Joule: Software–Defined Energy Metering

Roberto Riggio, Dejene Boru, and Tinku Rasheed
CREATE-NET, Italy

***Abstract*—In this demo we present *Joule*, a software enabling real–time power consumption monitoring of networking devices without having to deploy physical meters in the form of, for example, managed Power–Over–Ethernet supplies. We will demonstrate how *Joule* can estimate the power consumption of commercial WiFi Access Points by exploiting observable network statistics and under different workloads, e.g. browsing, video streaming, and background FTP traffic. An interactive web dashboard will allow attendees to have a first hand experience on the correlation between traffic and power consumption.**

I. Introduction

According to [1] it is envisioned that by 2015 wireless access networks alone will be responsible for 90% of the carbon footprint of the *entire* cloud networking ecosystem. Reducing the energy consumption of wireless access network becomes thus crucial in order to ensure the long–term sustainability of these technologies. The first steps toward this goal are (i) to acknowledge each component contribution to cloud energy footprint; and (ii) to devise energy efficient wireless networks and change how they are managed. Real time monitoring of the power consumption of each network element can provide operators with an increased insight into the relationship between network traffic and energy consumption, supporting an energy efficient evolution of their infrastructure. Nevertheless, deploying network–wide energy consumption monitoring solutions is often not viable due to the setup and management costs.

Drawing on these needs we designed and developed *Joule* a virtual energy consumption meter capable of estimating in real–time the power consumption of networking devices by jointly exploiting easily observable network statistics and empirical power consumption models [2]. In this demo we will demonstrate *Joule* operations over a small WiFi network and using different workloads, e.g., browsing, video streaming, and background FTP traffic. An *interactive* web dashboard will allow attendees to have a first hand experience on the correlation between traffic and power consumption.

II. System Architecture

This demo is built on top of *EmPOWER* [3] an SDN platform[1] for research and experimentation on programmable wireless and mobile networks. Following an SDN philosophy, *EmPOWER* allows new features and services to be deployed as applications. Network developers can exploit a set of high–level primitives to both control and query the status of the network. Such primitive are designed is such a way to relieve the developers from the implementation details specific to the underlying wireless technology, such as, in the case of a WiFi network, the association/de–association mechanisms, control frame exchange and status management. Network virtualization allows each application to be executed in a sand–boxed environment providing isolation and security. *EmPOWER* supports off–the–shelf WiFi Access Points and is currently being extended to include programmable wireless base stations (LTE and LTE-A).

[1]Online resources available at: http://empower.create-net.org/

978-1-4799-0913-1/14/$31.00 © 2014 IEEE

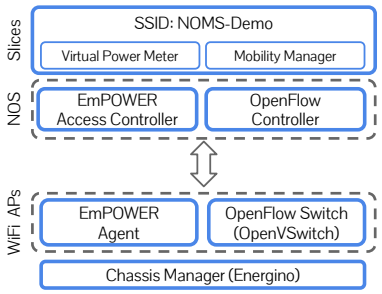


Fig. 1: The *EmPOWER* system architecture.

A sketch of the *EmPOWER* system architecture is provided in Fig. 1. As it can be seen, *EmPOWER* consists of a logically centralized Access Controller and multiple Agents running on each AP. The Access Controller, has a global view of the network in terms of clients, flows, and topology. Network services run within logically isolated slices and have full control of the resources allocated to them. From a end–user perspective each slice effectively corresponds to a WiFi network advertising its SSID (*NOMS–Demo*, in the figure).

Two applications running within the same slice have been implemented for this demo. The *Virtual Power Meter* gathers the network statistics and estimates the power consumed by the AP in a given observation period. The *Mobility Manager* handovers new clients to one of the available APs.

A. Implementation Details

From a technical standpoint, an *EmPOWER* application consist in a Python module that is dynamically loaded by the Access Controller at bootstrap. A REST interface is also available. In this demo we exploit the following primitives:

- Packets and bytes counters. Used to track the traffic generated and received by a certain wireless client. Traffic statistics are aggregated by frame length.
- RSSI triggers. Generate a callback to an user defined method when a condition has been met. Triggers are evaluated at each AP by the Agent.

Upon initialization, the *Mobility Manager* creates a new RSSI trigger in order to detect new clients joining the network. The trigger is created using the following statement:

```
T = rssi(relation='GT',
         value=-90,
         sta='ff:ff:ff:ff:ff:ff')
T.callback = handle_new_client
```

This trigger fires the first time a frame coming from any client is received by an AP with an RSSI higher than -90 dB (a value slightly above the detection threshold for WiFi

interfaces). The `handle_new_client` callback takes care of handing over the client to a random AP. Triggers are generated once when the condition is met and do not repeat while the condition remains verified. Consequently if it is necessary to be notified when the condition is no longer verified another trigger with the opposite conditions must be created. Once the handover has been performed, the *Virtual Power Meter* creates two periodic queries gathering the necessary traffic statistics:

```
C1 = packets_count(bins=[50,500,1460,8192],
                   sta=new_client)
C2 = bytes_count(bins=[50,500,1460,8192],
                 sta=new_client)
```

The queries above basically ask the controller to pool the AP `new_client` is communicating with and to gather the its statistics. Such statistics consists in the number of packets and bytes transmitted and received by the station aggregated by frame length into the specified bins. In the example above, 4 bins are defined (50, 500, 1460, and 8192 bytes). The *Virtual Power Meter* periodically polls the above counters in order to obtain the requested statistics, e.g.:

```
>>> C1.tx_samples
[10,  1, 500, 0]
```

This means that since joining the network, the station transmitted 10, 1, 500, and 0 packets with a frame length L, such that, respectively, $L \leq 50$, $50 < L \leq 500$, $500 < L \leq 1460$, and $1460 < L \leq 8192$. Similarly, $C2$ provides the number of bytes transmitted.

B. Power Consumption Models

In this demo we reuse the power consumption models developed in our previous work [2]:

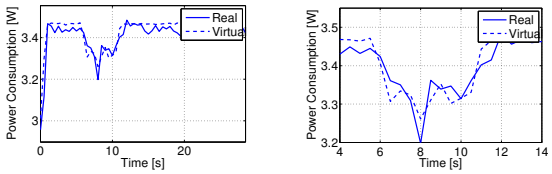
$$P = \alpha(d)sat(x,d) + \beta(d)\delta(x) + \gamma \tag{1}$$

where P is the AP power consumption in Watts, x is the offered load in Mbps, d is the datagram size in bytes, $\delta(x)$ is the step function, and

$$sat(x,d) = \begin{cases} x & 0 \leq x \leq x_{max}(d) \\ 0 & x \leq 0 \\ x_{max}(d) & x > x_{max}(d) \end{cases} \tag{2}$$

$$\alpha(d) = \alpha_0(1 + \alpha_1/d) \tag{3}$$

In this model, $sat(\cdot)$ represents the power consumption saturation regime reached when the offered load exceeds network capacity. $\alpha(d)$ captures the fact that the rate of increase in power consumption with offered load is observed to depend on datagram size. $\beta(d)$ captures the dependence of power consumption on datagram length when the offered load is fixed. γ captures the baseline power consumption when the AP is idle (no traffic besides the standard WiFi beacons). The dependence of $\alpha(d)$ on datagram size is primarily due to the contribution of fixed overheads (framing, contention, etc.), α_0 and α_1 are parameters. The specific values of all the model's parameters are determined during an initial training phase during which the system being modeled, the WiFi AP in this case, is subjected to a set of well designed workloads meant to represent all — or at least most of — the system's operating states and at the same time power consumption measurements are taken using suitable metering devices.



(a) Entire download. (b) During slowdown.

Fig. 2: Real (solid line) and virtual (dashed line) power consumption while download a large file using FTP.

By combining the statistics gathered by *EmPOWER* and (1) we can estimate the energy consumed by the AP. Notice that this is possible due to the fact the power consumption is highly correlated with traffic [4], [5].

III. Demo Scenario

The demo setup will consist of three APs and three Energino [4] power meters connected via a switch to a laptop acting as network controller. The APs are embedded boards running OpenWRT and equipped with one Ubiquiti SR71-A 802.11n interface. Each AP runs the Agent while the laptop runs the Access Controller. We will have a few wireless clients consisting of laptops, tablets, and mobile phones. The demo will display the following aspects of our system:

- Using a web based dashboard we will show in real–time the power consumption of the three APs as measured by the physical power meter and as estimated by *Joule*.
- We will start the download of a large file using FTP from one of the wireless clients and we will show how the virtual power meter closely tracks the physical one in the power consumption figures.
- We will force the wireless client to perform an handover in order to demonstrate how *Joule* can promptly react to network changes.
- Attendees will be able to use the wireless clients in order to perform normal Internet tasks, e.g. browsing the web, watching a video on YouTube, etc. allowing them to have a first hand experience on the correlation between traffic and power consumption.

Figure 2 reports the instantaneous power consumption of the AP while downloading a large file from a public server. The graph refers to one particular run where an artificially generated network slowdown happened between the 5th and the 15th second of the download. As it can be seen, the virtual power meter closely tracks the actual power consumption during the slowdown. The median values of the actual and estimated power consumption across the 10 runs are, respectively, 3.437W and 3.446W. The median absolute error is 28mW.

References

[1] "The Power of Wireless Cloud: An analysis of the energy consumption of wireless cloud," CEET, Tech. Rep., June 2013.

[2] Roberto Riggio and Douglas J.Leith, "A measurement-based model of energy consumption in femtocells," in *Proc. of IEEE WD*, 2012.

[3] Roberto Riggio et al., "EmPOWER: A Testbed for Network FunctionVirtualization Research and Experimentation," in *Proc. of IEEE SDN4FNS*, 2013.

[4] R. Riggio et al., "Energino: energy saving tips for your wireless network," in *Proc. of ACM SIGCOMM*, 2012.

[5] K. M. Gomez et al., "Energino: An hardware and software solution for energy consumption monitoring," in *Proc. of WinMee*, 2012.