What’s around me? Location analytics over Software–Defined WLANs

Roberto Riggio*, Iacopo Carreras† and Erinda Jaupaj†
*CREATE-NET, Trento, Italy; Email: rriggio@create-net.org
†U–Hopper, Trento, Italy; Email: name.surname@uhopper.com

Abstract—Software–Defined Networking is gaining increasing interest in the academic and the industrial communities alike. SDN principles call for commodity networking devices and for shifting all intelligence to a logically centralized controller. In this demo we build on a programmable enterprise WLAN platform in order to implement and deploy a location analytics solution to be used in shopping malls, airports, and similar venues for location–based advertisement and visitors profiling. Our solution can run on commodity devices and in many cases can be adapted to run over an existing WLAN infrastructure.

I. INTRODUCTION

Software–Defined Networking (SDN) is reshaping the way networks are controlled and managed opening the way to more flexible and manageable IT infrastructures. At its foundation, SDN relies on two main concepts: (i) control and data plane decoupling; and (ii) high–level programming primitives providing network developers with a programmatic interface to control and configure their SDNs. Similar concepts are also making their way into the wireless networking domain [1]. However, if OpenFlow has emerged as the de–facto standard for packet switched networks, an idempotent solution has yet to emerge for wireless networks. In fact, the flow abstraction on which OpenFlow relies does not account for: (i) the stochastic nature of wireless links (which are not equivalent to ports in Ethernet switches); (ii) the resource allocation granularity (the flow abstraction is too coarse for wireless networks); and (iii) the significant heterogeneity in the link and radio layer technologies (state management for network elements can differ significantly across currently deployed Radio Access Networks technologies). Preliminary programming abstractions for enterprise WLANs have already been proposed by the authors [1], [2]. Such abstractions tackle wireless client state management, resource provisioning, network monitoring, and network reconfiguration. A proof–of–concept controller as well as an SDK exposing the proposed abstractions have also been implemented.

In this demo we take a step forward toward a truly general purpose and programmable IT infrastructure by implementing a location analytics and mobile advertisement platform using our SDK and by deploying it on top of our programmable WLAN controller. This Network App, targeting shopping malls, airports, and similar venues, aims at computing statistics such as the average time spent by a visitor in a certain area, returning visitors, and hot–zones. Such information can then be leveraged for targeted advertisements and/or user profiling. As opposed to commercially available indoor localization solutions, which rely on proprietary hardware and/or controllers, our framework supports any WiFi Access Point (AP) capable of running an open firmware (e.g. OpenWRT) while our WLAN controller can run on any platform capable of running a Python interpreter (Linux, MAX OS, Windows, etc.).

II. SYSTEM ARCHITECTURE

We named our programmable enterprise WLAN platform EmPOWER [3]. An high level view of the EmPOWER system architecture is depicted in Fig. 1. The network is composed by a variable number of Wireless Termination Points (WTPs), i.e. the WiFi APs that form the WLAN providing clients with wireless connectivity. The SD–RAN Controller can run multiple virtual networks or slices on top of the same physical infrastructure. A network slice is a virtual network with a specific SSID and its own set of WTPs. Clients can opt–in a certain slice by associating to its SSID. Each Network App is instantiated in its own slice of and can only change the state of the clients in that slice. Network Apps exploit the programming primitives trough either a RESTful interface or a native Python API. Finally, the controller ensures that a Network App is only presented a view of the network corresponding to its slice.

III. CHANNEL QUALITY MAP

In this section we summarize the main features of the Channel Quality Map abstractions which is leveraged in this demo in order to implement the location analytics Network App. The Channel Quality Map abstraction provides network programmers with a full view of the network state in terms of channel quality between clients and WTPs. The Channel Quality Map is exposed to the network programmer by means of two data structures: the User Channel Quality Map (UCQM) and the Network Channel Quality Map (NCQM). Both are 3–dimensional matrices where each entry is the channel quality
over a certain frequency band between a client and a WTP in
the case of the UCQM, and between two WTPs in the case of
the NCQM. For example, the code below periodically queries
the specified WTP for its neighboring stations.

```python
ucqm({
    'addr': 'ff:ff:ff:ff:ff:ff',
    'block': ['04:00:21:09:f9:96', 36, L20]
}

Listing 1: UCQM query creation.
```

From the implementation standpoint, a monitor interface is
created on top of each physical radio available at each WTP
in the network. The RSSI readings reported by the wireless
driver for each decoded frame are then used as a measure of the
interference between the transmitter and the WTP. For each
For each neighbor within the decoding range, the
WTPs computes the average of the RSSI over windows of 500ms, an
exponential moving average ($\mu_{ema}$) and $N$–points
smoothing moving average ($\mu_{sma}$) are also maintained.

The query is executed periodically with the period set by the
`every` parameter (in ms)$^1$. Similarly, the RSSI
from neighboring WiFi Access Points can be tracked using
the `ncqm` primitive. In the above example specifying
`ff:ff:ff:ff:ff:ff` will return the RSSI of any station
within the decoding range of WTP 04:F0:21:09:F9:96 on the
legacy channel 36 (i.e., an 802.11a channel).

A sample output of the `ncqm` primitive is reported below.
In this case the station a0:d3:cl:a8:e4:c3 is a neighbor
of the WTP 04:F0:21:09:F9:96 on the 802.11a channel
36. The report includes, besides the previously described
averages, also the total number of frames received since
as a proximity radius (in m) to determine the
proximity of each WTP to the wireless client. The
Network App computes a set of aggregated statistics, namely, the average time spent by
visitors, and the most visited areas. During the demo we
will show real–time statistics from a 20 nodes deployment at
CREATE-NET premises (a 5–stories office building). A single
WTP setup will be staged during the demo showing real–time
statistics gathered from the demo floor.

## IV. Demo

Modern location–based applications and services rely on the
possibility to know in real–time the geographical position of
customers. While GPS–based localization can provide precise
and real–time geo–localization, its reliability drops dramatically
in indoor settings. Several indoor localization solutions
leveraging various technologies (WiFi, Bluetooth, acoustic,
etc.) are currently commercially available. While some of
them are characterized by sub–m precision, their cost could
be prohibitive for many deployments. Moreover, for several
use cases proximity based localization is sufficient instead of
precise indoor geo–localization. By proximity detection, we
refer to the capability of knowing if a certain wireless client
is within a few meters from an anchor point (a WTP in this
case). Notice that the assumption here is that anchor points are
deployed in close proximity of points of interests in a certain
venue, such as check–in desks or shops in an airport.

The RSSI tracking capabilities allowed by the Channel
Quality Map can be effectively leveraged to implement such a
proximity detection system. A simple RSSI tracking Network
App has been implemented as proof–of–concept. The Network
App tracks in real–time the RSSI of wireless clients at different
WTPs in the network. The Network App then uses the following
metrics in order to compute the proximity information:

- **Strength**, the average RSSI level observed in the last ob-
  servation window: WTPs that measure high RSSI values are
  considered to be closer to the wireless client.
- **Stability**, the standard deviation of the RSSI in the last
  observation windows: WTPs that experience less stable
  signals provide a less accurate proximity information.
- **Consistency**: WTPs that consistently reported RSSI mea-
  surements from a given client are consider to provide a
  more accurate proximity information.
- **Visibility**, the number of WTPs reporting RSSI measure-
  ments: receiving RSSI measurements from several WTPs is
  consider to reduce the accuracy.

The Network App exploits these metrics to build a list of
WTPs ordered in decreasing level of proximity (from the
closest to the furthest). For each WTP a proximity radius (in
m) is also reported: very close (< 6m) and close (< 10m).
Starting from this information, the Network App computes
a set of statistical metrics, namely, the average time spent by
visitor in proximity of each WTP, the number of returning
visitors, and the most visited areas. During the demo we
will show real–time statistics from a 20 nodes deployment at
CREATE-NET premises (a 5–stories office building). A single
WTP setup will be staged during the demo showing real–time
statistics gathered from the demo floor.

## REFERENCES

1. R. Riggio, K. M. Gomez, T. Rasheed, J. Schulz-Zander, S. Kulkinski, and
   M. K. Marina, “Programming Software–Defined Wireless Networks,” in

   in Software–Defined Wireless Networks,” in *Proc. of IEEE IM*, Ottawa,
   Canada, 2015.


$^1$Specifying `every = –1` will result in a single query being issued.