A flexible Monitoring-as-a-Service platform for IoT networks (Demo)

Petros Zervoudakis*, Nikolaos Karamolegkos*, Eleftheria Plevridi*†, Paraskevi Doulgeraki*,
Effie Karuzaki*, Stavroula Ntoa*, Eirini Sikianaki*, Nikolaos Partarakis*,
Pavlos Charalampidis* and Alexandros Fragkiadakis*
*Institute of Computer Science, Foundation for Research and Technology-Hellas, Heraklion, Crete, Greece
† Department of Computer Science, University of Crete, Heraklion, Crete, Greece
E-mails: {zervoudak, nkaram, eleftheria, vdoulger, karuzaki, stant, eirinisi, partarak, pcharala, alfrag}@ics.forth.gr

***Abstract*—Internet-of-Things (IoT) has drawn significant attention in recent years by encompassing a set of technologies that enables heterogeneous smart sensors to produce data for collection, storage, management and analysis. Commonly, IoT networks operate in remote or harsh environments, which inevitably means that sensors are prone to failure and malfunction. As a result, autonomous monitoring and maintenance of IoT systems is of paramount importance for rapidly detecting faults and minimizing the risk of system outages. This demo paper presents a platform that offers Monitoring-as-a-Service (MaaS) for the collection, storage and processing of monitoring data originating from heterogeneous IoT networks. The proposed solution incorporates blockchain and smart contracts technologies for data integrity verification, and application of automated workflows for the collected data filtering and labeling.**

***Index Terms*—Internet of Things, wireless sensor networks, cloud computing, blockchain, smart contracts, data integrity verification**

I. Introduction

The advent of the Internet of Things (IoT) technologies has shifted the traditional Internet networking paradigm into a pervasive interconnection between users, data and smart objects in a seamless way [1]. Based on this paradigm, a wide range of smart applications has become feasible, spanning from smart cities and smart buildings to precision agriculture and smart energy management [2], [3]. Modern IoT systems collect a massive amount of highly heterogeneous data via a large number of deployed sensing devices that are forwarded up to the cloud level for storage and further processing/analysis. Additionally, these IoT systems are characterized by high complexity and dynamicity that dictates the promotion of (semi-)automatic mechanisms for managing their diverse components.

IoT networks commonly operate in remote and harsh environments, which inevitably means that the sensors are prone to failure, malfunction or even rapid attrition. As a result, autonomous monitoring and maintenance is of paramount importance for minimizing the risk of outages and guaranteeing the compliance of Service Level Agreements (SLAs) between IoT application providers and consumers. Monitoring may be performed in either passive or active fashion and involves collecting and logging information from various subsystems of the network, for further processing and analysis tasks such as anomaly/outlier detection, fault diagnosis, predictive maintenance etc. [4].

Large volume monitoring data generated by modern IoT systems is commonly stored in cloud storage services (CSSs) for further retrieval and offline analysis. Although utilizing external CSSs minimizes the burden of local storage management and supervision, it also increases the risk of manipulation for data residing in remote servers. Unfortunately, data manipulation can severely degrade the quality and accuracy of any further processing or analysis and, subsequently, the effectiveness of consequent decision making. Thus, data integrity schemes are of great significance for effective cloud storage of monitoring data. Several techniques have been used for integrity verification, such as asymmetric cryptographic algorithms, erasure codes, hash functions, etc. [5], [6], [7]. Among them, *blockchain* technology has recently emerged for enabling a fully decentralized solution that can provide integrity verification without the need of a Third Party Auditor (TPA) [8]. Additionally, the concept of *smart contracts*, which are self-executing scripts deployed on a blockchain, can be used for creating and managing authoritative automated workflows for processing monitoring data [9].

In this demo, we present a Monitoring-as-a-Service platform that provides a solution for collecting, storing and processing monitoring data from heterogeneous IoT networks that support various smart applications. We leverage the FIWARE[1] open source framework for developing a flexible and scalable solution. In addition, we employ blockchain and smart contracts technology for providing an integrity verification mechanism for the collected monitoring data, as well as, apply automated workflows for the collected data filtering and labeling. In literature, other works achieving data integrity verification for IoT data in cloud storage applying methods use encryption techniques to protect data, relying on trusted TPAs [10]. Blockchain data integrity schemas can successfully overcome the trust problem of TPAs, however, they have to face the issues of large storage overhead [11], when the actual raw data are stored on a blockchain ledger. To address the issues outlined above, in our solution, IoT raw data is grouped into windows and verified tags are generated for integrity verifica-

[1] https://www.fiware.org

978-1-6654-6975-3/22/$31.00 ©2022 IEEE 487

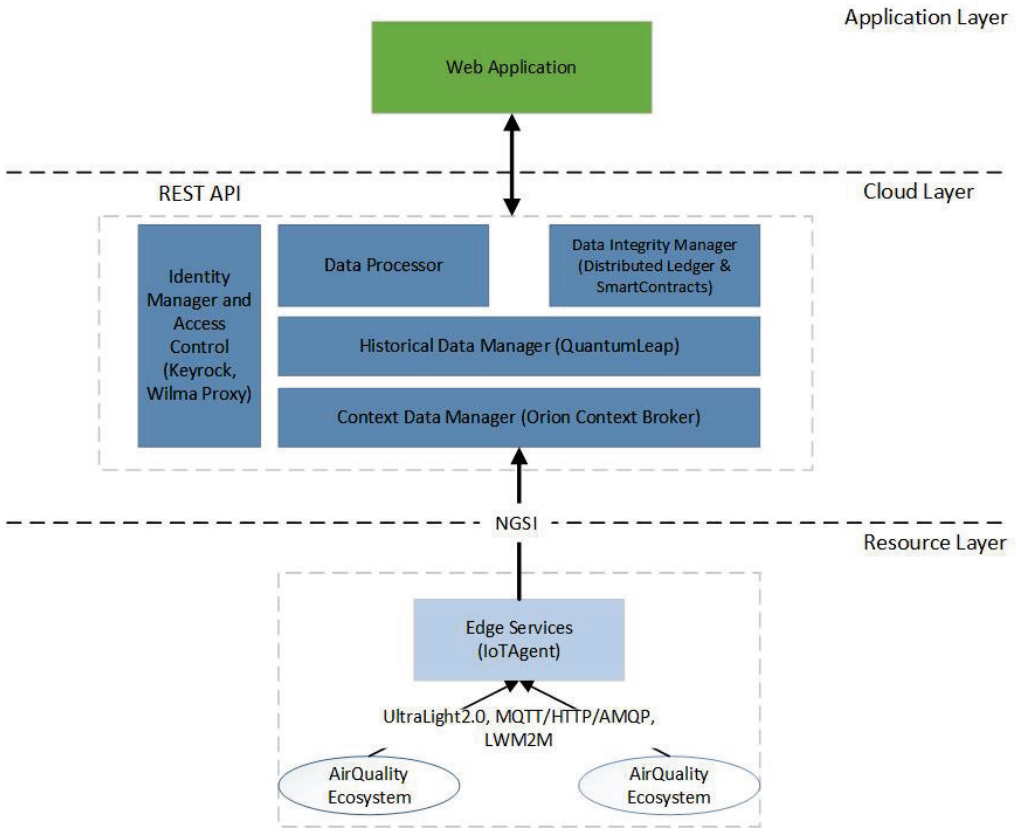

Fig. 1. Platform architecture

tion, allowing only those tags to be stored on blockchain with aim to reduce the cost of data storage needed.

II. System Architecture

The proposed architecture consists of three layers, as illustrated in Figure 1. The first layer is an *Application Layer* conserving application, interfaces and dashboards. The second layer is a *Cloud Layer* that involves various processes providing identity and authorization management, context information management, data management and data integrity verification. The third layer is a *Resource Layer* involving components that handle IoT infrastructure and data heterogeneity. The proposed architecture satisfies the main requirements defined in the IoT-A Architectural Reference Model [12] and enables cloud services to interact seamlessly with IoT devices.

A. Cloud Layer

This is a cloud environment that hosts all the services that enable to securely manage and ingest IoT data collected from IoT networks, named as *ecosystems* in the context of our platform.

Identity Management and Access Control is a component responsible for the authentication and authorization operations of the platform, including (i) identity and user management, and (ii) user authentication and access control. In this scope, Keyrock[2] based on the OAuth 2.0 in conjunction with Wilma PEP Proxy[3] enables authentication and authorization necessary for consuming resources of internal services. Wilma plays the role of the Policy Enforcement Point (PEP), an interceptor on incoming requests, while Keyrock acts as the Policy Decision Point (PDP) that evaluates access requests against a set of predefined authorization policies. In case authorization policies permit access, Wilma grants access to the requested resource.

Context Data Manager is a component, which acts as the core middleware service that follows a publish/subscribe paradigm. It is responsible for managing the entire lifecycle of context information, enabling the interaction of different processes and Application Programming Interfaces (APIs) across the platform. Context producers (e.g. IoT devices) publish their data to the Context Data Manager, through the FIWARE NGSI API[4]. Context consumers (e.g. historical data database) provide persistent storage of context information originating from the Context Data Manager, in external repositories for future use. In our platform, the FIWARE Orion Context Broker[5] plays the role of the Context Data Manager.

Historical Data Manager is a component which converts NGSI structured data into tabular format and automatically stores updated data persistently in a high-performance CrateDB[6] database. It also provides an interface for performing

[2]https://fiware-idm.readthedocs.io/en/latest
[3]https://fiware-pep-proxy.readthedocs.io/en/latest
[4]https://fiware.github.io/specifications/ngsiv2/stable
[5]https://fiware-orion.readthedocs.io/en/master
[6]https://crate.io

complex queries on historical data stored in the database (e.g. the latest N samples collected from a specific IoT resource, the minimum value over a time period per hour). This component is based on FIWARE QuantumLeap[7].

Data integrity is of high priority, and a very important contributor to this, is the *Blockchain* technology (BC) and the *Smart Contracts* (SCs). In general BC is a shared distributed and decentralized ledger, that facilitates the process of recording transactions and tracking assets in an immutable manner. Here, we build our solution on the Hyperledger Fabric (HLF)[8] framework, which has been developed and maintained with the support of Linux Foundation, IBM and Intel. Blockchain structure and data flow is carefully designed and implemented so as to achieve high availability of the service and minimize the risk of something being compromised. More specifically, it is a Cloud layer component implementing a BC network that consists of several peers, who participate in private peer-client connection channels for ensuring data privacy of participants. A distributed ledger is maintained for each client on those peers and the new blocks are added when consensus is met. The role of SCs, named as *chaincodes* in the context of HLF, is to automatically validate the compliance of collected IoT data against a set of configurable rules. Such rules assert, for example, that the temperature of a sensor inside a fridge should be between -10 and -6 degrees. If the data submitted to the chaincode does not fulfill these boundaries, it is labelled accordingly and this information is forwarded to the Context Data Manager. With the help of SHA-3 [13], the newest member of the secure hash algorithm standards, a fixed-size bit array, known as ”hash value” is calculated for the corresponding data. This hash is stored on the BC ledger in a key-value format, contributing to the data integrity process, at the retrieval stage. Hashing the data substantially decreases the retrieval time and reduces the space requirements in the BC.

As aforementioned, historical data are persistently stored and can be queried for future use via QuantumLeap API. Although QuantumLeap is a pretty solid choice for managing basic queries, it was not built for advanced data processing procedures, such as excessive computational complexity on raw data retrieval. To overcome this limitation, we include the *Data Processor* component, which is responsible for performing low-level analysis on collected data. Such a mechanism is necessary, since complex queries are typically required from data verification algorithms.

B. Resource Layer

This layer includes smart devices that send data updates for storage and processing to the Cloud Layer via appropriate adapters (i.e. IoT Agents) implementing the NGSI interface. An IoT Agent is a component that enables protocol translation from native IoT protocols (e.g. CoAP, MQTT, LwM2M, etc.) to NGSI protocol, and vice versa, enabling the seamless communication between the Cloud layer and the Resource layer of the described architecture.

[7]https://quantumleap.readthedocs.io/en/latest

[8]https://www.hyperledger.org/use/fabric

III. Web-based user interface

In this section, we present the *Application Layer* of the architecture, which is essentially a web-based user interface that facilitates the surveillance of IoT ecosystems. For its implementation, the Angular[9] development framework has been deployed that is an open-source, component-based framework for building scalable web applications, supported by Google and a large community of volunteers and companies. The main programming language used is Typescript[10], which is a superset of JavaScript and provides a syntax that allows optionally declaration of variable types. It also provides elements of object-oriented programming such as classes, interfaces, etc.

Finally, the HTML5 language and the Bootstrap[11] framework (version 5) were used for the implementation of the web pages. Bootstrap, being the most popular framework for developing responsive websites, provides mechanisms that control how the web-page elements will be shown, rearranged or hidden according to the screen size. As a result, it ensures that the best user experience and usability is offered on all screen sizes (computer screen, tablet, mobile phones). Finally, in order to control the appearance of web pages, the SCSS language is used, which is a superset of CSS that, among others, offers variables and nesting features for better code organization and finer control of the interface elements' appearance.

As shown in Figure 2, the user interface provides a meaningful representation of all IoT ecosystems tracked by the platform, with useful details about them (e.g. description, number of gateways, sensing devices and users appearing in it, data integrity verification schema option, etc.). Moreover, as illustrated in Figure 3, the user is able to retrieve additional information for each gateway belonging to a specific IoT ecosystem. A gateway dashboard comprises multiple widgets that visualize historical sensory data (e.g. temperature, humidity, volatile organic compounds, etc.) in the form of line plots for a given time period. Each line plot depicts historical data produced by a specific sensor, as retrieved from the cloud storage. Historical data visualized are labeled either as “Data integrity” (blue color) or as “No data integrity” (red color); in the first case, integrity of the data retrieved from the cloud database is verified, while on the second case data integrity has been compromised. Data integrity verification is performed by comparing the SHA-3 value of historical monitoring data retrieved from the cloud database, with the SHA-3 value calculated on the original data and stored in the BC ledger, as soon as it was collected from the IoT ecosystem. Finally, a widget in the gateway's dashboard informs the user on smart contract-based rules violation, by illustrating appropriate notifications.

[9]https://angular.io

[10]https://www.typescriptlang.org

[11]https://getbootstrap.com

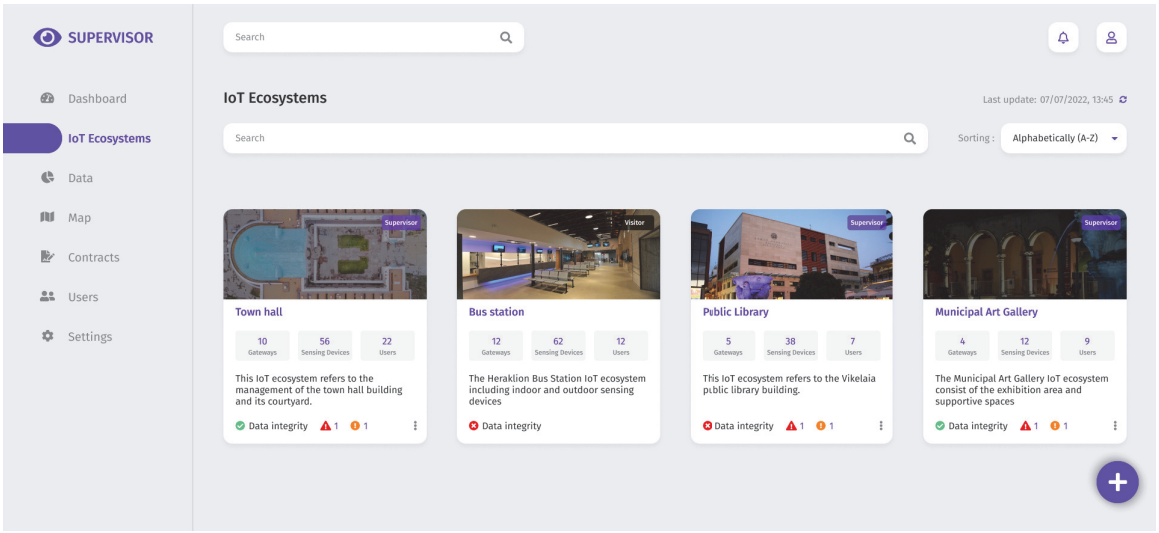

Fig. 2. Main view of IoT ecosystems

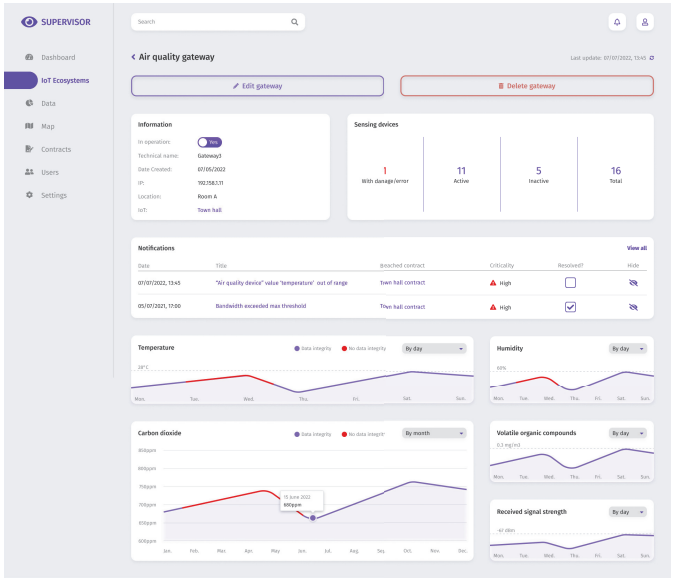

Fig. 3. Historical data and integrity verification for gateways

IV. Conclusion

In this paper, we introduced a platform that aims to provide Monitoring-as-a-Service for heterogeneous IoT networks. We presented a solution that incorporates blockchain and smart contracts technologies with the aim to provide data integrity verification and application of automated workflows for the collected data filtering and labeling. We provided an analysis on the architectural components of the platform and described a web-based graphical user interface, which enables the surveillance of IoT ecosystems. Further work includes various improvements and enhancements, such as the performance evaluation in terms of scalability, throughput and latency of the collection, storage, management and analysis process.

Acknowledgment

This research has been financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code: T1EDK-00070).

References

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[2] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-things-based smart cities: Recent advances and challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16–24, 2017.

[3] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[4] G. Stamatakis, N. Pappas, A. Fragkiadakis, and A. Traganitis, "Autonomous maintenance in iot networks via aoi-driven deep reinforcement learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2021, pp. 1–7.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 598–609.

[6] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *2010 proceedings ieee infocom*. Ieee, 2010, pp. 1–9.

[7] H. Zhu, Y. Yuan, Y. Chen, Y. Zha, W. Xi, B. Jia, and Y. Xin, "A secure and efficient data integrity verification scheme for cloud-iot based on short signature," *IEEE Access*, vol. 7, pp. 90 036–90 044, 2019.

[8] H. Wang and J. Zhang, "Blockchain based data integrity verification for large-scale iot data," *IEEE Access*, vol. 7, pp. 164 996–165 006, 2019.

[9] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016.

[10] B. Priyadharshini and P. Parvathi, "Data integrity in cloud storage," in *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, 2012, pp. 261–265.

[11] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for iot data," in *2017 IEEE International Conference on Web Services (ICWS)*, 2017, pp. 468–475.

[12] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. Van Kranenburg, S. Lange, and S. Meissner, *Enabling things to talk*. Springer Nature, 2013.

[13] M. J. Dworkin *et al.*, "SHA-3 standard: Permutation-based hash and extendable-output functions," 2015.