

ResFi: A Secure Framework for Self Organized Radio Resource Management in Residential WiFi Networks

Sven Zehl, Anatolij Zubow, Michael Döring and Adam Wolisz

{zehl, zubow, doering, wolisz}@tkn.tu-berlin.de

Telecommunication Networks Group (TKN), Technische Universität Berlin (TUB)

Abstract—In dense deployments of residential WiFi networks individual users suffer performance degradation due to both contention and interference. While Radio Resource Management (RRM) is known to mitigate this effects its application in residential WiFi networks being by nature unplanned and individually managed creates a big challenge.

We propose ResFi - a framework supporting creation of RRM functionality in legacy deployments. The radio interfaces are used for efficient discovery of adjacent APs and as a side-channel to establish a secure communication among the individual Access Point Management Applications within a neighborhood over the wired Internet backbone.

We have implemented a prototype of ResFi and studied its performance in our testbed. As a showcase we have implemented various RRM applications among others a distributed channel assignment algorithm using ResFi. ResFi is provided to the community as open source.

Index terms— Residential WiFi, radio resource management, distributed algorithms, self-organization

I. INTRODUCTION

In recent years we have seen a rapid growth in the use of wireless devices such as tablets and smart phones in all environments, e.g. enterprise and homes. Especially, the IEEE 802.11 (WiFi) wireless technology gained lot of popularity as a comfortable way to connect a multitude of devices. As applications like mobile HD video require high QoS, dense deployments of wireless technologies observed nowadays cause performance issues due to high contention and interference within the limited set of radio frequencies. In enterprise networks, remaining within a single administrative domain, this issue is commonly solved by installing a centralized controller which manages the usage of radio resources of all APs [1], [2]. The performance of this controller depends on the scope of information used – this is at least the sum of the traffic and channel usage observations by all the APs but gradually a trend to use also information provided by the end systems (e.g. 802.11k) becomes also visible. It has been widely demonstrated that the coordinated usage of radio resources has led to very significant improvement of the QoS, and in fact it is a fundamental condition to achieve satisfactory QoS in dense, heavily used environments. In contrast residential deployments usually consist of multiple autonomous APs remaining under administration of individual users. Indeed, each AP is usually

installed by a resident who due to lack of technical skills attempts to minimize the configuration effort. While in the past this led to the well-known phenomenon of using mostly the single, pre-set channel, manufacturers started increasing the scope of self-configuration functions provided. The scope of this self-configuration is, however, still limited to functions depending exclusively on local observations within this AP and local controls. In the residential deployment the individual APs - even located in close proximity - do not have a direct way to enter an organized information exchange and negotiations. In addition, the usual consumer electronic devices expected in an apartment usually do not support management features like those provided by 802.11k, so that no information can be obtained from them.

In this paper we present ResFi - a set of basic self-configuration functionalities enabling radio resource management in residential WiFi. ResFi offers the following functionalities:

- 1) Discovery of the immediate neighborhood – any active APs within the radio coverage.
- 2) Setting up secured point-to-point control channels between any pair of immediate neighbors over the wired Internet backbone.
- 3) Exchange of N-hop neighborhood information and continuous monitoring using the above channels.

ResFi is specified and implemented in form of platform independent source code which can be used on top of legacy APs. Up to our best knowledge this is a first attempt to suggest such a platform. We believe that this set of basic function creates a good foundation to develop management application algorithms which itself is explicitly a NON goal of this paper. As a proof of concept we provide however:

- A description, implementation and evaluation of a simple distributed AP channel assignment algorithm.
- A description and prototypical implementation of a simple distributed clustering algorithm to show the suitability of the proposed API.

The performance of the proposed approach is evaluated by means of experiments in real testbeds. Moreover, we provide an emulation in Mininet [3] to give developers an easy way to test own algorithms before deployment in a real testbed. Finally ResFi is provided as open source under a GPL on <https://github.com/resfi>.

II. RELATED WORK

WiFi enterprise networks are already widely deployed in companies, universities and public spaces like airports and fairgrounds. Commercial enterprise WiFi solutions mostly feature a centralized controller which performs RRM for all attached APs. For example the widespread CISCO solution [4] works as follows: each of the APs sends periodically on all the radio channels "neighbor search" messages including the Internet protocol address of their responsible controller and the identifier of the group they belong to within this controller. Neighboring APs forward the received "neighbor search" message including their own AP identifier to their responsible controller (frequently via the wired control connection). This enables the controller to build a hearing map, group APs in RRM groups or to elect a leader controller for the RRM process. Distributed approaches are less frequent - e.g. Aero-hive [5] uses a classical distributed leader election algorithm for RF channel assignment. If a newly started AP discovers other APs on his RF channel, it advertises its neighbor count via the wireless channel while listening for the advertisements of other APs. Finally the AP with the most neighbors wins the right to use the channel. All others switch to the next RF channel and the aforementioned procedure repeats.

Different options to optimize RRM in enterprise WiFi networks have been addressed in research papers. Again the use of a central controller using wireless propagation data, collected from all deployed APs [1], [2], [6], [7], [8] dominates the field. The centralized view is then used to make global decisions in terms of e.g. channel assignment. In addition, more advanced approaches also provide the possibility of load balancing and handover operations [1], [2], [7] or transmit power and rate adaption control [6].

A typical residential WiFi deployment usually consists of statically deployed APs and mobile client STAs. As the APs are not administered by a single authority but rather as each AP is independently managed by another unexperienced user, residential WiFi deployments can be assumed as chaotic [9]. The density of APs is highly correlated with the residential density and large-scale measurements [10] showed that the number of neighboring APs is relatively high in urban environments, i.e. each AP has on average around 16.8 neighboring APs in the 2.4 GHz band. In this chaotic deployment there does NOT exist a natural way to establish the information exchange between each of them. There does not exist a dedicated controller, and the skills of the "human administrator" are usually limited. Patro et al. [11] have postulated the use of a cloud-based controller for channel assignment and airtime management. They propose to run one controller per building either funded by an Internet service provider (ISP) or the building manager. Further, the interaction between the residential APs and the controller is enabled by an extended version of the OpenFlow protocol. This approach seems promising for single administered apartment houses (single ISP or single building manager) but due to the lack of an auto configuration possibility it has its difficulties for all other deployment

scenarios. Besides, the funding of the centralized controller and the payment of its operational costs is not easy to clarify. A controller-less solution would be favorable.

In RxIP [12] a novel approach: direct communication between neighboring APs is introduced for the first time. Each home AP transmits a globally-routable IP address through additional information embedded within the periodically broadcasted beacon frames. This allows passively listening neighbor APs to communicate with the transmitter over the wired Internet, thus featuring a P2P fashion of interaction. RxIP does not aim RRM in general but rather targets the specific use-case of hidden terminal discovery and mitigation of its effect. In dialog with its neighbors, each AP collects independently the information about potential hidden terminals related to him. Therefore the RxIP approach is by definition restricted to discovery of only those neighboring APs which use the same RF channel. Nevertheless a more global view seems to be desirable. Using large-scale measurement data from several cities Akella et al. [9] showed that end-client experience in home WiFi networks could be significantly improved by managing the transmit power in such chaotic wireless networks. Using their proposed load-sensitive rate fallback implementation in which transmitters reduce their transmit power even if it reduces their transmission rate, they were able to show significant throughput enhancement through interference reduction among neighboring APs in dense deployments by incorporating among others the traffic demands of neighboring APs. They did, however not provide suggestions how the relevant stations are to be selected and how should they exchange the necessary coordination information.

Finally numerous papers have addressed distributed radio resource management. For example in [13] power assignment in arbitrary wireless topologies has been assigned in a distributed way. Nevertheless none of these papers investigates specifically HOW to assure connectivity needed for information exchange among the involved nodes.

III. RESFI DESIGN PRINCIPLES

A. System Model

Our view of the "chaotically deployed" WiFi is presented in Fig. 1. Each static AP is assumed to have two network interfaces, namely, an IEEE 802.11 compliant air interface used for wireless communication towards STAs and a wired broadband access interface with a globally-routable IP address to connect to the Internet usually via DSL or cable modem. As we assume DSL/cable as the wired access technology there is significant last-mile latency on the wired link to the first hop inside the ISP's network [14]. We assume that the APs in a given neighborhood are deployed gradually (meaning they are switched on for the first time one by one), Any AP might also be switched off at any time – temporarily or for good. We assume also that every AP is controlled by an AP management unit (APMU) which consists of several functional blocks such as client access control and operational parameter setting (like beacon interval setting). In classical deployment each AP has some - rather simplistic - local radio resource management

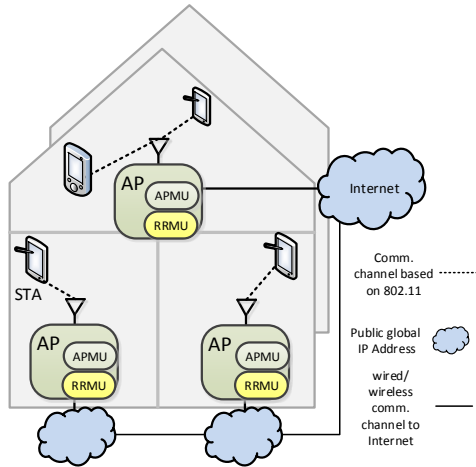


Fig. 1. A residential WiFi network consists of client Stations (STA) and Access Points (AP). Each AP is connected via wired broadband access to the Internet.

e.g. setting of a fixed transmission channel, or simple selection of the transmission channel. We postulate introducing in each AP a dedicated process called radio resource management unit (RRMU) which is assumed to have IP connectivity over the wired Internet backhaul. Moreover, the RRMU is assumed to have an API (called southbound API) making it possible to access radio statistics and parameters within the AP.

B. Principles of the ResFi Framework

The goal of ResFi is to define a self-organized creation of a secured connectivity among the RRMU of all APs within a given neighborhood without:

- Violating the assumptions of keeping each of the participating APs under separate local management
- Any changes in the hardware and drivers of commercially available access points

The approach can be presented in a nutshell as follows: During the boot-up phase of any AP a broadcast scan request including a ResFi vendor specific information element (IEV) containing so called "contact data" is triggered sequentially on each of the supported channels. Any AP within the coverage of this scan request is expected to answer with the respective "contact data" of the responder. This is possible by inserting a broadcast SSID within the probe request which triggers a response from all networks which have been able to receive this request. The contact data, embedded in a IEV of both the active scan probe and response consists of the globally-routeable IP address and port number of the AP's RRMU (on the fixed internet) as well as of a transient one-hop group encryption key and a public cryptography key individual to this RRMU. After having completed the scan and having received the answers, the RRMU of the newly booted AP can establish a secure, point-to-point control channel to the RRMUs of all the "discovered" APs over the wired backbone Internet. Placing the control channel into the wired connectivity has several advantages. Notably there is no additional load on the

wireless interfaces, and there is obviously a lower error rate. On the other hand longer message exchange delays have to be taken into account. This does not seem to be really a big issue, as the radio resource management does not take place in very short time scales. Thus coordination within one-hop neighborhood would be available at this point. It is, however, well known that RRM (e.g. channel selection) can achieve better efficiency if performed over a cluster of APs larger than one hop neighborhood. Therefore ResFi requires that each RRMU is able to act as a forwarder enabling to extend secure connectivity towards up to N hops (N can be set individually for every message sent via ResFi's northbound framework API). ResFi does not define the precise policy to create an RRM cluster within the scope of the connectivity borders mentioned above; neither does it feature a specific RRM approach. Both of these decisions are delegated to an RRM application which is not a part of the platform itself. We will provide in Section VII some examples of such applications.

The security of the control channel is not constrained to the establishment with the use of proper cryptographic keys; in addition the keys are occasionally exchanged (see the following part).

C. ResFi Security Model

Why do we care about the security of the control channel for cooperative RRM? The reason is very simple. Malicious action might severely harm the wireless access of some users, and lead to an unfair advantage of some AP owners. While "unfair cheating" behaviors cannot be completely eliminated (e.g. some AP might claim that there are numerous APs in his vicinity thus luring neighbors to leave a channel for him alone) we offer within our framework a set of measures leading to clear identification of the source and destination of any information as well as assuring the integrity of any information exchanged via the control channel. By the set of this means we can at least be sure, that the possible malicious behavior of any of the participants might be - after detection - uniquely traced back to this participant. And there will be no way this participant might claim his innocence. We will discuss below the threats we are considering - i.e. the security model - adding a "rough outline" of the countermeasures. The primary exchange of security material for establishing a secure control channel takes place over the wireless channel within the exchange of the IEV in the probe request and probe response frames. Therefore the possibility to get the security material is very constrained in space to the local observers.

1) *Thread: Eavesdropping or man in the middle attack on the wired control channel:* An attacker may be able to sniff the whole control traffic of multiple RRMUs which would allow him to get inside views of future behavior or configuration of the APs.

Countermeasure: The communication over the control channel is encrypted by utilizing a one-hop cryptography key. Every RRMU embeds its currently used symmetric group key within its probe request and response frames and uses this key for all outgoing traffic. Enhanced security between

distinct peers is achieved by encrypting unicast messages using a symmetric unicast key obtained on demand using an additional key exchange secured using the public keys which were exchanged during the discovery phase.

2) *Thread: Rogue Attack*: A malicious user may be able to drive through an area and collect the credentials to build up the wired control channel to multiple local RRMUs which would allow him to influence their behavior in a malicious way.

Countermeasure: ResFi RRMUs periodically change the utilized group encryption session keys in regular time intervals via the local wireless channel. The interceptor would have to place multiple local "spy devices" remaining in a continuous connection with him. Nevertheless, on the other hand in case of irregular or suspicious behavior his IP address could be checked.

3) *Thread: Spoofing Attack*: An attacker may try to masquerade as another valid RRMU by falsifying data.

Countermeasure: ResFi provides authenticity by the requirement that all outgoing ResFi messages sent via the wired backhaul have to be signed with the private key of the sender which allows the receiver to validate the signature with the corresponding public key exchanged during the discovery phase.

4) *Thread: Replay Attack*: If an attacker may be able to sniff control packets and send them unaltered but delayed to the original receivers this could result in confusion or misbehavior of the receiver APs RRMUs.

Countermeasure: All sent ResFi messages are equipped with a unique sequence number.

IV. RESFI – DETAILED SPECIFICATION

A. Bootstrapping

After an ResFi enabled residential WiFi AP has booted up, the ResFi agent is started, the first symmetric group key and the RSA key pairs are generated and the discovery process is initiated. For each detected adjacent AP a mutual key and public IP exchange is performed over the wireless channel. This process is also depicted in Fig. 2.

In a first step the ResFi agent of the newly booted up AP (AP_0) performs a full active scan on all available IEEE 802.11 RF channels. This step includes the sending of a probe request¹ including AP_0 's ResFi credentials (public IP of the RRM unit, currently used encryption key and public RSA key, embedded in an IE within the probe request) on each available RF channel which in turn triggers all ResFi APs in vicinity ($AP_{1..n}$) to send out their ResFi credentials embedded in an IE within a probe response¹ back to AP_0 , cf. Fig. 2 tag 1. AP_0 subscribes itself to the publish (Pub) sockets of $AP_{1..n}$ using the public IP provided by the Probe Responses and $AP_{1..n}$ subscribe themselves to the publish

¹ \approx 212 octets standard probe response size or \approx 64 octets standard probe request (depends on number of capabilities broadcasted in general by AP), plus each time the size of the vendor specific big ResFi IE (IE header 6 octets + transient group encryption key and IV 32 octets + 15 octets IP address + 162 octets DER encoded RSA public key)

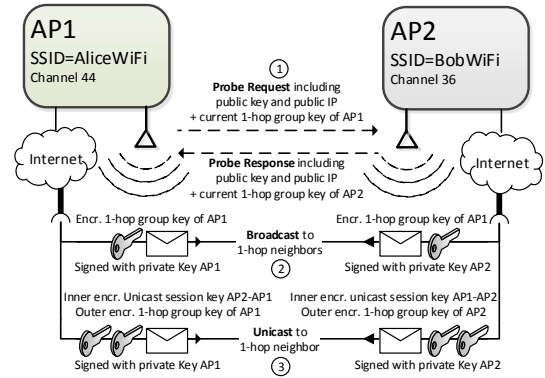


Fig. 2. Overview of the system architecture of ResFi: the wireless channel is used for exchange of configuration parameters (global IP of RRM unit, transient group encryption key and public RSA key) which are afterwards used for setting up the secure P2P out-of-band control channels over the Internet.

(Pub) socket of AP_0 using the public IP provided by the Probe Request. Now AP_0 is able to successfully receive, validate and decrypt all messages sent via the wired backhaul by $AP_{1..n}$ and $AP_{1..n}$ are able to successfully receive, validate and decrypt all messages sent via the wired backhaul by AP_0 . The formation of the secure bidirectional control channel is completed. Broadcast messages to all neighbors are encrypted using the transient symmetric group key, cf. Fig. 2, tag 2 while unicast messages are in advance encrypted using a symmetric unicast session key (obtained on demand via an additional key exchange encrypted using the public RSA keys of the corresponding neighbor AP RRMUs), cf. Fig. 2, tag 3. Moreover, all sent messages are signed using the private RSA key of the corresponding sender.

B. Standard Mode of Operation

In the standard mode of operation the secure bidirectional control channel was already successfully established. All participating ResFi AP RRMUs are able to broadcast messages encrypted with their own group session key and signed with their own private RSA key to its entire one-hop ResFi neighbor RRMUs via the backhaul overlay network. All participating ResFi neighbor RRMUs are able to decrypt these messages, verify their integrity and the authenticity of the sender AP as a result of the mutual configuration data exchange in the bootstrapping phase. In addition to the standard operation of encrypting and signing outgoing messages and decrypting and verifying incoming messages, ResFi APs enable to broadcast messages to N-hop neighbors by performing TTL based forwarding. Unicast messages which are in addition encrypted with a unicast session key can only be sent within the one-hop neighbor group. If needed, multi hop unicast messaging with end-to-end encryption can be implemented on application level, cf. Sec. VII-B.

1) *Transient Group Encryption Key*: During the standard mode of operation no specific control messages except the key change messages (KCM) have to be exchanged to enable the work of the distributed network. The object of the random

periodic KCM and therefore of the group encryption key change is twofold, first it provides confidentiality on the wired backhaul channel and second it ensures that every group participant is a real physical neighbor located in wireless transmission range. All ResFi agents have the obligation to periodically change their group encryption key and inform the other group members by sending a KCM as broadcast via the wired backhaul channel. The key change interval is bounded to KCM plus a randomly generated jitter. If a participating ResFi AP has not sent a KCM during $3 \cdot \text{KCM}$ all other group members are removing the subscription to its publish socket. A KCM always includes the current radio channel and the SSID of the sender to allow the other group members to use a single active frequency scan to obtain the new group session encryption key. For the KCM always the old group key is used, while all messages sent after the KCM are encrypted using the new group key. In advance the new group key is set within the probe response and probe request ResFi IEs for all new probe response and probe request messages. ResFi APs that receive a KCM perform a single frequency active scan for the given RF channel and the given SSID which results in the reception of the new group key as described in the bootstrapping section. For the single frequency scans during runtime an empty probe request is used to trigger the KCM sender to reply with a probe response including the new group session key². As ResFi relies on FIFO sockets and the scan procedure is blocking, all messages following the KCM, encrypted with the new group key, can always be decrypted successfully. Using the KCM scheme and single frequency/SSID scans performed by neighbor APs, the necessity of performing a new full active scan by the key changing AP is avoided. This prevents long deafness times due to active scanning on other RF channels.

2) *IP Address Change*: If the public IP address of a ResFi agent changes, the connectivity to all neighboring ResFi APs is broken. To overcome the connectivity loss, the affected ResFi agent repeats the bootstrap procedure described in Section IV-A.

3) *Radio Channel Change*: As the wireless channel after the boot-up phase is only used to obtain the symmetric group encryption key updates whose retrieval is always triggered by a KCM, which always includes the currently used radio channel, radio channel changing does not interfere the standard mode of operation of ResFi.

4) *Simultaneous Start-up of ResFi Agents*: Simultaneous scanning phases of ResFi agents would prohibit the mutual neighbor detection. ResFi solves this issue by utilizing a random delay between 0 and 100 scanning slots before performing the full scan. This reduces the probability of overlapping scanning phases of two concurrently booting ResFi agents with respect to the birthday paradox to less than 1 %.

Additionally, to consider worst case scenarios, e.g. after a regional power outage, which would cause the concurrent start-up of all neighboring ResFi agents, ResFi executes single

²small ResFi IE (IE header 6 octets + transient group encryption key and IV 32 octets)

frequency scans on all available channels, randomly and distributed over the first 24 h up time. This increases the neighbor detection probability with respect to the average number of 17 neighbors [10], 32 available RF channels (Europe) and a single frequency scan duration of 100 ms, cf. Sec.VIII-A, to more than 99 %.

C. North-bound and South-Bound API

The northbound (NB) application API provided by ResFi is shown in Table I. Using the API any application is able to disseminate JSON messages to either APs in direct wireless communication range or to perform a general N-Hop TTL based flooding operation. Furthermore, unicast communication to direct peers is also available. If a new message via the framework is received the message processing can be controlled by registering a callback. ResFi determines the wireless context transparently for the user.

The ResFi framework can be easily integrated in existing AP solutions by connecting the existing platform to the ResFi southbound (SB) framework API listed in Table II. While the framework SB API is mandatory, the SB API for the RRM is only a suggestion and can be extended to meet further application or algorithm needs. For this reason Table II only provides a subset of possible functions, in particular the RRM related part of the SB API shows the required functions needed for the example applications in Section VII.

V. RESFI – IMPLEMENTATION DETAILS

The ResFi implementation consists of the three components shown in Fig. 3. The ResFi framework agent is connected via the framework southbound API to a modified version of the software AP implementation *Hostapd* [15] via the also modified interface *Hostapd_CLI* which enables the embedding of additional IE(s) within probe responses, and the *IW tool* which is used as an interface to trigger a new WiFi scan and to retrieve its results. Further the retrieval of the probe request payloads is realized using inter process communication (IPC) between *hostapd* and the ResFi agent. For our prototype we utilized standard x86 machines running Ubuntu 14.04. As the ResFi agent is programmed using platform independent Python code, it can be easily ported to various platforms or may be used as reference implementation. As the southbound API prototype realization is Linux specific it can be easily installed on all Linux based systems e.g. OpenWRT APs.

A. ResFi Agent

The ResFi Agent is implemented in Python and runs in user-space. The publish/subscribe (Pub/Sub) sockets for the backhaul wired overlay network are implemented using the Python *ØMQ* library [16]. On top of *ØMQ* the *JavaScript Object Notation (JSON)* is used. Detection of IP address changes is implemented using Netlink events. To provide authenticity and integrity, 1024 bit RSA key pairs are used and for communication confidentiality, symmetric session encryption is performed using the *Advanced Encryption Standard (AES)* in *Cipher*

TABLE I
RESFI NORTH-BOUND API DESCRIPTION

North-bound general framework API	Description
<code>getNeighbors()</code>	returns list of current neighbor IDs.
<code>sendToNeighbor(nodeID, json_msg)</code>	sends a JSON message to particular neighboring AP additionally encrypted using symmetric unicast session key.
<code>sendToNeighbors(json_msg, TTL)</code>	sends JSON broadcast message to each direct neighboring AP, if TTL is used, flooding to N-Hop neighbors is performed.
<code>regCallbacks(rxCb, newLinkCb, linkFailureCb)</code>	register callback functions used to deliver data to application (rxCallback → new message for application, newLinkCallback → new neighbor detected, linkFailureCallback → neighbor was disconnected).
<code>registerNewApplication(naming_pattern)</code>	To handle parallel ResFi applications, name space separation for message handling is used.
<code>getResFiCredentials(param)</code>	if <code>param == 1</code> returns public IP of RRMU, if <code>param == 2</code> returns public RSA key
<code>usePrivateRSAKey(data, mode)</code>	enables to utilize the private key of the RRMU. If <code>mode == 1</code> , returns signature computed over <code>data</code> , if <code>mode == 2</code> , function decrypts <code>data</code> and returns plaintext.
North-bound RRM API (suggestion)	Description
<code>getNetworkLoad(type)</code>	returns current network load: 1=number of served STAs, 2=total TX Bytes in DL, etc.
<code>getChannels()</code>	returns available RF channels.
<code>setChannel(chan)</code>	set (primary) RF channel to be used
<code>setTxPower(mac_addr, dbm)</code>	set transmit power towards STA with <code>mac_addr</code>
<code>injectFrame(data)</code>	inject raw 802.11 frame
<code>enableRTSCTS(mac_addr, bool)</code>	enable usage of RTS/CTS towards STA with <code>mac_addr</code>
<code>startVAP(ssid, rxcb)</code>	start virtual AP with SSID, rxcb callback delivers received raw 802.11 frames.
<code>deauthenticateSTA(mac)</code>	deauthenticate currently associated STA

TABLE II
RESFI SOUTH-BOUND API DESCRIPTION

South-bound framework API	Description
<code>getWiredInterface()</code>	enables ResFi to get the wired interface with IP access to backhaul Internet.
<code>subscribeToProbeRequests()</code>	enables ResFi to retrieve the probe request payload from incoming probe requests.
<code>addIEtoProbeResponses()</code>	enables ResFi to add/modify additional IE(s) to probe responses
<code>performActiveScan()</code>	enables ResFi to start full/single active scan, takes add. IE which is added to probe req.
South-bound RRM API (suggestion)	Description
<code>{set get}RfChannel()</code>	get/set currently used RF channel
<code>{set get}txPower(mac_addr)</code>	get/set transmission power to be used to STA <code>mac_addr</code>
<code>{set get}ClientInfo()</code>	get information about associated STAs (e.g. MAC, capabilities, RSSI, RX/TX count) or modify settings (e.g. disconnect, priority, RTS/CTS usage, (dis)associate STA, (un)blacklist STA)
<code>{getRx getTx}Stats(mac_addr)</code>	get information about sent/received packets and bytes towards STA <code>mac_addr</code>
<code>injectRawFrame(data)</code>	inject raw 802.11 frame into wireless interface
<code>startVAP(ssid, buffer)</code>	start new virtual AP with given SSID, all incoming data is saved in buffer.

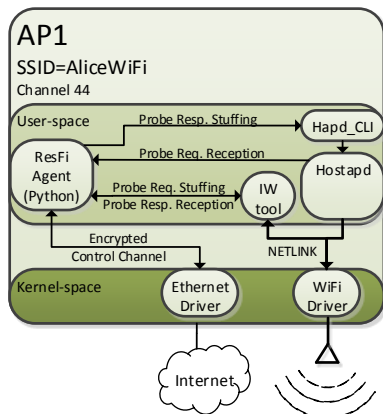


Fig. 3. Overview of components in the ResFi prototype.

Feedback Mode (CFB) with 128 bit key size. All security related functionality has been implemented by utilizing the *PyCrypto Library* [17] and can be easily adapted to the needed purpose (e.g. different key-size, cipher mode or algorithm).

B. Hostapd, Hostapd_CLI and the IW Tool

Hostapd is responsible for performing all the AP management functionality on Linux based platforms. This includes the handling of probe requests and sending the probe responses. We modified hostapd in version 2.1 and the runtime interface `hostapd_cli` to enable first, the embedding of additional IE(s) to all probe responses and second, the retrieval of the IE(s)

from all received probe requests. Besides also the retrieval of the current AP parameters is enabled. The ResFi Agent calls `hostapd_cli` to embed the public IP and the security keys into the probe response frames and to read the AP parameters. The probe request payload is retrieved using an additional `ØMQ Pub/Sub` socket to allow IPC between `hostapd` and the ResFi agent.

The IW tool [18] can be used to configure the WiFi driver. IW internally uses Netlink communication to enable user-space / kernel-space communication. We utilized the IW tool in version 4.3. The ResFi Agent calls IW to start and retrieve the results of an active WiFi scan on a single or over multiple channels and for a specific or unspecific SSID. Moreover, the IW tool is used to embed the additional ResFi IE(s) within the probe request messages used during the boot process.

VI. RESFI MININET EMULATION

In order to offer the application developer an easy way to test own RRM algorithms, before deploying them in a real testbed, the ResFi framework allows the emulation of typical residential networks taking both the wireless access as well as the wired backbone network into account. This is achieved by running ResFi in Mininet [3], a container-based emulation which is able to emulate large network topologies on a single computer. Specifically, we emulate the wired Internet backhaul using the reported last-mile latencies and throughput values from [14]. Moreover, the wireless channel which is

used by ResFi for exchanging wireless management frames is also emulated. This is achieved using the following model: all APs in mutual wireless reception range are connected via a bidirectional link with fixed bandwidth (6 Mbps in case of 802.11g/a), delay (depending on distance) and loss characteristics (configurable parameter) to the same switch. Finally, the AP density which defines the wireless topology is a configurable parameter. Note, any application code which was tested in the emulation environment can be used afterwards to be deployed on real hardware without any modifications.

VII. RESFI APPLICATION EXAMPLES

Next we present examples for applications supported by ResFi as a showcase.

A. Network Clustering

In order to reduce the information update overhead and to optimize the use of the network bandwidth, obtaining a hierarchical organization of the residential AP network is desired. This can be achieved by clustering algorithms that partition the AP nodes of the network into clusters [19]. Clustering is crucial for controlling the spatial reuse of the shared wireless channel (e.g., in terms of time division or frequency division schemes). As a proof of concept we implemented both the Distributed Clustering Algorithm (DCA) and the Distributed Mobility-Adaptive Clustering (DMAC) proposed by Basagni [19] as applications in ResFi.

B. End-to-End Security for N-Hop Neighbors

Basically, ResFi provides one-hop broadcast encryption by utilizing a group encryption key which is obtained through the initial exchange over the wireless channel and one-hop unicast encryption using a symmetric neighbor specific key derived on demand by an additional key exchange secured using the public keys of the peers. If end-to-end security between N-hop neighbors is needed, this functionality can be implemented as ResFi application.

E.g. to enable encrypted communication in an established cluster, cf. Sec. VII-A, the cluster head can utilize the *getResFiCredentials()* function to obtain its public key and can then propagate the key to all cluster nodes and vice versa. As then all participants know the public keys of each other, end-to-end signing and en/decryption (requires prior symmetric unicast session key generation and exchange) of messages is possible using the function *usePrivateRSAkey()*, see Table I.

C. Dynamic Channel Selection

The ResFi framework allows easy implementation of distributed dynamic channel selection schemes for WiFi APs. According to the approach proposed by Mishra et al. [20] each AP may periodically inform its direct neighbor APs about its network load (e.g., number of served clients or flows), recent airtime utilization on different channels, the presence of WiFi and non-WiFi networks and its own radio channel. Such information can be combined at each AP to select the least congested channel. As a proof-of-concept we implemented the aforementioned algorithm (Lst. 1).

Listing 1. Distributed channel assignment implemented using ResFi.

```

from common.resfi_api import AbstractResFiApp
class ResFiApp(AbstractResFiApp):
def __init__(self, log, agent):
    AbstractResFiApp.__init__(self, log, 'distchan', agent)
    self.Hc = {}; self.Mc = {}; self.Sc = {}
    self.jitter = 10; self.nbMap = {} # neighbor map
    self.ch_lst = self.getAvailableChannels(True)
    for ch in self.ch_lst: # init phase
        self.Hc[ch] = 0; self.Sc[ch] = 0; self.Mc[ch] = 0

def run(self):
    rnd_wait_time = random.uniform(0, self.jitter)
    time.sleep(rnd_wait_time) # wait random time
    while not self.isTerminated():
        my_msg = {}; my_msg['pl'] =
            {'ch':self.getChannel(),'ld':self.getNetworkLoad()}
        self.sendToNeighbors(my_msg, 1) # API call
        rnd_wait_time = random.uniform(0, self.jitter/2)
        time.sleep(rnd_wait_time)

def rx_cb(self, m): # ResFi rx callback
    # analyze received message
    self.nbMap[jd['originator']] =
        {'ld':float(m['pl']['ld']),'ch':int(m['pl']['ch'])}
    my_load = self.getNetworkLoad() # my own network load
    # channel assignment algorithm of Mishra et al.
    wmax = 0 # calc wmax
    for entry in self.nbMap: # for each neighbor
        edge_weight = self.nbMap[entry]['ld'] + my_load
        if edge_weight > wmax:
            wmax = edge_weight
    # calc Hc as proposed in Hminmax algorithm:
    for ch in self.ch_lst: # for each channel
        self.Hc[ch] = 0 # reset to zero
        for entry in self.nbMap: # for each neighbor
            if self.nbMap[entry]['ch'] == ch: # same channel
                # select the max() weight; here load
                self.Hc[ch] = max(self.Hc[ch],
                    my_load + self.nbMap[entry]['ld'])
    # mark colors with the max conflict weight
    for ch in self.ch_lst: # for each channel
        self.Mc[ch] = 1 if self.Hc[ch] >= wmax else 0
    # weight sum of all edges to AP whose nb has color c
    for ch in self.ch_lst: # for each channel
        self.Sc[ch] = 0 # reset to zero
        for entry in self.nbMap: # for each neighbor
            if self.nbMap[entry]['ch'] == ch: # same channel
                self.Sc[ch] = self.Sc[ch]
                    + my_load + self.nbMap[entry]['ld']
    # choose color with min sum conflict among all
    # unmarked colors
    best_ch = None; best_val = float("inf")
    for ch in self.ch_lst:
        if self.Sc[ch] < best_val and self.Mc[ch] == 0:
            best_ch = ch; best_val = self.Sc[ch]
    if best_ch is not None and
        self.getChannel() != best_ch:
        self.setChannel(best_ch) # found better sol.

```

D. Interference Management

The well-known hidden terminal problem [8] causes severe co-channel interference (and thus packet loss) in dense WiFi networks with multiple APs operating on the same radio channel. While the use of virtual channel reservation has a potential to reduce the number of hidden nodes it creates significant overhead by exchange of 802.11 RTS/CTS packets. Therefore, an adaptive RTS/CTS scheme activated only on wireless links suffering from hidden terminal problem would be favorable. This can be easily achieved using our ResFi platform. For this purpose each AP could perform passive hidden terminal detection as proposed in [21] and inform its neighboring APs about links potentially affected by hidden terminals for which the RTC/CTS handshake would be enabled, see function *enableRTSCTS()* in Table I.

E. Virtual Access Points (VAP)

The spatial area covered by a single WiFi AP is limited especially when using the 5 GHz ISM band with unfavorable propagation characteristics. In dense residential areas there is a high probability that a significant parts of a residential apartment is in excellent coverage of neighbor’s AP rather than within the range of its own home AP [22]. A way to utilize the neighboring AP is to deploy on-demand a virtual AP on the neighboring AP and to tunnel all encrypted WiFi traffic to the home AP [23]. This allows the client devices to always authenticate against the home AP using the WPA passphrase already stored in the device. There is no registration process; no software to install on the device; not even any settings to change. The on-demand deployment of VAPs can be easily achieved using the ResFi framework. Specifically, each AP has to disseminate information about the configured SSIDs in its home AP to the neighboring AP where dynamically a VAP is configured, see function *startVAP()* in Table I.

F. Client Handover for Load Balancing and Mobility Support

The BIGAP approach [2] enables soft handover operations in centralized enterprise WiFi networks to enable client STA mobility and load-balancing support without network outage. Such soft handover can also be achieved in residential environments between the home AP and a neighboring AP by combining the BIGAP approach with the ResFi VAP application (cf. Sec. VII-E). If the client STA supports dynamic frequency selection (DFS) and both, the current AP and the target AP are operating on different RF channels, soft handover operations are possible by injecting an additional beacon frame including a channel switch announcement IE with the RF channel of the target AP via the function *injectFrame()* executed on the current AP. If no DFS support on the client STA is available, hard-handover using the function *deauthenticateSTA()* on the current AP enables a controlled handover. All aforementioned functions are part of ResFi’s NB API, see Table I.

VIII. EVALUATION

The evaluation section is divided into two parts. First, we evaluate the efficiency of some critical parts of our framework. To this point we evaluate the scanning duration in a small WiFi testbed at our premises experimentally and the control overhead in the wireless channel analytically. The latter is compared with the approach proposed in [12] extended to multi-channel environments to which we refer as RxIP++. Second, we conduct a holistic evaluation of ResFi and two of the example applications from Sec. VII-C and VII-A in the ORBIT radio grid testbed which is characterized by a very dense deployment of wireless nodes.

A. Active vs. passive Scanning

During the start-up phase, ResFi performs a full WiFi scan over all available RF channels. The long deafness duration caused by this scan is only negligible during the start-up phase. During the standard mode of operation only single frequency scans for single SSIDs are utilized to keep the AP deafness

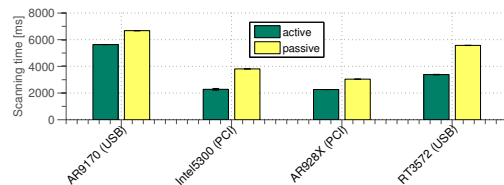


Fig. 4. Scanning duration of a full scan (performed over all available WiFi channels, errorbar shows standard deviation).

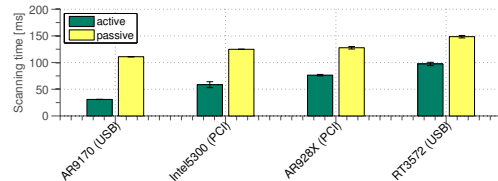


Fig. 5. Scanning duration on single channel (errorbar shows standard deviation).

as short as possible. In this experiment we measure the duration of different active and passive scanning procedures using diverse commercial off-the-shelf WiFi hardware. The results are used in Sec. VIII-B to calculate the reconfiguration overhead caused by ResFi’s periodic scanning procedures.

1) *Methodology*: The utilized WiFi chips are listed in Fig. 4 and 5. We used x86 machines with Ubuntu 14.04 and the scanning calls were executed using the tool iw (cf. Sec. V-B).

2) *Results*: The durations of full WiFi scans are shown in Fig. 4. We see that independent of the scanning mode, the scanning durations strongly vary between different WiFi chips and interconnection technology. In general, chipsets connected via PCI show shorter scanning durations while USB connected chips are slower. However, when the two different scanning modes (passive or active) are evaluated it is obvious that active scanning is always superior to passive scanning w.r.t. to the scanning latency. The results of the latency experiment of single frequency scans are depicted in Fig. 5, interestingly the connection type whether USB or PCI does not affect the scanning latency. Nevertheless, for single frequency scans active scanning is also always faster than passive.

Takeaways: Single frequency scans always provide the shortest latency in comparison to full scans (30 ms vs. 6.3 s for AR9170). Active scanning is always faster than passive.

B. Reconfiguration Overhead

In the following we analyze the overhead in the wireless channel due to periodically changing the transient symmetric group encryption key.

1) *Methodology*: The overhead in the wireless channel is due to the transmission of probe request and reply packets which are sent on a basic bitrate (e.g. 6 Mbps in 802.11a/g). Moreover, during a scanning operation for neighboring nodes the AP is deaf and cannot handle data transmissions of its associated client stations and hence is wasting valuable airtime. Moreover, an associated station may disassociate if it misses too many beacon frames. As shown in Section VIII-A,

the duration of a single active WiFi scan for a given SSID on a particular radio channel takes between 30 ms and 100 ms depending on the hardware. Hence, there is a tradeoff between the rate at which the reconfiguration takes place and the airtime available on the wireless channel for application data transfer. Because in ResFi a reconfiguration at a single AP triggers the scanning operation of each neighboring AP the expected AP density plays a major role. We analyzed the data provided by the large-scale measurement campaign of Biswas et al. [10] whereas the number of neighboring APs in the 2.4 and 5 GHz band is on average 16.8 and 5.1 respectively. To consider virtual WiFi networks in which one physical AP broadcasts multiple SSID and BSSIDs, we have included only BSSIDs into the average neighbor computation in which the RSSI, the Organizationally Unique Identifier (OUI) and the 802.11 capabilities are different. Note, the overhead in 2.4 GHz is also larger than in 5 GHz because the management frames are sent on a lower PHY bitrate, i.e. 1 vs. 6 Mbps.

Next, we give a detailed description of the overhead analysis for both the beacon stuffing approach used in RxIP++ which serves as baseline and the ResFi approach using probe request and response management frames.

a) *RxIP++*: When using the approach from RxIP for dissemination of configuration data the overhead is due to the transmission of additional IEs in the beacon frames and the required scanning overhead in multi-channel environments. For a network of N co-located, i.e. in communication range, APs the relative overhead for each AP can be computed as follows:

$$O_{\text{RxIP}} = \frac{1}{C} \times N \times T_{\text{Beac-IE}} \times R_{\text{Beac}} + (N-1)T_{\text{scan}} \quad (1)$$

where C is the total number of channels available, N is the number of neighboring APs, $T_{\text{Beac-IE}}$ and R_{Beac} are the additional beacon overhead and beacon interval (10 Hz) respectively. The first term represents the overhead due to the additional transmission of IE in beacon frames. Note, that due to multi-channel environment the APs are operating on different radio channels, hence to get the overhead per channel we have to divide the first term by the number of channels. The second term represents the overhead due to scanning deafness.

b) *ResFi*: The overhead of the ResFi approach:

$$O_{\text{ResFi}} = (N-1)(T_{\text{PReq}} + T_{\text{PRep}}) + (N-1)T_{\text{scan}} + \frac{1}{C} \times (N-1) \times (N-2) \times (T_{\text{PReq}} + T_{\text{PRep}}) \quad (2)$$

where the first and third term represent the overhead due to transmission of probe request and reply messages and the second term accounts for deafness due to scanning procedure.

2) *Results*: Using equations 1 and 2 we are able to calculate the overhead for different AP densities, i.e. number of neighboring APs. Here we assume that each AP performs a single group encryption key update. Fig. 6 shows the relative available airtime in the wireless data channel with an update interval of 60 s, i.e. $1 - O_{\text{proposed}}$ and $1 - O_{\text{Beacon}}$ respectively. The results can be summarized as follows. In the 2.4 GHz and 5 GHz band the overhead for a single reconfiguration is highest

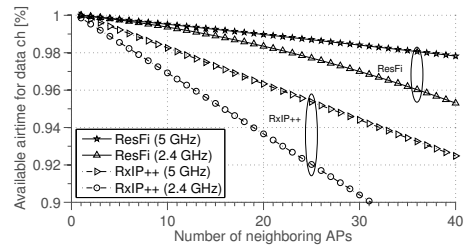


Fig. 6. Impact of periodic group encryption key change ($\Delta = 60$ s) on the available airtime in the data channel.

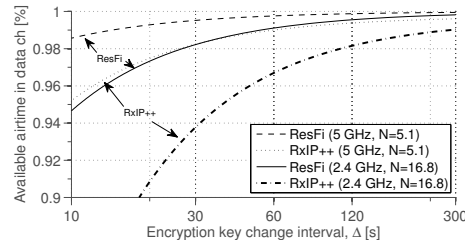


Fig. 7. Tradeoff between encryption key change interval and available airtime in the data channel (N represents the number of neighbors each AP has).

with RxIP++, whereas using the proposed ResFi approach which relies on probe request and probe response frames is superior in both bands for a reconfiguration period of 60 s and AP densities between 0 and 40. Next, we analyze the impact of the reconfiguration rate on the available airtime in the wireless data channel. The results are shown in Fig. 7. From a practical point of view a maximum overhead of 1% is tolerable. Hence the maximum reconfiguration rate is pretty low, i.e. update every 60 and 20 s for the 2.4 and the 5 GHz band respectively. However, for the envisioned residential AP scenario it is still sufficient as we suggest to change the group encryption key every minute. Again ResFi approach is superior in both bands.

Takeaways: There is a clear tradeoff between reconfiguration rate and overhead in the wireless channel. The beacon-stuffing approach (RxIP++) is not efficient in real residential deployments with high AP densities.

C. Reconfiguration Latency

1) *Methodology*: In this experiment we analyze reconfiguration latency in ResFi due to changing configuration data, e.g. group encryption session key. The reconfiguration latency is composed of the delay due to transmission of the key change message (KCM) over the wired out-of-band control channel as well as the scanning delay due to active scanning on a particular channel and given SSID. We considered two different wired backhaul technologies. First, Gigabit Ethernet as a very low latency backhaul which we use in our testbed. It serves as a baseline. Second, the typically used backhaul technology in residential WiFi deployment, i.e. cable/DSL. For the latter we used the traffic control tool [24] to emulate the last-mile latency in residential WiFi deployments as reported by [14]. Note, the last-mile latency is the latency to the first hop inside the ISP's network and hence captures the latency of the access link (DSL/cable). According to [14] most users of cable ISPs are in the 0–10 ms interval whereas a significant

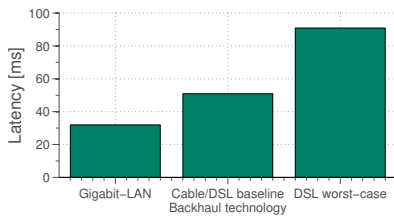


Fig. 8. Reconfiguration latency due to changing encryption key (Conf. \geq 95%).

proportion of DSL users have last-mile latencies of more than 20 ms, with some users seeing last-mile latencies up to 60 ms. For the experiments we used x86 machines with Ubuntu 14.04 and Linksys AE1000 WiFi USB as APs.

2) *Results:* The results are shown in Fig. 8. We see that in the worst case, i.e. DSL, the reconfiguration latency is around 165 ms which is 58 % higher as compared to Gigabit LAN.

Takeaways: Even with a DSL backhaul access the group encryption key can be changed at most 6 times per second which is more than sufficient to achieve the targeted level of security.

D. Large Scale Testbed Evaluation

The goal of the following experiments is twofold. First, we want to evaluate the performance of ResFi in very dense network deployments. As metric we measure the achievable TCP uplink throughput of simultaneous transmitting client STAs in a very dense environment. Second, we want to analyze ResFi's two basic example applications, i.e. namely distributed channel assignment and network clustering.

1) *Methodology:* To mimic a very high density residential scenario consisting of multiple apartments and co-located public hotspots, e.g. coffee shops, we evaluated ResFi in the ORBIT radio grid testbed [25] where 15 and 42 nodes where configured as APs and client STAs respectively. All nodes are in one single collision domain, i.e. in mutual wireless communication range. This enables us to emulate 12 apartments each with a single AP serving a single client STA and three co-located public hotspots each with a single AP serving 10 client STAs.

For the evaluation of the AP channel assignment algorithm, cf. Sec.VII-C, the APs together with the ResFi agents were sequentially started and configured on a randomly selected RF channel. Thereafter the client STAs were connected to corresponding APs. We limited the number of available RF channels to four, i.e. channels 36, 48, 149 and 165. We gave the ResFi agents time to settle down. Thereafter, the achievable TCP/IP uplink throughput from all concurrently transmitting client STAs was measured using iperf. As baseline we selected a random channel assignment algorithm. Finally, the implementation of the ResFi network clustering application, cf. VII-A, was tested.

2) *Results:* As expected all ResFi agents reliably detected each other and every ResFi agent established a secure point-to-point connection to each of its neighboring APs.

The ResFi channel assignment algorithm has set the highly

loaded APs on exclusive RF channels whereas the lightly loaded APs shared the same RF channel. The achieved average uplink TCP throughput for each client STA is presented in Fig. 9 as bar-plot. Moreover, the TCP uplink throughput using the random channel assignment is presented as boxplot after 25 repetitions. In addition, Fig. 10 shows the uplink TCP throughput of all STAs of both algorithms aggregated as boxplots. We observe that the controlled channel assignment using ResFi increases the overall fairness between the client STAs, i.e. the median throughput is increased by 97% as compared to baseline.

The ResFi clustering application was able to successfully built clusters where all nodes within a cluster were able to agree on the same cluster head.

Takeaways: ResFi performs well even in environment with high AP density, i.e. every AP detects all participating ResFi agents in radio range and enables secure communication with them. Moreover, even a simple channel assignment algorithm (< 50 lines of code) provides significant improvement.

IX. CONCLUSIONS AND AREAS FOR FURTHER RESEARCH

Up to our knowledge we have presented the first proposal of a holistic platform supporting automatic establishment of secure connectivity within a definable scope of neighborhood and set of resource management supporting functions for residential WiFi networks. Our proposal allows usage of legacy hardware and avoids violation of the existing management borders following out of the fragmented ownership structure. ResFi was prototypically implemented and the source code is provided to the community as open source. We believe that there is a clear need for such a solution.

On our side the following further areas of work on this framework have been already identified:

- 1) Many RRM require rather tight time synchronization among the nodes. So far ResFi relies on Network Time Protocol (NTP [26]) to time synchronize over the Internet backhaul which achieves only an accuracy of 10s of ms in WAN networks. We intend to extend ResFi to provide over-the-air time synchronization using either 802.11 beacons [12] or using 802.11 management frames for exchanging IEEE 1588 Precision Time Protocol (PTP) frames [27].
- 2) The semantics of the network load - a notion introduced in our API - is not unique. Different function of the air time utilization, number of neighbors etc. have been used in the past in this context. While in the actual version we consider the air time utilization on the actually used channel as the metric of the network load, we consider offering a possibility to introduce in a flexible way a definition of this parameter.
- 3) The notion of one-hop neighborhood is not unique, either. At this moment we include in the one-hop neighborhood any AP which provides a decodable probe response to a probe request broadcasted with the lowest bit rate. This notion might be generalized by attributing

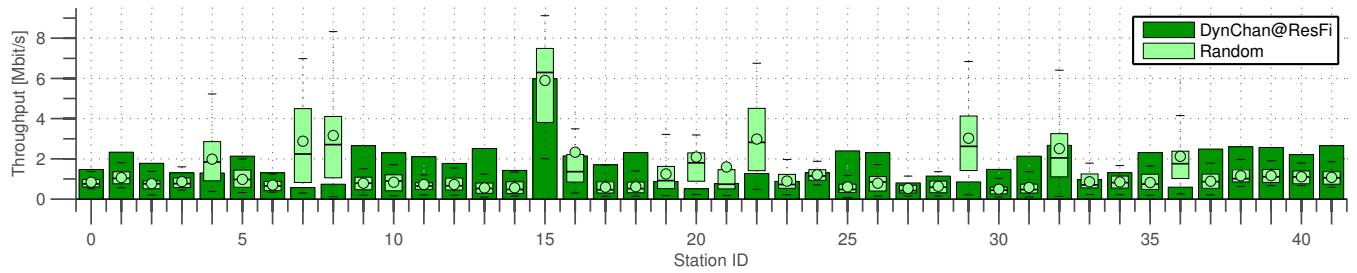


Fig. 9. TCP throughput (uplink) of 42 client STAs, RF channels of the 15 APs assigned by ResFi in comparison to randomly assigned channels

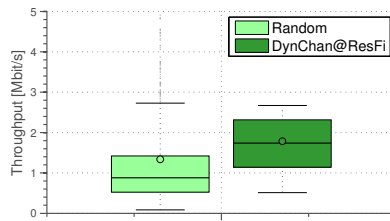


Fig. 10. TCP throughput (uplink) of all 42 client STAs as boxplots

to the probe exchange some constraints on power with which this exchange is performed.

While we believe to have covered a reasonable set of requirements while keeping the solution relatively simple, we have so far verified its merits only using a few very simple cases studied. We hope that the usage of this framework (enhanced by the open source approach) for more complex RRM functions might lead to its further improvement.

ACKNOWLEDGMENT

This work has been supported by the SDN WiFi Community project funded by Deutsche Telekom and European Union's H2020 research and innovation programme (GA No 688116).

REFERENCES

- [1] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill, "Designing high performance enterprise wi-fi networks." in *NSDI*, vol. 8, 2008.
- [2] A. Zubow, S. Zehl, and A. Wolisz, "BIG AP – Seamless Handover in High Performance Enterprise IEEE 802.11 Networks," in *Network Operations and Management Symposium, 2016 IEEE*, April 2016.
- [3] <http://mininet.org/>.
- [4] Cisco, "Radio Resource Management under Unified Wireless Networks," <http://www.cisco.com/c/en/us/support/docs/wireless-mobility/wireless-lan-wlan/71113-rrm-new.html>, Tech. Rep. 71113, May 2010.
- [5] A. Networks, "Radio Resource Management in HiveOS," http://docs.aerohive.com/pdfs/Aerohive-Solution_Brief-Radio_Resource_Management_in_HiveOS.pdf, Tech. Rep., 2011.
- [6] R. N. Murty, J. Padhye, A. Wolman, and M. Welsh, "An architecture for extensible wireless lans." in *HotNets*, 2008, pp. 79–84.
- [7] Y. Yiakoumis, M. Bansal, A. Covington, J. van Reijendam, S. Katti, and N. McKeown, "Behop: a testbed for dense wifi networks," in *Proceedings of the 9th ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*. ACM, 2014.
- [8] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "Centaur: realizing the full potential of centralized wlans through a hybrid data path," in *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 297–308.
- [9] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in chaotic wireless deployments," in *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, ser. *MobiCom '05*. New York, NY, USA: ACM, 2005, pp. 185–199.
- [10] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-E, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 153–165.
- [11] A. Patro and S. Banerjee, "COAP: A Software-Defined Approach for Home WLAN Management Through an Open API," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 18, no. 3, pp. 32–40, Jan. 2015.
- [12] J. Manweiler, P. Franklin, and R. R. Choudhury, "RxIP: Monitoring the health of home wireless networks," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 558–566.
- [13] M. Kubisch, H. Karl, A. Wolisz, L. Zhong, and J. Rabaey, "Distributed algorithms for transmission power control in wireless sensor networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 1, March 2003, pp. 558–563 vol.1.
- [14] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband internet performance: a view from the gateway," in *ACM SIGCOMM computer communication review*, vol. 41, no. 4. ACM, 2011, pp. 134–145.
- [15] J. Malinen, "hostapd: Ieee 802.11 ap," <https://w1.fi/hostapd/>, January 2013, accessed: 2015-10-28.
- [16] iMatix Corporation, "Zmq - code connected," <http://zeromq.org/>, January 2014, accessed: 2015-10-28.
- [17] D. Litzemberger, "Pycrypto - the python crypto toolkit," <https://www.dlitz.net/software/pycrypto>, October 2015, accessed: 2015-10-29.
- [18] J. M. Berg, "Iw - wireless configuration tool," <http://git.kernel.org/cgit/linux/kernel/git/jberg/iw.git>, October 2015, accessed: 2015-10-28.
- [19] S. Basagni, "Distributed clustering for ad hoc networks," in *Parallel Architectures, Algorithms, and Networks, 1999.(I-SPAN'99) Proceedings. Fourth International Symposium on*. IEEE, 1999, pp. 310–315.
- [20] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for WLANs," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 3, pp. 19–31, 2005.
- [21] F. Y. Li, A. Kristensen, and P. Engelstad, "Passive and active hidden terminal detection in 802.11-based ad hoc networks," in *Proceedings of IEEE international conference on computer communications*, 2006.
- [22] J. Shi, L. Gui, D. Koutsonikolas, C. Qiao, and G. Challen, "A little sharing goes a long way: The case for reciprocal wifi sharing," in *Proceedings of the 2Nd International Workshop on Hot Topics in Wireless*, ser. *HotWireless '15*. New York, NY, USA: ACM, 2015.
- [23] J. Vestin, P. Dely, A. Kessler, N. Bayer, H. Einsiedler, and C. Peylo, "CloudMAC: towards software defined WLANs," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 4, 2013. <http://lartc.org/manpages/tc.txt>.
- [24] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *IEEE Wireless Communications and Networking Conference, 2005*, vol. 3, March 2005, pp. 1664–1669 Vol. 3.
- [25] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," Tech. Rep., 2010.
- [26] K. Lee, J. C. Eidson, H. Weibel, and D. Mohl, "Ieee 1588-standard for a precision clock synchronization protocol for networked measurement and control systems," in *Conference on IEEE*, vol. 1588, 2005, p. 2.