Wi–Not: Exploiting Radio Diversity in Software–Defined 802.11–based WLANs

Estefanía Coronado*, Davit Harutyunyan‡, Roberto Riggio‡, José Villalón* and Antonio Garrido*
*High-Performance Networks and Architectures (RAAP). University of Castilla–La Mancha, Albacete, Spain
Email: {Estefania.Coronado, JoseMiguel.Villalon, Antonio.Garrido}@uclm.es
‡FBK CREATE-NET, Trento, Italy
Email: d.harutyunyan,rriggio@fbk.eu

***Abstract*—The increasing demand for live streaming and for remote sensing applications is bringing renewed interest on uplink performances in Wi–Fi networks. Radio diversity can improve the performance of such applications by opportunistically receiving mobile users' traffic at multiple attachment points. However, radio diversity techniques can not be used in standard Wi–Fi networks due to backwards compatibility problems. In this paper we present *Wi–Not*, a novel SDN–based solution for exploiting radio diversity in software–defined WLANs. *Wi–Not* allows mobile terminals to be associated to multiple Wi–Fi APs in the uplink direction improving frame delivery probability in uplink–constrained applications. *Wi–Not* does not require changes to the mobile terminals and can be easily deployed with minimal changes to the network infrastructure. An experimental evaluation carried out over a real–world testbed shows that this approach can deliver an improvement of up to 80% in terms of UDP goodput and up to 60% of TCP throughput. We release the entire implementation including the controller and the data–path under a permissive license for academic use.**

***Index Terms*—Software Defined Networking, IEEE 802.11, WLANs, load–balancing, resiliency**

I. Introduction

Wi–Fi networks are already vastly popular and their usage is only set increase in the coming years. As a matter, of fact rather than being marginalized by 5G systems, Wi–Fi is expected to play a key role as traffic offloading technology. Nevertheless, while so far wireless, and in particular mobile networks, have been designed around the requirements of the downlink (i.e., cell or access point selection is performed using downlink signal strength), in the recent years we have witnessed a mushrooming of new uplink–centric applications such as Machine Type Communications (MTC) and Internet of Things (IoT), and Vehicle to Infrastructure (V2I).

This calls for a paradigm shift where the traffic originated from a mobile terminal is received by one node while the traffic destined to the same mobile terminal is transmitted by another node. This kind of network setup is usually referred to as uplink/downlink decoupling and, in its most general form, can consist of two possibly non overlapping sets of transmitting and receiving nodes. On the other hand, the 802.11 protocol has turned a naturally broadcast medium, i.e. the wireless medium, into an unicast media. Mobile terminals have to select one and only one point of attachment, i.e. the Access Point (AP) to the network, and use only that in both the downlink and uplink direction. Furthermore, the wireless link is asymmetrical so it can be beneficial to relying on multiple attachment points in the uplink direction.

Software Defined Networking (SDN) has recently emerged as a new way of refactoring network functions. By clearly separating data–plane from control–plane and by providing high–level programming abstractions, SDN allows to implement traditional network control and management tasks on top of a *logically* centralized controller. However, albeit SDN is already an established technology in the wired domain, with OpenFlow playing the role of de–facto standard [1], equivalent solutions for wireless and mobile networks have only recently started to appear [2], [3].

The contribution of this paper is twofold: (i) we introduce *Wi–Not*, an SDN–based solution capable of exploiting uplink radio diversity in 802.11 WLANs, and (ii) we propose an algorithm that dynamically selects which AP must generate the L2 acknowledgement. In fact, if multiple attachment points are selected it is fundamental to make sure that only one of them generates the acknowledgement for each frame. Based on a real–world testbed evaluation we have been able to demonstrate an improvement of up to 80% in terms of UDP goodput and up to 60% of TCP throughput. *Wi–Not* does not require changes to the mobile terminals and can be easily deployed with minimal changes to the network infrastructure. We release the entire implementation under a permissive APACHE 2.0 license[1] for academic use.

The rest of this paper is structured as follows. In Sec. II we discuss the related work. We delve into the *Wi–Not* design in Sec. III, whereas in Sec. IV the implementation details are presented. Section V describes the evaluation methodology and discusses the results of the measurements. Finally, Sec. VI draws the conclusions pointing out future work.

II. Related Work

The authors of [4] exploits rate adaptation and partial packet recovery to enhance the network efficiency in terms of robustness and performance. The partial retransmission of the erroneous frames is also used in Wireless Sensor Networks [5]. Alternatively, in [6] the receiver computes the checksum of the corrupt packets and generates negative acknowledgements.

[1] http://empower.create-net.org/

978-1-5386-3416-5/18/$31.00 © 2018 IEEE

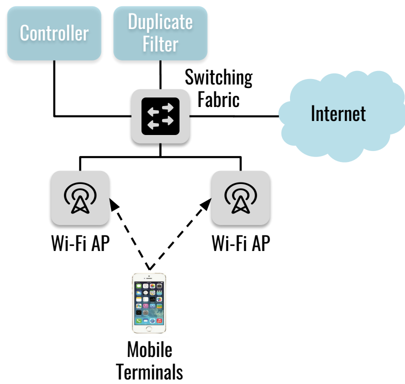

Fig. 1: *Wi–Not* System Architecture.

In this way, the sender only retransmits the incorrect blocks indicated in the checksum. The approach presented in [7] relies on the buffering of several copies of the same corrupt packet to recover the original frame. Multi–radio diversity is used in [8] to coordinate the reception of multiple erroneous copies of a given frame through several links to recover it without retransmissions. Lastly, authors in [9] introduce a link–layer protocol that selects relay nodes based on the notion that the relays with the best link quality have a higher chance than the source of successfully delivering a packet.

III. System Design

A. Overview

SDN has emerged as a new paradigm capable of addressing the limitations of current networking technologies by introducing a fully programmable network architecture, and allowing to implement control and management tasks on top of a (*logically*) centralized controller. Figure 1 depicts the high–level reference system architecture used by *Wi–Not*. As can be seen, it consists of three main elements: the SDN controller, the duplicate filtering module, and data–plane (i.e. switches and APs). The SDN controller is in charge of configuring the switching fabric *and* the Wi–Fi APs. The Wi–Fi APs serve the mobile terminals while the switching fabric delivers the traffic originating at the mobile clients to the duplicate filtering element where redundant frames are eliminated. Notice how, our solution does not depend on a particular controller implementation and can be effectively deployed using platforms such as 5G–EmPOWER [2] or Odin [3]

B. Data–plane

In *Wi–Not* each mobile terminal is attached to one and only one AP in the downlink direction and to one or more APs in the uplink direction. We name the AP providing downlink connectivity to the mobile terminals *Master AP*, while we name the APs providing uplink connectivity to the mobile terminals *Slave APs*. Notice that the *Master AP* also support uplink connectivity and is the AP in charge of generating ACKs. Slave APs on the other hand do not generate ACKs. Downlink connectivity in *Wi–Not* is no different from the standard single radio case, i.e. frames addressed to mobile terminals are dispatched by the switching fabric to the *Master AP* where they are regularly transmitted. Conversely, uplink frames are received at possibly multiple APs.

Figure 2 illustrates, by using a sequence diagram, an example of a frame exchange among the various *Wi–Not* components in the uplink direction. As can be seen the single uplink packet transmitted by the mobile terminal is successfully received by two Wi–Fi APs. Only the *Master AP* however generates the L2 ACK. The uplink traffic is then dispatched by the switching fabric (pre–configured by the SDN controller) to the duplicate filtering element. Here the redundant frames are eliminated and the unique frames are delivered to their intended destination.

C. Duplicate Filtering

The duplicate Filtering module must have access to the sequence number in the Wi-Fi header. However, Wi-Fi frames cannot be directly transported over the backhaul, since their format would not be recognized by the OpenFlow switches. Instead, before entering the wired backhaul, Wi-Fi frames must be first encapsulated into a suitable transport protocol such as the Lightweight Access Point Protocol (LWAPP) [10]. LWAPP frames can then be carried over Ethernet. This process of encapsulation is transparently performed by the AP.

For each mobile terminal active in the network, the duplicate filtering module maintains a simple data structure, named *duplicates table*, consisting of: the MAC address of the mobile terminal, a circular buffer storing the last N sequence numbers transmitted by the mobile terminal, and the number of duplicates filtered so far (for statistical purposes). The size of the circular buffer for our implementation has been set to $N = 3$. The trade–off here is between fast lookup and probability of letting a duplicate frame pass without detecting it. We have empirically found that storing the last 3 sequence numbers is enough in order to avoid duplicates in all our tests.

Frames delivered to the duplicate filtering module are first decapsulated from the LWAPP header, then the module lookups for the mobile station MAC address in the *duplicates table*. If an entry is found, then the duplicate filter checks if the sequence number is present in the circular buffer. If the check is positive, the frame is dropped and the duplicates counter is increased. Otherwise, the frame sequence number is pushed into the circular buffer and the frame is sent back to the backhaul. Notice that, before being delivered to the backhaul, the Wi–Fi frame is first converted into an Ethernet frame.

From the execution complexity standpoint the duplicate filtering module performs just a lookup operation, which is typically very fast if hash tables are used, and then a linear search over the circular buffer. Moreover, centralizing the duplicates filtering function means that the Wi–Fi APs do not have to perform it any more. We remind the reader that regular Wi–Fi APs do implement a similar duplicate filtering functionality in that, due to lost acks, it is possible that a mobile terminals sends the same frame twice.

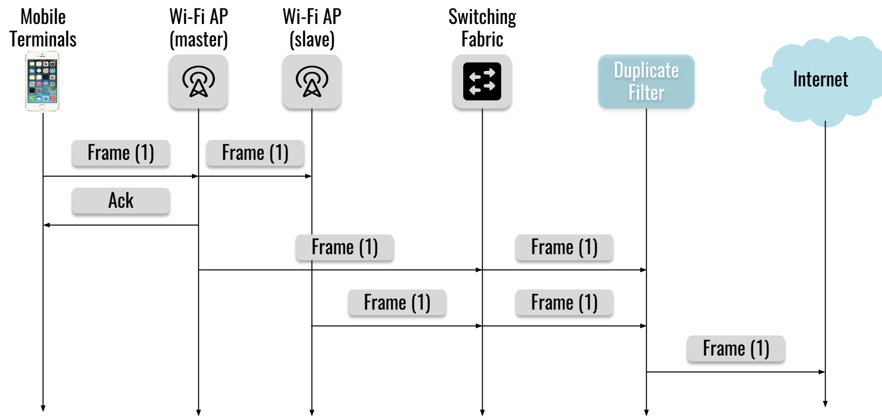

Fig. 2: Frame exchange among the *Wi–Not* components.

Finally, we would like to notice that, albeit in the current implementation only a single *duplicate filtering* module is present in the network, in principle multiple instances can be deployed for load balancing purpose.

D. Master AP Selection

The Wi–Fi medium is intrinsically broadcast, i.e. frame are received and processed by all APs within decoding range of a given mobile terminal. After decoding it, a Wi–Fi APs verifies if the frame checksum is valid and only then it checks if the frame destination address matches its own address. If any of those checks fails the frame is dropped. Conversely, if the checks are passed the AP replies with a L2 ACK and forwards the frame to the backhaul (as an Ethernet frame typically). In order to avoid collisions, in *Wi–Not* only one AP generates the L2 ACK. The selection of such an AP is performed by exploiting the global network view exposed by the SDN controller. In particular, the SDN controller polls all the APs in the network in order to retrieve the list of mobile stations in their neighbourhood together with their signal strength levels[2]. A graphical representation of the network channel quality map built using this information can be found in Fig. 3 where for each pair mobile terminal and AP a measure of the signal strength is reported. The AP with the best signal strength is selected as *Master AP*. The reason for this choice is that the AP with the highest signal strength is likely to successfully received the highest fraction of uplink frames.

IV. Implementation details

To validate the usefulness of *Wi–Not* in real–world settings, we implemented it over the 5G–EmPOWER [2] platform.

A. Data–path Implementation

APs are composed of one OpenvSwitch [11] instance for the wired backhaul and one Click modular router instance [12] for the 802.11 data–path implementation. Click is used to handle the clients/APs frame exchange, while the remaining network intelligence is managed by the 5G–EmPOWER controller.

[2]In this work we use the RSSI as a measure of the signal strength.

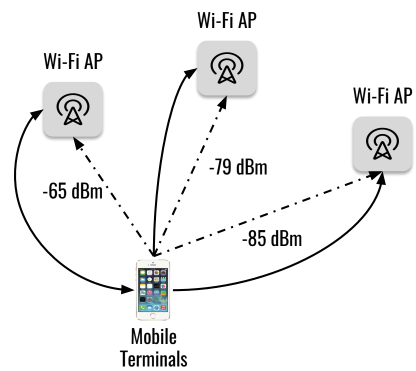

Fig. 3: Channel quality map used to select the *Master AP*.

The AP data–path needs to ensure that an ACK is generated for each frame that the mobile terminal delivers to the AP. ACK frames are handled by the hardware because they have a strict real time constraint. In our APs we use wireless cards based on the Atheros (now Qualcomm) chipset. More specifically we use cards which are supported by the ath9k driver [13]. ath9k–based cards use a hardware register called BSSID mask. This mask is used by the hardware to decide when incoming frames should be acknowledged. The value that this mask holds is equal to the common bits of all addresses that for which a L2 ACK must be generated.

Being based on Odin [3], 5G–EmPOWER creates a virtual AP, named Light Virtual Access Point (LVAP), for each mobile terminal attached to an AP. In this work, we extended then the 5G–EmPOWER platforms in order to allow an application running on top of the controller to create an LVAP at multiple APs and to specify which AP should generate the L2 ACK (by setting the BSSID mask only for that AP).

B. Statistics gathering

The 5G–EmPOWER platform provides a full set of programming primitives to network developers trough a Python–based SDK [2]. Such primitives are used by *Wi–Not* to

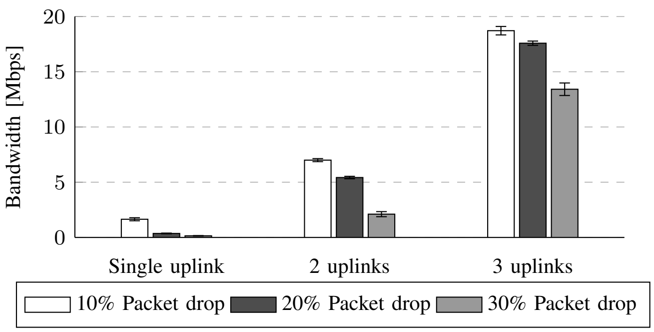

Fig. 4: Maximum bandwidth for TCP traffic and different packet loss ratios: single vs multiple uplinks.

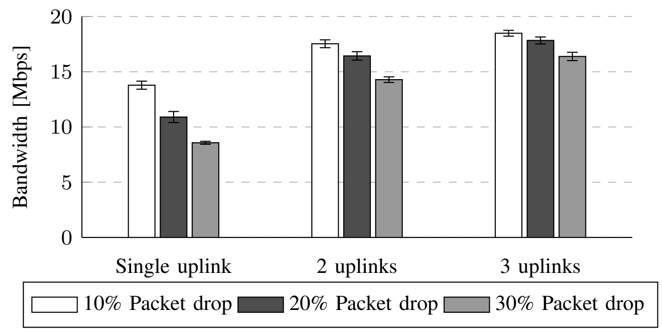

Fig. 5: Maximum bandwidth for UDP traffic and different packet loss ratios: single vs multiple uplinks.

collect the Exponentially Weighted Moving Average of the signal strength between mobile terminals and all APs within decoding range in the network. This information is then used by the *Wi–Not* application implemented on top of the 5G–EmPOWER to select the *Master AP*.

V. Performance evaluation

In order to evaluate the performance of *Wi–Not* we have conducted a set of experiments in a real testbed based on the 5G–EmPOWER platform. The testbed is composed of 3 APs, an OpenFlow switch, and a central controller. All the APs are based on the PCEngines ALIX 2D (x86) processing board with Atheros AR9220 Wi–Fi cards and run OpenWRT 15.05.01. We have introduced in the 802.11 data–path a Click element that drops packet according to a certain probability in $[0, 1]$. For these tests, probabilities of 0.1, 0.2 and 0.3 have been used. A Dell laptop with Ubuntu 16.10 transmits uplink traffic to the machine running the controller using Iperf, which has been run in both TCP and UDP modes with trials of 30 seconds. The maximum bandwidth obtained by a single uplink is compared to the one achieved when involving 2 and 3 uplinks. Each experiment has been repeated 6 times.

The average bandwidth achieved by a single uplink when performing TCP transmissions is compared to the one obtained in the case of multiple uplinks in Fig. 4. This Figure plots how the system behaves upon an increasing packet drop probability. It is shown how the use of multiple uplinks not only leads to an increase in the bandwidth, but also to an improvement in the network reliability. In fact, the inclusion of several uplinks has more than doubles the bandwidth for each additional uplink. This scenario is replicated in Fig. 5 for UDP traffic. As was previously the case, it is demonstrated that the use of multiple links significantly outperforms the use of a single one.

VI. Conclusions

In this paper we presented an SDN–based solution for exploiting wireless links diversity in software–defined WLANs. The proposed solution has been implemented and tested over a real–world wireless SDN platform. Results show that the proposed approach can deliver an improvement of up to 80% in terms of UDP goodput and up to 60% of TCP throughput.

Acknowledgements

This work has been supported by the Spanish Ministry of Economy under Grant Agreement BES-2013-065457, by the European Union's MINECO/FEDER funds under project TIN2015-66972-C5-2-R, and by the H2020 Research and Innovation Action under Grant Agreement H2020-ICT-671639 (COHERENT).

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, "Programming abstractions for software-defined wireless networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 146–162, 2015.

[3] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards Programmable Enterprise WLANS with Odin," in *Proc. of ACM Workshop on Hot Topics in Networks*, New York, 2012.

[4] A. P. Iyer, G. Deshpande, E. Rozner, A. Bhartia, and L. Qiu, "Fast resilient jumbo frames in wireless lans," in *2009 17th International Workshop on Quality of Service*, July 2009, pp. 1–9.

[5] R. K. Ganti, P. Jayachandran, H. Luo, and T. F. Abdelzaher, "Datalink streaming in wireless sensor networks," in *Proc. of ACM SenSys*, Boulder, Colorado, USA, 2006.

[6] B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller, "Maranello: Practical partial packet recovery for 802.11," in *Proc. of USENIX NSDI*, San Jose, California, 2010.

[7] H. Dubois-Ferrière, D. Estrin, and M. Vetterli, "Packet combining in sensor networks," in *Proc. of ACM SenSys*, San Diego, California, USA, 2005.

[8] A. Miu, H. Balakrishnan, and C. E. Koksal, "Improving loss resilience with multi-radio diversity in wireless networks," in *Proc. of ACM MobiCom*, Cologne, Germany, 2005.

[9] M.-H. Lu, P. Steenkiste, and T. Chen, "Design, implementation and evaluation of an efficient opportunistic retransmission protocol," in *Proc. of ACM MobiCom*, Beijing, China, 2009.

[10] "Lightweight Access Point Protocol," Internet Requests for Comments, RFC Editor, RFC 5412, 2010. [Online]. Available: https://www.ietf.org/rfc/rfc5412.txt

[11] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending Networking into the Virtualization Layer," in *Proc. of ACM Workshop on Hot Topics in Networks*, New York, 2009.

[12] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000.

[13] "ath9k linux wireless driver," 2012. [Online]. Available: http://linuxwireless.org/en/users/Drivers/ath9k