Internet of Things as a Service (iTaaS): Challenges and Solutions for Management of Sensor Data on the Cloud and the Fog

Euripides G.M. Petrakis[a,*], Stelios Sotiriadis[b], Theodoros Soultanopoulos[a], Pelagia Tsiachri Renta[a], Rakumar Buyya[c], Nik Bessis[d]

[a] *School of Electrical and Computer Engineering, Technical University of Crete (TUC), Chania, Crete, Greece*
[b] *Department of Computer Science and Information Systems, Birkbeck, University of London, London, UK*
[c] *School of Computing and Information Systems, University of Melburne, Melburne, Australia*
[d] *Department of Computer Science, Edge Hill University, Ormskirk, UK*

Abstract

Building upon cloud, IoT and smart sensors technologies we design and develop an IoT as a Service (iTaaS) framework, that transforms a user device (e.g. a smart phone) to an IoT gateway that allows for fast and efficient data streams transmission to the cloud. We develop a two-fold solution, based on micro-services for the IoT (users' smart devices) and the cloud side (back-end services). iTaaS includes configurations for (a) the IoT side to support data collection from IoT devices to a gateway on a real time basis and, (b) the cloud back-end side to support data sharing, storage and processing. iTaaS provides the technology foreground to enable immediate application deployments in the domain of interest. An obvious and promising implementation of this technology is e-Health and remote health monitoring. As a proof of concept we implement a real time remote patient monitoring system that integrates the proposed framework and uses BLE pulse oximeter and heart rate monitoring sensing devices. The experimental analysis shows fast data collection, as (for our experimental

*Corresponding author
Email addresses: `petrakis@intelligence.tuc.gr` (Euripides G.M. Petrakis), `stelios@dcs.bbk.ac.uk` (Stelios Sotiriadis), `tsoultanopoulos@gmail.com` (Theodoros Soultanopoulos), `ptsiachri@gmail.com` (Pelagia Tsiachri Renta), `rbuyya@unimelb.edu.au` (Rakumar Buyya), `Nik.Bessis@edgehill.ac.uk` (Nik Bessis)

Preprint submitted to Elsevier *September 17, 2018*

setup) data is transmitted from the IoT side (i.e. the gateway) to the cloud in less than 130ms. We also stress the back-end system with high user concurrency (for example with 40 users per second) and high data streams (for example 240 data records per second) and we show that the requests are executed at around 1 second, a number that signifies a satisfactory performance by considering the number of requests, the network latency and the relatively small size of the Virtual Machines implementing services on the cloud (2GB RAM, 1 CPU and 20GB hard disk size).

Keywords: Cloud computing; Internet of Things; Fog Computing; Remote patient monitoring; Survey

---

1. Introduction

Internet of Things (IoT) and cloud computing are getting prominence over the recent years, due the increasing usage of smart devices and sensors in many application areas (e.g. healthcare [1], assisted living [2], environmental monitoring, industrial systems [3]). The widespread use of smart devices and sensors acting as data aggregators connected to the Internet, in conjunction to the emerging use of large computer platforms (notably the cloud) where the data are transferred for permanent storage and analysis (e.g. using data analytics platforms such as Apache Spark), shape a promising future and a great opportunity for new businesses to increase their clients and support new functionality. This is a massive and still expanding market that is expected to reach the 11 trillion U.S. dollars by 2025 [4].

The idea of the Internet of Things combined with cloud computing, opens new horizons in the field of real time data collection and analysis [5, 6, 7, 8]. cloud computing emerges as the key platform for IoT data storage, processing and analytics due to its simplicity, scalability and affordability (i.e. no up-front investment, low operation costs). Companies deploy applications and systems on the cloud to avoid infrastructural expenses, operational and maintenance costs. A promising application domain for this technology is e-Health and remote patient monitoring. Accessible monitoring, control, alert and smart intervention solutions can provide invaluable assistance to the wellness, safety and convenience of chronic patients, of the elderly and other user categories. Such a solution will increase users autonomy and confidence and enable self-managing their condition with the help of caregivers remotely. As a result, this new model will reduce the need for patients to organize and attend face-to-face appointments with doctors and might reduce the amount of medication and days in hospital. All of these outcomes will increase the care efficiency and the cost effectiveness of the solution. Key challenges are the on-the-fly data discovery, collection and analysis as the IoT system might produce enormous volumes of data that is collected and need to analyzed as close to real time as possible.

iTaaS is a two-fold solution, based on microservices for the IoT and the gateway (users' smart phones) and the cloud side (that includes the back-end services for data storage and analytics). Firstly, we target on real time IoT data collection from smartphones, with a particular interest on Bluetooth Low Energy (BLE) devices. These are low power sensors that support mobile application connectivity and connections to handheld devices which are already connected to the Internet though WiFI or GSM networks. Secondly, we target on a cloud system deployment with scalable data storage capabilities and interfaces to external users and systems. The vision of the iTaaS is to transform smart phones to gateway (fog) platforms for IoT data collection and processing. Captured data are encrypted and streamed to the cloud back-end system in order to be available to users based on subscriptions. The gateway (mobile device) establishes a two-way communication between the front-end (users carrying wearable sensors and the mobile device connected to these sensors), and a back-end (the cloud running services for data monitoring (by authorized users subscribing to specific users and sensor information), data analysis and permanent storage.

Inspired by the concept of Service Oriented Architectures (SOA) [9], the iTaaS solution makes use of modular services implementing fundamental functionalities communicating with each other and offering important benefits, such as scalability, re-usability multi-tenancy, increased accessibility and security through powerful APIs for seamless application integration. iTaaS front-end is implemented in native Android. iTaaS back-end services are implemented on OpenStack[1] and FIWARE[2], an open-source distributed cloud infrastructure funded by the EU. iTaaS reference architecture encompasses IoT-A [10, 11] design principles in an attempt to develop an innovative IoT platform that supports generic services and IoT devices (i.e. independent of connectivity and not coupled to specific IoT protocols).

iTaaS patient monitoring application brings high level personalisation to the coaching suggestions by caregivers. It is easy to define abnormal pattern detection rules regarding physical (e.g. walk ability, tremor), medical (e.g. heart-rate, oxygen saturation in blood), socio-emotional (e.g. relatives to rely on, anxiety) activities of the patient. These rules are defined by a formal caregiver (e.g. a physician) based on the actual needs of the patient, they are embedded into the health monitoring scenario and operate on a publish - subscribe model: Vital measurements transmitted by sensors are recorded and only subscribed users (e.g. a physician who monitors the patient) have access to this information. In this scenario, caregivers assign sensors to patients and set lower or upper values for sensor measurements. The caregivers get notified when these values are violated (using a messaging service). The proposed coaching solution keeps patients and caregivers in the loop tracking user's progress, providing a monitoring overview highlighting suspicious patterns and coaching instructions.

We focus on BLE sensors and we develop a data schema to allow for universal BLE support so that, BLE sensors can be easily linked to an ITaaS application for data pushing (without worrying for sensor specific processes for handling data). The gateway supports dynamic and automatic discovery and registration of new sensors: each sensor is declared by its XML schema and this information is registered at the back-end. The gateway runs a service for

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

synchronizing sensors schema information with schema information stored on a local database (so the gateway gets updated on new sensors). For experimental purposes, we utilized the Nonin Onix[3] device (that produces the peripheral capillary oxygen saturation (SpO2) and pulse rate measurements), and the Polar H7[4] (that produces pulse rate measurements). Although developed to handle BLE sensors, iTaaS can accommodate any kind of sensor protocol such as (ZigBee, Zi-wave) that can be supported by a smart device (the gateway) since the only component that is affected by the decision to select a specific protocol is the BLE scanner for connected sensors. The rest of the system (i.e. besides the BLE scanner) is sensor agnostic since all data are communicated and processed in a sensor agnostic format (i.e. JSON).

Summarizing, ITaaS focuses on a) a generic IoT data collection framework based on gateways and BLE protocol (the industry standards in sensor communications), (b) a SOA design approach for both the gateway and the cloud, (c) a proof of concept solution in e-Health implementing a real medical scenario for remote patient monitoring, and (d) an extensive experimental analysis to demonstrate effectiveness of our implementation. iTaaS shows how to collect sensor data fast and on-the-fly on the gateway, the utilization of real world sensors and IoT workloads, how processing of user and sensor data takes places on a gateway with fog capabilities, how data are transmitted securely to the cloud, how to support persistent and trusted processing and storage of data on the cloud and, how to manage use profiles and user subscriptions to data and services. iTaaS is expandable by design as more services can be added (e.g. data analytics) or modified on the fly without requiring that the system be stopped or re-designed.

iTaaS solution (design and its implementation) is based on previous work by the authors presented in one conference [12] and two workshops [13, 14]. The architecture of the iTaaS gateway for on the fly data collection is shown

[3]http://www.nonin.com/Finger-Pulse-Oximeter/
[4]https://support.polar.com/

in [12, 14]. Design and implementation of the back-end solution is shown in [13]. Based on our past experiences, we identified the need for an integrated iTasS framework, where data of sensor devices can be easily collected to the cloud and be distributed to interested users based on subscriptions. At the same time, essential configurations at both ends of the architecture can be easily adapted (e.g. setup of thresholds, notifications etc.). The integration of all concepts referred to above has not been presented elsewhere. This paper extends and unifies our previous work and presents an integrated architectural and implementation view of all iTaaS concepts. Besides system architectural issues and a remote health monitoring application, this paper is also a thorough survey of the related literature emphasizing on IoT, the cloud and the fog, discusses topics for future work and research challenges.

Related work in IoT data management is discussed in Sec. 2, with a particular interest on IoT data collection. Emerging technologies and technological challenges that will shape the IoT landscape in the near future are discussed in Sec. 2. In Sec. 4 we present the reference architecture and the description of the front-end and back-end services. In Sec. 5, we demonstrate the experimental analysis and the significance of our results, followed by conclusions, system extensions and issues for future research in Sec. 6.

2. Related Work and Background

This survey focuses on (a) synergies between the sensing environment and the cloud, (b) device to platform connectivity and data aggregation at network end, (c) data persistent storage and analysis (on the cloud), and also, (d) use cases with particular emphasis to e-Health. An almost orthogonal issue with large impact in today's IoT design and implementation is security and trust.

Several approaches to IoT systems design are known to exist and many of them have been implemented in commercialized IoT platforms[5] facilitating the

[5] https://iot-analytics.com/product/iot-platforms-white-paper/

implementation of applications or, in custom solutions tailored to the needs of a particular application domain such as manufacturing [15], monitoring of users activity [16], assisted living [2], e-Health [17] and remote health monitoring [18, 12, 13, 1, 19] etc. Interesting surveys covering important aspects of IoT system design and implementation have been published in [5, 20, 4, 21, 3, 7].

Table 1 summarizes all these issues, provides pointers to related work and summarizes the contributions of the present work in relation to these issues (last column). Compared to existing systems, iTaaS is a two-fold service-oriented event architecture, based on microservices for the IoT and the Fog (users' smart phone devices acting as data aggregators) and the cloud side (that includes the back-end services for storage and analytics). iTaaS is driven by the key requirements of today's IoT systems for adaptability, security, low-cost and scalability, modularity and expandability.

Biswas and Giaffreda [5] discuss the IoT and cloud convergence challenges and focus on the identification of requirements for ubiquitous accessibility, connectivity, dynamic user management of users and scalability. A key requirement is efficient data collection. Focusing on e-Health and health care, Theummler, Paulin and Lim [17] focus on the infrastructure requirements for supporting remote patient monitoring taking into account patient monitoring parameters, the nature of data (e.g. images and multimedia), out-hospital interactions (e.g. between patients and health providers). Emerging infrastructure models for supporting high quality services in IoT and health care are considered as well including, narrow-band IoT such as Low Power Wide Area Networks (LPWAN) and protocols (e.g. LoRa[6]), Edge computing [30] and 5G[7]. 5G will not only support mobile telephony but also services through net slicing: Different bands of the spectrum are assigned to support services in application domains (e.g. remote health monitoring and personalized medicine in the narrow band). Telecom providers can operate services such as Software Defined Networks (SDN)

[6] https://lora-alliance.org

[7] https://5g.co.uk/guides/what-is-5g/

Table 1: Related work and contributions of iTaaS approach

Architectural Properties	**Existing Systems**	**iTaaS**
Surveys & IoT Design	[5, 17, 20, 21, 3, 7, 4]	
High-Level Architectures	*Cloud*: [22, 6, 23, 24] *IoT*: [10, 11, 25] *Fog*: [8, 26]	√ [*Cloud, Fog*]
Service Oriented Architectures (SOA)	[23, 12, 14, 13, 27]	√ [*REST*]
Fog & Gateway Architectures	[28, 29, 23, 12, 30, 14, 13]	√ [*Android,IoS*]
Data Aggregation & Protocol	*BLE*: [31, 25, 29, 23] *LTE*: [32]	√ [*BLE*]
IoT Storage Architectures	[33, 34, 3, 28]	√ [*MongoDB*]
IoT – Cloud Integration	[22, 15, 25, 12, 14, 13]	√ [*FIWARE*]
Security	[35, 22]	√ [*By Design*]
Use Cases	*Manufacturing*: [15] *Farm Data*: [36] *Access System*: [16] *Assisted Living*: [2] *e-Health*: [18, 17, 1, 19, 12, 14, 13]	√ [*e-Health*]

and establish Virtual Private Network (VPN) services specifically for healthcare. Along the same lines, Orsino et al. [32] focus on the IoT and energy efficient data collection in 5G in a smart city scenario.

Anderson, Fierro and Culler [6] consider the problem of heterogeneity of components in an IoT ecosystem and explore issues of synergy and interoperability at all levels starting from hardware and firmware and towards the level of device connectivity and service discovery. This paper gives concrete guidelines for IoT platform and application design, learned from building a full-stack synergistic IoT platform. Kobialka et al. [25], present a work to link networks of sensors with computing systems in a distributed way. In close analogy to OCCI[8], Ciuffoletti [22] introduces an API interface framework and specifications for interacting with IoT infrastructures based on REST[9] . Their proposed framework is simple open and expandable.

Jiang et al. [33] discuss a storage framework for IoT for structured, unstructured data and their processing in Apache Hadoop. Similar to previous work by Li et al. [34], authors present a NoSQL storage management solution for IoT data and they provide an ontology for data sharing between different IoT applications. Both solutions focus on IoT data storage, in contrast to our work that includes services for data collection on the IoT side. Cecchinel et al. [28], focus on a storage architecture addressing issues of sensor management and IoT data collection.

Boualouache et al. [31] propose a system architecture to collect data from IoT environment using BLE technology and smart phones as gateways. They focus on the performance of communication and on issues of energy consumption. Guoqiang et al. [29] propose a configurable smart IoT gateway that is pluggable, supporting different communication protocols. Similar to iTaaS, data acquired using different protocols in different formats are translated to a uniform format (e.g. JSON) that is understandable by all services. In our earlier work [23] we

[8] http://occi-wg.org

[9] https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style

demonstrate a solution for benchmarking IoT devices. We introduced micro and macro-benchmarking techniques for IoT gateway devices. However, this work is based on a custom CoAP[10] benchmarking utility.

In regards to use cases, Tao et al. [15] demonstrate an IoT and cloud service architecture for the manufacturing field. They address connection, communication, computing and control aspects of services in manufacturing and they discuss their approach in three applications namely, applications in the workshop, applications in the enterprise and applications among enterprises. They present an analysis of the requirements for hardware, firmware, services and human perception of IoT functionality along with key enabling technologies. Xu et al. [18] discuss a service to handle data for medical devices in cases of emergences, with data management and inter-operation support. They focus on locating medical actors (i.e. staff and ambulances) in a mobile environment and introduce a data annotation framework facilitating the access to data using ontologies. Additional related work on stand-alone systems (not exploiting cloud technology) in e-Health, includes recognition of daily activities in a living environment for specific user cases such as Alzheimer's patients [1] or the elderly [19]. They apply spatio-temporal reasoning on location or body activity signals received from body-worn or ambient sensors. Recognition is restricted to a set of predefined activities (models) described using rules involving spatio-temporal constraints.

Balabanis et al. [2] focuses on monitoring users life by monitors their actions in a indoors environment. We presented an e-health provision system for patient monitoring using motion sensing devices (such as the Microsoft Kinect device). The system exhibits significant benefits towards assisted living IoT environments including reduced costs by cutting unnecessary visits to physicians. It also provides significant benefits towards improved patient experience and safety. For example, in rehabilitation environments proactive treatment allows real time prevention by continuous monitoring. It also improves the overall treatment

[10] http://coap.technology

process by providing assistance to people who are hospitalized or at home and their condition requires uninterrupted monitoring.

There are a lot of approaches for IoT data collection and management (some of them theoretical) without providing proof of concept and performance analysis of real world implementations. Others are focusing on a single technological aspect (e.g. data management on the cloud and data analytics, sensor data collection and communication with the cloud etc.). A holistic approach that puts existing technologies in place and produces an end-to-end-system supporting the desirable properties (i.e. features in Table 1), followed by a real-world evaluation is not known to exist (to the best of our knowledge). Most of the works referred to above lack of experimental demonstration or proof of concept implementation. iTaaS provides a coverage of all these features.

3. Research Challenges

Lately, IoT systems have become increasingly complex because of the use of novel technologies and the complexity and size of modern use cases. Existing IoT architectures have been conceptualized and implemented to address certain challenges based on single domain and single use case-oriented requirements, thus not considering issues of openness, scalability, interoperability and use-case independence. As a result, they are less principled, lacking standards or are vendor or domain specific. Most importantly, they are hardly replicable in the sense that, most of the times, the same architecture cannot be used in many use cases.

Future work in IoT systems design should go beyond these limits and towards more open and dynamic configurable IoT platforms. This in turn goes beyond architectures that are vertically closed so forming many "Intranets of Things" [37]. Some first steps towards such IoT foundational technologies and architectures have been taken (e.g. IoT-A [10]). IoT-A proposes an Architecture Reference Model (ARM) defining the principles and guidelines for generating IoT architectures, providing the means to connect vertically closed systems in the communication layer (i.e. how devices interact with the system) and service layer (i.e. how services are integrated). IoT-A compliant architectures may assure that generated knowledge will be modular and reusable across domain or use-case specific boundaries. IoT-A addresses the architecture design problem and does not focus on whether existing cloud platforms can offer the tools and services to support the implementation of IoT-A compliant IoT systems. An attempt to address this problem is the work in [11].

The basic challenge that future work in IoT systems should address, relates to the lack of technological components to aid and simplify the development of cross-domain IoT applications in a fast and secure way. Among these, security, openness, interoperability through advanced connectivity and usability (i.e. applicability) issues may play an important role in deploying interconnected solutions. To our understanding, technology advances mainly in security, IoT communications and IoT systems interoperability are the main pillars for breaking the silos of single sector deployments and allowing the development of cross domain applications that are open, secure and scalable.

3.1. Fog Computing

Even though a cloud may offer virtually unlimited compute resources, internet bandwidth may impede application performance or, business or regulatory requirements of the application mandate for hosting resources in a specific place. This is typically the case with user data where severe restrictions in regards to data transfer and storage to public locations may apply (e.g. military, medical, people activity monitoring applications). Another limitation is with regards to security: Although many attempts towards increasing the security on the cloud have emerged lately, users are not always willing to send their data to the cloud due to the unknown storage location and also, due to the perceived "distance" between them and their data. In general, the user or application owner has limited control on cloud infrastructures that are managed only by the service providers. This distance is also the main reason for the long delays that are experienced sometimes between the client devices and the services locations [8].

Examples for delay intolerant applications are physical access control systems, health monitoring and factory automation. To address these limitations, the paradigm of fog computing has lately emerged, starting from Cisco [38, 39].

Fog computing can be assumed as an extension of cloud computing, bringing virtualized services closer to the edge of the network (i.e. close to the user on gateways or on user devices). Fog brings the benefits of low latency (due to the close proximity), location awareness and increased security, privacy and availability. Efforts to standardize architectures and platforms for extending the cloud to support functional edge nodes are currently underway (e.g. OpenFog [26]). However, since the paradigm of fog computing has emerged only lately, architectures and platforms for extending the cloud to support functional edge nodes have not been fully designed yet. In a cloud-fog scenario, fog-nodes[11] are installed close to the network edge. A fog node ingests data from IoT devices, runs mission critical functionality and communicates the results of this processing to the cloud. It is a responsibility of the application designer to decide on the optimal place for data processing (i.e. the fog node or the cloud). The most time-sensitive data must be processed on a fog node closest to the IoT devices.

A fog-node can be realized as a separate (probably virtualized) infrastructure running services ranging from data collection, storage, encoding, decoding, communication, processing etc. This is feasible for nodes connected to a sustainable power source. Data are processed and stored on the fog nodes (temporarily or permanently); only aggregated results (e.g. simple statistics, visualization charts) and sent to the cloud for permanent storage and analysis (e.g. big data analytics to gain business insights and for generating new application rules based on these insights by applying machine learning and data mining). Alternatively, implementation of fog nodes may rely a lightweight solution receiving feeds from IoT devices and providing feedback in real time based in a time-critical scenario

[11] https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

(e.g. malfunctioning devices due to hardware failure, compromised devices in a security breach scenario, fraud detection on ATMs). The functionality of a fog node is constraint by the energy consumption restrictions of the device (e.g. a fog node on a smart device running on batteries in a mobile user scenario). The analysis is applied to most time sensitive data only (e.g. for detecting signs of problems and preventing failure by sending control commands to actuators).

3.2. Security

Future IoT system design should devote significant effort to deal with security, privacy and trust in decentralized cloud and foain computing infrastructures. Securing IoT applications that are distributed over several IoT and cloud infrastructures is a challenging task. Regulatory entities (e.g. US FTC[12] and the article 20 working party of the EU[13]) recommend that the principles of Security by Design and Security and Privacy by Default[14] should be applied in IoT. Security and privacy by design suggests taking security and privacy into account since the design phase of a system or service and apply configurations that assure the highest security and privacy level. Identifying the main actors (e.g. users, devices, systems, and services) and associating actors (i.e. their profiles) to access rights for services and data is of paramount importance.

The Industrial Internet Security Framework by IIC[15] and the security pillar of the reference architecture of the OpenFog consortium[16] highlight the need for security monitoring. They emphasize the need for monitoring devices, networks and applications at the edge of the network (i.e. on devices and edge nodes) and in the cloud. For example, the security pillar of the OpenFog reference

[12] https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf

[13] https://ec.europa.eu/info/law/law-topic/data-protection_en

[14] https://www.ipc.on.ca/wp-content/uploads/Resources/pbd-privacy-and-security-by-design-oracle.pdf

[15] https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf

[16] https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf

architecture specifies the FaaS (Fog-as-a-Service) security monitoring functionality for devices at network end. They acknowledge the need to identify proper placement for monitoring functions and the need to deal with security at the scale of big data.

Although cloud systems are considered to be more secure for deploying IoT applications, users and data are exposed to many risks as IoT is operating in the periphery of the cloud, it is open to many users and is generally less protected than the cloud itself. This fact opens-up new research challenges for methodologies ensuring security and for methodologies for detecting and for dealing with the cause and point of system failure if security fails [40]. The EU's GDPR[17] has a significant impact on IoT systems design. Data protection in e-health in particular is crucial, as potential intrusion may not only lead to vulnerable personal data theft (e.g. patient data in hospital databases) but may also lead to risks in human life (e.g. disruption of important sensors monitoring patient etc.).

Due to the size and complexity of modern IoT systems, security threats can be detected in many aspects of system operation and relate mainly to malicious user's behavior detection which is expressed as (a) fraud detection in which case, authorized of unauthorized users operate the system for the purpose of unfair or unlawful gain or, (b) intrusion, in which case, unauthorized users are attempting to disrupt normal system operation. Similar behavior is now detected in virtualized environments such as the environment of a cloud provider (now affecting the operation of the system in scale and a large number of users) with certain economic and operational impact.

A key problem for IoT applications that collect large amounts of data, is the on-the-fly and real time solutions for anomaly detection either for system failures, for improving QoS or for detecting security leaks and vulnerabilities. When errors and faults occur, when hardware resources are faulty or configured or utilized in a way that causes application performance degradation, prompt

[17]https://www.itgovernance.eu/en-ie/gdpr-report-ie

action should be taken in order to ensure high quality of service. Traditional large-scale solutions for malicious behavior or malfunction detection, suggest either continuous monitoring of the state of fog or cloud components or periodically monitoring of system logs, or both. In large-scale cloud and fog-distributed infrastructures, system logs are becoming big data as time passes. The monitoring of large system logs resorts to intro or retrospective analysis of big data [41, 42]. The risks have been highlighted in several application domains and may take the form of Distributed Denial of Service (DDOS) attacks[18] (e.g. car hacking[19]). An IoT deployment should use latest virtualisation and security technologies to enhance the performance of IoT platforms [43]. Anonymization, pseudonymization and data protection techniques can be applied to avoid exposing data to unauthorized third parties. The privacy of the user sensitive data can be supported also by new techniques such as blockchains[20] which are currently emerging.

Fog architectures may help enhance security however, new security issues emerge, mainly due to the fact that fog nodes may be untrusted. Given that it is often easier to hack into client software and because of the proximity of fog devices to end users, fog nodes should be first to provide access control and encryption, contextual integrity and isolation mechanisms over sensitive data before it leaves the node. Techniques for distributed and hybrid identity management can be applied (e.g. access to data and services protected by OAuth2.0 protocol[21]). Techniques for supporting the confidentiality and integrity of data need to be developed. Off-the shelve fog or private cloud devices providing the desired functionality are currently becoming available on the market at affordable prices (e.g. SixSQ's NUVLA Box[22]).

[18] https://www.symantec.com/connect/blogs/iot-devices-being-increasingly-used-ddos-attacks

[19] https://www.theguardian.com/technology/2016/aug/28/car-hacking-future-self-driving-security

[20] https://www.blockchain.com

[21] https://oauth.net/2/

[22] https://sixsq.com/products-and-services/nuvlabox/overview

The world is moving towards machine-type communication, where anything from a smart sensor to products in super-markets will be connected to the internet. Web of Things (WoT) is an initiative and framework towards unifying the interconnected worlds of Things (i.e. devices, sensors) into a single architecture [44]. WoT is far from being a reality to date, mainly due to the fragmentation of technologies that are currently being applied for the design and implementation of IoT systems. A common work around to this problem is to allow each Thing become part of the existing Web. Then each device can be published on the Web (i.e. advertise its identity and contents), be discovered by Web search engines and be used by humans or applications just as any other Web site. Then a Mashup service[23] will allow application developers to compose new applications in significantly less time minimizing the effort required to maintain the system each time a device or service is added, re-moved, or updated. Node-RED[24] and WireCloud Mashup service[25] of FIWARE support this functionality.

Although lightweight Web servers[26] can be embedded in small devices in order to enable WoT functionality, they feature limited resources and the solution is not optimal in terms of battery life time, sensor autonomy and cost. A work around to this problem would be to deploy a Web Proxy, that runs on the cloud and keeps the virtual image of each Thing. Through Web Proxies, not only Things can communicate with other entities or with the cloud but also, the Things (their descriptions, data and services) become part of the Web, so that they can be published, consumed, aggregated, updated and searched for. Mapping any device into a Web Proxy makes the integration (configuration of new applications or re-configuration of existing ones) much easier just like Web sites can be created and published on the Web.

Essential parts of the WoT proxy is an API interface to allow connections

[23] soyl12
[24] https://nodered.org
[25] https://catalogue-server.fiware.org/enablers/application-mashup-wirecloud
[26] https://www.linux.com/news/which-light-weight-open-source-web-server-right-you

with the outside world and a directory. Sense2Web[27] is an implementation of a WoT directory and is implemented as a service in FIWARE cloud[28]. The service acts as a meeting point for the IoT context producers that intend to register the availability of their Things and sensor devices, and the IoT context consumer applications that intend to discover them using SPARQL. All devices are declared as NGSI-9 entities[29]. In order to allow the users to understand what data or services are offered by an IoT entity, the directory will include an additional human readable component of the offered services (e.g. OpenAPI[30]). The WoT Model[31] provides definitions for Web Things proposes a REST Web API for Things.

The Semantic IoT or Semantic Web of Things (SWoT)[32] concept emerged recently and relates to technologies for designing inter-operable domain or cross-domain Web of Things Applications. SWoT is an extension of the Web of Things that enables applications to share content and services beyond their boundaries or, even more important, to create, new applications as a composition of existing ones (e.g. using mashups). The main idea is to accomplish these tasks automatically or with minimum human intervention. To enable the vision of SWoT, tools that are capable of understanding the meaning of IoT applications (i.e. data and services) and reason over their content must be applied. Semantic Web (link) technology is a solution to this need. In Semantic Web[33], formal definitions of concepts and their properties form ontologies, which are defined using the RDF, RDFS and recently the OWL language[34]. IoT ontologies, in particular, comprise of definitions of IoT concepts (e.g. sensors, services) and of their properties (e.g. measurements) by means binary relations. The Seman-

[27]http://info.ee.surrey.ac.uk/Personal/P.Barnaghi/doc/LinkedData.pdf
[28]https://fiware-iot-discovery-ngsi9.readthedocs.io/en/latest/
[29]http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/NGSI-9/NGSI-10_information_model
[30]https://www.openapis.org
[31]http://model.webofthings.io/#web-things-model
[32]https://semantic-web-of-things.appspot.com/?pĪSWC2017Tutorial
[33]https://www.w3.org/standards/semanticweb/
[34]https://www.w3.org/OWL/

tic Sensor Network (SSN)[35] ontology by W3C is applicable to a wide range of applications. Query languages such as SPARQL[36] can be used for querying information in ontologies and reasoners such as Pellet[37] can be applied for finding inconsistencies of for inferring new information from information represented in the ontology.

Approaches towards implementing the SWoT vision are currently becoming available and some have demonstrated their potential in experimental set-ups. To the best of our knowledge, SWoT methodologies are not commercialized yet or applied in real-world or large-scale implementations. Most IoT ontologies are not widely accepted yet or have not proved to be scalable (as huge amounts of data are generated in typical IoT application). Real-time annotation of IoT data (i.e. deriving their meaning) and instantiating this information to ontologies would require a huge amount of resources. In addition, the completeness of information in IoT ontologies (which is related to the expressivity and complexity of the representation) and speed of processing, are traded-off. Therefore, existing solutions to the problem of scalability rely on lightweight ontologies such as IoT-lite [45] or Sensor Domain Ontology (SDO) [46]. Both solutions are defined as subsets of SSN ontology and promise sensor data annotation and instantiation in real time. The IoT-A ontology [47] extends SSN ontology to represent more IoT-related concepts such as services in addition to sensor devices. Sense2Web [48] is also an implementation of these ideas which is available as service in FIWARE.

3.4. Low Power Wide Area Network (LPWAN) Connectivity and 5G

IoT manufacturers have developed products (e.g. sensors), which typically use short range protocols such as Bluetooth and later Bluetooth Low Energy (BLE) to connect to a gateway (a smartphone or tablet) and then to the cloud via some backhaul (e.g. a cellular network and the internet). However, the

[35]https://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[36]https://www.w3.org/TR/sparql11-query/
[37]http://semanticweb.org/wiki/Pellet

commercial success of this solution is questionable as the number of IoT devices (sensors) that can be connected to a smartphone using Bluetooth is limited (especially for sensors and gateways running on battery and unless the gateway is connected to a sustainable power source). Bluetooth and BLE, the same as WiFi and ZigBee, are not suited for long-range performance, while cellular networks are costly, consume a lot of power, and are expensive.

Low Power Wide Area Network (LPWAN) technologies fill the gap between mobile (3G, 4G, LTE) and short-range wireless (e.g. Bluetooth, WiFi and ZigBee) networks. The trade-off is the achievable data and error rate (i.e. LPWAN protocols do not guarantee delivery of data packets). LPWAN technology falls short in terms of QoS compared to cellular standards and this means that an operator cannot use it to provide the kind of Service Level Agreements (SLAs) that are critical for customers. However, LPWAN specifications fits well the requirements of many IoT applications (e.g. smart cities, home automation, industrial automation, environmental monitoring, e-Health) who need to transmit small quantities data periodically over a long range, while maintaining long battery life (e.g. a wearable health sensor that only transmits when vital measurements exceed some predefined threshold).

LPWAN networks are being deployed now because the cost to deploy the network in unlicensed bands requires much less capital than the cost of its 3G, 4G counterparts. SigFox[38] and LoRa[39] are the main competitors in this landscape targeting similar applications. The same do other technologies such as Weightless, Ingenu, NB-IoT, and LTE-M[40], which also enable long rage devices to be connected to telecommunication networks with the latter two been standardized within 3G and 4G respectively. Typically, an IoT application can be deployed only if the network is already there. However, for vendors that need to deploy IoT applications on their own and run the network by themselves, LoRa

[38]https://www.sigfox.com/
[39]https://www.lora-alliance.org/
[40]https://www.link-labs.com/lpwan

is a good option. LoRa supports 2-5Km ranges in urban areas (entire cities can be covered with a few LoRa antennas) and up to 15Km in suburban areas. It works in the unlicensed spectrum below 1Ghz which come at no cost. It is an asynchronous protocol, which is optimal for battery lifetime and cost. There are no royalty issues with LoRa (except of the LoRA modulation chip which is produced by Semtech[41]).

5G emerged as a need to go beyond the limits of its predecessor cellular technologies. 5G focus on dividing the network up into slices to fit different services for different use cases while ensuring QoS. 5G will act as an enabler to IoT service providers and end-user organizations who have to face the reality of managing a diverse range of connections, notably cellular and LPWAN, in addition to short range connections, such as BLE, ZigBee or WiFi. Leveraging on 5G capabilities (ubiquity, integrated security and network management) a network of LoRa devices can be developed anywhere, without installation of additional network equipment and without the need for network management (which is offered by the cellular network). Alongside, 5G technology comes with its own authentication, authorization, and accounting framework thus minimizing the effort to supporting this functionality in LPWAN IoT networks. Soon, telecom providers will support LoRa functionality and synergy within 5G. LoRa has been adopted by major EU telecom providers including among many others Orange (France), KPN (the Netherlands), Proximus (Belgium), Netzicon (Germany), Unidata (Italy). LoRa capabilities will eventually be integrated into their 5G service base stations [49].

4. IoT as a Service (iTaaS)

iTaaS is a two-fold solution, based on micro-services for the IoT (users' smart devices) and the cloud side (back-end). We followed a valid system design approach [10, 50] that identified (a) the functional components and their

[41] https://www.semtech.com

interaction, (b) the information that is managed and how this information is acquired, transmitted, stored and analysed, (c) the software entities that support the functional and information activities, (d) the requirements for assuring data, network and user security and privacy. iTaaS (reference) architecture [12, 14, 13] is described by a set of UML diagrams including (a) information (class) diagrams describing information that is handled by the system, (b) activity diagrams describing flowcharts for several types of user actions the most important of them being, system login request (user authentication and authorization), request for new account, request to access a facility or area and, event handling (i.e. handling cases of overcrowded areas and critical events) and, (c) an architecture diagram.

iTaaS architecture is organized in two processes, the data collection from the IoT system and the back-end cloud. The sensors are embedded or connect to user devices (e.g. smartphones) that send data to the cloud. The services are REST-based [27] and integrate the system functionalities that are grounded upon four zones as shown in Fig. 1.

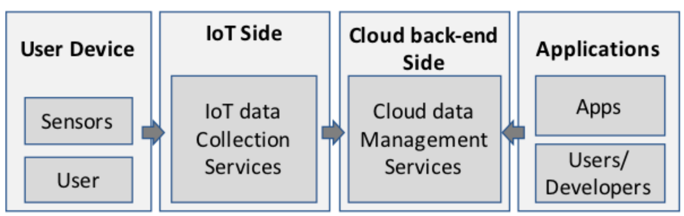

Figure 1: Generic architecture for data capturing and cloud data management

The "User Device" includes the IoT data source generators notably users, sensors and smartphones. Furthermore, users can utilize the front-end interface to send data to the back-end wherein third party users (e.g. doctors and physicians) can access it. Finally, they are able to receive updates or notifications when required (e.g. a situation is critical). To develop our solution, we utilized two types of BLE sensors; the Polar H7[42] and the Onyx II[43] that include the following features.

- The polar H7 is a BLE transmitter recording in real time accurate the heart rate levels. The device is attached around the chest with an elastic belt and it detects heartbeat. It gives a timing reference for a specific heart rate measurement and transmits the information to the back-end.
- The Onyx II (model 3230) provides an oximetry monitoring solution that allows monitoring vital data such as the oxygen saturation range and heartbeats per minute.

The "IoT side" is a fog node that includes various modules such as device pairing, data collection, a lightweight database, linking to the cloud, connectivity, data filtering, event processing and notification services. It implements the following actions:

- Discovery and registration of new BLE sensors using the generic sensor schema.
- Data filtering mechanisms for reducing communication overhead.
- Encryption for secure connection with the back-end and anonymization for ensuring user privacy (only access tokens are transmitted).
- Local data caching, for handling situations of poor or no bandwidth connection and data synchronization with back-end when connection is established.
- Emergency mode operation that allows prioritization of data collection. This allows messages to be exchanged among the subscribed users for taking actions.

[42] http://www.polar.com
[43] http://www.nonin.com/Onyx9560

- Personalized coaching by associating coaching rules with the actual condition of the patient. The rules are defined by the physician on the cloud and are uploaded to the gateway. This way, decisions are taken locally on gateway minimizing communication response times. The gateway synchronizes and gets updates on rules defined on the cloud.

The "cloud back-end side" implements services for storage, big data processing, publication and subscriptions, event management, user authentication, messaging, services for establishing secure network connections with gateways or users (including encoding / decoding of user data). The event management is used for decision making and user rule management. The publish/subscribe service is for data streaming from the IoT to the subscribed applications and users. The big data processing module is responsible for big data analytics (e.g. using Apache Hadoop or Spark). This module provides API interfaces for such systems to connect and perform analytics. The "applications" or end-users can be (a) physicians (or caregivers) who subscribe to user data, have access to sensor measurements and are entitled to provide coaching instructions to patients and, (b) administrators or technical specialists that define schemas for new sensors, can create new users and define their access rights. Both, physicians and administrators can access the system using a Web application.

The storage system is a MongoDB database[44]. The NoSQL format of MongoDB makes it easy to store semi and unstructured data and supports permanent storage for users, sensors, messages, rules, events (e.g. rule violations) and actions undertaken by health care personnel in response to events. For IoT applications dealing mostly with massive volumes of sensor measurements (e.g. JSON formatted data), NoSQL databases are prevailing due to their scalability, easy maintenance (they require less management by supporting automatic repair, data distribution and data model adaptation. Many commercial solutions are available and selection depends mainly on application domain and data type.

[44]https://www.mongodb.com/scale/nosql-database-comparison

Among them, Apache Cassandra, Redis, and MongoDb are very popular with the latter being particularly good for JSON data storage (such as data in IoT). Compared to relational databases, NoSQL databases are extremely performant and scalable (i.e. they can handle very large data sizes while maintaining very good performance) allowing both, common operations such as key indexing, queries by individual keys, and also non-common ones, such as text searching (e.g. MongoDB), values expiring (e.g. Redis), map/reduce or distributed storage (e.g. Cassandra).

4.1. Implementation

In the following, we present a detailed discussion of the IoT side services (in Sec. 4.1.1), the back-end cloud services (in Sec. 4.1.2) and their integration, interaction and data flows (in section 4.3.3). Each service is implemented as a REST API. Fig. 2 shows the iTaaS system implementation for IoT device data management on the cloud.

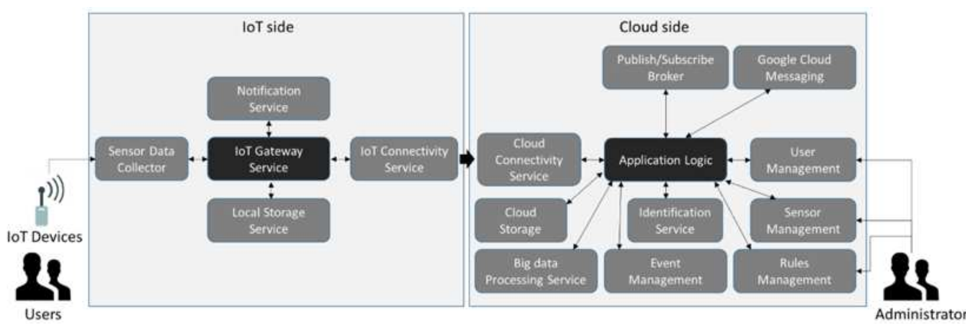

Figure 2: iTaaS system implementation for IoT device management on the cloud

4.1.1. IoT side services

The implementation is based on BLE standard[45] to support low latency and short-range networks (less than 50m) for low-power devices. BLE implements the Generic Access Profile (GAP) that defines the mechanism for BLE devices

[45] https://developer.android.com/guide/topics/connectivity/bluetooth-le

to communicate with each other, by broadcasting and observing data. This mechanism makes a device visible and allows other devices to connect with it. BLE supports also two core protocols referred to as ATT (Attribute Protocol) and GATT (Generic Attribute Profile). GATT (Generic Attribute Profile) defines how data are formatted and exchanged. GATT is built on top of ATT (Attribute Protocol) and defines how devices that follow the BLE protocol can transfer data when paired. It uses the attributes provided by ATT, organizing them into a hierarchy of data structures. It further divides information into logical pieces (e.g. heart rate service) and “features” that are information specific to a particular sensor (e.g. heart rate measurement).

The gateway is dynamic allowing automatic discovery and registration of new devices or sensors. Each device or sensor is declared by its XML schema which is stored in the cloud. Only devices whose XML schema has been defined on iTaaS platform can connect to the gateway. Once a device is registered to iTaaS gateway (running GAP service) its XML schema is downloaded to the gateway from the cloud. Then, its data format and identity become protocol independent by associating its GATT profile with a generic XML schema (encoded as Java Object) that defines the characteristics and data that can be processed by the iTaaS platform (not all GATT characteristics of a BLE device or sensor need to be mapped to the XML schema). The XML schema allows for recognition of attribute values of the sensors. It’s a responsibility of the system administrator to define XML schemas for new devices. iTaaS has the ability to recognize and register new characteristic values of BLE sensors (e.g. pulse rate in addition to blood-oxygen saturation) by editing their XML schema (for adding new characteristics).

Formal XML schema for two BLE sensors is presented in the Appendix. The description of the XML schema includes:

- Information about sensor name (tagged as “sensors”): The sensor information model for the specific BLE device for data collection.

- Device name (tagged as “device name”): The device (sensor) name (e.g.

NoninOnix, Polar H7).

- Unique identification per sensor (tagged as “uuid”): The unique id per sensor (e.g. the MAC address).
- Values to collect (tagged as “values”): It characterizes the sensor name and the units of the sensor data (e.g. percentages of data that is about to be collected, FORMAT_UINT8 that is, an 8-bit integer).
- Format (tagged as “format”): It characterizes the data type of the data to be collected (for example collecting float data).
- In some cases, BLE sensor manufacturers define complicated functions for decoding their features (tagged as “position” and “multi”).

The IoT side (gateway) services are organized as shown in Fig. 2 and includes:

Application Logic. The application logic is the centralized module that orchestrates services running on the gateway: Determines in what sequence the services run and how there are synchronized (e.g. when the user requests data to be send to the cloud, application logic activates the connectivity service for sending data). In addition, the service runs

- The management module for handling (i.e. comparing with thresholds) the measurements produced by the sensors according to patient specific rules which are defined by the caregivers and are downloaded from the cloud,
- A module for processing measurements that calculates the average values and periodically,
- Determines the conditions for collecting and transmitting data to the cloud: allows a user to select the condition for data collection (on time-interval, on demand, and on rule violation, as explained in Sec. 4.1.3).

Sensor Data Collector. It activates the mechanism to determine which peripheral devices advertise data to the service and establish a GAP connection with authenticated BLE devices (i.e. sensors in our case). Before that, application logic service downloads (from the cloud) the XML schema of the sensors which are entitled to connect and transmit data to the service. The XML data of sensors are stored in the Local Storage. The XML of the connected sensor is converted to a Java object through a Java document object model parsing process with the aim of identifying attribute values of sensors. Sensor authentication is carried out by comparing the MAC address of the sensor with that defined in the XML. The XML also provides information on what measurements are useful for the application. Sensor data are then translated to JSON and passed to application logic for processing on the gateway. Fig. 3 illustrates this process.

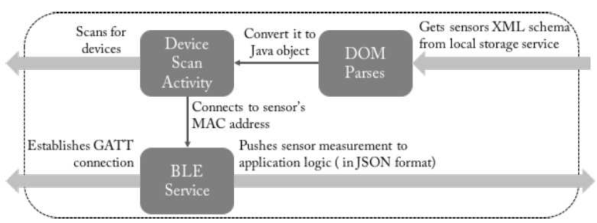

Figure 3: Sensor Data Collector internal processes

Local Storage. The local storage allows insertion, deletion and update of data and it includes the database manager, the data encryption and decryption module and the SLQLite[46] database engine as shown in Fig. 4 The database manager defines the appropriate tables for different types of data and predefined operations such as the "add sensor measurement" which stores the measurement of a sensor, the "delete sensor measurement" for deletion of a specific sensor, the "return sensors" that returns the identifier of a sensor. The "data encryption and decryption" is a component that runs on top of local storage for decod-

[46]https://www.sqlite.org/index.html

ing/encoding data based on the Advanced Encryption Standard (AES). Finally, the SQLite database[47] includes tables such as the "user device" comprised of the user id and the paired device (email and device registration id), the "user rules" that includes the user id (email) and sensors (device name id) along with the rule thresholds, the "sensor measures" that store data locally in case of failure in communication. The service contains the so called "asset folder" that includes the XML format of each sensor. Sensor data can be stored locally in the SQLite database and are deleted upon transmission to the JSON cloud storage. For example, data can be stored on local SQLite storage when WIFI connection is lost and transmitted to the back-end when connection is up again.

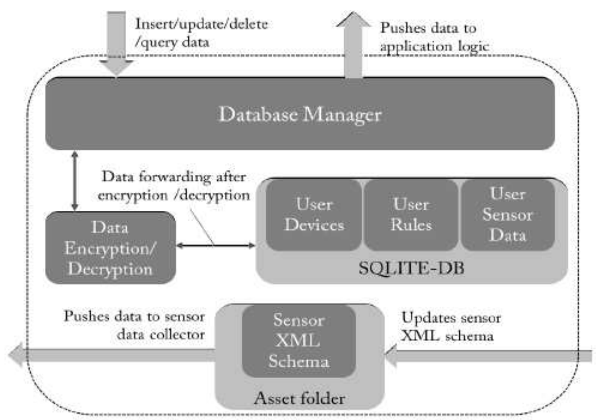

Figure 4: The Local Storage internal processes

Connectivity Service. The Connectivity Service is the communication channel between the gateway service and the cloud. Data is transformed to JSON and then with the suitable asynchronous calls is sent securely to the cloud. The data exchanged is packed in JSON format and data (Base64) encryption (or decryption) is applied before transmission. The connectivity service operations include (a) the recovery of device registration identifier from the cloud (b) it sends the registration id to the cloud using the appropriate HTTP headers, (c)

[47] https://www.sqlite.org/index.html

it downloads rules per user using a post API call so data is forwarded to the application logic, which in turn transfers it to the local storage, (d) it sends sensor data using the on time interval, the on "demand" or the "violation rule" functions (explained in Section 4.1.3) and (e) when logging into the application to verify the identity and access rights of the visitor: It checks the local storage for user's credentials (i.e. login data or session key) and attempts to automatically connect the user with the back-end.

Google Cloud messaging (GCM). The Google Cloud messaging[48] is a free service that allows developers to send messages to multiple platforms including Android and iOS. It is comprised by a GCM server deployed in the cloud and a GCM client that is part of the front-end service. The steps to send a message are as follows:

- The smart phone service sends an HTTP request to the GCM server using credentials designated by Google via AppEngine. The Google Connection server responds to the device with a message and assigns a unique Registration ID to it.
- Then the client service sends the Registration ID to the backend (application logic) for later use. The server stores the registration ID of the device to local storage until the user decides to send a message.
- Using the Registration ID along with the text message, it makes a request to the GCM server to send in information as message to the application using a secured way as it can keep the message until the device is available online.

Notification Service. The notification service informs the users with appropriate prompts namely "toast messages" for Android devices and "alerts" for important changes occurring in the flow of operation of the service. The notification

[48] https://developers.google.com/cloud-messaging/

service works independently from GCM service that handles user prompted communication. The gateway service in implemented on native Android and runs on a mobile device (e.g. a smartphone). The interface is implemented in Javascript, HTML, jQuery and CSS. The asynchronous communication with the cloud is implemented using AJAX calls.

4.1.2. Back-End Cloud services

iTaaS back-end is a composition of autonomous RESTful services [27] communicating with each other over HTTP. Individual services or groups of services are deployed on the same or different Virtual Machines (VMs) on the cloud. Network delays are expected due to the nature of this design. However, the experimental results (Sec. 5) demonstrate that iTaaS is capable for responding in real time even under heavy workloads. Nonetheless, the advantage of this Service Oriented Architecture (SOA) is, system modular design, ease of configuration (that best suits the need of an application), ease of maintenance and expandability. For example, more services can be added at-run time or, any service can be moved to a different VM (on the same or different cloud) with minimum overhead (i.e. only the IP of the service will change). Access to data and services is protected by an OAuth2.0 mechanism. Some of the service modules in iTaaS architecture are available as reusable Generic Enablers (GEs) on FIWARE catalogue[49]. iTaaS back-end implements the following services:

Application Logic. In line with application logic on the front-end, its purpose is to orchestrate, control and execute services running on the cloud. Threshold violations (reported by the Event Management service below) are typically managed by this component. When a request is received (from a user or service), it is dispatched to the appropriate service. For example, services regarding user accounts and access rights are dispatched (through application logic) to user management service. It is tightly related with the connections and the context

[49]https://catalogue.fiware.org

broker services (below) which form the communication channel with the gateway. Finally, application logic performs basic security controls (e.g. checking if a session between the mobile application and the cloud has been initiated).

Connectivity service. in line with connectivity service running on the front-end, it implements secure asynchronous communication between the front and back-end. Encoded data are formed in JSON and are exchanged using AJAX calls.

Publication and Subscription service. This service acts as a mediator for the data sent to the back-end and the end-users or applications. Using this service, applications or users can subscribe to data produced in the front-end. Sensors are listed as "public entities" and physicians are subscribers to these entities. Each time a new sensor is registered to the system, this component is updated in order to update its context (e.g. sensor name) and its content (e.g. data to be collected such as heart rates). It is implemented using Orion Context Broker[50], a ready to use service of FIWARE. It is responsible for the management of publications and subscriptions related to sensors and measurements. When a new patient-specific sensor is used, a new entity is created in the broker. Similarly, when a measurement becomes available by a sensor, a notification is sent to all entities (e.g. users or serviced) subscribed to this information.

Event management service. This module gets input from the Publication and Subscription service (above). When the sensor Data collection service creates a new measurement, the corresponding entities in Publication and Subscription service is updated as well. The Event Processing service subscribes to this information and gets a notification about the change. This triggers the execution of rules set by the physician (which are stored in the Storage Service) to govern the monitoring of the patient wearing the sensor. If the sensor values are within the limits of these rules no action is taken; otherwise (e.g. heartrate exceeds its

[50]https://catalogue-server.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker

threshold), the Event Monitoring Service notifies the physician to take action (e.g. an alert indication is send to the patient using GCM service).

Google Cloud Messaging (GCM) service. In close analogy to the GCM service running in the front-end, allows text messages to be managed easily and to be sent the notification message on the IoT side.

User Management service. implements functionality for user management, including creating, editing, deleting users and their profiles, their access rights to data and services and access history. This information is stored in the database. For example, using this service, application logic service checks the database whether to authorize users to access and use the front-end. It implemented using the KeyRock Identity Management[51] service of FIWARE. It provides also a Single Sign On (SSO) service for secure access to services, data and networks. It applies (a) identification services for users (b) management of their profiles and, (c) authorization services supporting access control based on user roles and access policies based on OAuth2.0[52] mechanism.

Big Data Processing service. Allows big data platforms such as Apache Spark[53] to perform data analytics on the collected data. Data could be analysed on real time. This component is particularly useful as more and more data is stored in the cloud. The big data processing service will handle analytics for useful feedback (e.g. historic data analysis).

Sensor Management Service. It is a service responsible for importing, editing, viewing and deleting available sensor devices. In particular, the administrator imports an XML file for each sensor that essentially has the descriptions of all metadata information about the sensors. With this service in place, new sensor formats (their XML) can be defined and used in the system. An example XML scheme for the Polar H7 and NoninOnix 3250 sensors is shown in Appendix.

[51] https://catalogue-server.fiware.org/enablers/identity-management-keyrock
[52] https://oauth.net/2/
[53] https://spark.apache.org

There are also services that implement functionality for medical users (e.g. physicians):

Rule Management Service. The physician sets rules for each patient that is monitored. The rules are basically the thresholds (upper and lower threshold) as they offer a maximum and a minimum permissible value where the intermediate level of the values determines the normal state of a patient. For example, a user may have a maximum threshold of 120 and a minimum of 80 heart rate. If the patient's measurement value is higher than these limits then, the user violates this rule set for him by his physician and the condition is now considered critical.

Measurement Management Service. The physician can access the history of the patients she/he is following. That is the most recent and past measurements, including the name of the measurement, its start and end time.

Patient Management Service. The physician can access the patients she/he is following along with their condition (normal or at risk) and can choose to send a message to a patient.

Storage service. This is the shared database where data generated from all services referred to above are stored. It is implemented using MongoDB[54] and stores user information, rules for patient management, sensors and history of data sent by a sensor in conjunction with the date of measurement and patient status.

The back-end is implemented on a FIWARE platform running on OpenStack[55]. They communicate asynchronously using Slim PHP[56] (for handling REST requests and for communicating with the database), cURL[57] for communicating data and AJAX requests between services and, finally, MongDB for the database.

[54] https://www.mongodb.com
[55] https://www.openstack.org
[56] https://www.slimframework.com
[57] https://curl.haxx.se

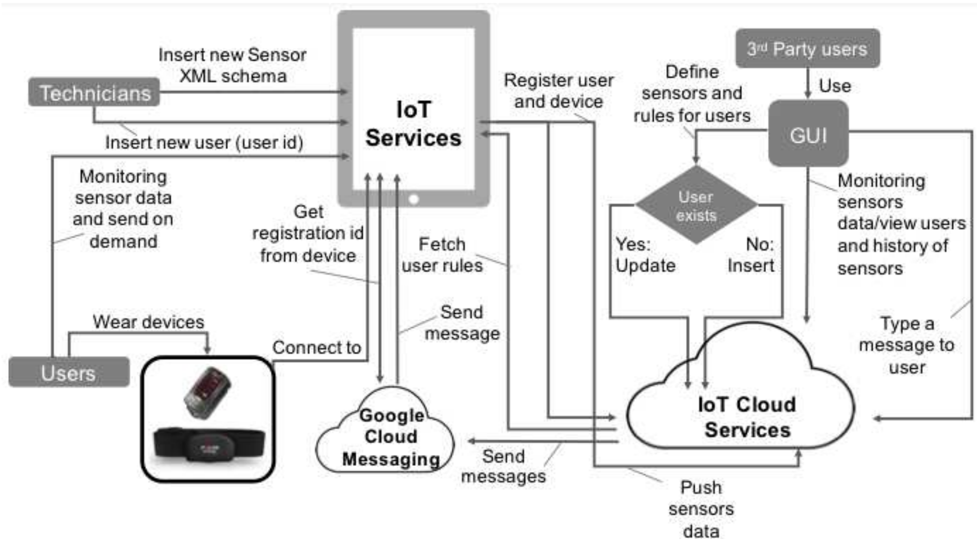

Figure 5: iTaaS Data flow

Fig. 4.1.3 presents the data flow and data transfer among components in patient monitoring scenario (mainly focused on how data is collected from the user sensors and is transmitted to the cloud). The patients are the service users and the doctors are the consumers. In this scenario, the following actions take place in a logical sequence:

- Registration of a new sensor: An XML schema describes each sensor. The system administrator accesses and modifies the XML schema of the sensors (as presented in Appendix). These hold information for the sensors so to be recognized by the gateway service.

- Registration of a new user: The system administrator uses the authentication mechanism to register new users to the service using the email as registration identifier (id). The user's profile is stored locally and in the cloud authentication service.

- Request to the GCM service: The service provides identification per device using a device identifier (id) from the GCM service.

- User and device registration in the cloud: The service uses the local storage for storing data associated with a user identifier (i.e. user email address). The data are forwarded to the cloud using an HTTP post command (following the gateway REST API).

- Sensor assignment and user rules: This is used to associate sensors to user and to assigns rules applicable to each sensor. For example, in a healthcare scenario, medical personel can define sensor category (e.g. Nonin pulse oximeter), and set lower and upper values for sensor measurements, thus to be notified when these values are violated using GCM service.

- Rules for mobile devices: The service transfers the rules from back-end to front-end based on an HTTP post request using as parameters the user id, and an appropriate identification code to get rules and sensors related to the specific user and store them in the device's local storage.

- User Interaction: The user does not require making any configuration, as the service is automated and dynamic. He/she only has to wear a portable BLE sensor and carry a mobile device. The sensor is automatically paired to mobile device. Then the service can monitor sensor data in real time.

- Data forwarding to the cloud: The data that is forwarded to the cloud include Device Name identifier (i.e. the sensor identifier name) and Measurements (i.e. name and value of the measurement).

- User interaction: Average sensor values are transmitted to back-end per regular intervals or on demand. We distinguish between two modes operation namely a) the smooth mode: The measurement values generated by sensors are within the range of the maximum and minimum threshold and are transmitted to back-end per regular time intervals and b) the risk mode: The measurement values generated by sensors violate one of the rules and are transmitted to back-end immediately. In addition, the use (patient) has to option to transmit data to back-end on demand (on his/her own initiative).

- The violation rule: If theres is a rule violation, the service activates the risk mode (above) automatically and sensor measurements are transmitted per second.
- Monitoring measurements in real time: One of the most important functions of the system is the fact that the end-user (e.g. a physician) has the option to monitor real time measurements of the users.
- Measuring Time: It refers to the date format (year- month-day) and time (hours: minutes: seconds) information model including the measurements interval.
- Messaging to user: GCM forwards messages to the user in real time. It offers a messaging middleware for information exchange (e.g. text messages from third party users to end users).

5. Performance Evaluation

The experimental analysis focuses on the performance evaluation of services belonging to the following categories:

- The IoT side that responds to the phases of data collection and data pushing on the cloud.
- The back-end cloud system which in turn, includes (a) services for data storage in the NoSQL database and (b) services related with context and event management.

5.1. IoT Side System Evaluation

To measure the effectiveness of the front-end system we compute the times required for data to be transmitted from the IoT side to the back-end cloud system (HTTP request/response times). Each transmission phase includes a number of interacting sub-phases such as BLE device discovery, data collection, local storage and data forwarding to the back-end. For this experiment, we use the Polar H7 to measure the heart rate pulses and the Nonin Onyx (device 3250) to measure oxygen saturation levels and heart rate pulses. The experimental configuration includes (a) sensor data collection on 1 second interval, (b) average over 100 user's transmission phase executions and (c) use of SQLite database for local storage on the gateway. The results of this experiment are summarized below:

- The process on a gateway starts with the "user registration" method, where the system registers a new person. For each registration, data is sent to the cloud (including user identification and other details). This action requires 230ms on the average.
- A request to the back-end storage requires an average of 145ms.
- Local storage actions are quite fast (4ms) and includes accessing the smartphone SQLite database for temporary storage of sensor data.
- Each time a new sensor measurement is produced, it is evaluated according to the rule-based system, so violated thresholds can be captured. This process requires 275ms.
- The IoT service includes data encoding to a JSON like information model (2ms).
- Data streams are then send to the cloud (125ms).
- The time for data stream transmission to the cloud takes 130ms. However, in cases of messages using the GSM service, an extra 400ms are spent.

The iTaaS mobile application runs on an ordinary Android smartphone (Android 5.1 ARM Quad-Core CPU 2.2GHz, 2GM RAM, 32GB Flash storage) consuming less than 20% of the CPU time and 100MB RAM.

5.2. Back-end System Evaluation

We run an exhaustive set of experiments and we analyze the performance limits of the services running on the cloud. We study system scalability (i.e.

how system response time increases with the number of connected users). All services run on the same VM (core VM), with the exception of the Publish/Subscribe Orion Context Broker that runs on a separate (second) VM, the JSON storage service (MongoDB database) that runs on a third VM and, the Identification – Authorization service (Keyrock Identity Management) that runs on a fourth VM. Therefore, the cloud back-end runs on four VMs. The Identity Management service is not expected to cause any performance bottlenecks as it is addressed only once by each user at login. Therefore, in the experiments below, the time for system login is not taken into account. However, all other VMs may receive up to a large number of simultaneous requests as the number of users and their requests increase. All measurements of time below account also for the time spent for the communication between VMs or between services within the same VM.

All VMs, with the exception of Orion Context Broker that runs on a shared VM and is installed on a remote FIWARE node (FIWARE is a federation of distributed cloud infrastructures), are hosted on the FIWARE node of TUC and have the following features: One virtual processor (x86_64 processor architecture, 2,800 Mhz, first level cache size 32 KB, second layer 4,096 KB cache size), 2,048 MB RAM, 20 GB hard drive capacity. Each VM runs Ubuntu Operating System 14.04 and Apache HTTP server. The computational resource usage metrics are taken using the Linux htop command[58].

We execute the following actions:

- HTTP POST to storage service in order to save data to the database. These are data related with the management of rules associated with users and their sensors. An example dataset includes the creation of a new rule for heart rate thresholds.
- HTTP PUT to the JSON storage service in order to update data related

[58] https://www.howtogeek.com/howto/ubuntu/using-htop-to-monitor-system-processes-on-linux/

with rules. An example is the update process of the heart rate thresholds.

- Data management based on context/content for data distribution on third party users. This includes content collection per sensor data in order to allow other users to be subscribed on their context. An example includes the subscription of the medical personnel to the heart rate measurements of a particular patient.

- HTTP POST to Event service in order to evaluate patient measurements (i.e. check if the sensor measurements are in the permissible range). It accesses the database where the rules are stored. As a result, it sets new status for the patient.

- HTTP PUT to Event service. It is similar to above. As a result, it updates the status of a patient.

Table 2 shows the HTTP request/response time for the above calls in milliseconds (that reflect the service completion time). It should be mentioned that we observed a delay in the JSON storage when new rules are created per user.

Table 2: Completion time of service execution for different requests in the back-end

Type of request	*JSON storage (POST)*	*JSON storage (PUT)*	*Publish - Subscribe*	*Event Service (PUT)*	*Event Service (POST)*
Time / request	670ms	400ms	420ms	430ms	695ms

The purpose of the following experiment is to measure the performance of the core VM. We used ApacheBench[59] to issue multiple simultaneous requests to the core VM. In ApacheBench we are opted to define the total number of requests and how many of them will be executed simultaneously. Table 3 shows

[59]https://httpd.apache.org/docs/2.2/en/programs/ab.html

the results of this experiment. All measurements of time below account also for the time spent for the communication between VMs or between services within the same VM.

Table 3: Core VM response time and resource usage for 2,000 requests

Concurrency	1	40	80
Time (ms)	1.08ms	1.38ms	6.80ms
CPU usage (%)	30%	60%	95%
RAM usage (MB)	130MB	180MB	350MB

We notice a very low resource usage when concurrency = 1. This is expected, as only one request is executed at any time. Response times improve drastically with the simultaneous execution of 40 requests (i.e. the Apache HTTP server switches to multitasking). Processing capacities may increase exponentially or raise restrictions for space or bandwidth for concurrency > 80. The best values are obtained with concurrency = 40, hence we conclude that the system optimal performance is on around 40 concurrent users based on this OpenStack configuration.

iTaaS may produce big amounts of data and requests, requiring large processing capabilities, which may surpass the capacities that our experimental system set-up is able to provide. In this set-up, most iTaaS core services are implemented in a single VM thus overloading the VM when the number of concurrent service requests exceeds a limit. An obvious solution to dealing with performance would be to employee additional VMs each running a single service (or a small group of services). Alongside, we can allocate additional VMs implementing the same service (or groups of services) thus having more than on VM sharing the load. In all measurements above, the performance of the mobile device is not taken into account.

6. Conclusions and Future Work

Recent developments in the area of smart and wearable sensors, make cloud, Internet of Things (IoT) and big data processing the cutting-edge technologies of this time. These highlight new requirements, such as real time data collection, on-the-fly big data storage and analytics that current systems are unable to support, mainly because of the need for increased scalability. As an extra feature data can be collected from low cost and affordable sensors that support low energy consumption rates. Hollistic approaches based on the integration of recent technological advances in sensors, protocols and showing seamless integration of recent edge, cloud technologies and platforms are still missing. iTaaS is a contribution towards this direction.

This work introduces iTaaS reference architecture and its implementation showing proof of concept in a remote health monitoring scenario. iTaaS is a twofold solution, based on micro-services for the IoT (users' smart devices) and the cloud side (that includes back-end services). Building upon principles of Service Oriented Architectures (SOA) design and driven by the key requirements of today's IoT systems for adaptability, low-cost and scalability, iTaaS architecture is modular and expandable.

The iTaaS framework supports optimal sensor data handling exhibiting low bandwidth usage which includes data filtering (for reducing communication overhead), local data caching for processing data locally, postponed data uploading for handling situations of poor or no bandwidth connection and, finally synchronization with data on the cloud. It supports universal usage over BLE sensors; BLE sensors can attached easily (on the fly) to any mobile device (running the gateway services) with automatic sensor discovery and coupling with the gateway. It is dynamic as captured data streamed to the cloud back-end system in real time and allows for decoupling the system functionality into the front and the back-end (a mobile device / gateway and the cloud respectively). iTaaS framework is therefore generic allowing easy adaptation to any IoT scenario.

iTaaS framework provide assurances of security, privacy, scalability, and reliability at both its cloud and IoT components. iTaaS enables the engineering of reliable and secure coaching systems able to respond to dynamic and complex situations. iTaaS service framework will support health monitoring solutions both, both indoor and outdoor. Its ambition is to increase acceptance of the technology solutions by the older adults and people with minimum exposure to technology as all communication with the system is via user friendly interfaces and other effective communication means (e.g. instructions by narration and iconic interfaces).

The future research direction is the definition of patterns to specific sensor data, in order to allow users to be notified according to different patterns and not only static conditions and thresholds. Extending iTaaS framework to support emerging service provision models dictated by the adoption of new generation telecommunication protocols (5G and LPWAN respectively) as well as implementation of an inter-operable Semantic Web of Things IoT feamework, are challenging directions for future research.

Appendix

The next XML file shows registrations of two devices used widely in the market (the Nonin Onix and the Polar H7).


```
<? xml version=‘‘1.0” encoding=‘‘UTF–8” ?>
 <sensors>
        <device name=‘‘NoninOnix’’>
            <uuid id=‘‘0aad7ea0–0d60–11e2–8e3c–0002a5d5c51b”>
                <values name=‘‘SPO2” type=\‘‘%’’>
                    <format>FORMAT_UINT8</format>
                    <position>7</position>
                    <multi>1</multi>
                </values>
                <values name=‘‘pulse_Rate’’ type=‘‘bpm’’ picture=”1”>
                    <format>FORMAT_UINT8</format>
```

```
                    <position >[8,9]</position>
                    <multi >[256,1]</multi>
                <values>
            </uuid>
        </device>
        <device name=‘‘Polar H7”>
        <uuid id=‘‘00002a37−0000−1000−8000−00805f9b34fb”>
                <values name=‘‘pulse_Rate’’ type=‘‘bpm”>
                    <format>FORMAT_UINT8</format>
                    <position >1</position>
                    <multi>1</multi>
                </values>
            </uuid>
        </device>
</sensors>
```


References

[1] K. Bouchard, B. Bouchard, A. Bouzouane, Spatial recognition of activities for cognitive assistance: Realistic scenarios using clinical data from alzheimer?s patients, Journal of Ambient Intelligence and Humanized Computing 5 (5) (2014) 759–774. doi:https://doi.org/10.1007/s12652-013-0205-8.

[2] S. Balabanis, S. Sotiriadis, E. Petrakis, Internet of things platform in cloud computing for enhanced living, Cloud Computing for Enhanced Living Environments: Algorithms, Architectures and Platforms (Cloud4ELE), IEEE Cloud Computing 3 (6) (2016) 28–34. doi:http://doi.ieeecomputersociety.org/10.1109/MCC.2016.128.

[3] L. Xu, W. He, S. Li, Internet of things in industries: A survey,

IEEE Transactions on Industrial Informatics 10 (4) (2014) 2233–2243. doi:10.1109/TII.2014.2300753.

[4] McKinsey, Company, The internet of things: Mapping the value beyond the hype (Jun. 2015).
URL http://fognetworks.org/news/McKinsey-report/

[5] A. R. Biswas, R. Giaffreda, Iot and cloud convergence: Opportunities and challenges, in: IEEE World Forum on Internet of Things (WF-IoT), Seoul, South Korea, 2014, pp. 375–376. doi:10.1109/WF-IoT.2014.6803194.

[6] M. P. Andersen, G. Fierro, D. E. Culler, Enabling synergy in iot: Platform to service and beyond, in: IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), Berlin, Germany, 2016, pp. 1–12. doi:10.1109/IoTDI.2015.45.

[7] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: State-of-the-art and research challenges, Journal of Internet Services and Applications 1 (1) (2010) 7–18. doi:https://doi.org/10.1007/s13174-010-0007-6.

[8] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the first edition of the MCC workshop on Mobile cloud computing (MCC'12), Helsinki, Finland, 2012, pp. 13–16. doi:10.1145/2342509.2342513.

[9] T. Erl, SOA Principles of Service Design, Prentice Hall, Upper Saddle River, NJ, USA, 2007.
URL https://dl.acm.org/citation.cfm?id=1296147

[10] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. van Kranenburg, S. Lange, S. Meissner, Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model, Springer, Heidelberg, 2013.
URL https://www.springer.com/gp/book/9783642404023

[11] S. Sotiriadis, K. Stravoskoufos, E. Petrakis, Future internet systems design and implementation, cloud and iot services based on iot-a and fiware, in: V. Angelakis, E. Tragos, A. Kapovits, H. Poehls, A. Bassi (Eds.), Designing, Developing and Facilitating Smart Cities: Urban Design to IoT Solutions, Springer, 2017, Ch. 10, pp. 193–207.
URL https://www.springer.com/gp/book/9783319449227

[12] T. Soultanopoulos, S. S., P. E., C. Amza, Data management of sensor signals for high bandwidth data streaming to the cloud, in: IEEE Sarnoff Symposium, Newark, New Jersey, USA, 2016, pp. 53–58. doi:10.1109/SARNOF.2016.7846764.

[13] P. Tsiachri-Renta, S. Sotiriadis, E. Petrakis, Healthcare sensor data management on the cloud, in: Workshop on Adaptive Resource Management and Scheduling for Cloud Computing (ARMS-CC 2017), ACM Symposium on Principles of Distributed Computing 2016 (PODC 2017), Washington, DC, USA, 2017, pp. 25–30. doi:10.1145/3110355.3110359.

[14] T. Soultanopoulos, S. S., P. E. E., C. Amza, Internet of things data management for bluetooth low energy (ble) devices, in: Workshop on Adaptive Resource Management and Scheduling for Cloud Computing (ARMS-CC 2016), ACM Symposium on Principles of Distributed Computing 2016 (PODC 2016), Chicago, Illinois, USA, 2016, pp. 25–28. doi:10.1145/2962564.2962568.

[15] F. Tao, Y. Cheng, L. D. Xu, L. Zhang, B. H. Li, Cciot-cmfg: Cloud computing and internet of things-based cloud manufacturing service system, IEEE Transactions on Industrial Informatics 10 (2) (2014) 435–1442. doi:10.1109/TII.2014.2306383.

[16] F. Antonopoulos, S. Sotiriadis, E. Petrakis, N. Bessis, A physical access control system on the cloud, in: International Conference on Ambient Systems, Networks and Technologies (ANT 2018), Vol. Pro-

cedia, Computer Science, 130, Porto, Portugal, 2018, pp. 318–325. doi:https://doi.org/10.1016/j.procs.2018.04.045.

[17] C. Theummler, A. Paulin, A. Lim, Determinants of next generation e-health network and architecture specifications, in: IEEE International Conference on e-Health Networking, Applications and Services (HEALTHCOM 2016), Munich, Germany, 2016, pp. 1–6. doi:10.1109/HealthCom.2016.7749460.

[18] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, F. Bu, Ubiquitous data accessing method in iot-based information system for emergency medical services, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1578–1586. doi:10.1109/TII.2014.2306382.

[19] R. Nirmalya, A. Misra, D. Cook, Ambient and smartphone sensor assisted adl recognition in multi-inhabitant smart environments, Journal of Ambient Intelligence and Humanized Computing 7 (1) (2016) 1–19. doi:dx.doi.org/10.1007/s12652-015-0294-7.

[20] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future Generation Computer Systems (FGCS) 27 (7) (2013) 1645–1660. doi:10.1016/j.future.2013.01.010.

[21] D. Miorandi, S. Sicari, F. D. Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges, Ad Hoc Networks 10 (7) (2012) 1497–01516. doi:https://doi.org/10.1016/j.adhoc.2012.02.016.

[22] A. Ciuffoletti, Occi-iot: An api to deploy and operate an iot infrastructure, IEEE Internet of Things Journal 4 (5) (2017) 1341–1348. doi:10.1109/JIOT.2017.2734068.

[23] K. Douzis, S. Sotiriadis, E. Petrakis, C. Amza, Modular and generic iot management on the cloud, Future Gen-

eration Computer Systems (FGCS) 78 (1) (2018) 369–378. doi:https://doi.org/10.1016/j.future.2016.05.041.

[24] A. Ikram, A. Anjum, R. Hill, N. Antonopoulos, S. S. L. Liu, Approaching the internet of things (iot): A modelling, analysis and abstraction framework, Concurrency and Computation: Practice and Experience 27 (8) (2016) 1966–1984. doi:https://doi.org/10.1002/cpe.3131.

[25] T. Kobialka, R. Buyya, C. Leckie, R. Kotagiri, A sensor web middleware with stateful services for heterogeneous sensor networks, in: International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2007), Melbourne, Australia, 2007, pp. 491–496. doi:10.1109/ISSNIP.2007.4496892.

[26] O. A. W. Group, Openfog architecture overview (Feb. 2016).
URL https://www.openfogconsortium.org/wp-content/uploads/OpenFog-Architecture-Overview-

[27] S. Schreier, Modeling restful applications, in: ACM International Workshop on RESTful Design (WS-REST?11), Hyderabad, India, 2011, pp. 15–21. doi:10.1145/1967428.196743.

[28] C. Cecchinel, M. Jimenez, S. Mosser, M. Riveill, An architecture to support the collection of big data in the internet of things, in: IEEE World Congress on Services (SERVICES), Anchorage, AK, USA, 2014, pp. 442–449. doi:10.1109/SERVICES.2014.83.

[29] S. Guoqiang, C. Yanming, Z. Chao, Z. Yanxu, Design and implementation of a smart iot gateway, in: IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, Beijing, China, 2013, pp. 720–723. doi:10.1109/GreenCom-iThings-CPSCom.2013.130.

[30] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, X. Yang, A survey on the edge computing for the internet of things, IEEE Access 6 (2017) 6900–6919. doi:10.1109/ACCESS.2017.2778504.

[31] A. E. Boualouache, O. Nouali, S. Moussaoui, A. Derder, A ble-based data collection system for iot, in: International Conference on New Technologies of Information and Communication (NTIC), Mila, Algeria, 2012, pp. 1–5. doi:10.1109/NTIC.2015.7368748.

[32] A. Orsino, G. Araniti, L. Militano, J. Alonso-Zarate, A. Molinaro, A. Iera, Energy efficient iot data collection in smart cities exploiting d2d communications, Sensors 18 (6). doi:10.3390/s16060836.

[33] L. Jian, L. D. Xu, H. Cai, Z. Jiang, F. Bu, B. Xu, An iot-oriented data storage framework in cloud computing platform, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1443–1451. doi:10.1109/TII.2014.2306384.

[34] T. Li, Y. Liu, Y. Tian, S. Shen, W. Mao, A storage solution for massive iot data based on nosql, in: IEEE International Conference on Green Computing and Communications, Besancon, France, 2012, pp. 50–57. doi:10.1109/GreenCom.2012.18.

[35] E. Bertino, K. Choo, D. Georgakopolous, S. Nepal, Internet of things (iot): Smart and secure service delivery, ACM Transactions on Internet Technology, Special Issue on Internet of Things (IoT): Smart and Secure Service Delivery 16 (4) (2016) 22–22:7. doi:10.1145/3013520.

[36] A. Rousalis, S. Sotiriadis, E. Petrakis, Management of farm data on the cloud, in: IEEE International Conference on Advanced Information Networking and Applications (IEEE AINA 2018), Cracow, Poland, 2018.

[37] X. Wang, W. Cheng, P. Mohapatra, T. Abdelzaher, From today's intranet of things to a future internet of things: A wireless and mobility-related view, Wireless Communications 17 (6) (2010) 44–51. doi:10.1109/MWC.2010.5675777.

[38] Cisc18, Cisco fog computing solutions: Unleash the power of the internet of things

(2018).
URL https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-solutions.pdf

[39] D. Linthicum, Edge computing vs. fog computing: Definitions and enterprise uses (2018).
URL https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html

[40] T. Garfinkel, M. Rosenblum, A virtual machine introspection-based architecture for intrusion detection, in: Proceedings of Network and Distributed Systems Security Symposium, 2012, pp. 191–206.
URL https://suif.stanford.edu/papers/vmi-ndss03.pdf

[41] K. Nagaraj, C. Killian, J. Neville, Structured comparative analysis of systems logs to diagnose performance problems, in: Proceedings of the 9th USENIX conference on Networked Systems (NDSI'12), San Jose, CA, USA, 2012, pp. 22–26.

[42] A. Bhattacharyya, S. Jandaghi, S. Sotiriadis, C. Amza, Semantic aware online detection of resource anomalies on the cloud, in: IEEE CloudCom, Helsinki, Finland, 2016, pp. 134–143. doi:10.1109/CloudCom.2016.0035.

[43] P. Massonet, S.Dupont, A. Michot, A. Levin, M.Villari, Enforcement of global security policies in federated cloud networks with virtual network functions, in: International Symposium on Network Computing and Applications (NCA 2016), Cambridte, MA, USA, 2016, pp. 81–84. doi:10.1109/NCA.2016.7778597.

[44] D. Guinard, V. Trifa, Building the Web of Things, Manning Publications Co., Greenwich, CT, USA, 2016. doi:https://webofthings.org/book/.
URL https://dl.acm.org/citation.cfm?id=1296147

[45] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, K. Taylor, Iot-lite: A lightweight semantic model for the internet of things and its use with dynamic semantics, Personal and Ubiquitous Computing 21 (3) (2017) 475—487.
URL https://doi.org/10.1007/s00779-017-1010-8

[46] M. Al-Osta, B. Ahmed, G. Abdelouahed, A lightweight semantic web-based approach for data annotation on iot gateways, in: Proceedings of the 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) - Volume 113, 2017, pp. 186–193.
URL `https://doi.org/10.1016/j.procs.2017.08.339`

[47] W. Wang, S. De, R. Toenjes, A comprehensive ontology for knowledge representation in the internet of things, in: Proceedings of IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, 2012, pp. 1793—-1798. `doi:10.1109/TrustCom.2012.20`.

[48] P. Barnaghi, M. Presser, Publishing linked sensor data, in: Proceedings of the 3rd International Conference on Semantic Sensor Networks - Volume 668, SSN'10, 2010, pp. 1–16.
URL `http://dl.acm.org/citation.cfm?id=2878789.2878790`

[49] N. Le, M. Hossain, A. Islam, D.-Y. Kim, Y.-J. Choi, Y. Jang, Survey of promising technologies for 5g networks (12 2016). `doi:10.1155/2016/2676589`.

[50] N. Rozanski, E. E. Woods, Software Systems Architecture: Working With Stakeholders Using Viewpoints Addison-Wesley Professional, 2005.
URL `https://www.viewpoints-and-perspectives.info`