Towards a Scalable and Near-Sighted Control Plane Architecture for WiFi SDNs

Julius Schulz-Zander
TU Berlin
Berlin, Germany
julius@inet.tu-berlin.de

Nadi Sarrar
TU Berlin
Berlin, Germany
nadi@inet.tu-berlin.de

Stefan Schmid
Telekom Innovation
Laboratories / TU Berlin
Berlin, Germany
stefan@inet.tu-berlin.de

ABSTRACT

Not much is known today about how to reap the SDN benefits in WiFi networks—a critical use case given the increasing importance of WiFi networks. This paper presents *AeroFlux*, a scalable software-defined wireless network, that supports large enterprise and carrier WiFi deployments with low-latency programmatic control of fine-grained WiFi-specific transmission settings. This is achieved through *AeroFlux*'s hierarchical design. We report on an early prototype implementation and evaluation, showing that *AeroFlux* can significantly reduce control plane traffic.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network management; C.2.1 [**Network Architecture and Design**]: Wireless Communication

Keywords

SDN, Virtualization, OpenFlow, WLAN, WiFi, IEEE 802.11

1. INTRODUCTION & ARCHITECTURE

In the near future, WiFi is expected to be the predominant Internet access technology, and scalability issues will become more important. Moreover, besides control plane communication overhead, latency in WiFi control plane operations is critical, rendering indirections via a remote controller (in the cloud) out of question. This paper initiates the study of the design of a WiFi SDN, called AeroFlux, where transmissions are optimized through a fine-grained and near-sighted control loop.

We first review the AeroFlux architecture (cf. [4] for more details). AeroFlux is based on a 2-tiered control plane that handles frequent, localized events close to where they originate, i.e., close to the data plane, by so-called near-sighted controllers (`NSC`), e.g., located close to the Access Points (APs). Global events, which require a broader picture of the network state, are handled by the global controller (`GC`), the logically centralized part of the control plane, e.g., a set of redundant controllers deployed in a data center.

The `GC` handles events which are not time-critical [1], or events belonging to inherently global tasks [3]. Examples include authentication, wide-area mobility management, global policy verification (including loop-free forwarding sets), client load-balancing, and applications for intrusion detection or network monitoring. In addition, the global controller is best suited to manage middleboxes (`MBs`[1], such as firewalls), including their instantiation and the steering of flows for `MB` traversal. The `GC` also instantiates and controls the `NSCs`, by offloading selected control plane functions to the `NSCs` whilst monitoring their load. This is important in the presence of client mobility, and also when attempting to realize per-flow application- and traffic-aware optimizations, where `MBs` need to perform a deep packet inspection (DPI) for flow classification. Besides reducing the network's control plane traffic load, this also ensures a low latency for control plane operations between the `NSCs` and WiFi APs: an important property given the non stationary characteristic of the wireless channel, which requires timely adjustments.

Essentially all functions exported by the Radio Agent (`RA`) on the APs can be used by control plane applications running on the `NSCs`. Moreover, in AeroFlux, a software daemon running on each AP (henceforth referred to by `RA`), maintains the AP's Light Virtual Access Points (LVAPs) [5]. LVAPs keep per-client association and authentication state, and they store per-client OpenFlow and WiFi Datapath Transmission (`WDTX`) rules. `WDTX` rules extend OpenFlow rules by defining per-flow transmission settings. `WDTXs` are kept on a WiFi AP and are linked to OpenFlow flow table entries, while a `WDTX` entry can relate to one or multiple `OF` rules. Through the `AF` protocol, `NSCs` (and the `GC`) manage the `WDTX` entries on an AP. In our current prototype, a `WDTX` entry allows to control the following 802.11-specific settings: transmission rate, number of retries, transmission power, and the usage of ACKs and RTS/CTS. `WDTX` rules can be extended to enable flow-based control over the transmission chain or the antenna used for the transmissions.

AeroFlux exports an interface that allows applications to specify *control application constraints*, informing the `GC` which apps need to be handled by `NSCs` (e.g., due to their stringent latency requirements). The `GC` will use these constraints to decide, where in the control plane (on the `GC` itself, or on an `NSC`) to run the application. As a result, AeroFlux enables a centralized per-flow control of the Wifi datapath, which supports application-aware service differentiation. Specifically, AeroFlux introduces a more fine-grained control by adapting the power, transmission rate and retry count on a per-slice, per-client, or per-flow level. Furthermore, AeroFlux allows infrastructure controlled handovers, authentication and the automated routing of flows through middleboxes [2].

[1]We can safely assume that sooner rather than later, most network functions in enterprise and operator networks are provided by software running on standard server hardware, as envisioned by Network Function Virtualization (NFV).

HotSDN'14, August 22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2989-7/14/08.
http://dx.doi.org/10.1145/2620728.2620772.

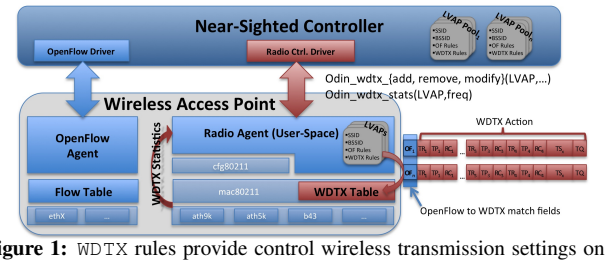

Figure 1: WDTX rules provide control wireless transmission settings on a per-flow level.

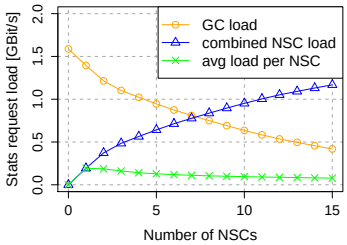

Figure 2: Near-sighted control reduces control plane load.

2. PROTOTYPE

Our early AeroFlux prototype builds upon our existing Odin-framework [5]. The prototype consists of the following architecture building blocks.

Global Controller: The GC is based on the Odin controller. The Odin controller in turn is based on the open source Floodlight controller, which, as of today, does not support a distributed deployment. By relying on Odin, AeroFlux inherits an implementation of the LVAP concept. The GC currently implements two southbound interfaces, the OpenFlow protocol for the wired and a separate datapath control interface for the wireless portion of the network. Odin introduced a separate southbound interface for the WiFi specific transmission settings, since accommodating that extra functionality into the OpenFlow protocol does neither yield any specific advantages nor does it simplify the prototyping.

Near-Sighted Controller: The NSC interacts with the RA via a south-bound protocol. The NSC manages pools of LVAPs that belong to a network slice. The communication interface between GC and NSC is currently under development. A basic stand-alone variant of the NSC exists as proof-of-concept prototype to investigate the feasibility and performance of RA's components.

Radio Agent: Each AP hosts an RA which implements an interface for controlling WiFi-specific transmission settings and for gathering wireless related measurement data such as link and channel statistics, e.g., adjusting per-flow transmission power level or export channel utilization. Furthermore, the RA hosts Light Virtual Access Points [5], which are virtual, per-client APs, that abstract the client's state. associations, authentication, handovers and slicing. The NSC and its interactions with the RA is shown in Fig. 1.

WDTX rules: The WDTX rules are implemented within the mac80211 framework of the Linux Kernel to benefit from the WiFi driver abstraction and the Linux rate control algorithms (Minstrel and MinstrelHT) to maintain transmission rate statistics. WDTX rules can overwrite the per-flow physical transmission rate (TR_n), power (TP_n) and retry count (RC_n). Rules can either contain fixed and/or meta transmission settings, including *best probability rate*, *best throughput rate* and *basic rate*, which can be set for the device multirate retry chains. WDTX rules are bound to OF rules (Fig. 1) by tagging of all packets belonging to an OF flow at the ingress port. We are currently investigating the possibility to assign transmission bounds to WDTX rules, e.g., a minimum transmission success probability or maximum duration including frame retransmissions.

3. EARLY EVALUATION

To get an idea of the scalability characteristics of AeroFlux, we conducted simulations based on data from an existing enterprise campus network, serving up to $9,000$ clients every day. The network statistics indicate six popular areas with more than 400 clients, two with more than a thousand clients. Roughly 60 percent of the clients are served by less than 10 hotspot locations. We consider a power and rate control application: This application requires frequent communication (statistics requests) with the RA and does not rely on a global view of the network state; it is hence an ideal candidate for near-sighted control.

To estimate the control plane load, we will only take into account *statistics requests*. This yields a lower bound on the actual control plane overhead and hence on the benefit of the near-sighted control. The frequency at which WDTX entries are updated depends on the implemented rate and power control algorithm. Analogous to the Linux default rate control implementation, our application collects up-to-date per-client statistics once every $100ms$. A client's statistics status update is 2.3kB in size on average when supporting 802.11n 2x2 MIMO transmission rates.

Fig. 2 shows the reduction in statistics request load on the GC when incrementally deploying NSCs in a greedy manner: The location of a next NSC is chosen to be the most popular NSC-less location, i.e., the location with the greatest number of associated clients that is not yet served by an NSC. Alternative algorithms performing more rigorous optimizations will only improve the benefits of the near-sighted control further.

Fig. 2 shows that with zero NSCs deployed, i.e., when GC is responsible for performing rate and power control for all clients, the statistics-related control plane traffic alone consumes a bandwidth of 1.6 GBit/s. Deploying NSCs helps: With NSCs at the eight most popular locations, the load on the GC is halved to about 800 MBit/s, while the average load on the NSCs is only 111 MBit/s. With our greedy deployment strategy and with only very few (one or two) NSCs deployed in the network, the average NSC load is below 200 MBit/s (easy to handle) while the benefit in terms of reduced GC load is already significant.

4. CONCLUSION

This paper initiates the study of the design of a WiFi SDN where transmissions are optimized through a fine-grained and near-sighted control loop. In particular, we have presented a 2-tiered control plane architecture, called AeroFlux, and sketched our prototype implementation and early evaluation.

Acknowledgments: The authors would like to thank Anja Feldmann for useful inputs. Research supported by the EIT ICT project *Mobile SDN* and Federal Ministry of Education and Research (BMBF) (Reference number 01IS12056).

5. REFERENCES

[1] S. Hassas Yeganeh and Y. Ganjali. Kandoo: a framework for efficient and scalable offloading of control applications. In *HotSDN '12*.

[2] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *ACM SIGCOMM '13*.

[3] S. Schmid and J. Suomela. Exploiting locality in distributed SDN control. In *HotSDN '13*.

[4] J. Schulz-Zander, N. Sarrar, and S. Schmid. Aeroflux: A near-sighted controller architecture for software-defined wireless networks. In *Proc. Open Networking Summit (ONS)*, 2014.

[5] J. Schulz-Zander, L. Suresh, N. Sarrar, A. Feldmann, T. Hühn, and R. Merz. Programmatic orchestration of wifi networks. In *USENIX ATC '14*.