Energy Consumption of Networked Embedded Systems

Nadir Cherifi, Gilles Grimaud, Thomas Vantroys, Alexandre Boé

Energy Consumption of Networked Embedded Systems

Nadir Cherifi
Worldline e-payment services
IRCICA/Cristal, CNRS UMR 9189,
Univ. Lille 1, France
Email: nadir.cherifi@worldline.com

Gilles Grimaud
Thomas Vantroys
IRCICA/Cristal, CNRS UMR 9189,
Univ. Lille 1, France
Email: gilles.grimaud@univ-lille1.fr
thomas.vantroys@univ-lille1.fr

Alexandre Boe
IRCICA/IEMN, CNRS UMR 8520,
Univ. Lille 1, France
Email: alexandre.boe@univ-lille1.fr

***Abstract*—The Web of Things (WoT) defines the idea of addressing and requesting any surrounding connected device through web applications. These devices are for a large part, tiny, wireless and mostly battery-powered. Therefore, energy represents a critical limiting resource for their large scale deployment in real life applications, such as smart cities or smart buildings. Consequently, the ability to measure, track and finely profile energy consumption of such devices and correlate it with the application driving the device functionalities is a big challenge to improve the development of WoT embedded applications. In this paper, we present our recent ongoing works on energy consumption measurement of networked embedded applications for the WoT area. More particularly, we focus on measuring energy consumption of the well-known Contiki Operating System HTTP web server using two measurement methods, a simulation based method and a real world software based measurement method. Doing this, we quantify and model the gap existing between the simulation and the real world regarding network embedded applications, such as embedded web servers. Afterwards, we describe our aims in using a real world hardware energy consumption measurement method and profile a home-made embedded web server prototype called Smews. This latter could theoretically improve performances and power consumption of WoT applications relying on TCP/IP protocol.**

I. Introduction

We call Internet of Things (IoT) the idea or the description of a system where surrounding devices in the physical world are connected directly, without a translation gateway, to the traditional internet via wireless or wired connections. Achieving this objective needs to introduce IP inside the connected network devices, allowing a real interoperability at the network layer. The Web of Things (WoT) extends this idea a step further by adding the internet interoperability at the application layer. WoT promotes the use of Web services for the application level in order to fully integrate device network systems with existing IT systems and allow them to evolve as the application use case grows.

Focusing most on battery-operated and wireless connected devices, energy consumption and efficiency become among the main concerns in the software development of these devices. In addition, this latter takes a critical role when speaking about WoT, due to the use of traditional existing internet protocols which have been designed without power-efficiency in mind. Therefore, the design of energy-efficient embedded software for the WoT needs a fine-grained power consumption profile of the devices in order to detect the hot-points of consumption and correct them. Regarding power consumption measurement of tiny networked devices, many estimation and measurement methods are proposed. We argue for a real execution conditions measurement methods, permitting us to capture most of the device activities in real life execution conditions. This paper contributes to give an energy consumption profile of the Contiki OS embedded HTTP web server. We show and explain the energy usage induced by using an application layer protocol such HTTP relying on TCP and its guaranteed delivery principals. Then, we experimentally show and quantify, using numerous network configurations, that the gap existing between a real environment and a simulated one can be really significant.

The paper is structured as follow. First, in Section II, we give the necessary background concerning power consumption measurement methods and highly constrained embedded communication stacks. in Section III, we describe the experimental methodology for our work in progress about power consumption. Then, in Section IV, we present and discuss the experimental results of this work. Finally, in Section V, we conclude this paper and give an overview of our further works.

II. Background

Performing an energy profiling of a networked embedded system requires a good knowledge in various domains. Among them, energy measurement techniques take, of course, the great part because of the importance of correct energy measurement figures. Furthermore, knowing the wireless sensor world, we can easily admit the fact that most of the sensor energy consumption is caused by the radio transceiver, Thus, a good understanding of the concepts which cover the different communication stack layers driving the radio module, represents the main way to explain energy consumption hot-points in wireless connected devices.

Acknowledgements: This work has been partially supported by Worldline e-payment services company.

A. Energy measurement methodology

Energy consumption of low-power wireless devices is difficult to measure. We can globally divide measurement and estimation methods into three distinct categories: hardware based, software based and simulation based.

Concerning hardware measurement, most of the techniques rely on measuring the voltage drop over a shunt resistor (high precision resistor) in series with the target device to profile. Based on that, the data acquisition is performed using a digital oscilloscope or a dedicated board as it is the case for FlockLab [1], which is a testbed for timing and power consumption profiling which uses the on board GPIO of target devices to record logical events and correlate them to the energy drain curve. Alternative hardware methods of using shunt resistor exist. One of them use special capacitors, called GoldCaps [2], with very large capacities (1 Farad) for powering the target device instead of using a traditional battery. This enables short-term experiments permitting a prediction of the device overall lifetime.

Some of hardware measurement platforms could be embedded directly on the target device, giving this latter the possibility of measuring *in-situ* its own energy consumption. This knowledge allows the device to perform energy-aware functionalities. ICount [3] adds energy metering for free (no hardware overhead) to systems that have a built-in DC-DC boost converter in their power supply and a dedicated hardware counter (Eg. timer) by just counting the switching cycles of the regulator. Quanto [4], takes advantage of the on-board energy meter ICount and instruments the underlying device OS to track the energy consumption of different device activities. Despite the obvious advantage of these on-boards energy metering modules, their low accuracy makes them unsuitable for fine-grained energy consumption profiling.

In contrast with hardware measurements methods, software based techniques do not require any additional materials to estimate the energy drain of a device. Powertrace presented in [5] tracks system power states by measuring the time during which the device components (CPU, Radio, Flash, etc.) are in each power state (CPU Active/Sleep, Radio TX/RX, etc.). Device drivers are instrumented to record a timestamp when a component enters a new state. When the component leaves its state, the time difference is computed, then added to the global corresponding component time. Like on-board measurement hardware, Powertrace lacks of precision when it is used to profile a precise, fine-grained and local device activity.

A last category of measurement methods takes the choice of estimating device energy consumption by simulation. We generally call them, power simulator. PowerTOSSIM [6] is a power simulator that extends the initial TOSSIM, an event-driven simulation environment for TinyOS applications. It instruments each peripheral components in each state, and employs a code-transformation technique to estimate the number of CPU cycles executed by each node, avoiding the need to expensive instruction-based simulation. In contrast, E-Simu [7] chooses to model device energy consumption by taking into account energy per instruction and per instruction switching. E-Simu also adds a characterization of several peripherals. This allows to model the energy profile of an entire platform. Power simulators represent good tools for quick energy consumption estimation, as they could easily be integrated in the development processes. However, power simulators fail to provide fine-grained estimations of devices in real world execution conditions, due to the unpredictable hardware energy activities and radio interferences to name only a few. Finally, simulators are based on platform architecture models, which is for most of the time, very difficult to construct, develop and validate.

B. Energy and MAC

MAC layer is the last software layer of the communication stack before the device radio driver on typical connected sensor nodes. It drives the activity of the radio in order to perform several and various objectives. When targeting energy consumption, the radio transceiver embedded by every wireless devices is certainly, the most energy consuming component [8]. Therefore, to save and manage energy, wireless connected objects power off their radios as often as possible [9]. MAC Protocols used to these aims are called power-saving MAC protocols or Radio Duty Cycle (RDC) protocols.

Radio must be efficiently duty cycled in order to reach a good trade-off between energy consumption and latency. To reach this goal, many power-saving MAC protocols were proposed [9], [10], [11]. ContikiMAC is a duty cycle MAC protocol [8]. It uses a periodic wake-up scheme, where multiple copies of the radio packet to be sent are transmitted as a wake-up strobe. ContikiMAC has three important energy optimization features. First, ContikiMAC uses a power-efficient wake-up mechanism that relies on very precise timing between transmissions, thanks to the real time library of the underlying operating system. Second, it implements a fast-sleep feature allowing the radio to quickly go back to sleep with an efficient discernment between a real packet transmission and radio noise. Finally, ContikiMAC implements a phase-lock mechanism maintaining at the node level a list of neighbours and wake-up phases, allowing a sender to know about the wake-up moments of its neighbour receivers. Thus, a sender can begin its successive transmissions just before the receiver is expected to be awaken, saving a great amount of energy.

C. Energy and IP

The rise of IoT leads to a new approach consisting in using traditional IP as the dedicated protocol for tiny objects. This induces new issues on various points and especially concerning the energy overhead, because IP protocol was not originally designed with energy efficiency in mind. Despite these issues, the embedded IP approach was proven, to be feasible by multiple scientific works [12], [13]. uIP and lwIP [14] proposed by Adam Dunkels, represent the first real consistent works leveraging the use of IP through WSN radio mediums. These works propose various approaches for solving the IP-on-WSN energy overhead. IP Header overhead and IEEE 802.15.4

bandwidth limitation which both impact latency and energy consumption of the device, were solved by applying to the headers compression techniques. Finally, lwIP and uIP propose lightweight IP routing approaches that allow to mitigate the network workload on the high constrained wireless sensor nodes.

More recently, with IPv6 being promoted as the next global network protocol for the new Internet, the Internet Engineering Task Force (IETF) proposes 6LoWPAN [15] in order to enable large IPv6 packets over constrained and low power wireless networking as IEEE 802.15.4. Compared to IPv4, header length are doubled in IPv6, and therefore, header compression was used to minimize the energy wasted on transmission. 6LoWPAN specification defines for this aim various compression mechanisms targeting different ratio and time compressions. Network packet fragmentation schemes were also proposed to support, in a fast way, the 1280 Bytes MTU of IPv6 over the constrained IEEE 802.15.4 radio medium. Regarding implementations, SICSlowpan (for Contiki OS) [16] and b6loWPAN or blip (for TinyOS) [17] represent the most mature 6LoWPAN implementations at that time. They support header compression and fragmentation, including a mechanism for stateless auto-configuration, simplifying and alleviating the IPv6 address assignment.

D. Energy and TCP

On lossy and low power wireless networks, the User Datagram Protocol (UDP) is often preferred due to its low overhead. Nevertheless, for being compliant with some historic application protocols, such as HTTP or SSH, and to fit to prerequisites of various application types (Medical, Military, etc.), packet delivery reliability is needed. For this goal, adapting the heavy Transmission Control Protocol (TCP) on lossy networks has been the center of various works. In [18] Adam Dunkels et al. present a distributed TCP caching mechanism in order to avoid an end-to-end retransmission. In this scheme, each device on the wireless route is able to cache a single TCP segment enabling single-hop retransmissions. Other approaches focus on the reduction of acknowledgement (ACK) number by leveraging the delayed ACK mechanism included in most of traditional TCP/IP stacks [19]. The main idea of these approaches is to delay the transmission of an ACK until the reception of multiple TCP segments. For example, in [20], the authors studied the impact of the delayed ACK mechanism on wireless sensors, and proposed a dynamic one to improve throughout and energy consumption over the whole network.

III. Methodology

By the use of two energy estimation techniques, we want to quantify the differences between simulation-based and real world software-based methods, then show experimentally the weaknesses of a simulation environnement. We present in this section, the methodology used to achieve this objective.

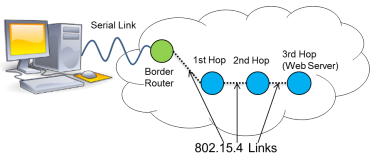

Fig. 1: Test Bed description.

A. Test Bed Description

We perform our experiments on the Tmote Sky platform. It embeds a 16-bit MSP430 MCU and a low power CC2420 radio chip compliant with the IEEE 802.15.4 standard. As a test bed, we set up as shown in Figure 1, a small, linear 3-hop network with static routes, in order to avoid inadvertently measuring the effect caused by a dynamic routing protocol. Another Tmote Sky implements an IPv6/IPv4 border router which is used to connects the IP sensor network to a laptop running Linux. This Tmote use Serial Line IP (SLIP) to send and receive IP packets to/from the Laptop. The 802.15.4 node radio is configured to use the channel 15, which underlies more WiFi interferences. For measuring energy footprint of the Contiki's HTTP embedded web server, we run various experiments using our test bench. Explicitly, the laptop initiates an HTTP request asking for a payload ranging from 1 to 1040 bytes (which is the maximum payload size usable with IPv6/6LoWPAN on our hardware platform) from a HTTP web server located at the third hop node, the two nodes remaining are only used to route the requests and responses. We use the Linux CURL command line tool to launch the HTTP requests.

B. Test Bench Configurations

We decide to run multiple experiments using two Contiki network configurations. We use uIPv4 and uIPv6 stack implementations in Contiki OS. For uIPv4 We use the default Contiki Maximum Segment Size (MSS) of 48 Bytes. In contrast, concerning uIPv6, we use first a MSS of 1220 Bytes, enabling in the same time the fragmentation process of the Contiki's 6LoWPAN layer represented by the SICSlowpan implementation. In a second time, we choose to bypass the 6LoWPAN layer by fixing the MSS at 48 bytes as for the uIPv4 stack. Regarding other TCP parameters, we use the default parameters of Contiki. The uIPv4 and uIPv6 Contiki's software implementations do not include any TCP energy-specific optimization. So, there are no TCP distributed caching and the delayed acknowledgement is not used. Every single TCP segment needs to be acknowledged.

C. Test Bench Measurement Environment

To quantify the motes energy consumption, we run our tests using two different environments : in the first, we deploy our test bed in a real environment represented by our office. Then we use the Contiki Cooja Simulator which gives us an ideal experiment environment, avoiding all kind of interferences. To characterize and compute the energy consumption for every node on the route when processing a HTTP request, we use the Contiki's built-in energy profiler Powertrace [5].

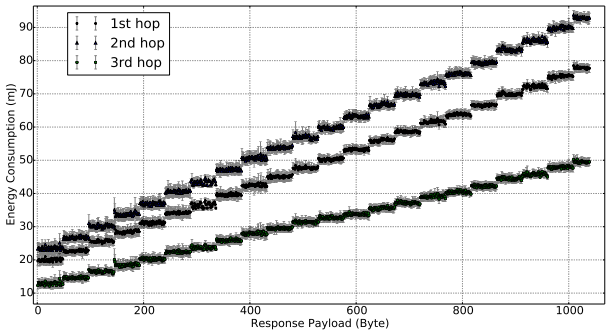

Fig. 2: Simulation with IPv4: Energy consumption of every node involved in the request. 3rd hop (green) represents the requested server, 1st (red) and 2nd (blue) hops represent the routers.

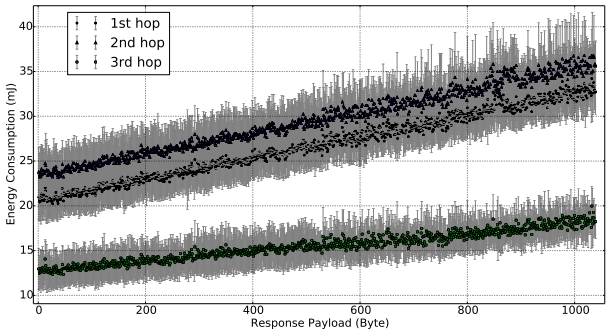

Fig. 3: Simulation with IPv6: Energy consumption of every node involved in the request. 3rd hop (green) represents the requested server, 1st (red) and 2nd (blue) hops represent the routers.

Particularly, every node triggers two Powertrace timestamps, one at the beginning of a request by detecting the TCP opening connection flag (SYN), and the second at the end by detecting the TCP closing connection flag (the last FIN-ACK flag). The results displayed in all the figures are averaged over 50 runs and error bars, when displayed, are in gray color and show the standard deviations.

IV. Experimentations

In this section, we propose to experimentally quantify the gap which exist between the simulation and the real world, and highlight the deficiencies of the simulation-based environnement.

A. Simulation

We first ran the previously described experiment in a simulation environment, using the Contiki Cooja simulator. Also, the 3-hops motes implements as a RDC protocol, the 8 Hz frequency ContikiMAC. We use both IPv4 and a full MSS IPv6 as transport layer. Figure 2 and Figure 3 shows the energy consumption according to the payload size of the 3 nodes involved in the request (3rd hop represents the server node. 1st and 2nd hops are routers) for both IPv4 and IPv6 experiment

We can first notice, in both experiments, the differences between energy consumption of every motes involved in the request. The two routers are more energy intensive than the server due to their two sides transmissions and receptions. Particularly, the first hop router is less energy consuming than the second hop due to the RDC protocol of the neighbours nodes. The border router having its radio always on, the first hop node - in contrast of the second hop - doesn't loose energy in transmitting multiple strobes waiting for the awakening of this latter.

Concerning the IPv4 experiment, we can easily notice, on every node, the step-wise increment of the energy consumption as the number of packets increase (In our configuration, every 48 bytes). Thus, the energy consumption is more related to the number of transmitted IP packets than the number of bytes. In addition, the standard deviation describing the variation on every run, is small and constant (about 1.5 milliJoules) trough all the different payload sizes. This permit us to simply quantify and model the energy consumption behaviour using a linear function model. We note E_{v4_i} the energy consumption, in milliJoules (mJ), when using IPv4 for the i-hop node on the route. The x variable represent in all the next equations, the payload size in bytes. We give as example the global equation for the 1st-hop node (eq 1). For the other nodes, we find $\alpha_2 = 3.5$, $\beta_2 = 22.7$, $\alpha_3 = 2$ and $\beta_3 = 12$

$$E_{v4_1} = \alpha_1 \lceil \frac{x}{48} \rceil + \beta_1 = 3.5 \lceil \frac{x}{48} \rceil + 19 \quad (1)$$

In IPv6 experiment, the energy consumption is not really linked to the number of IP packets or radio fragments, but more with the number of transmitted bytes. We remind that, the use of a large MSS of 1220 bytes and the 6LoWPAN layer, permits the web server and the nodes on the route to transmit all the requested payload sizes in a single IP packet. This feature induces a lower global energy consumption, and a light increase of this latter according to the payload size, comparing to the use of the IPv4 protocol. However, the standard deviation is similar to IPv4, low and constant (almost 2.5 mJ). We can model and quantify the energy consumption by mean of a linear function. We note E_{v6_i} the Energy consumption, in mJ, when using IPv6 for the i-hop node on the route. We give as example the global equation for the 1st-hop node (eq 2). For the other nodes, we find $\gamma_2 = 0.012$, $\varphi_2 = 23.2$, $\gamma_3 = 0.006$ and $\varphi_3 = 12.7$

$$E_{v6_1} = \gamma_1 x + \varphi_1 = 0.010x + 20.3 \quad (2)$$

B. Real World

To show the real world impact on the energy consumption, we perform the same experiments as in the simulation case, in our office with real nodes. Figure 4 and Figure 5 shows the energy consumption according to the payload size of the 3 nodes involved in the request for both IPv4 and IPv6 experiments, respectively.

Focusing on the IPv4 case, we can conclude that unlike the simulation based experiments, the energy consumption

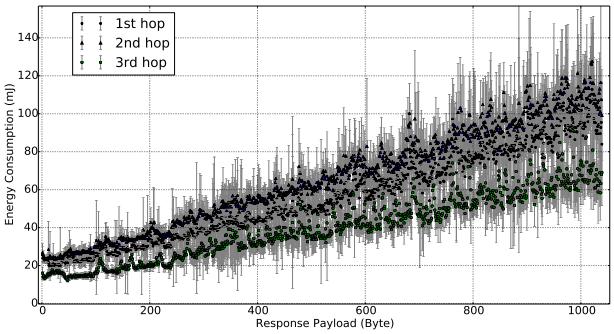

Fig. 4: Real World with IPv4: Energy consumption of every node involved in the request. 3rd hop (green) represents the requested server, 1st (red) and 2nd (blue) hops represent the routers.

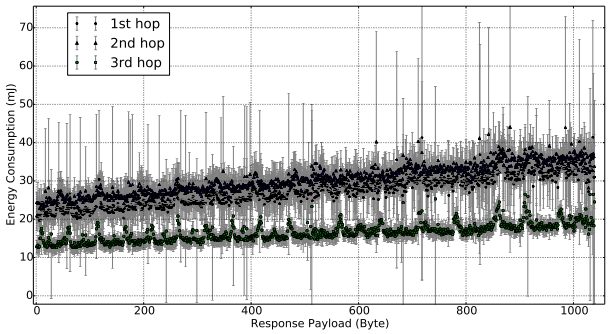

Fig. 5: Real World with IPv6: Energy consumption of every node involved in the request. 3rd hop (green) represents the requested server, 1st (red) and 2nd (blue) hops represent the routers.

behaviour in the real world, is more related to the payload size than discretely to the number of transmitted IP packets. First, the energy consumption is higher in the real world experiment by a non negligible amount. Furthermore, the standard error deviation is larger when compared to the simulation based one, following a non constant function curve. Without taking into account, at a first stage, the high variation of the energy drain on every node, we can approximate this latter by a linear function. We give as example the global equation for the 1st-hop node (eq 3). For the other nodes, we find $\lambda_2 = 0.084$, $\psi_2 = 23.8$, $\lambda_3 = 0.049$ and $\psi_3 = 13.6$

$$E_{v4_1} = \lambda_1 x + \psi_1 = 0.073x + 21 \tag{3}$$

As quoted, due to a large energy consumption variation, the predictive potential of our linear model in the real world case, is not as powerful as for the simulation based case. Figure 6 shows the standard error deviation of the energy consumption on the second hop node. We show only the second hop node of our test bed, which represents the most frequent case encountered on nodes within a traditional WSN. The standard deviation grows according to the requested payload size, reaching a really high value of 36 mJ at 1000 bytes. Moreover, the values of the dispersion is increasingly

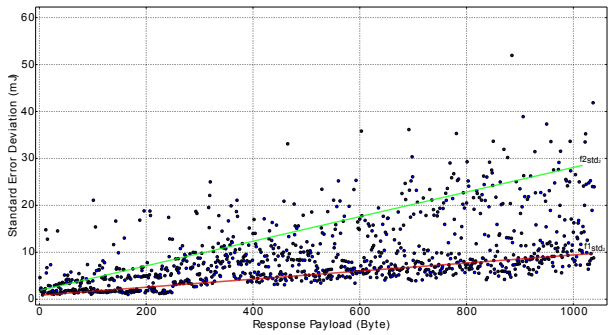

Fig. 6: Real World: Standard error deviation of energy consumption in IPv4 on the 2nd-hop node. $f1_{std_2}$ (red) and $f2_{std_2}$ (green) approach the two sets of the standard deviation.

significant trough the increase of the payload size, and we can observe that the standard deviation is split into two sets, highlighted by the straight lines, $f1_{std_2}$ and $f2_{std_2}$ on the Figure 6. We use this behaviour, in order to approach the IPv4 energy consumption standard error deviation. We note Std_2 the standard error deviation, in mJ, of the 2nd-hop node (eq 4) and $f1_{std_2}$ and $f2_{std_2}$, the two linear functions printed on Figure 6.

$$Std_2 = \begin{cases} f1_{std_2} = \eta_a x + \mu_a = 0.01x + 1 \\ f2_{std_2} = \eta_b x + \mu_b = 0.028x + 1.6 \end{cases} \tag{4}$$

In summary, working on the real world rather than on the simulated one, requires from the WoT application developer, to handle with an increasing and unpredictable system energy consumption as the transported payload by the WSN nodes grows. This unpredictable and high error deviation on the node energy usage is mainly caused by the interferences found in real execution conditions. Indeed, our experiments were run in an office with WiFi interferences causing a highly varying radio link quality. Furthermore, these interferences increase the packet loss rate during the experiment, causing, in the case of uIPv4 Contiki implementation a highly energy penalizing end-to-end transport layer retransmissions all over the network route.

For IPv6 experiment, the nodes energy consumption behaviour described in Figure 5 is not as perfect as for the IPv4 experiment. However, the values of this latter are quite similar to the simulated case, and could be predicted and modelled using nearly the same previously presented IPv6 simulation model. Concerning the standard error deviation described in Figure 7, despite the presence of several high values near the 20 mJ limit, and a small dispersion bringing the energy consumption variation at almost 10 mJ, the global variation of the experiment is practically constant. The coherence between simulated and real tests can be explained by the energy efficiency of the IPv6 layer, and particularly, the 6LoWPAN adaptation layer which leads to low energy communication primitives for large packets, using the 802.15.4 burst sending feature. This feature also permits to globally reduce the bad

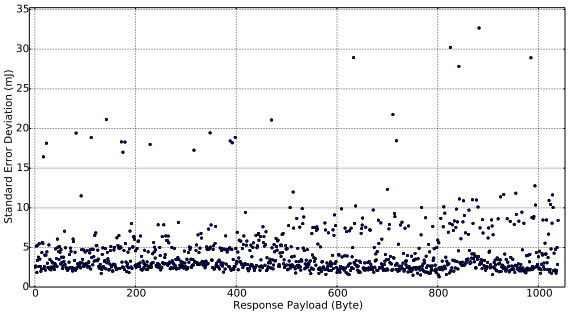

Fig. 7: Real World: Standard error deviation of energy consumption in IPv6 on the 2nd-hop node.

effect of surrounding interferences on radio fragment loss. Specifically, the possibility to send really large packets through the radio medium reduces logically the probability of IP packet loss and so, the probability of network end-to-end retransmissions. In addition, a radio fragment loss has a low impact on both energy consumption because of the 6LoWPAN layer, which will handle this loss with a node-to-node retransmission manner.

V. Conclusion and Future Work

In this paper, a global energy profile of the embedded ContikiOS HTTP web server has been presented. We experimentally evaluated this web server on the energy efficiency metrics with using different network configurations and deployment environments. We showed that, the simulation cannot be a substitute to a real life scenario. To name only few reasons, we quoted WiFi interferences which greatly impact the radio link quality of IEEE 802.15.4 channels, inducing overheads and a non negligible variations on energy consumption, specifically when using uIPv4 with large packets. Finaly, we give approached models wich permit an embedded developer to easily predict the global energy consumption of his WoT application.

In future work, we plan to explore two important points. The first, is focused on the set up of a real hardware-based measurement method, that will allow us to finely profile energy consumption of wireless devices, with catching the real world condition impacts. Moreover, this would lead to a fine grain source code analysis, giving developers the ability to track and detect hot energy consumption points in their embedded applications. The second, is focused on the enhancement of the latency and energy consumption of embedded HTTP web servers. We plan to adapt on Contiki OS, a home-made web server for constraint devices, called Smews [21]. Smews web server argues for the idea of adapting stack behaviour to application contents properties, by setting up various strategies fitting with each web content type. In addition, it has been proven to be more efficient than the state of art in term of request latency and memory charge. Smews has the ability to manage multiple TCP Segments in fly, enabling a quick TCP stream and fewer acknowledgement numbers thanks to delayed acknowledgement, and TCP window mechanism. Smews was initially designed to work with Ethernet and SLIP links. Consequently, a great work has already begun and is expected to adapt it to the Contiki OS to work on a lossy and low throughput radio mediums. Although, we have strong hopes that Smews web server could dramatically improve the latency and the energy efficiency of Web of Things applications.

References

[1] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2013.

[2] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso, "Experimental evaluation of lifetime bounds for wireless sensor networks," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, 2005.

[3] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler, "Energy metering for free: Augmenting switching regulators for real-time monitoring," in *Information Processing in Sensor Networks (IPSN), 2008.*, April 2008.

[4] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, "Quanto: Tracking energy in networked embedded systems," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2008.

[5] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level power profiling for low-powerwireless networks," SICS, Tech. Rep., 2011.

[6] V. Shnayder, M. Hempstead, B.-R. Chen, and M. Welsh, "Power-TOSSIM: Efficient Power Simulation for TinyOS Applications," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[7] N. Fournel, A. Fraboulet, and P. Feautrier, "Embedded software energy characterization: Using non-intrusive measures for application source code annotation," *Journal of Embedded Computing*, Jul. 2009.

[8] A. Dunkels, "The contikimac radio duty cycling protocol," 2011.

[9] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2004.

[10] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2006.

[11] D. Moss and P. Levis, "Box-macs: Exploiting physical and link layer boundaries in lowpower networking," Tech. Rep., 2008.

[12] J. W. Hui and D. E. Culler, "Ip is dead, long live ip for wireless sensor networks," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*. New York, NY, USA: ACM, 2008.

[13] C. Westphal, "Layered ip header compression for ip-enabled sensor networks," in *IEEE International Conference on Communications (ICC)*, June 2006.

[14] A. Dunkels, "Full TCP/IP for 8 Bit Architectures," in *Proceedings of the First ACM/Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*, San Francisco, May 2003.

[15] G. Mulligan, "The 6lowpan architecture," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, New York, NY, USA, 2007.

[16] A. Dunkels, "Sicslowpan - internet-connectivity for low-power radio systems," SICS, Tech. Rep., 2008.

[17] J. Granjal, E. Monteiro, and J. S. Silva, "Network-layer security for the internet of things using tinyos and blip," *International Journal of Communication Systems*, 2014.

[18] A. Dunkels, T. Voigt, and J. Alonso, "Making TCP/IP Viable for Wireless Sensor Networks," in *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session*, Berlin, Germany, Jan. 2004.

[19] R. Braden, "Requirements for internet hosts – communication layers," 10 1989, rFC 1122.

[20] E. Altman and T. Jimnez, "Novel delayed ack techniques for improving tcp performance in multihop wireless networks," 2003.

[21] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, "Serving embedded content via Web applications: model, design and experimentation," in *Proceedings of the International Conference on Embedded Software (EMSOFT)*, Grenoble, France, 2009.