COAP: A Software-Defined Approach for Managing Residential Wireless Gateways

Ashish Patro, Suman Banerjee
Department of Computer Sciences, Univeristy of Wisconsin Madison
{patro, suman}@cs.wisc.edu

ABSTRACT

In dense residential deployments, such as apartment buildings, neighboring wireless gateways (e.g., Access Points) share the same unlicensed spectrum by deploying consumer-grade access points in their individual homes. In such environments, WiFi networks can suffer from intermittent performance issues such as wireless packet losses, interference from WiFi and non-WiFi sources due to the increasing diversity of devices that share the spectrum. *In this demo, we present an OpenFlow-based vendor-neutral cloud-based centralized framework called* COAP *to configure, co-ordinate and manage individual home wireless gateways using an open API implemented by these gateways.*

1. OVERVIEW

Wireless gateways (e.g., Access Points, wireless Set-Top Boxes) act as the primary mode of internet access in most residential network deployments today. A diverse set of WiFi-capable devices access Internet-based services through these gateways, e.g., laptops, tablets and other handhelds, game controllers (XBox, Wii), media streaming devices (Apple TV, Google TV, Roku), and many more. Given its central role in these home networks, the performance and experience of users at homes depend centrally on efficient and dynamic configuration of these gateways.

Using SDNs for residential Wireless Gateways. Most enterprise WiFi solutions today (including some recent SDN-style efforts [4]) adopt a centralized approach (including vendor-specific cloud based solutions [2]) for managing a set of homogeneous Access Points in a uniform and coordinated manner. We believe that a SDN based approach (using an open API) is important in home environments where each wireless neighborhood has a diverse set of these wireless gateways. Unlike enterprise environments, homogeneity in such environments is likely hard to achieve.

In our proposed service, called COAP (Coordination framework for Open APs), participating wireless gateways (e.g., Access Points) are configured to securely connect to a cloud-based controller (Figure 1). The controller provides all necessary management services that can be operated by a third-party (potentially distinct from the individual ISPs). In the context of large apartment buildings, we envision that the apartment management contract with a single controller service and all residents are asked to utilize the designated controller service within the building. This service would be no different than many other utilities distributed to residents, e.g., water, electricity, etc., which is arranged by the apartment management. Individual residents can also pick different controller services to realize many of our proposed benefits. *However, some advanced features, e.g., interference management and mitigation, are better served if neighboring gateways participate through the same service.*

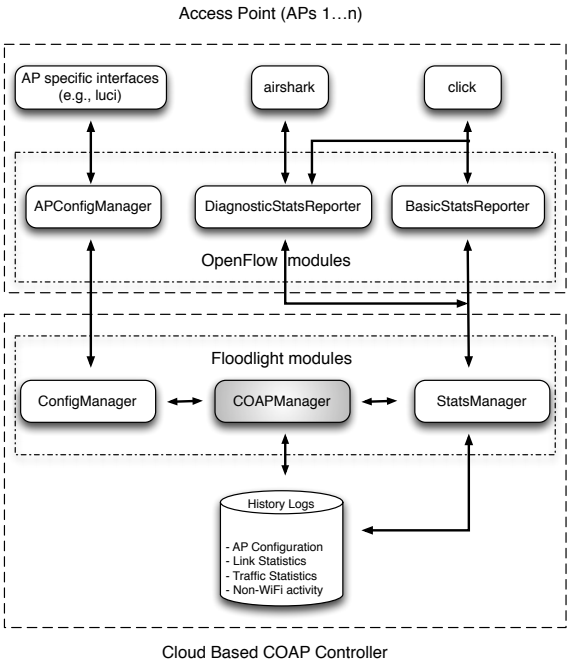

Figure 1: Components of the COAP framework. The different modules implemented by the home gateways (e.g., Access Points) and the controller are shown.

2. FRAMEWORK COMPONENTS AND IMPLEMENTATION

To participate in the COAP framework, home gateways need to expose a vendor-neutral open API to communicate with the COAP controller. In this model (Figure 1), gateways need to implement three different modules that expose a differ-

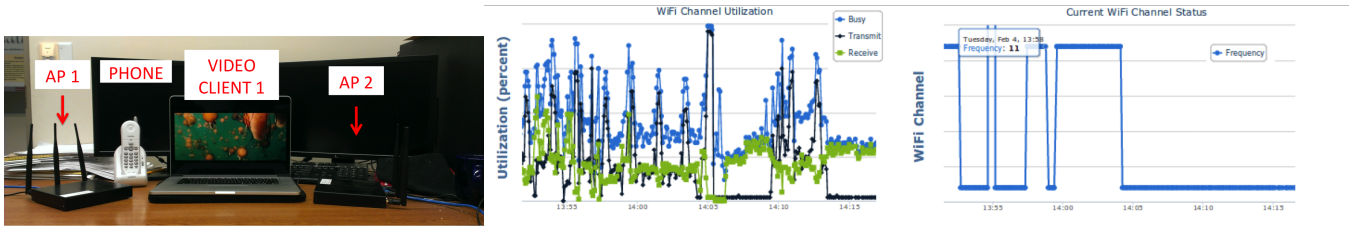

Figure 2: Devices used for the demo (left), Web UI snapshots for timeseries airtime usage (center) and channel (right).

Function	Description
SetAP	Configures the channel and/or transmit power.
ManageAirtime	Manage the airtime access by throttling airtime or slotting the transmissions.
UpdateFlowPriority	Update a flow's priority level to manage contention at the gateway's transmit queue.

Table 1: Examples of configurations performed at the COAP APs.

ent set of functions (related to configuration, diagnostics and statistics collection) to the controller. Following are additional implementation details about the COAP framework.

Controller. The COAP controller is implemented over the Java based open source SDN controller, Floodlight and currently runs on a standard linux server for our deployment. All the COAP controller modules (StatsManager, COAPEngine and ConfigManager) are implemented as modules within Floodlight. The controller uses OpenFlow with COAP-specific protocol extensions to collect data from the gateways as well as apply the configuration updates.

Protocol extensions for OpenFlow. The OpenFlow communication protocol currently consists of capabilities to exchange switch related statistics (e.g., statistics per switch port, flow, queue etc.). We augment this feature to also exchange different COAP related wireless specific statistics (e.g., airtime usage, link performance, non-WiFi activity). We omit the details here for the sake of brevity. To transmit wireless configuration updates to gateways (e.g., switch channel, throttle airtime), we extend the OpenFlow protocol to use a mechanism analogous to the one used to send switch configuration updates (e.g., adding a flow-related rule).

Wireless Gateways. In our current implementation using OpenWrt based WiFi gateways, we use the open-source click [1] framework to implement the WiFi and non-WiFi statistics gathering capabilities of the *BasicStatsReporter* and *DiagnosticStatsReporter* modules. Airshark [3] provides the non-WiFi device detection capabilities using commodity WiFi cards. The OpenFlow module interfaces with Airshark and click as well as communicates with the SDN controller.

We have instrumented the ath9k wireless driver to support APIs related to airtime management. For example, to throttle the airtime access of a gateway, we disable the transmit queue (using the AR_Q_TXD register) to block packet transmissions for the required duration. Across vendors, the underlying implementation of this feature can be driver specific and transparent to the COAP API.

The *APConfigManager* module interfaces with the OpenWrt configuration tool (*luci*) to perform AP level configurations (e.g., channel, transmit power and traffic shaping). This interface can easily be modified for other platforms.

3. DEMO APPLICATIONS

In the demo, we plan to present two motivating applications enabled by the COAP framework using configurations described in Table 1. Figure 2 shows the demo components and couple of UI snapshots: a laptop acts as a WiFi client for video streaming and the OpenWrt based gateways use the COAP API to communicate with the controller. The cordless phone acts as a non-WiFi interferer.

1. **Mitigating non-WiFi interference.** In this scenario, a WiFi link streaming an HD video is interfered by a neighboring cordless phone on the same WiFi channel. The cordless phone's transmissions causes a high degradation of the link's video quality. One or more neighboring COAP enabled gateways detect the presence of the cordless phone and report the instance to the COAP controller. The controller leverages this information to infer link degradation due to the cordless phone's activity and switches the affected link's channel.
2. **Airtime management.** We demonstrate a technique for mitigating WiFi interference by coordinating airtime access between neighboring gateways. In this example, the performance of a WiFi link serving video traffic degrades in the presence of a neighboring link due to excessive airtime utilization and/or packet losses due to interference. The controller mitigates this problem by *scheduling transmissions of the two links in non-overlapping time slots (e.g., 10-20 ms)*. This example motivates one aspect of how co-operation and co-ordination between neighboring gateways can mitigate WiFi interference.

4. REFERENCES

[1] Click modular router. http://www.read.cs.ucla.edu/click/click.
[2] Meraki. Enterprise cloud management. http://www.meraki.com/products/wireless/enterprise-cloud-management.
[3] S. Rayanchu, A. Patro, and S. Banerjee. Airshark: Detecting non-WiFi RF devices using commodity WiFi hardware. IMC '11.
[4] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao. Towards programmable enterprise WLANs with Odin. HotSDN '12.