SWAM: SDN-based Wi-Fi Small Cells with Joint Access-Backhaul and Multi-Tenant Capabilities

Matteo Grandi*, Daniel Camps-Mur*, August Betzler*, Joan Josep Aleixendri*, and Miguel Catalan-Cid*
*i2CAT Foundation
{matteo.grandi, daniel.camps, august.betzler, joan.aleixendri, miguel.catalan}@i2cat.net

***Abstract*—In this paper we present SWAM, a system that builds on commodity Wi-Fi routers with multiple wireless interfaces to provide a wireless access infrastructure supporting multi-tenancy, mobility, and integrated wireless access and backhaul. An infrastructure provider can deploy inexpensive SWAM nodes to cover a given geographical area providing on-demand connectivity for Mobile Network Operators. Our main contribution is the design of the SWAM datapath and control plane, which are inspired by the overlay techniques used to enable multi-tenancy in data-center networks. We prototype SWAM in an office wireless testbed and validate experimentally its functionality.**

I. Introduction

Densifying the wireless access is seen as the most promising approach to deliver the capacities required in the future 5G network [1]. Wireless backhauling, based on mmWave radios and technologies operating below 6 GHz, is seen as a key solution to address such a challenge. The multi-tenancy paradigm currently used by cloud providers should be applied to future outdoor Small Cell deployments in order to serve multiple Mobile Network Operators (MNOs), as proposed by the 5G vision of *slicing* [2].

In this paper we present *SWAM*, a system that builds on inexpensive commodity Wi-Fi routers to provide a wireless access infrastructure supporting multi-tenancy, mobility and integrated access and backhaul capabilities. SWAM can be used by an infrastructure provider to provision on-demand connectivity for different MNOs, i.e. *tenants*.

Our main contributions in this paper are: i) The SWAM datapath and control plane, which allow to instantiate a set of per-tenant virtual access points (*vaps*) and backhaul them wirelessly until the tenant home network, ii) a SWAM prototype based on commercial wireless cards and an embedded single board computer running Linux and iii) an experimental evaluation using an indoor office testbed composed of five SWAM nodes, faithfully representing a realistic SWAM deployment.

II. System Design

In a typical implementation, a SWAM node may be equipped with three or four wireless Network Interface Cards (NICs). One or two wireless NICs are used to provide access traffic instantiating *vaps* serving a particular tenant, hence enabling multi-tenancy, while the others are used to provide backhaul functionality. In addition, some nodes, hereafter referred as SWAM gateways, are also equipped with one Ethernet connection to reach the Tenants' Home Network (HN) through tunnel interfaces. Connectivity services, including IP address allocation, are hosted in each tenant's respective HN. Hence, a customer, e.g. STA B1 in Figure 1, connects through the *vaps* of a given tenant, receives an IP address from the tenant's HN, and this IP is maintained while the tenant roams across the network.

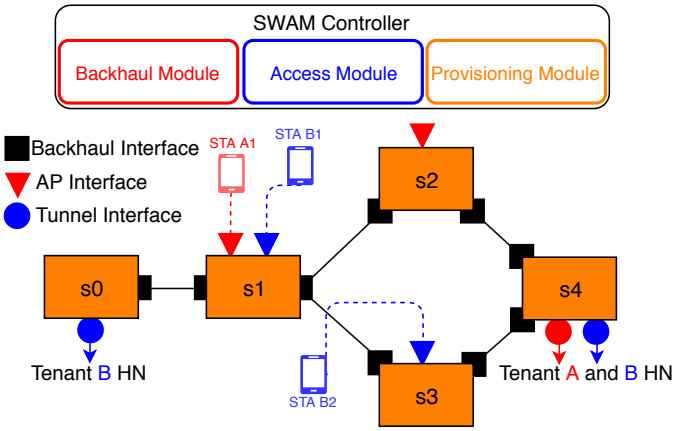
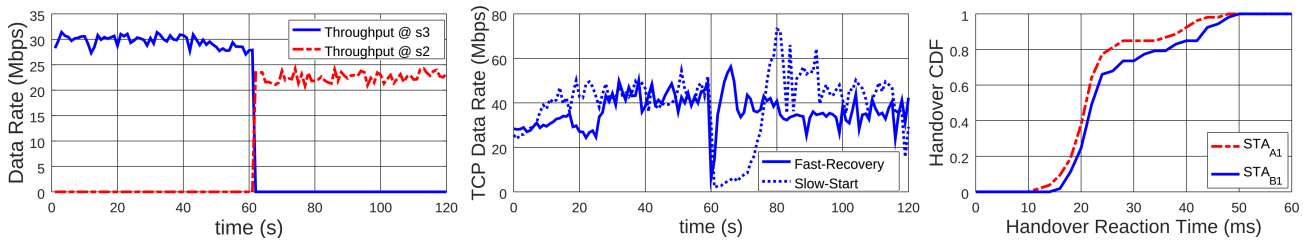


Fig. 1. A set of SWAM nodes in a kite topology instantiate per-tenant *vaps*, depicted with a colored triangle, per-tenant tunnel interfaces, depicted with a colored circle, and backhaul interfaces, depicted with a dark square.

In SWAM, control and management of the access and backhaul resources are accomplished using the SWAM controller illustrated in Fig. 1, which performs three main tasks: i) steering traffic through the wireless backhaul, ii) multiplexing access traffic into the wireless backhaul, and iii) dynamically provision the required virtual interfaces for each tenant.

The goal of the SWAM architecture is to process packets coming from the tenants' customers (*vap* interfaces) and deliver them to the appropriate SWAM gateways through the wireless mesh backhaul. The core idea is to achieve a logical separation between the access and the backhaul. The job of the wireless backhaul is to forward packets along a set of end-to-end tunnels, whereas the job of the access side is to match traffic coming from the tenants' *vaps* to the appropriate backhaul tunnels. In SWAM, a backhaul tunnel is defined using a VLAN tag, and provides an unidirectional connection between two interfaces of a per-tenant access bridge. The SWAM datapath is composed of three software switches, which we describe next:

Per-tenant access bridge: One per-tenant access bridge for each tenant that has presence in that node. The per-tenant bridge manages the following interfaces: i) all the *vaps* for



(a) (b) (c)

Fig. 2. Data collected during the SWAM validation tests, showing (a) the throughput measured during path reallocation, (b) TCP traffic monitored during a gateway relocation, and (c) the CDF of the handover reaction time.

this tenant, ii) any tunnel interface for this tenant, and iii) a set of virtual interfaces connecting to the other SWAM nodes where this tenant has presence. The per-tenant bridges behave as traditional MAC learning bridges.

Integration bridge: It maintains a mapping between each backhaul-facing port of the per-tenant access bridge and their corresponding backhaul tunnel. This allows to map the traffic from the tenant's access and its HNs to the backhaul tunnels. The forwarding table in the integration bridge is populated by the SWAM controller using OpenFlow [3].

Backhaul bridge: The backhaul bridge switches backhaul tunnels based on their VLAN identifier. We adopt the SDN wireless architecture proposed in [4], whereby each potential destination reachable through a physical wireless interface is represented using a virtual Ethernet interface.

In addition, SWAM contains control plane mechanisms that provide three main features: i) avoid loops in the resulting per-tenant overlays, ii) enable the possibility of using multiple gateways for a given tenant to balance load across the wireless backhaul, and iii) support client mobility, whereby when a client device hands over from one tenant *vap* to another, the backhaul tunnels are quickly re-configured appropriately. In this latter case the system is able to maintain the client connectivity while roaming through the network, and connectivity is maintained. The interested reader is referred to [5] for a detailed description of the SWAM datapath and control planes.

III. Experimental evaluation

We validate SWAM in a physical indoor testbed composed of five SWAM nodes built in a logical *kite* topology, and instantiate two tenants with their respective *vaps* and tunnel interfaces connecting to their home network (c.f. Fig. 1). We connect three laptops (clients) to different per-tenant *vaps* of the network. Three consecutive experiments evaluate how the SWAM architecture enables the following features:

SDN-based fast route repair: The first experiment shows how SWAM reacts when a backhaul link carrying an end-to-end tunnel between two per-tenant access bridges breaks. To enable fast recovery, SWAM proactively installs backup paths and reallocates the end-to-end backhaul tunnel. Interested readers are referred to [6] for details on this fast recovery scheme. In average, we measure a path reallocation time of around 246 ms (Fig. 2(a)) during which packets are dropped.

Load balancing mechanisms: In order to alleviate network congestion, the SWAM controller may decide to balance traffic flows adding additional SWAM gateways for a given tenant, and redirecting the traffic from certain *vaps* to this new gateway. We observe that when relocating a gateway, ongoing sessions are only interrupted during 201 ms on average, and ongoing TCP sessions can also quickly recover (Figure 2(b)).

Client handover: The last experiment showcases how SWAM deals with client mobility. In order to determine how fast SWAM can react to a client that switches from one *vap* to another, two cases are analyzed: a case where the client continues to use the same gateway after the handover, and a case where it does not. Figure 2(c) depicts that in both cases the network reaction time is minimal and lies in the range of 20 ms. This demonstrates that SWAM is a very agile architecture, capable of quickly reconfiguring backhaul flows when dealing with client mobility. A detailed discussion on the performance of SWAM can be found in [5].

IV. Conclusions

In this paper we have introduced SWAM, a system supporting multi-tenancy, client mobility, and integrated wireless access and backhaul. SWAM recovers from failures in the wireless backhaul without affecting ongoing traffic, relocates per-tenant gateways in approximately 200 ms, and reconfigures backhaul tunnels upon a handover in about 20 ms.

V. Acknowledgements

This work has been supported by the EC under grant N 762057 (5GPICTURE), and by the Spanish ministry under grant TEC2016-76795-C6-2-R.

References

[1] Qualcomm, *Rising to Meet the 1000x Mobile Data Challenge*. Available at: https://www.qualcomm.com/media/documents/files/rising-to-meet-the-1000x-mobile-data-challenge.pdf

[2] 5GPPP Architecture Working Group, *View on 5G Architecture*, June 2016. Online. Available at: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf

[3] OpenFlow. Available at: https://www.opennetworking.org

[4] Hurtado-Borras, A. et al. (2015, June). SDN wireless backhauling for Small Cells. In Communications (ICC), 2015 IEEE International Conference on (pp. 3897-3902). IEEE.

[5] Grandi, M. et al. (2018, April). *SWAM: SDN-based Wi-Fi Small Cells with Joint Access-Backhaul and Multi-Tenant Capabilities*. Available at: http://arxiv.org/abs/1804.07106

[6] SODALITE Deliverable 3: *Description and initial validation of SDN algorithms, and initial experimental evaluation* Deliverable 6.10 (2015), FLEX Project, FP-7-ICT-2013-10 , Grant Agreement 612050