A Fog-enabled IoT Platform for Efficient Management and Data Collection

Pavlos Charalampidis, Elias Tragos, Alexandros Fragkiadakis
Institute of Computer Science
Foundation for Research and Technology-Hellas (FORTH-ICS)
Heraklion, Crete, Greece
email: {pcharala, etragos, alfrag}@ics.forth.gr

***Abstract*—One of the most promising emerging technologies, Internet-of-Things, refers to the interconnection of thousands (or even millions) of smart objects, supporting a large number of applications like environmental monitoring, smart agriculture, e-health, etc. Research groups in both the academia and industry, have proposed and/or developed a significant number of IoT architectures and platforms, however not focusing on platform management-related issues. Furthermore, the vast amount of the sensory data and the rapid proliferation of the smart devices (i.e. sensors) make a new approach regarding efficient data collection and storage inevitable. Fog Computing (FC) is an environment where data are stored and pre-processed before transmitting them to the cloud, having a number of advantages like scalable real-time services, fault detection and isolation, enhanced security and privacy, etc. In this work, we present a fog-enabled IoT platform used for sensory data collection, presenting several metrics that can be used as the basis for a Management-Platform-as-a-Service, able to efficiently monitor the IoT platform and predict potential failures.**

I. Introduction

Internet-of-things (IoT) generally refers to the interconnection of things (e.g. sensors, smart phones, home appliances, etc.). Several IoT platforms like IoT-A [1], BUTLER [2], iCORE [3], etc., have been proposed addressing issues like security, performance, energy-efficiency, etc. The evolution of these platforms, however, created isolated systems with limited inter-operability. A new generation of IoT projects (i.e RERUM [4], [5], FIESTA [6], INTER-IoT [7]) provided solutions to security-by-design, inter-operability, and several as-a-service capabilities like platform-as-a-service, testbed-as-a-service, etc.

Despite these advances, and by considering the rapid proliferation of the IoT platforms and the smart devices, there is the need for proper storage and processing of vast amount of data generated and collected by distributed IoT networks. Fog Computing (FC) is an environment where data are stored and pre-processed before transmitting them to the cloud, so that computation, storage and networking services can be performed locally. FC has a number of advantages, within the IoT concept, like scalable real-time services, fault detection and isolation, enhanced security and privacy, mobility support, geo-distribution, location awareness, low latency, etc.

In this paper, we present a three-layer IoT architecture that employs fog-enabled gateways (GWs). More specifically, the presented architecture consists of: (i) sensor nodes (SNs) that lie on the network edge, (ii) fog-enabled GWs that aggregate sensory data and perform several other operations like SN registration, etc., and (iii) an IoT middleware used as back-end cloud. The proposed architecture is capable of collecting not only sensory data but management data as well, like the device uptime, the energy consumption of the SNs, etc. This can be used as the first step towards the implementation of a Management-Platform-as-a-Service with inter-operability capabilities. This service will support the collection of various management-related data like network statistics, energy consumption, health status of the SNs, aiming to detect and predict potential failures at all layers of the infrastructure, thus making feasible a self-healing IoT platform.

Although cloud computing and storage has been used as a solution to support dynamic scalability in several IoT applications, the deployment of a large number of nodes in future smart cities needs among others, location-awareness and low latency, requirements that can be satisfactorily met with FC. In [8], the importance of fog-cloud interplay and the role of FC in the context of IoT is highlighted, while in [9], a programming model to support large-scale IoT applications through mobile FC is proposed, built with an eye on service provisioning to geographically distributed, latency-sensitive applications. Furthermore, the authors in [10] explore the suitability of FC to serve applications in the IoT context and perform a comparative analysis between fog and cloud computing in terms of resource consumption metrics. All these works are mainly focused on the principles and basic notions of FC. On the contrary, in this work we are presenting a real implementation and deployment of a fog-enabled IoT system. In [11], a hierarchical distributed FC architecture for big data analysis in smart cities, along with a prototype for smart pipeline monitoring, is presented. Other extensive deployments that focus on service provisioning and experimentation infrastructures for smart-cities applications are described in [12], [13], [14]. Although sharing common ground with them, we are also focusing on metrics that can be used as the basis for a Management-Platform as-a-Service.

The rest of the paper is organised as follows. In Section II we provide an overview of the IoT platform describing its main

software and hardware components. The evaluation metrics and the corresponding results are presented in Section III. Finally, conclusions and further work appear in Section IV.

II. System overview

Conceptually the IoT system presented here follows a three-tier hierarchical architecture comprising of three distinct components. These are: (i) the resource-constraint IoT SNs that lie on the network's edge, equipped with various sensors that produce data characterised by locality, (ii) the GW that plays the role of the fog layer, positioned in proximity to the SNs and autonomously processes data aggregated by the nearby network edge, and (iii) the MW that plays the role of the back-end cloud responsible for virtualisation, data storage and service provisioning. Next, we describe in more detail the components lying in all three tiers.

A. IoT Sensor Nodes

The IoT Sensor Nodes used here are resource-constraint devices (in terms of processing power, memory and storage) responsible for real-time sensor sampling and measurements communication. They are built around a Zolertia Re-Mote platform [15] that hosts an ARM M3-Cortex running at 32 MHz, 32 KB RAM and 512 MB of Flash Memory and can operate in two frequency bands, namely ISM 2.4 GHz and ISM 863-950 MHz under the IEEE 802.15.4 standard. A number of analog and digital sensors is attached to the Re-Mote platform and is able to sense variables describing e.g. weather conditions (ambient temperature, relative humidity, ambient light), air quality (PM10 and gases concentration), noise, etc. An example of a physical device can be seen in Figure 1.

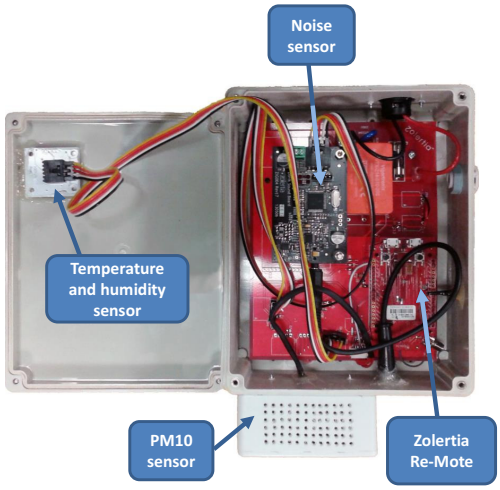

Figure 1: Sensor device

Regarding the software part, the SNs host the Contiki OS that is tailored to networked, resource-constrained and low-power wireless IoT devices. On the application layer, the Constrained Application Protocol (CoAP) [16] is used for communication with the GW. CoAP is a constrained web protocol specialised to M2M requirements based on the request-response model and supporting UDP transport with application layer reliable unicast. Thus, the measurements collected through the sensor drivers are exposed as CoAP resources by a CoAP server running on the SN. Apart from the sensory measurements, self-monitoring resources that report network statistics, device hardware/software info and power consumption, are also exposed.

B. IoT Gateway

The IoT Gateway (GW) plays the role of the bridge between the SNs and the MW and is responsible for offering functionalities, such as SN registration and management, network and protocol translation, measurements aggregation and forwarding to the MW. Essentially, it hosts two different network interfaces. On the one side, there is an IEEE 802.15.4 interface offered by a Zolertia Re-Mote that acts as a border router and enables connectivity to the SNs. On the other side, connectivity to the MW is provided by the Ethernet or WiFi interface of a Raspberry-Pi 3 running Raspbian OS.

The GW has fog characteristics as it is used to register/re-register SNs and also collects and aggregates data from the SNs. More specifically, it ensures that the registration of the devices to the MW is performed in an easy, transparent and adaptive way. In particular, a CoAP server running at the GW plays the role of the registrar that handles registration messages received from the SNs, stores necessary identity information (eg. ID, IPv6, etc.) in a local database and forwards registration messages to the MW. Furthermore, the GW implements a mechanism for per device data collection activation/de-activation by translating appropriate HTTP requests received from MW into CoAP requests for registration/de-registration to CoAP asynchronous notifications' mechanism (OBSERVE). Additional security and reliability enhancing techniques include connectivity between the MW and the GW realised using a VPN connection, as well as local logging and monitoring that ensures the data transmission the MW is not disrupted by i.e. a reset of the GW or a reset of the devices.

C. IoT Middleware

The IoT Middelware (MW) acts as the back-end where services are invoked, and data streams are managed. Specifically, it performs functionalities for virtualisation, data processing and service provisioning, along with additional security and privacy related ones. The implementation of the MW is based on the middleware developed as part of the EU-FP7 project OpenIoT [17], leveraging its open-source nature and design choices that follow the IoT-A architecture. Its functional components include: (i) the Service Manager that is responsible for handling the service requests from the applications by identifying the Virtual Entities (VEs) that are of interest for the application and matches them to predefined templates existing

in a local registry, (ii) the Generic Virtual Object (GVO) manager that creates and manages the digital representations of the Sensor Devices, (iii) the Federation Manager that creates and manages federations of SNs, which provides advanced services to the end-users, i.e. for enabling devices to cooperate by performing service composition and orchestration, and (iv) the Data and Context manager that handles and processes data gathered from the SNs in order to send them to the applications by tailoring them to their needs (i.e. performing averages, filtering data, etc.).

III. Deployment evaluation

The IoT system described in Section II was deployed in several geographically distinct installation sites in the city of Heraklion, Crete, Greece. The installations were performed in both indoor and outdoor spaces (municipal buildings, parks and squares). In each site, a set of SNs along with a GW were installed and the measurement types depicted in Table I were collected with the reported rate. Here, we present results on the evaluation of the system in terms of the device availability, secure transmissions and energy efficiency.

Measurement type	Rate (secs)
Current	30
Humidity	120
Ambient light	30
PM10	60
Power consumption	10
Barometric pressure	120
Gas	30
Weather	30
Noise	30

TABLE I: Measurement types and their corresponding rates

A. Availability of the devices

1) Uptime: The uptime of an SN is defined as the time since it has started operating, expressed in seconds. The uptime is made available through a COAP resource, and here it is leveraged to monitor SNs availability. By availability we consider the time SN is up and running, and not necessarily if the SN is properly functioning or if it is transmitting data. The uptime can be affected only by: (i) shortage of power supply in case of battery depletion or power outage, (ii) hardware reset when there is a bug in SN's firmware, and (iii) software reset requested by the GW for proper network initialisation. If any of the above happens, the uptime resets to zero, and then starts counting again. Figure 2 shows an example of the uptime collected by one of the SNs.

Observe that there are periods of time when no observations exist. This is because the client that issues the COAP requests for the uptime collection operated periodically. Furthermore, observe that the uptime is not increasing as expected because two software resets were issued by the GW. In general, if the

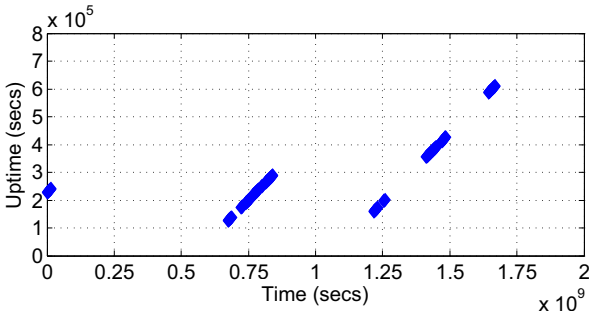

Figure 2: SN uptime when two software resets are issued

SN has not rebooted, then the uptime increases monotonically, thus when no reboot has taken place, the availability of an SN can be computed using the *expected uptime* Up_{exp}, defined as follows:

$$Up_{exp}(i) = Up_{i-1} + (T_i - T_{i-1}), \quad (1)$$

where Up_{i-1} is the uptime provided by the SN during observation $i-1$, T_i is the current timestamp, and T_{i-1} the timestamp referred to the previous observation.

Now assume that the COAP observations arrive at times T_i, with $i \in [1, N]$, and M is a subgroup of consecutive observations. To define SN's availability, we compute within each subgroup, the difference between the reported uptime and the expected one:

$$\Delta_i = Up_i - \alpha \times Up_{exp}(i), \quad (2)$$

where α is the constant that defines a tolerance on the expected uptime, to compensate for delays due to congestion in the network or overloading of the SN. The availability of the SN after M consecutive observations is now defined as follows:

$$Av_M = 100 \times \left(1 - \frac{f(\Delta > 0)}{M}\right), \quad (3)$$

where $f(\cdot)$ is a decision function that counts the number of times Δ is greater than zero. Referring again to the SN with the two reset requests, its availability for $M = 10$ and $\alpha = 0.8$ is shown in Figure 3. When the SN reboots, its availability drops from 100% to 90%.

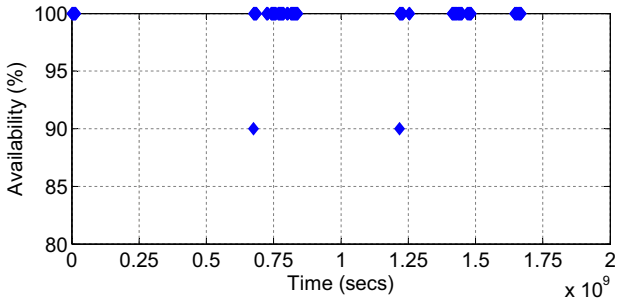

Figure 3: Uptime-based SN availability with two resets and $M = 10$

2) Inter-arrival time of measurements: As stated before, by using the device uptime as an evidence of availability cannot guarantee that the SN is properly functioning and that it is transmitting data. In order to overcome this limitation, we investigate the availability of the devices based on the inter-arrival time (IAT) of the collected measurements (e.g. ambient temperature, light, gas, etc.). During the measurements collection, one would expect that the IAT of each measurement would be equal to its expected rate. This is not always true because packet collisions and delays (which are highly possible in wireless environments with significant interference) can significantly increase IAT.

Assume that IAT_e is the expected IAT for a single type of measurement. If IAT_r denotes the IAT measured during an experiment, then $d = \frac{IAT_r - IAT_e}{IAT_r}$ gives the difference between IAT_r and IAT_e. A small d denotes that the specific measurement suffers no high delays or packet losses. At this point, we define the device availability (DA) as $Av = 100 \times (1-d)$. Next, we report DA for measurement traces collected from the GWs in one indoor (IN-1) and one outdoor (OUT-1) installation. We note here that DA is computed at GW level, meaning that we first compute the DA for each separate measurement, and then we aggregate them into a single vector for computing an aggregate empirical cumulative density function (CDF).

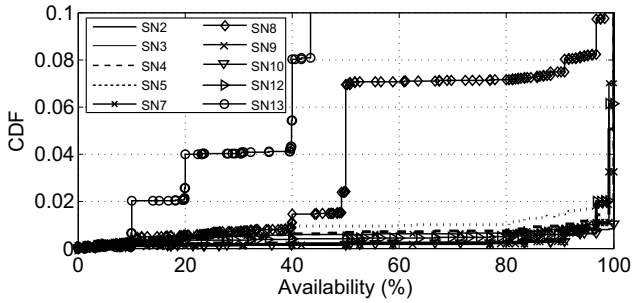

Figure 4: CDF of data availability in IN-1

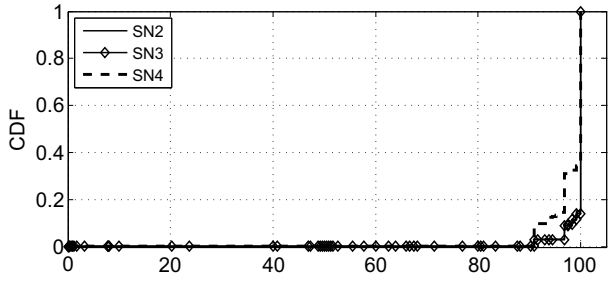

Figure 5: CDF of data availability in OUT-1

Figure 4 shows the data availability for ten SNs deployed in the IN-1 installation. All data from the devices, except those from SN8 and SN13, have a very high availability (over 90%). Data from SN8 and SN13 have a lower availability because they are located in rooms where multi-hop connections with the GW are created, so the delay increases. The data availability in OUT-1 installation is depicted in Figure 5. In this case, observe that for all SNs, the availability stays over 90%.

B. Compression and encryption using Compressive Sensing

Compressive Sensing (CS) acts as a lightweight mechanism for simultaneously compressing and encrypting a block of sensory samples collected by an SN. Specifically, it computes a reduced dimensionality set of linear projections of the sampled data vector on a random subspace, through the multiplication of the data vector with a random matrix, known as *sensing matrix*. This set of random projections is further transmitted to the GW, where the original samples are recovered by solving an appropriate ℓ_1-norm minimization problem. As a result, it is possible to: (i) decrease the number of packets transmitted by the SN that is further translated to savings in transmission energy and (ii) achieve confidentiality of transmitted data that is based on the random nature of the sensing matrix and that comes at limited extra cost [18].

Here, we evaluate the performance of CS with regards to the energy savings in two different installations. Like before, we report results of one indoor (IN-2) and one outdoor (OUT-2) installation. At each site, we use four SNs to measure ambient temperature and power consumption values (as reported by the *powertrace* power profiler [19]) that are exposed as resources by the CoAP server of each device. The first device (SN1) provides uncompressed samples of ambient temperature (denoted as NON-CS) while the other three (SN2, SN3, SN4) employ CS with compression rate (CR) of $50\%, 75\%$ and 87.5%, respectively. The temperature sensor is sampled every 30 seconds and the CS-enabled SNs use a Bernoulli sensing matrix for encrypting the acquired data (in blocks of 64 samples), and transmit them to the GW. The decryption of the CS measurements is performed at the GW, by using the OMP algorithm [20] that is proven fast and computationally efficient. The acquisition of all data covers a period of 24 hours, during which approximately 2900 samples of ambient temperature and 8600 of power consumption were collected.

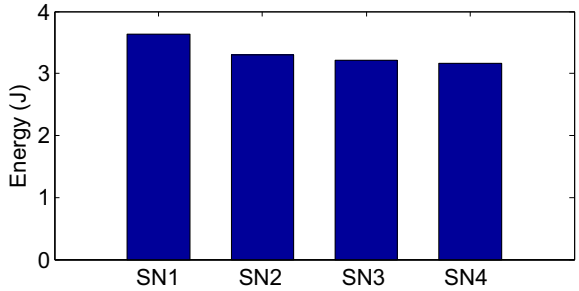

Figure 6: Total transmission energy in IN-2

Figure 6 and Figure 7 illustrate the total transmission energy of each device, for installation IN-2 and OUT-2, respectively. As expected for both sites, the employment of the CS scheme reduces the required transmission energy, compared to the

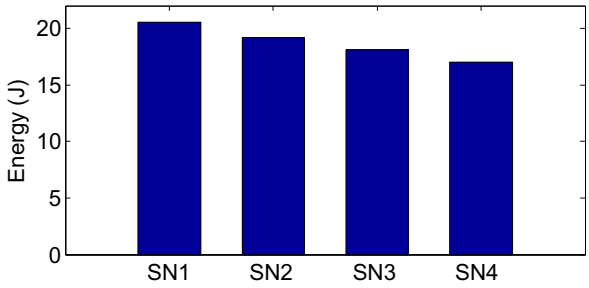

Figure 7: Total transmission energy in OUT-2

NON-CS case. Additionally, by increasing the CR, leads to an increased energy benefit, since as CR increases, less packets are transmitted from the SN, thus its RF transceiver is draining less energy. After 24 hours, transmission energy savings of almost 12% and 18% are achieved at best (SN4) for IN-2 and OUT-2, respectively. It is noted that, although considerable, we would expect the energy savings to be more pronounced in the absence or limitation of background traffic, which here exists in the form of power consumption reporting packets. Finally, the difference in the total energy consumption between the two installations is easily explained by considering the corresponding difference in the data rates. The indoor installations operate at 250 Kbps, while outdoor ones at nominally five times smaller rate, namely 50 Kbps. In the second case, since the RF transceiver remains active for a longer time in order to transmit the same data volume, it is normal to consume more energy, compared to the faster first case.

C. Network security using DTLS

Datagram Transport Layer Security (DTLS) [21] is used for securing network traffic in a way that does not depend on reliable message transfer. By incorporating a mechanism that allows packet retransmissions and reordering, whenever it is necessary, it can be used with unreliable datagram transport, like UDP. Thus, it can be bound to CoAP and act as a security protocol for authentication, automatic key management and data encryption between IoT devices, in a transparent way for end applications. CoAP over DTLS is termed as secure CoAP (CoAPs).

Here, we evaluate the energy efficiency of DTLS protocol by using two SNs to measure ambient temperature values exposed as CoAP resources by each device. The first device (SN1) sends measurements over the non-secure CoAP protocol, while the second one (SN2) sends measurements using CoAPs. A client running at the GW observes the temperature resource for both devices and logs measurements every 30 seconds. TinyDTLS [22] implementation (an optimized implementation of DTLS v1.2 for embedded devices) is leveraged at the SN side, providing the necessary DTLS functionality on the server side. The cipher suite selected uses a pre-shared key (PSK) with AES operating in CCM mode with 8-byte long authentication tags (`TLS_PSK_WITH_AES_128_CCM_8`), since this is the most efficient in terms of the operational cost. The gathering of the measurements was performed over a period of 14 hours. Like before, we measure the transmission power consumption of each device using *powertrace*. The power consumption is exposed as a CoAP/CoAPs resource by the SN, which the client software running at the GW observes with an interval of 10 seconds. Thus, a total number of around 5000 traces is logged.

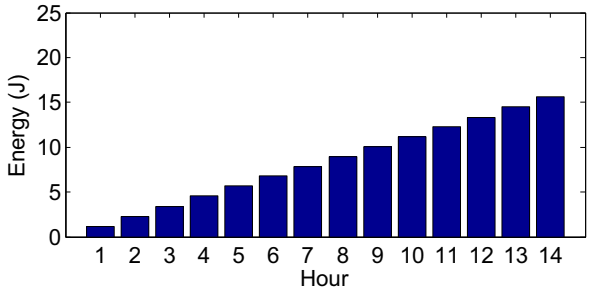

Figure 8: Cumsum of transmission energy (CoAPs - SN1)

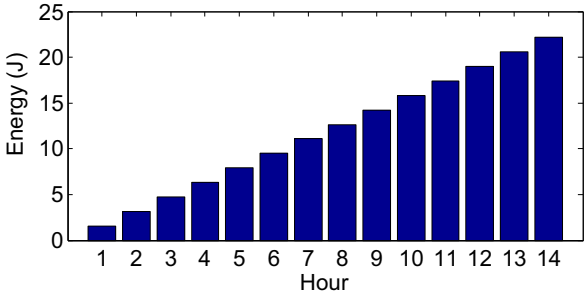

Figure 9: Cumsum of transmission energy (CoAPs - SN2)

In Figure 8 and Figure 9 we present the cumulative sum (cumsum) of energy consumption for transmission of the acquired measurements over CoAP and CoAPs, respectively. It is noted that the cumulative values are computed after the transmission energy, for each hour of the day, by the summation of the aggregated *powertrace* values that refer to the *Transmit* operation. Thus, each bar in the plot refers to the total amount of energy spent for transmission till the end of the corresponding hour. By comparison of the figures, we conclude that, the data encryption provided by DTLS comes at a cost in the energy consumption. This is obvious during the entirety of the acquisition period. In particular, after 14 hours, SN1 that transmits unencrypted measurements has consumed 15.6 J, while SN2 that sends measurements over CoAPs has consumed 20.1 J, which translates to an increase of almost 29%.

Next, we investigate the overhead DTLS creates over an insecure CoAP connection in terms of the round-trip time (RTT). We assume that a secure connection is already established by a handshake and the communicating parts execute the DTLS record protocol to ensure the confidentiality and integrity of the application data. It is noted that although the

DTLS handshake essentially adds to the total overhead, it is only executed at the beginning of a DTLS session that can remain active for a long time (several hours). Thus, in this analysis, we do not consider handshake latency and quantify the DTLS overhead by measuring and comparing the RTT of a CoAP request for the secure and the insecure connections. In particular, for each device (SN1 and SN2) we repeatedly transmit, every ten seconds, a confirmable CoAP POST request with a size equal to 59 bytes, and report the minimum, average and standard deviation of the request RTT after 500 repetitions. We execute the experiment for both frequency bands the SNs are able to utilize, namely the 868 MHz and 2.4 GHz.

	868 MHz			2.4 GHz		
	min	**average**	**st. deviation**	**min**	**average**	**st. deviation**
CoAP	52	1199	730	26	1196	739
CoAPs	213	1237	799	130	1261	725

TABLE II: Round-trip time (ms) for CoAP and CoAPs

Observe in Table II that the overhead of CoAPs compared to that of CoAP, is by average only 38 ms for the sub-GHz band and 65 ms for the 2.4 GHz band. Interestingly though, in the minimum numbers of the RTT the difference between CoAP and CoAPs is much higher. Nevertheless, these results show that adding security to the connection is not costly once an active session has been established, hence a session should be kept alive as long as possible to avoid repeated handshakes.

IV. Conclusions

In this paper we presented a fog-enabled three-layer IoT platform, capable of collecting not only sensory data but management data as well, like the device uptime, the energy consumption of the SNs, etc. This can be used as the first step towards the implementation of a Management-Platform-as-a-Service with inter-operability capabilities. The evaluation results, based on the data collected from the SNs, show that the uptime metric can be efficiently use for characterising the availability of the SNs, that can be further used to detect potential failures, misconfigurations and network-related problems like bandwidth limitations and interference. In addition, the data availability measurements that are computed based on the inter-arrival time of the collected measurements can reveal similar inefficiencies.

The energy consumption measurements show that a significant amount of energy can be saved when CS is used as fewer packets are sent due to compression. These measurements can be used to compute the remaining energy of the SNs and take necessary actions for prolonging network's lifetime. This can be done using various methods, like replacing SN's battery as soon as a remaining threshold has been reached, or by adding more intelligence in the network, as for example to perform routing based on the SNs' remaining energy.

The secure COAP measurements reveal that the overhead of COAPs is not significant, given that the sensory data are not collected very often. These results show that adding security to the connection is not costly once an active session has been established, hence a session should be kept alive as long as possible to avoid repeated handshakes.

Acknowledgment

This work has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 612361.

References

[1] "Iot-a project," http://www.iot-a.eu/.
[2] "Butler project," http://www.iot-butler.eu/.
[3] "icore project," http://www.iot-icore.eu/.
[4] "Rerum project," https://ict-rerum.eu/.
[5] G. Moldovan, E. Tragos, A. Fragkiadakis, H. Pohls, and D. Calvo, "An IoT middleware for enhanced security and privacy: the RERUM approach," in *Proc. of NTMS*, 2016.
[6] "Fiesta project," http://fiesta-iot.eu/.
[7] "Inter-iot project," http://inter-iot-project.eu/.
[8] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 169–186.
[9] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. ACM, 2013, pp. 15–20.
[10] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, 2015.
[11] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," *Proceedings of the ASE BigData & SocialInformatics 2015*, pp. 28:1—-28:6, 2015.
[12] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "Smartsantander: Iot experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.
[13] T. Ojala, "Open urban testbed for ubiquitous computing," in *Communications and Mobile Computing (CMC), 2010 International Conference on*, vol. 1. IEEE, 2010, pp. 442–447.
[14] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh, "Citysense: An urban-scale wireless sensor network and testbed," in *Technologies for Homeland Security, 2008 IEEE Conference on*. IEEE, 2008, pp. 583–588.
[15] "Zolertia re-mote," http://zolertia.io/product/hardware/re-mote.
[16] Z. Shelby, K. Hartke, and C. Bormann, "Rfc 7252the constrained application protocol (coap)," *Internet Engineering T ask Force (IETF)*, 2014.
[17] P. Dimitropoulos, J. Soldatos, N. Kefalakis, J. Bengtsson, and A. Giuliano, "D2.3 - OpenIoT Detailed Architecture and Proof-of-Concept Specifications," Technical Report, Tech. Rep., 2013.
[18] A. Fragkiadakis and E. Tragos, "Enhancing compressive sensing encryption in constrained devices using chaotic sequences," in *Proc. of SmartObjects (Mobicom)*, 2016.
[19] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level power profiling for low-power wireless networks," 2011.
[20] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.
[21] K. Hartke and O. Bergmann, "Datagram transport layer security in constrained environments," 2012.
[22] "Tinydtls project," https://projects.eclipse.org/projects/iot.tinydtls.