi networks rely upon multiple network functions, henceforth called *building blocks*. Example functions are handling of client associations, authentication, frame deduplication, traffic en/decryption, and transmission control. LegoFi aims to translate these features into *Lego-like* building blocks, in order to realize flexible WiFi deployments where building blocks can be located (and possibly service-chained) where virtualized compute/network resources are available. However, the different functional building blocks come with different requirements: some are real-time critical (*e.g.*, ACK or RTS/CTS generation) while others are latency sensitive (*e.g.*, fine-grained transmission control [14] or responding to probe messages), and should hence be allocated close to the user's end devices. Other functions can benefit from an increased visibility and ability to optimize network functions across multiple APs (*e.g.*, applications supporting mobility). Finally, security critical applications (*e.g.*, the RADIUS authorization service or storage for crypto keys) should be located in a safe building.

MobiArch'16, October 03-07 2016, New York City, NY, USA

ISBN 978-1-4503-4257-5/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2980137.2980142

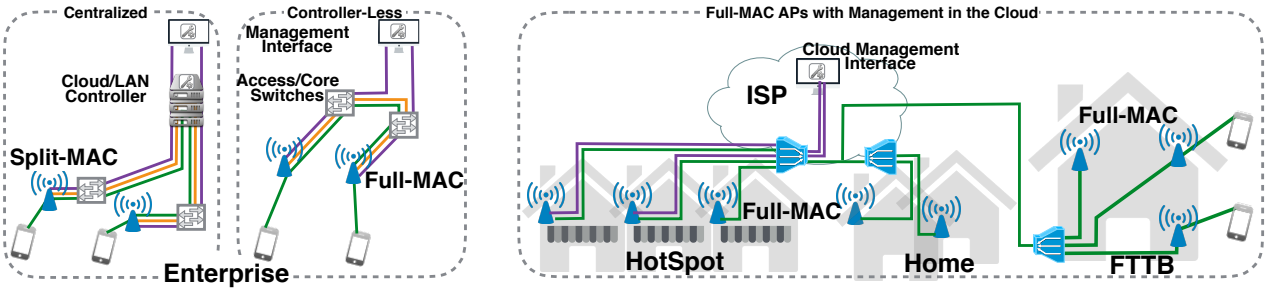

Figure 1: Illustration of Split-MAC and Full-MAC architectures and their deployment scenarios. Enterprise deployments typically separate the management (*purple*), control (*orange*), and data plane (*green*). In most enterprise WiFi deployments, the control plane is either handled by a centralized controller or distributed between the APs. Generally, management functions are out-sourced to a cloud or built into the device.

2.1 Deployment Options and Background

In general, choices for function deployment today consist of the Access Point (AP), LAN, and the Cloud (reachable over the WAN). Two architectures predominate in today's networks, either the *Split MAC* (*e.g.*, in enterprise deployments) or the *Full MAC* (*e.g.*, in home deployments). In the former, WiFi network functions are distributed between a centralized data center (generically Cloud)- or LAN-based WiFi controller and *Thin APs*. In the latter, all WiFi functionality is built into the AP with no remote control plane channel. However, in either case the non-WiFi related management functions such as firmware life-cycle and configuration management or security monitoring are sometimes outsourced to a centralized Cloud- or server-based management system. The AP can provide low latency, which is attractive for fine-grained transmission control or for reacting quickly to packet losses. Deployment in the Cloud imposes significantly higher latency and results in higher communication costs. On the other hand, the Cloud can be more secure than a physically accessible AP, and can give the network operator a single point where they can more easily manage many different WiFi deployments. See Figure 1.

A third possibility has opened recently with the advent of programmable network equipment [3, 1, 9], where the switches or routers are programmable at the first hop in the operator's network but near the WiFi access point. For example, *Fiber to the x* (FTTX), including fiber to the building (FTTB) or distribution-point (FTTdp), offers a programmable switch at a location close to the AP but far enough back in the network so that multiple APs can be served by the same network function. Moreover, a pico data center consisting of a virtualized server in the basement of a building could be placed in close proximity to the user. These deployments offer a platform for WiFi building block deployment while increasing the operator's ability to manage the network, similar to a Cloud deployment, but with lower latency, similar to an AP deployment.

2.2 Use-Cases

Use Case 1: Deduplication.

Dense deployments, with many APs operating on the same channel and users moving around, offer a unique optimization opportunity for LegoFi, for example a football stadium equipped with a large number of access points that provide Internet connectivity to tens of thousands of spectators (*cf.* Figure 2), or dense deployments with many APs operating on the same channel in cities, airports or train stations. In such dense scenarios, the collision probability is high and the performance suboptimal [5]. WiFi usually implements Layer-2 reliability through a two way handshake for each unicast data transmission. More specifically, each successfully received unicast data frame is normally acknowledged (ACKed) by the transmitter. However, if the transmission of an WiFi ACK frame fails, the sender will retry, which can cause duplicate frames at the destination. Today's WiFi access points filter out duplicate frames in order to prevent problems with TCP.

With LegoFi, we can exploit the capture effects across multiple APs occurring in dense settings. As the collision probability is likely to vary across access points (*e.g.*, due to different signal strengths), several access points may successfully "overhear" packets, and could forward them to a nearby, centralized network function performing deduplication. In other words, with a LegoFied deduplication, we can increase the receive probability. Hence, by consolidating this filtering of duplicate frames to a more centralized point in the network, and composing a couple of APs in a hearing domain, we can increase the receive probability and ensure a reduced RTT on the uplink. In case of group related traffic such as broadcast or multicast, which lacks a Layer-2 reliability mechanism, this can also significantly increase the receive probability. LegoFi's *deduplication building block* could be placed on the AP itself, behind a group of APs, *e.g.*, on the connecting switch as in Figure 2 or a centralized controller. This was demonstrated at [12] by introducing a new type of the *Light Virtual Access Point* abstraction [15] which just forwards a client's data frames (*i.e.*, by not generating layer 2 (L2) IEEE 802.11 ACK frames) while solely one LVAP was generating L2 ACKs.

Use Case 2: Load Balancing.

Another use case involves centralized client association management to more evenly balance clients across APs. WiFi networks use IEEE 802.11 *management frames* to handle associations, authentication, capability information exchange, and discovery. Clients need to authenticate and associate with an AP first, before sending data frames. This process begins with the discovery phase, where a client either actively scans for APs by generating probe requests, or passively learns about neighboring APs through beacon frames. During an active scan, APs that respond with probe response messages become potential candidates for the client

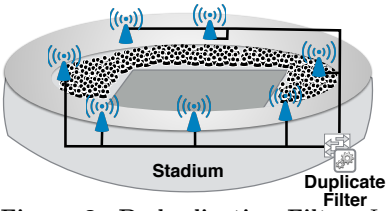

Figure 2: Deduplication Filter: In extremely dense WiFi networks, the collision probability differs among the APs. Enabling the possibility overhearing frames at multiple locations increases the likelihood of a successful frame reception.

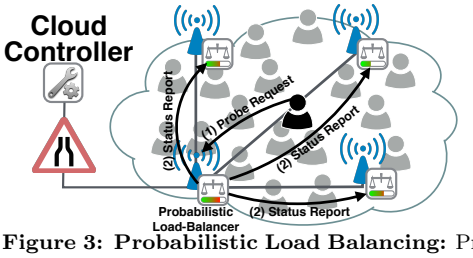

Figure 3: Probabilistic Load Balancing: Probabilistic load balancing allows an almost *controller-less* client-based load balancing at an off-site location when the uplink is a bottleneck. Moreover, this can prevent the controller from getting DDoSed from several off-site locations though WiFi management frames, which are typically handled by a WiFi controller.

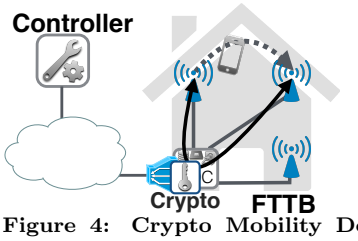

Figure 4: Crypto Mobility Domain: Data encryption and decryption of WiFi frames in a trusted environment can improve security while extending the mobility domain. In particular between untrusted access points, *e.g.*, a neighbor's AP.

to associate with. The client then decides which AP to associate with via a locally made choice. Specifically, during an active scan a client will send a probe request frame on a channel and then wait for probe responses from neighboring APs on that channel before switching to the next channel and trying again.

In most WiFi networks, probe frames are handled by the AP itself, since the *handling of probe frames underlies time constraints*. In enterprise networks, however, a copy of the received probe request can be forwarded to the centralized controller in order to build a global hearing map. While responding to probe frames directly on the AP may provide the best latency, with LegoFi it is possible to detach this functionality from the AP and move it further back into the network. This enables the network to influence the client's network view and thus, the association decisions of clients through suppression of probe response messages at the APs. The result is a kind of "near-sighted" load-balancing.

A LegoFied *probe responder building block* can be coupled with the authentication and association building block to manage client associations in accordance with a hearing map across APs. This building block can be placed behind a group of APs, *e.g.*, on the connecting switch or programmable network equipment. To provide a timely feedback loop, we coordinate the responses across access points, and hide APs from clients when they perform active scan. Note, that clients can still learn about the neighboring APs by passively listening to *Beacon* frames.

For instance, a simple *controller less* client-based load balancer can be realized through a probabilistic probe responder (Figure 3). The association is controlled by blacklisting clients at particular APs before the client actually performs the association, with the blacklisting controlled by the client to AP ratio. Accordingly, the probe responder learns statistics periodically from the surrounding APs via the centralized controller or broadcasting. Since the probe responder must be located close to the APs, this deployment scenerio works as an almost *controller-less* alternative and is especially suited for high latency uplinks.

Use Case 3: En/Decryption Mobility Domain.

AAA functions are not as time-critical but highly security-critical. These functions today restrict client mobility: most traffic is en/decrypted directly at the AP or forwarded to the centralized controller. We see a potential for consolidating this functionality across a number of access points as shown in Figure 4. Thus, mobility domains are enlarged, and decryption of WiFi data frames in a trusted environment can also aid to in improving the security of the system. Especially in dense urban deployments, we can leverage the WiFi AP functionality of a neighboring AP with a better reception rate. For instance, a pico datacenter in the the ISP network, *e.g.*, in the basement of a building can easily provide this functionality.

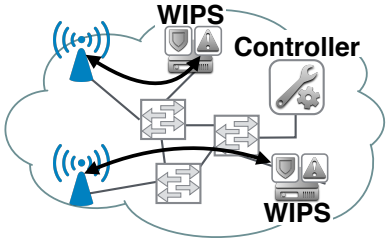

Figure 5: Wireless Intrusion Prevention System: Decoupling this feature from Enterprise WiFi Controllers allows scaled flexibility. WIPS functionality can be placed as a VNF somewhere in the data-path on commodity computing hardware.

Use Case 4: Wireless Intrusion Prevention.

A Wireless Intrusion Prevention System (WIPS) is a typical building block of enterprise WiFi deployments. The key role of WIPS is to monitor the radio spectrum for the presence of wireless network attacks and to take countermeasures automatically. Typical features range from prevention against rogue APs, detection of spectrum jammers, or intrusion detection. WIPS are typically implemented by the wireless AP controller and are not designed as a network function that can be deployed in a flexible manner. This, however, raises scalability concerns for upcoming Cloud-based WiFi deployments based on cheap off-the-shelf APs, as found in today's user's premises.

In order to provide enterprise like WIPS over the Internet for residential and Hotspot WiFi deployments, the WIPS needs to be decoupled from the Controller and modularized. This allows more flexible deployment strategies, *e.g.*, located close to a group of WiFi Hotspot. Moreover, this also allows more flexible placement in enterprise networks, since this feature can be placed somewhere on commodity computing hardware, similarly to today's intrusion detection and prevention systems in wired networks. Furthermore, by consolidating this functionality for a number of WiFi APs,

Table 1: Requirements of the WiFi Building Blocks

Capability	Feature	Latency	Bandwidth	Visibility	Placement	Plane
Radio Resource Management	Channel Selection	medium	low	group	LFL	Control
	Co-channel and RF Interference	medium	low-medium	group	LFL	Control
	Transmit Power Control	low	low	group	TFL	Control
	Transmit Rate Control	low	low	direct	TFL	Control
Client Association	Beaconing	realtime	-	direct	AP+HW	Control
	Probe Handling	realtime-low	low	group	TFL	Control
	Association Handling	medium	low	group	all	Control
Client Authentication	MAC ACLs	high	low	global	all	Control
	PSKs	high	low	global	all	Control
	802.1X	high	low	global	LFL	Control
	RADIUS Accounting	high	low	global	LFL	Control
	Captive Portal / Web Login	high	-	global	LFL	Control
Control Messages	Acknowledgements	realtime	-	direct	AP+HW	Data
	RTS/CTS	realtime	-	direct	AP+HW	Data
Encryption and Roaming	Key Caching	low	low	direct/group	DP/TFL	Control
Data Transport/Forwarding	Duplicate Filter	low-medium	low	direct/group	DP/TFL	Data
Monitoring	Fault/Rogue AP Monitoring	high	medium	direct/group	LFL	Management
	Security Surveillance	high	medium	group	LFL	Management

standard security features can be applied to today's residential and Hotspot networks and scaled flexibility in enterprise deployments achieved (see Figure 5).

2.3 Design Space

A possible decomposition of the WiFi architecture into network function building blocks is summarized in Table 1. Due to space considerations, not all possible network functions are included, and some of those included represent aggregations of functions that, in a more detailed analysis, could be broken down further. The functional requirements are indicated by the columns *Latency* and *Bandwidth*. The *Placement* column indicates where the candidate function can be placed, based on its latency and bandwidth constraints. AP indicates that the function must run on the AP with HW further indicating that it will most likely be implemented in hardware. TFL, for *Tight Feedback Loop*, indicates that the function can run either on the AP or potentially further back in the network but topologically close to the AP, like on a pico datacenter (virtualized rack or server in the basement). LFL, for *Loose Feedback Loop*, indicates that the function can run at an operator data center or other location topologically distant from the AP. Finally, *Datapath* (DP) indicates that the function can run on a programmable switch, like a programmable DSLAM, Open Network Linux switch [3] or the like.

Control frame generation functions such as ACK or RTS/CTS generation need to be deployed on the AP, preferably in hardware, because they must obey realtime constraints specified by the 802.11 standard. Some other functions, such as beacon generation, also need to be deployed on the AP because they involve specific timing constraints. Other functions are latency sensitive, for example probe and association control, but not sensitive enough that they have realtime constraints. These functions can be deployed further back in the network, giving them more *Visibility* over multiple access points, for example for better load balancing in the case of association control. Radio resource management functions such as channel selection and security functions such as 802.1x do not need such tight coordination with the radio layer but do require global visibility across multiple access points and potentially multiple deployments, including APs in a neighboring building in the case of radio resource management, so they can be allocated into operator data centers. Finally, data plane functions such as duplicate filtering can be allocated onto programmable data plane elements close to the AP, if such elements are available.

Deployment of the envisioned LegoFi function blocks can be controlled by an orchestration layer that is run out of the operator data center with agents on the APs and pico datacenters. Each function block needs to define the requirements for the orchestration layer as shown in Table 1. The orchestration layer then deploys the function block in accordance with the constraints.

3. THE LEGOFI APPROACH

In this section, we present a collection of new abstractions for VNF composition and a prototype implementing the LegoFi architecture.

3.1 Abstractions

The main abstraction of LegoFi is the *Lightweight Virtual Network Function* (*LVNF*). A *LVNF* encapsulates a piece of control or data plane functionality necessary for deploying a network service and exposes a high level, declarative interface suitable for that functionality, in addition to an interface for composing with other *LVNF*s. *LVNF*s can be composed to perform complex operations on data plane traffic, or to implement various control plane operations. In LegoFi, everything, including radio access, is implemented as a *LVNF*. Notice however that, some *LVNF*s may require dedicated hardware. For example radio *LVNF*s require wireless NICs and/or Software Defined Radios (SDRs). Moreover, a hardware cryptography engine can be beneficial when performing en/decryption tasks.

The *LVNF* abstraction covers three fundamental elements of VNF composition, namely: layer management (*i.e.*, how to form a virtual network), state management (*i.e.*, how to migrate VNFs from one node to another), and network monitoring (*i.e.*, how to expose the network status):

- **State Management:** Programmers need not be exposed to the details of handling VNF state, nor should they have to deal with the details of polling the network elements for statistics. The *LVNF* management framework (discussed in the next section) tackles this requirement.
- **Layer Management:** Programmers should simply specify the logical sequence of VNFs that traffic should traverse, leaving the details of the routing to an underlying runtime system. The *Virtual Port* abstraction (described below) addresses this requirement.
- **Network Monitoring:** Network managers should have a means to easily query the status of the network using high-level primitives. Such information includes network statistics and topology changes, *e.g.*,

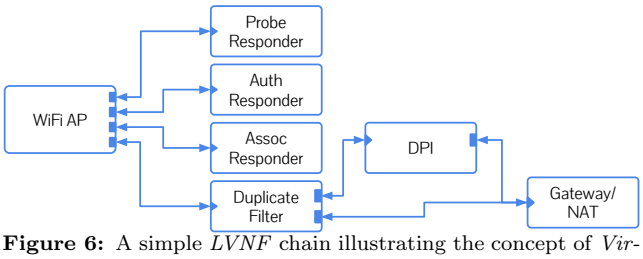

Figure 6: A simple *LVNF* chain illustrating the concept of *Virtual Port*.

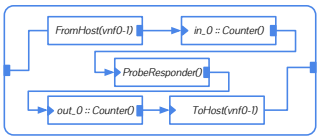

Figure 7: Internal structure of the probe responder *LVNF*.

nodes/links going offline or becoming congested. The *Network Graph* abstraction [10] handles this requirement.

A *Virtual Port* links an output port on one *LVNF* to the input port on another. The link is formed by specifying the portion of the flowspace that must be routed across the ports. For example, Figure 6 depicts a *LVNF* chain in which a WiFi AP is deployed as a collection of *LVNF*s. The composed WiFi AP can be viewed itself as a *LVNF*. This *LVNF* is responsible for handling the pure packet TX/RX operations including rate control, ACK, or RTS/CTS generation, and power control with some of these functions offloaded to the Wireless NIC firmware. The radio *LVNF* has four output *Virtual Ports*. The first three emit, respectively, Probe Requests, Authentication Requests, and Association Requests and are on the control plane, while the fourth output emits data frames and is in the data plane. Management frames are then forwarded to three different *LVNF*s handling the generation of the corresponding response messages, while data frames are forwarded to a duplication filter *LVNF*. HTTP traffic is forwarded to a DPI *LVNF*. Finally, all traffic is redirected to a gateway *LVNF* (Gateway) which connects the WiFi network to the Internet.

3.2 Architecture & Prototype

LegoFi is realized within the EmPower framework [11, 13]. It provides a proof-of-concept *LVNF* management and orchestration framework by combining different virtual infrastructure managers and software defined network controllers. A *Virtualized Infrastructure Manager* (VIM) [2] is a software module that manages the allocation, placement, and control of virtual infrastructure. The goal is to integrate all VIMs under a common north-bound API. While our prototype currently targets 802.11 networks, we believe the LegoFi architectural concept and possibly also the *LVNF* API may also be applicable to 5G wireless networks as well.

The LegoFi architecture currently accounts for three kinds of resources: (1) basic forwarding nodes (*i.e.*, OpenFlow switches), (2) packet processing nodes, and (3) radio access nodes. The latter, in addition to the features supported by the packet processing node, also embed specialized hardware in the form of one or more 802.11 Wireless NICs. Our prototype builds upon Click [6] as a single advanced packet processing engine. Click allows complex VNFs to be built using simple and reusable components, called elements. Click includes over 300 elements supporting functions such as packet classification, access control, and deep packet inspection. Finally, Click is easily extensible with custom processing elements making it possible to support features that are not provided by the standard elements. However, standard Click does not allow to build distributed network functions, since each network function is represented by a single Click script which usually chains several Elements.

Wireless Virtual Network Functions.

*LVNF*s are instantiated starting from a template called an *Image*. Each network function corresponds to an *Image* and consists of a Click script together with a manifest file containing additional information such as the number of input/ output ports used by the *Image*, and the list of Click handlers[1] exposed by the *Image*. A simple *Image* which does not perform any modification on the traffic is shown in the listing below.

Listing 1: An *Image* implementing a null network function.

```
image = Image("in_0 -> Null() -> out_0")
```

The *Image* consists of a very simple Click script and uses one input and one output port. Notice how `in_0` and `out_0` are the names of two Click elements in charge of the network I/O. Upon deployment, the *Image* will be extended with the missing elements declaration; the final Click script deployed on the packet processing nodes will be the following:

Listing 2: An *LVNF* instance implementing a null network function.

```
vnf_in_0_0 :: FromHost(vnf0-0);
in_0 :: Counter();
vnf_in_0_0 -> in_0;
vnf_out_0_0 :: ToHost(vnf0-0);
out_0 :: Counter();
out_0 -> vnf_out_0_0;
in_0 -> Null() -> out_0
```

Notice how $vnfx-y$ is the name of the virtual interface to be created on the target packet processing node. The specific value for x is automatically generated when the *LVNF* is deployed, while y is the actual port id. Moreover, *LVNF* developers do not need to care about the specific name of the virtual interface. Instead, they can perform *LVNF* chaining by using just the port ids y.

Control Plane.

The wireless virtual infrastructure management controller is implemented as a federation of controllers handling virtual infrastructure management for the wired and wireless parts of the network. The Ryu controller [4] handles configuration of the switching fabric in the wired network using OpenFlow [8]. The *LVNF*s are handled by the EmPOWER

[1]Handlers are access points through which users can interact with elements in a running Click router or with the router as a whole.

controller [12, 10] for both the wireless and the wired packet processing functions. The EmPOWER controller presents a REST interface implementing basic CRUD (Create, Read, Update, Delete) operations.

4. CONCLUSION

Motivated by the rapidly proliferating deployment scenarios for WiFi networks, we propose to decompose, open and virtualize the WiFi networks in their basic function blocks, making it possible to deploy them when and where they are most useful. The resulting flexibilities can be exploited to introduce innovative new network services, such as a smart client association management or efficient packet deduplication. Indeed, while we agree on the need to standardize the radio link protocol, we are convinced that the network functions should be *open* to innovation.

The source code implementing the LegoFi control and the data planes is made available to the research community under an APACHE 2.0 license.
Visit: http://empower.create-net.org/

5. ACKNOWLEDGEMENTS

Julius Schulz-Zander was supported by the DFG project Gottfried Wilhelm Leibniz-Preis 2011 FKZ FE 570/4-1. Roberto Riggio was supported by the European Union's H2020 Research and Innovation Action under Grant Agreement H2020-ICT-671639 (COHERENT). James Kempf contributed most of his time to this paper during his sabbatical at TU Berlin.

6. REFERENCES

[1] Cumulus Networks. http://cumulusnetworks.com.
[2] ETSI GS NFV-MAN 001 V1.1.1. http://tinyurl.com/hdgkk85.
[3] Open Network Linux. https://opennetlinux.org.
[4] Ryu SDN Framework. http://osrg.github.io/ryu/.
[5] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*.
[6] Click modular router project. http://read.cs.ucla.edu/click.
[7] Y. J. Gwon, J. Kempf, R. Dendukuri, and R. Jain. VoIPv6 over IEEE 802.11b Wireless LAN. In *Proc. WiNMee*, 2005.
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM CCR.*, 38.
[9] R. Narayanan, S. Kotha, G. Lin, A. Khan, S. Rizvi, W. Javed, H. Khan, and S. Khayam. Macroflows and microflows: Enabling rapid network innovation through a split sdn data plane. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 79–84, Oct 2012.
[10] R. Riggio, A. Bradai, T. Rasheed, T. Ahmed, K. Slawomir, and J. Schulz–Zander. Virtual network functions orchestration in wireless networks. In *11th International Conference on Network and Service Management (CNSM)*, Barcelona, November 2015. IFIP/IEEE, IFIP/IEEE.
[11] R. Riggio, T. Rasheed, K. Slawomir, J. Schulz–Zander, and M. Marina. Programming abstractions for software–defined wireless networks. *IEEE Transactions on Network and Service Management (TNSM)*, May 2015.
[12] R. Riggio, J. Schulz-Zander, and A. Bradai. Virtual network function orchestration with scylla. In *SIGCOMM '15 (Demo)*, New York, NY, USA. ACM.
[13] R. Riggio, I. G. B. Yahia, S. Latré, and T. Rasheed. Scylla: A language for virtual network functions orchestration in enterprise wlans. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 401–409, April 2016.
[14] J. Schulz-Zander, C. Mayer, B. Ciobotaru, S. Schmid, and A. Feldmann. Opensdwn: Programmatic control over home and enterprise wifi. In *Proc. ACM Sigcomm Symposium on SDN Research (SOSR)*, 2015.
[15] J. Schulz-Zander, L. Suresh, N. Sarrar, A. Feldmann, T. Hühn, and R. Merz. Programmatic Orchestration of WiFi Networks. In *Proc. USENIX ATC '14*.
[16] S. Sen, N. K. Madabhushi, and S. Banerjee. Scalable WiFi Media Delivery through Adaptive Broadcasts. In *Proc. NSDI*, 2010.
[17] S. Taylor, A. Young, and A. Noronha. What do consumers want from wi-fi? *Insights from Cisco Internet Business Solutions Group (IBSG) Consumer Research*, 2012.