Chapter 2
Fog of Things: Fog Computing in Internet of Things Environments

Leandro Andrade, Cleber Lira, Brenno de Mello, Andressa Andrade, Antonio Coutinho, and Cássio Prazeres

2.1 Introduction

The Internet of Things (IoT) has matured in recent years allowing market solutions to emerge using the technology in different directions. Several domain scenarios such as Smart City, Smart Transportation, Connected Vehicles, Smart Health, Smart Building, Industrial Internet, Smart Farming, Smart Supply Chain, and others have been the focus for products and applications. The increasing demand for more local processing and more ways to protect the data before it goes to the cloud has made Fog Computing more relevant.

In a typical implementation of IoT, the data collected from sensors is stored and processed in cloud servers. Although this type of approach is commonly used, it has some limitations according to Abdelshkour [2]: connectivity to the cloud is a precondition and some IoT systems need to be able to work even when connection is temporarily unavailable; high demand for bandwidth, as a result of sending every bit of data over cloud channels; and slow response time (high latency) and limited scalability as a result of dependency on remote servers hosted in centralized data centers.

L. Andrade (✉) · B. de Mello · A. Andrade · C. Prazeres
Federal University of Bahia (UFBA), Salvador, Brazil
e-mail: leandrojsa@ufba.br; brenno.mello@ufba.br; dsandrade@dcc.ufba.br; prazeres@ufba.br

C. Lira
Federal Institute of Bahia (IFBA), Salvador, Brazil

Federal University of Bahia (UFBA), Salvador, Brazil
e-mail: cleberlira@ifba.edu.br

A. Coutinho
State University of Feira de Santana (UEFS), Feira de Santana, Brazil
e-mail: acoutinho@uefs.br

© Springer Nature Switzerland AG 2020
V. Roesler et al. (eds.), *Special Topics in Multimedia, IoT and Web Technologies*,
https://doi.org/10.1007/978-3-030-35102-1_2

In order to overcome the aforementioned Cloud Computing limitations, Bonomi et al. [7] proposed a Fog Computing paradigm which brings some operations of computing and storage close to the edge of the network. The adoption of Fog Computing does not exclude Cloud Computing and brings characteristics such as low latency, location awareness, support for mobility, a strong presence of streaming and real-time applications, and scalability for a large number of fog nodes. Based on the Fog Computing paradigm for IoT systems, Prazeres and Serrano [21] introduced the Fog of Things (FoT) paradigm. It proposes the cooperative use of network edge processing capability with Cloud servers to perform data processing and service delivery on devices, small local servers, and gateways (very small servers).

The SOFT-IoT platform is a concrete implementation of the FoT paradigm [21], that uses microservice infrastructure distributed along devices (gateways, local servers and cloud servers) in the IoT system. The microservices of SOFT-IoT are deployed on an Enterprise Service Bus (ESB) infrastructure based on the OSGi, which is a specification for middlewares that combines the functionality of a Service-Oriented Architecture (SOA) and modularity [22]. This chapter presents the Fog of Things paradigm and the SOFT-IoT platform. In addition, it describes the characteristics and architecture of FoT and the technologies used to implement SOFT-IoT.

The chapter is organized as follows. Section 2.2 describes FoT paradigm. Section 2.3 presents FoT implementation following IoT architecture divided in different layers. Section 2.3 introduces the SOFT-IoT platform in a conceptual perspective. Section 2.5 introduces some topics of research related to SOFT-IoT platform. Lastly, we present final remarks and future works in Sect. 2.6.

2.2 Fog of Things (FoT)

The FoT was proposed with the objective of taking advantage of the benefits that Fog Computing can bring to the IoT. In the FoT paradigm part of the data processing capacity and service delivery operations are processed locally on small servers. The Fog of Things paradigm goes beyond Fog Computing in the following aspects: (1) using all the edge processing capabilities of the network through data processing and service delivery on devices; (2) defining IoT services at the edge network; (3) distributing the IoT services on the edge of the network through a message and service-oriented middleware.

IoT platforms based in FoT paradigm can be deployed in a hybrid fog in which some of the IoT services will be deployed in the gateways and others in one or more servers. These implanted in the servers will be responsible for the self-organizing management of system, security management (authentication and identification), and storage. These deployed at gateways will be responsible for the basic services of an IoT platform, such as device access, discovery, composition, location, and others.

Some of the concepts used in FoT are based on the Architectural Reference Model (ARM) [5] for the IoT. The ARM was developed by the partners of the European FP7 Research Project IoT Architecture (IoT-A) with the technical objective of creating a generic architecture reference that could be useful to build a real IoT system. The IoT-A project was ran between 2010–2013, providing several resources (models, views, best practices, etc.) and bridging existing developments in the IoT domain.

As shown in Fig. 2.1, the FoT paradigm is composed of components, such as applications, devices, gateways, servers, messaging-oriented middleware, and security providers. In Prazeres and Serrano [21], the FoT is presented with the following organization characteristics:

- **FoT-Device:** identified as “D” in Fig. 2.1 reuses the concept proposed by ARM IoT [5] in which “devices are technical artifacts that perform real-world integration with the digital world of the Internet.” Thus, Bassi et al. define three types of IoT devices: sensor (e.g., temperature sensor), actuator (e.g., switch), and label (e.g., RFID). They also transform raw data into structured content following Linked Data guidelines;
- **FoT-Gateway:** the gateway (“G” in Fig. 2.1) is the basic communication node in the network managed by FoT. Its basic aim is to translate communication layer protocols (Ethernet, WiFi, ZigBee, Bluetooth, etc.) to the HTTP protocol and to the rest of the IoT system. FoT offers access to FoT-Devices and other IoT services. Therefore, the applications can abstract the protocol of communication with the devices and access them in a standardized way for each type of device, as happens in most Web applications;

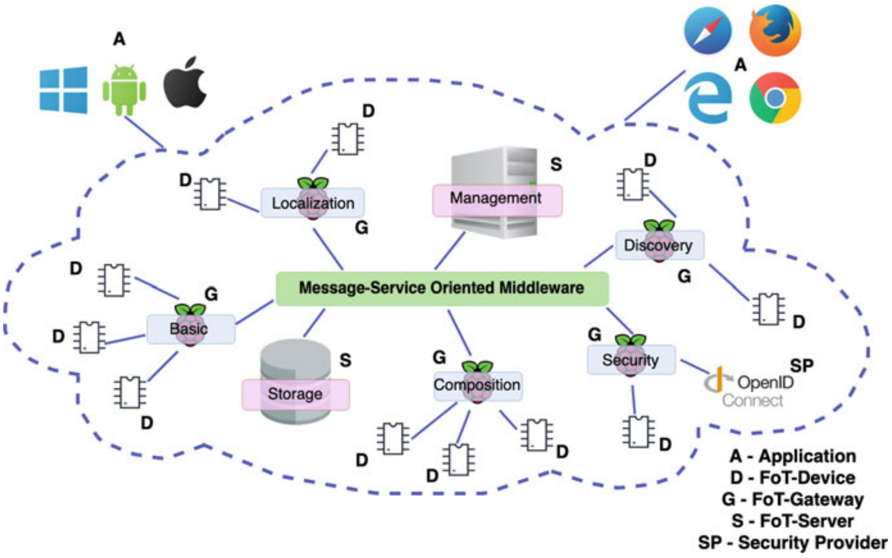


Fig. 2.1 Fog of Things paradigm

- **FoT-Server:** the server “S” in Fig. 2.1 can be two types: a special type of gateway that has platform management features or a special feature type (see definition of feature below) that provides specific information such as historical device data that is not supported by the gateways. It is important to note, as shown in Fig. 2.1, that FoT-Gateways can provide storage features, such as data transformation, but in general they have a lower capacity of storage and processing than FoT-Servers. FoT-Server is called a server with the name of the main functionality it offers in FoT. For example, “management server,” “storage server,” “authorization server,” etc.;
- **Enterprise Service Bus (ESB):** an ESB is a distributed infrastructure based on open standards which combines messages, Web Services, data transformation, intelligent routing, invocation, and service mediation to facilitate the integration of securely and reliably distributed applications and services [9];
- **Application:** identified as “A” in Fig. 2.1, it is any type of application based in HTTP, which is provided by the FoT-Gateways or FoT-Servers to access the FoT-Devices and provide interaction with these devices for the user. Thus, applications can be Web, mobile (Android, iPhone, or Windows Phone), or even traditional desktop applications.
- **Security Provider:** a security provider (“P” in Fig. 2.1) is a FoT-Server and is treated separately given the importance of the security aspect for an IoT platform.

In addition to the previously described components, for a better understanding of the operation of the FoT, some concepts are defined below.

- **Resource:** reuses the concept of the IoT-A ARM [5], where resources are software components that provide data to or from devices. In the IoT-A ARM, there is a distinction between “on-device resources” and “network resources.” The first, as the name suggests, are software components which are deployed on the device to provide access to it. The second are resources available somewhere on the network. For example, a database for storing historical data of the FoT-Devices;
- **IoT Service:** it is based in the concept of IoT ARM [5], where IoT service provides an open and standardized interface, which offers all the necessary functionality to interact via network with IoT resources or devices. On the FoT, all IoT services will be implemented as RESTful Web Services;
- **User:** is someone who uses IoT services of the FoT. In this way, a user can be, among other things, a person, an application, or another service;
- **Profile:** as can be seen in Fig. 2.1, each node in the FoT has a label: basic, discovery, composition, management, etc. These labels are the names of the node profiles which define a set of functionalities that the node must offer, via an IoT Service, to the platform as a whole or to external applications/services. The purpose of profiles is to facilitate and optimize platform management dynamically and in a self-organizing way.

Profiles were defined to guarantee the functionalities of the FoT to support dynamic, self-organizing capabilities. Other profiles can be defined and deployed even after the platform development and deployment. The Basic Profile implements

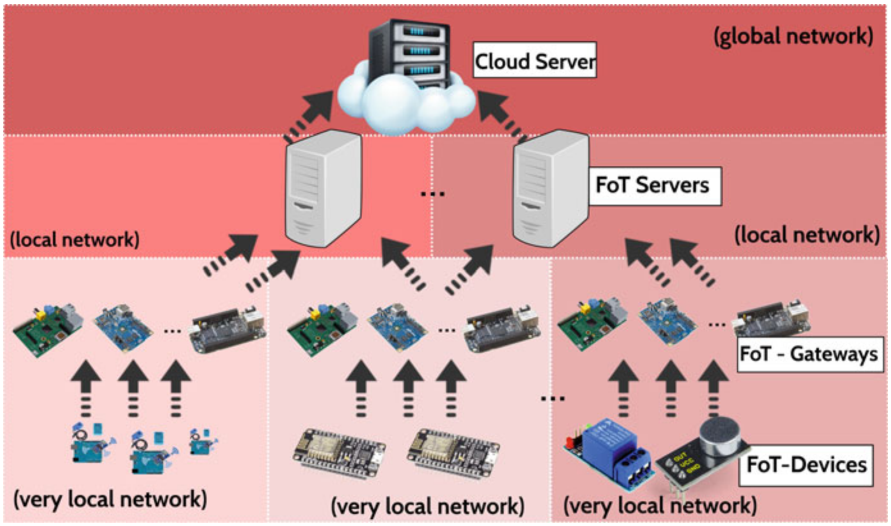


Fig. 2.2 Fog of Things paradigm, based in [4]

the basic functionalities that are: communication protocol mapping for HTTP; access to devices (for example, provide access to device features such as getting temperature, turning on/off a lamp, and changing the temperature of the air conditioner); automatic device configuration; and automatic publishing (exposing device features such as a RESTful Web Service). The FoT-Gateway which has only the basic features belongs to the Basic Profile. However, if necessary, this gateway can receive new features to incorporate one of the other profiles such as Discovery, Composition, Localization, Storage, Security, and Management.

Figure 2.2 presents the flow of transition of data in a FoT architecture, from the IoT devices to the cloud server. It shows that the flow of data ascends from very local networks (from IoT Devices to FoT-Gateway) to a global network in cloud servers. This structure is suitable to support mobility in devices, faster answers in layers of local network, and autonomy to keep the IoT system in operation when there is no communication with the cloud server.

2.3 IoT Architecture with Fog of Things

IoT development depends on the design of new applications and business models. In recent studies, Khan et al. [15] and Al-Fuqaha et al. [3], divided the structure of IoT into five layers: (1) Perception Layer, (2) Network Layer, (3) Middleware Layer, (4) Application Layer, and (5) Business Layer. Figure 2.3 shows IoT Architecture following the FoT paradigm. In the remainder of this section, we explain each one of these layers and some security aspects under development in the FoT.

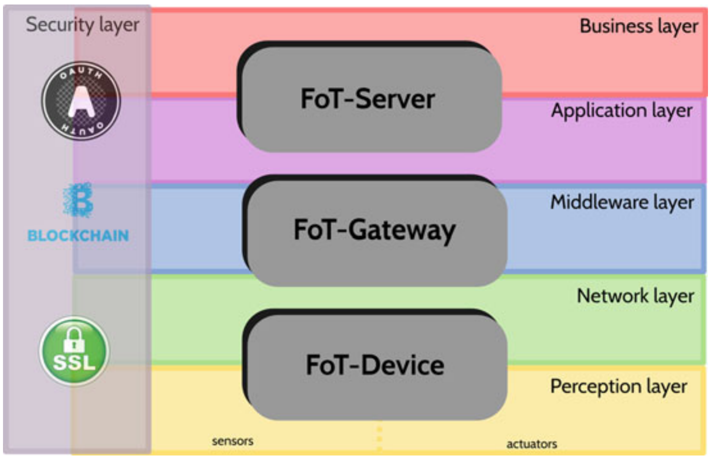


Fig. 2.3 IoT architecture with FoT

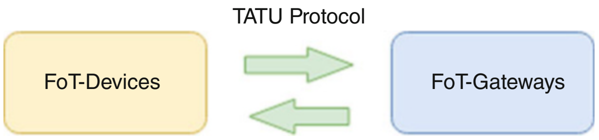


Fig. 2.4 Communication between FoT-devices and FoT-gateways

2.3.1 Perception and Network Layers

The Perception Layer contains the FoT-Devices. These devices can be categorized, as described by Bassi et al. [5], as actuators, identifiers, and sensors. Actuators can change the environment, for example, devices with relays (electrically operated switches) that can turn other devices on/off. Identifiers can identify people and collaborate with the system to allow people access. Sensors can collect data from the environment, for example, devices with a DHT11 sensor able to collect the temperature and humidity of the environment. The FoT-Devices interact with the other elements in the FoT through the Network layer. This layer transfers the information from the Perception Layer to the Middleware Layer. On SOFT-IoT the transmission can be wireless or wired. Also, we can use technology 4G, Bluetooth, and infrared depending on the device.

Figure 2.4 is based on the Fig. 2.3, but with a focus on the communication between FoT-Devices in the Perception Layer, the FoT-Gateways in the Middleware Layer, and the use of The Accessible Thing Universe (TATU) protocol through the Network Layer to connect these layers.

TATU[1] arose from the need to develop a messaging pattern so that communication between devices was facilitated even for devices that did not use Message Queuing Telemetry Transport (MQTT), it functions as an extension of MQTT,[2] developed as a set of solutions to make the Perception Layer simpler and more intuitive. We call this protocol TATU Thing Protocol for Internet (TATU TPI). This protocol follows the JavaScript Object Notation (JSON) format.

2.3.1.1 TATU Methods

In the FoT, as the IoT, there are many types of data flow, generally used to meet data analysis demands and real-time data presentation, e.g., temperature and luminosity by a dashboard.[3] TATU protocol offers two methods for data requisition, GET and FLOW [6].

The traditional GET requisition pattern (see Fig. 2.5a) is useful when the FoT-Gateway or application needs to collect some data in real-time, so either the application or the FoT-Gateway triggers the communication by sending a GET request. Then, the FoT-Device collects the required data and returns its value to whoever demanded it.

The FLOW requisition pattern (see Fig. 2.5b) was proposed to avoid continuous communication between FoT-Gateway/application and an FoT-Device, in this data flow pattern, the FoT-Gateway/application sets up a time range and sample amount (one array with multiple values) which it expects to receive periodically. So, the FoT-Device will behave proactively, sending the arrays automatically to the requester as soon as the parameters are set up. The time between requisitions can be changed at any time as a request from either application or FoT-Gateway.

The GET pattern can perform the same functions provided by the FLOW data flow pattern. However, when compared to FLOW, the GET pattern tends to demand more network utilization to transfer the samples, this is due to the GET data flow pattern being based on request and responses, whereas the FLOW pattern is mostly based on responses.

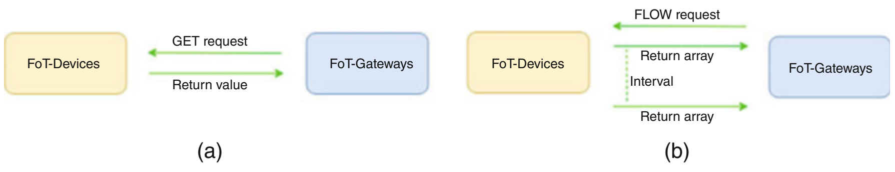


Fig. 2.5 TATU's methods GET and FLOW. (**a**) GET requests. (**b**) FLOW requests

[1]TATU is available on: https://github.com/WiserUFBA/TATUDevice.

[2]MQTT Protocol: http://mqtt.org/.

[3]A tool for management and monitoring of metrics and indicators, projected to ease the comprehension and decision-making.

2.3.2 Middleware Layer

The Middleware Layer provides an interface between FoT-Device (sensors and actuators) and the rest of the IoT system through FoT-Gateways. In the FoT paradigm the Middleware Layer is divided in two parts with different functionalities and orientations: the message-oriented part, and the service-oriented part.

The message-oriented part provides communication between the FoT-Devices using the TATU protocol. This functionality implements a virtual communication between FoT-Devices through TATU protocol and offers uniform access to the sensor data for FoT-Gateways in the Middleware Layer.

The service-oriented part provides access to the collected sensor data for the Middleware and Application Layer in the FoT based IoT systems. It is also responsible for gathering the sensor data obtained from the message-oriented part and storing it in a local database. The service-oriented part offers an interface for the other modules in the application layer to have access to the collected data.

2.3.3 Application and Business Layers

The Application Layer is responsible for providing the services requested by the users. Applications can be deployed on devices with limited capacity (e.g., Raspberry Pi FoT-Gateways) and servers that are located on the edge network (FoT-Server) or servers located in the cloud (cloud server). In FoT paradigm an application is any kind of application that uses the HTTP-based REST API to access IoT services and offer interaction with services to the user.

In FoT, applications run on a service bus. The service bus enables the dynamic deployment of software through a base infrastructure, which supports communication between applications. Applications in the FoT paradigm adopt the Microservice architectural style. Microservices enable the creation of a system from a collection of small and isolated services capable of managing their own data [13].

The academic interest in Microservices for the development of IoT applications is recent [10]. Newman [19] lists some benefits, discussed below, when adopting Microservices as a solution in the development of applications.

- **Technology Heterogeneity:** each part of the application can be implemented with different technologies. So if a part of the application needs to improve its service quality it is possible to decide to use a different technology stack that is more adequate to achieve the required Quality of Service (QoS) levels. For example, in Fig. 2.6 (side b), each application can be built with different technologies to meet each objective.
- **Scaling:** with Microservices it is possible to scale parts of the application according to the need. Thus, it is possible to execute other parts of the application on a device with less computational power.
- **Ease of Deployment:** with Microservices it is possible to make a change to a single service and deploy it independently of the rest of the system. If a

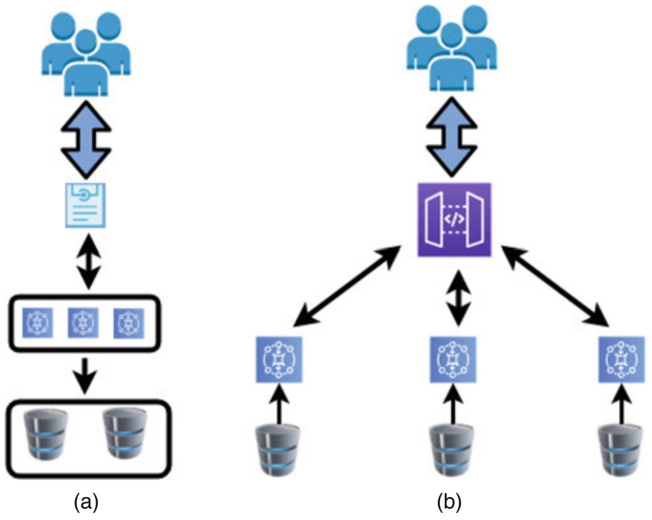


Fig. 2.6 Monolithic versus microservices architecture [13]. (**a**) Monolith. (**b**) Microservices

problem does occur, it can be isolated quickly to an individual service, making fast rollback easy to achieve.

- **Organizational Alignment:** with Microservices, it is possible to better align the application architecture with the organizational structure.
- **Composability:** One of the key issues in service-oriented architectures and in distributed systems is the possibility of improving application reusability. With Microservices, it is possible for a functionality to be consumed in different ways for different purposes.

The Business Layer manages the general activities and services of an IoT system. The responsibilities of this layer are to build a business model, graphs, flowcharts, etc., based on data received from the Application Layer. In the FoT paradigm, the aim of the Business Layer is to enable the creation of different data visualizations that take into account the hierarchical levels in an architecture involving fog and cloud. In this context, Pinto [20] presents a model for data visualization that organizes the presentation at different levels of abstraction. This model intends to provide multiple forms of visualization over the same dataset from its generation in the FoT-Devices until its storage in the cloud, passing through the FoT-Gateways and FoT-Servers.

2.3.4 Security Layer

Current IoT systems integrate physical objects, sensor data, and computing resources into a large network over the Internet. IoT security is an area that aims to guarantee the privacy, confidentiality, and availability offered by an IoT ecosystem.

In such environments, potential security vulnerabilities and privacy violations need to be addressed based on trust and suitable mechanisms which developers can use to build secure, scalable, and reliable distributed solutions. It involves ensuring the security of IoT infrastructure components such as data, network, services, and devices.

Each layer of the IoT architecture with the FoT paradigm shown in Fig. 2.3 can employ mechanisms for addressing related security issues. In the following sections, general models, security threats, and examples of mechanisms used to improve trust, security, and privacy at every level of the IoT architecture are discussed.

2.3.4.1 Security Models and Concepts in IoT

Major reference architectures such as the Architectural Reference Model (ARM) [5] and the Industrial Internet Reference Architecture (IIRA) [18] offer standards-based architectural templates which enable IoT system architects to design solutions based on a common framework and concepts. Also, these architectural models: (1) define essential concepts and properties such as trust, security, and privacy; (2) discuss potential security issues and approaches for IoT architecture; and (3) define security models and functionality for IoT systems that serve as solid foundations upon which it is possible to build complex solutions that guarantee those properties.

The ARM consists of three interconnected parts: the IoT Reference Model (RM), the IoT Reference Architecture (RA), and a set of guidelines or best practice. The RM provides a set of models that are used to define certain aspects of the architectural views such as IoT domain model, information model, communication model, functional model, and finally models for security, trust, and privacy.

Based on the RM, the RA consists of a set of views that represent structural aspects of the system and perspectives that focus on the quality of the system. Figure 2.7 shows the IoT-A ARM functional view that proposes a layered model of functional groups which maps the concepts introduced in the ARM domain model together with a set of essential functional components that an IoT system should provide [8].

The following functional components were proposed in the RA security group:

- **Authorization (AuthZ)**—The AuthZ component is a front end for managing policies and performing access control decisions based on access control policies. This access control decision can be called whenever access to a restricted resource is requested. For example, this function is called inside the IoT service resolution component to check if a user is allowed to perform a look-up on the requested resource. This is an important part of the privacy protection mechanisms.
- **Authentication (AuthN)**—The AuthN component is involved in both user and service authentication. It checks the credentials provided by a user, and, if valid, it returns an assertion as a result, which is required to access the IoT services.

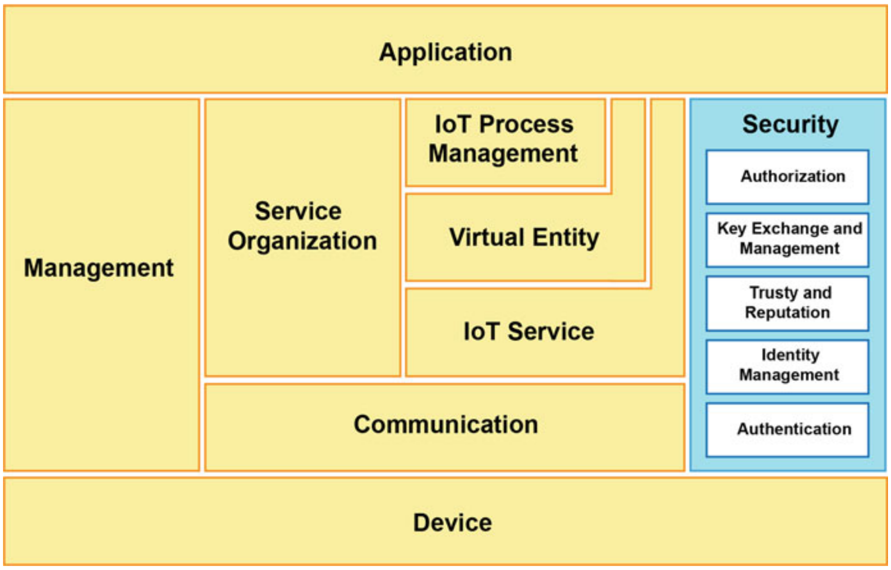


Fig. 2.7 IoT-A ARM functional view and security components

Upon checking the correctness of the credentials supplied by a newly joining node, it establishes secured contexts between this node and other entities.

- **Identity Management (IM)**—The IM component addresses privacy questions by issuing and managing pseudonyms and accessory information to trusted subjects so that they can operate (use or provide services) anonymously.
- **Key Exchange and Management (KEM)**—The KEM component is involved in enabling secure communications between two or more IoT-A peers that do not have initial knowledge of each other or whose interoperability is not guaranteed, ensuring integrity and confidentiality.
- **Trust and Reputation Architecture (TRA)**—The TRA component collects user reputation scores and calculates service trust levels. Its functions can be invocated at a given remote entity to request or provide (recommendations or feedback) reputation information about another entity.

The Industrial Internet of Things (IIoT) has to consider safety more heavily than in the standard IoT. Information leaks in IIoT can cause not just a loss of reputation and money but also the loss of human lives. Critical incidents can be a result of system operations that do not occur in a timely and correct manner.

The IIRA model was first published in 2015 by the Industrial Internet Consortium (IIC), an open membership organization founded by AT&T, Cisco, General Electric, IBM, and Intel to accelerate the IIoT technology adoption. It focuses on the industrial sector where cyber-physical systems and other objects have fast become Internet-enabled, and security is an overall critical problem.

The industrial area introduces the concept of Operational Technologies (OT), that is the hardware and software that detects or causes a change through the direct monitoring and or control of physical devices processes and events in the enterprise. The Information Technology (IT) and OT convergence are critical in IIoT security because it has given remote access to control systems where conventional control systems usually are on an isolated network. An example is giving a Programmable Logic Controller (PLC) an IP address and then connecting it to the Internet.

Also, the concept of brownfield deployments is prominent in factory environments, where solutions need to be able to coexist and interoperate with existing legacy systems with no security features. Instead of replacing industrial types of equipment that are tough to change, industrial deployments often have these new and old systems side-by-side and work with each other.

As part of the IIRA model, the ICC members have defined the Industrial Internet Security Framework (IISF), a common security approach to assessing cybersecurity in the IIoT systems. The IISF presents fundamental security concepts that lay the foundation and architectural decisions for IIoT platforms.

These concepts are based on three main security definitions widely known as the CIA triad: confidentiality, integrity, and availability. The CIA triad is a well-known model designed to guide policies for information security within an IT organization. In this context, confidentiality is a set of rules that limits access to information, integrity is the assurance that the information is trustworthy and accurate, and availability is a guarantee of reliable access to the information by authorized people.

However, IoT systems pose extra challenges to the CIA triad due to the large amount of transmitted data, the variety of data formats, the heterogeneity of devices and network technologies, and the continuously growing number of data sources. This new computer scenario demands a high level of privacy, which is related to the right of an individual or group to control who can access or manage system components and information. This concept ties with confidentiality, as individual entities should be able to see specific data while others should not. It can have stakeholder specific requirements based on different commercial markets. Also, it should be clearly defined to users, so they know how their data is being used.

The IISF model also emphasizes safety, reliability, and resilience as essential and related concepts. In industrial environments, where system failures can lead to different types of security risks, safety concerns the necessity of a cyber-physical system to operate without directly or indirectly causing damage to the health of users. Reliability is the ability of a system to perform its required functions at a needed time, and the guarantee that security protocols do not interfere with system functions. Resilience is the ability of a system to avoid, absorb, or manage dynamic adverse conditions, keeping their state under control.

Figure 2.8 presents the related security concepts in IoT. All these related concepts lead to the central concept of trustworthiness, which is the degree of confidence that the system will perform as intended concerning all key system characteristics.

The IISF also defines functional and other concepts involved in the development of a security framework which can also apply to generic IoT systems. The application of these functional concepts into practical ideas involves providing

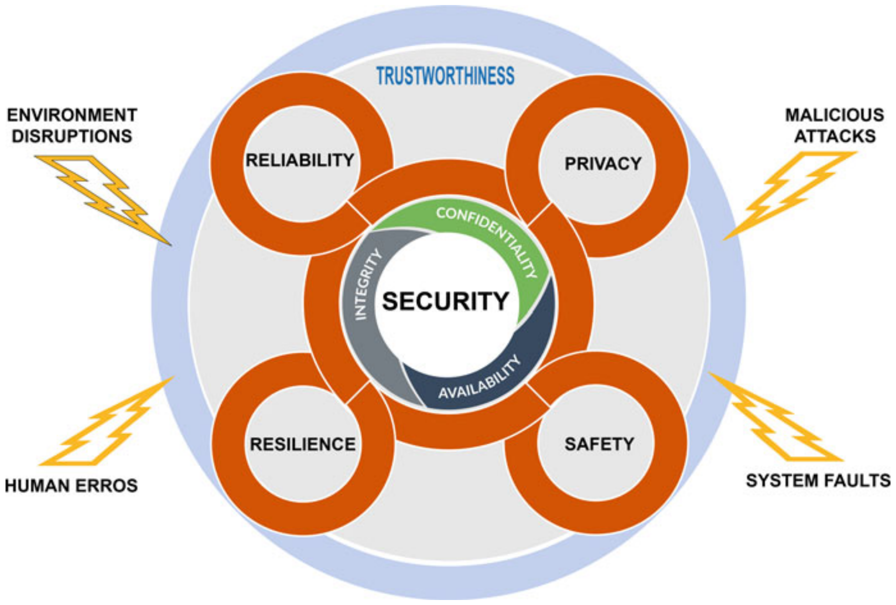


Fig. 2.8 Security concepts in IoT

seamless mechanisms to endpoint protection, communications protection, security monitoring, and security management. Other concepts employed in the proposed framework, such as public key infrastructure and change control are a driving force for providing security in real-world IoT platforms.

2.3.4.2 Security in Perception and Network Layer

The Perception Layer may involve different devices that perform actions on the physical environment based on collected data. There are several kinds of sensors and actuators, sensing technologies and types of data transmission. All these possibilities could also be a target of cybersecurity threats at the Perception Layer.

The IoT devices are vulnerable to a variety of attacks that may try to capture or replace it with a malicious node and to compromise the security of the IoT application [16]. Most of the current IoT literature treats the IoT security from the network or software perspective. However, embedded IoT systems are designed to perform specific applications based on a mix of hardware and software components. In the Perception Layer perspective, the root of trust coming from hardware is always the best and most reliable solution. If the edge devices are compromised, then the entire system may be compromised. Dedicated security modules in the microcontroller/SoC used in the IoT edge devices can help in designing better protection mechanisms [16].

The connection to the Internet in IoT is mandatory, where several heterogeneous devices can communicate with each other in the ubiquitous network. Tamper resistance and encryption schemes to protect sensitive data are used to deal with a range of communication security issues from the Perception Layer to the cloud data center.

MQTT and COAP are the most used protocols to access IoT edge devices. Neither of these protocols use any security mechanisms by default. Although the option to add an optional security layer in the form of TLS/SSL for MQTT and DTLS for COAP is possible, it creates additional overhead in terms of processing and bandwidth.

In the FoT paradigm, the primary function of the Network Layer is transmitting the information received from the FoT-Devices at the Perception Layer for processing in the FoT-Gateways, FoT-Servers, or cloud datacenters. To enable this, there are many communication standards, such as 5G, WiFi, WPAN, and Bluetooth. Also, several connectivity technologies can be used at different levels in the same IoT application, such as Zigbee, 6LOWPAN, NFC, and RFID. Each of these edge network technologies may or may not provide security features for data transmission.

Due to the different technologies and number of network channels used in the IoT infrastructures, the Network Layer is highly vulnerable to different types of attacks [14], and there are several security challenges that the IoT deployments are currently facing. In this sense, digital certificates and authentication protocols are fundamental to provide secure and reliable communication between the involved entities. Concerning the risk of different attacks, paradigms such as Fog Computing and Blockchain can provide different solutions to overcome those security threats.

2.3.4.3 Security in Middleware Layer

The Middleware Layer includes different components, such as brokers, persistent data stores, and machine learning processing. Although the Middleware Layer is suitable for providing a reliable and robust IoT application, it is also susceptible to security threats. Different attacks can take control of the entire IoT application or damage the environment by corrupting middleware security. The security of databases and FoT nodes is a critical challenge in the Middleware Layer. A well-defined reference framework and standard for an end-to-end IoT application is not yet available.

In the Middleware Layer, the gateway is a broad component that has an essential role in connecting devices, services, and applications. It accomplishes security functions such as decrypting and encrypting IoT data and translating security protocols for communication between different layers. Also, IoT constrained devices do not have the capabilities to download and install the firmware updates. In this regard, gateways are used to download and apply the firmware updates.

This procedure permits different forms of attack that can be avoided by checking the hash of downloaded code and validity of the signatures for secure firmware updates. Therefore, security challenges for IoT gateway involve protecting encryption keys. Services and functionalities should be restricted for unauthorized users to avoid backdoor authentication or information breach.

2.3.4.4 Security in Application and Business Layers

The Application Layer has specific security issues that are not present in other layers, such as data theft and privacy issues. Also, the security issues in this layer are specific to a domain of applications. A combination of techniques and protocols such as data encryption, data isolation, privacy management, user and network authentication can be used to protect IoT applications against data thefts.

In the Application and Business Layer, access control and authorization mechanisms are critical security functions since the access is compromised, then the complete IoT application becomes vulnerable. Due to potential security issues present in the lower layers, the industry-standard protocols still focus on providing specific security authorization flows for end-to-end cloud-based applications that run in desktop, mobile phones, and living room devices.

The access to constrained resources remains a blocking concern, where conventional solutions already accepted for both Web and cloud applications cannot be directly used in this context. The generic HTTP is a heavyweight protocol and incurs a significant parsing overhead. There are many alternate protocols accessible at the Application Layer that have been deployed for IoT environments such as MQTT, SMQTT, CoAP, XMPP, AMQP, M3DA, and JavaScript IoT. However, as discussed in Sect. 2.3.4.2, these protocols have limited security and authentication features.

OAuth [17] is an example of a protocol that allows users to have access to resources on a website without exposing their credentials. It is an authorization framework, currently in version 2.0, that allows applications to reach user accounts over an HTTP, such as Facebook and GitHub. It works by delegating the authentication of users to the service that hosts the user account and authorizing third-party applications to access the user account. The OAuth-IoT [24] can be an alternative for a flexible authentication and authorization framework for the IoT.

2.3.4.5 Blockchain-Based Security Solutions for the IoT

Currently, IoT trust, security, and privacy services such as identity management, access control, and authentication focus on a client–server architecture [27]. These services are based on the assumption that IoT users have to put all trust in their so-called trusted third parties which possess personal data of users and can see all transactions between users and service providers.

The advance of Blockchain presents an approach to ensure trust, security, and privacy solutions in decentralized systems [1] such as IoT. However, the integration of IoT with Blockchain is recent and needs further research. In [14], a review of Blockchain-based solutions for IoT systems is presented. In Sect. 2.5.3, the Blockchain-related initiatives being taken within the SOFT-IoT project are discussed.

2.4 SOFT-IoT Platform on Fog of Things

The Fog of Things paradigm and its concrete implementation in the SOFT-IoT platform supports deployment in several domains such as sensor networks, smart homes, autonomous vehicles, and medical assistance. In addition, SOFT-IoT allows for the use of different architectures in Fog Computing and/or Cloud Computing [4]. It enables the following computational architectures: personal area sensor networks; personal area sensor network combined with gateway management, thus creating a local area network; personal area sensor network with gateways and servers, thus creating a more robust local area network infrastructure; cloud servers managing local devices; deployment of all infrastructure from the global network to the personal area network. Thus, the SOFT-IoT architecture is flexible and configurable to meet specific or generic infrastructure and application requirements.

In SOFT-IoT, the FoT-Devices are sensor nodes or actuators embedded with a driver (TATUDevice). It is implemented from a lightweight protocol TATU defined by Fog of Things. The gateway in SOFT-IoT has the function of providing sensor data, perform data processing and transformation, or indicating actions to the actuators. These actions are obtained by a communication protocol, in the form of Web Services, making the access to the devices transparent. On the other hand, a SOFT-IoT fog server provides specific information, such as historical device data, that is not supported by the gateways.

SOFT-IoT has as the main component for the implementation of its applications a service bus. ServiceMix, based on the OSGi specification, is used as a service bus on the SOFT-IoT platform. Apache ServiceMix[4] is open source software, implemented in Java, where native ESB/OSGi architecture services provide all the infrastructure necessary to support SOFT-IoT. ServiceMix allows services (also called bundles) to be deployed at run time, also allowing services to share data and objects through relationships and dependencies.

Figure 2.9 introduces Apache ServiceMix and its main modules. ServiceMix is lightweight and portable, with a basic requirement for Java 1.7 support. Also, it supports Spring Framework[5] and Blueprint[6] and is compatible with Java SE or with a Java EE application server. Karaf is the core of ServiceMix and offers the concept of features where package collections can be installed as a group in a running

[4]http://servicemix.apache.org/.

[5]https://spring.io/.

[6]https://aries.apache.org/modules/blueprint.html.

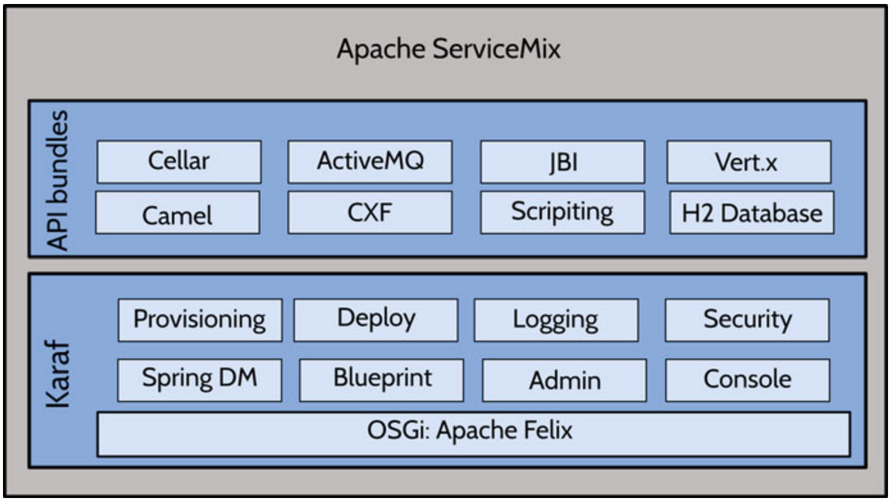


Fig. 2.9 Apache ServiceMix components

OSGi environment. ServiceMix uses ActiveMQ to provide messaging service, CXF to support RESTful services, Cellar for communication, and interactions between different ServiceMix and Camel installations for integration and exchange of data through routes. Finally, ServiceMix contains additional modules such as the H2 Database, which manage the file-based relational database system.

2.4.1 SOFT-IoT Devices

FoT-Devices in the SOFT-IoT platform are sensors/actuators that use the TATU driver. These devices can have low processing and cost, such as Arduino boards.[7]

This type of board does not have a communication module directly soldered to the board, but there are additional options that allow this device to be categorized as sensors or actuator, as set out in Sect. 2.3.1.

The devices also have a semantic description based on ontology. The Zeroconf[8] protocol is implemented in devices and is responsible for enabling devices to have their functionality exposed by RESTful Web services. The TATU driver allows for the communication of the devices with the gateways, allowing the use of the TATU protocol by the gateways through a RESTful Web service. Also, the transformation from raw data to Linked Data content is carried out on devices with an extension

[7]Arduino: https://www.arduino.cc/.

[8]Zeroconf: http://www.zeroconf.org/.

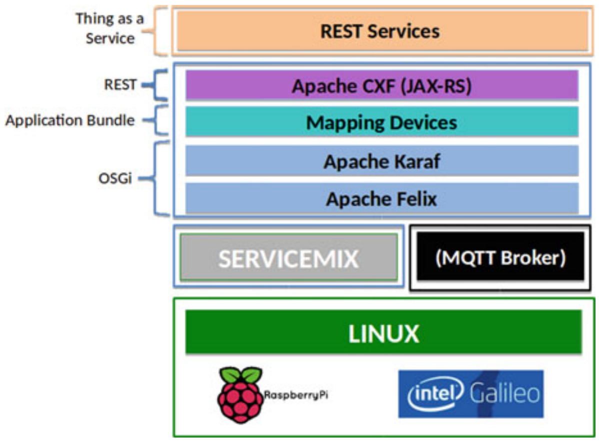


Fig. 2.10 Gateway SOFT-IoT technologies

to TATU protocol. Therefore, that data messages between devices and gateways are using JSON-LD3 format, which is a size-reduced format for representing Linked Data content (Fig. 2.10).

2.4.2 *SOFT-IoT Gateway*

The SOFT-IoT gateway implements the message and service-oriented middleware. Service-oriented middleware provides communication between applications and services deployed at gateways. On the other hand, the message-oriented part provides communication between gateways and devices. To allow communication between gateways and devices using the TATU protocol, the MQTTDriver was developed. The MQTTDriver helps the developer in the task of implementing communication with the devices by enabling virtual communication between the service and the device. Some of the features implemented in the MQTTDriver are: change the value of an actuator; read the value of a sensor; obtain device properties such as IP; and edit the device properties. Figure 2.11 shows the Middleware Layer, its modules, and interactions.

The gateway on the SOFT-IoT platform is generally a low-cost device with limited processing and memory resources. The gateway uses a reduced version of the Linux operating system, a Java virtual machine, an implementation of the OSGi (service-oriented part) specification, and an MQTT server (message-oriented part). The technologies used in the gateway are shown in Fig. 2.10. As shown in this figure, the service-oriented Middleware Layer provides interfaces for applications to access devices via RESTful Web Services (TaaS) using Apache CXF technology. The MQTT broker aims to provide communication through messaging between the gateway and the devices and also between the various gateways. Finally, there is the Mapping Devices component that aims to intermediate the communication between

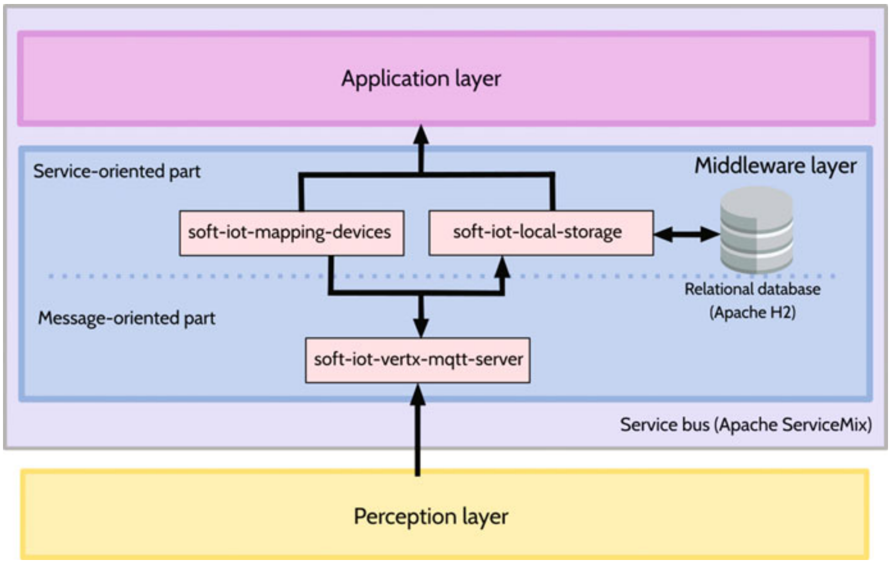


Fig. 2.11 SOFT-IoT middleware layer components and interactions

the devices and the gateways, being implemented following the OSGi specification and deployed to the Apache Karaf server.

The SOFT-IoT platform has a specific module for mapping the devices and translating the TATU communication messages between the FoT-Device and FoT-Gateway. The module **soft-iot-mapping-devices**[9] implements the virtual interfaces of sensors and actuators connected to the platform. This module also works as a translator of TATU messages exchanged between sensors/actuators devices (FoT-Device) and FoT-Gateway, which allow request and response to requests for sensor data and actuator interactions.

Soft-iot-mapping-devices instantiate a set of objects that represent FoT-Devices. These objects contain information related to devices such as unique identifier, device type (sensor and/or actuator), and geolocation data. In addition, they are available for access by the other modules of the platform, serving as interface for accessing the sensors and/or actuators connected to the platform.

The MQTT Broker component provides a server that is capable of handling connections, communication, and message exchange with remote MQTT clients. In the SOFT-IoT platform the MQTT broker is implemented in the module **soft-iot-vertx-mqtt-broker**,[10] an OSGI-based implementation, as a ServiceMix bundle. This module is responsible for enabling communications with remote MQTT clients, which in the case of the SOFT-IoT platform are connected devices. In

[9]https://github.com/WiserUFBA/soft-iot-mapping-devices.
[10]https://github.com/WiserUFBA/soft-iot-vertx-mqtt-broker.

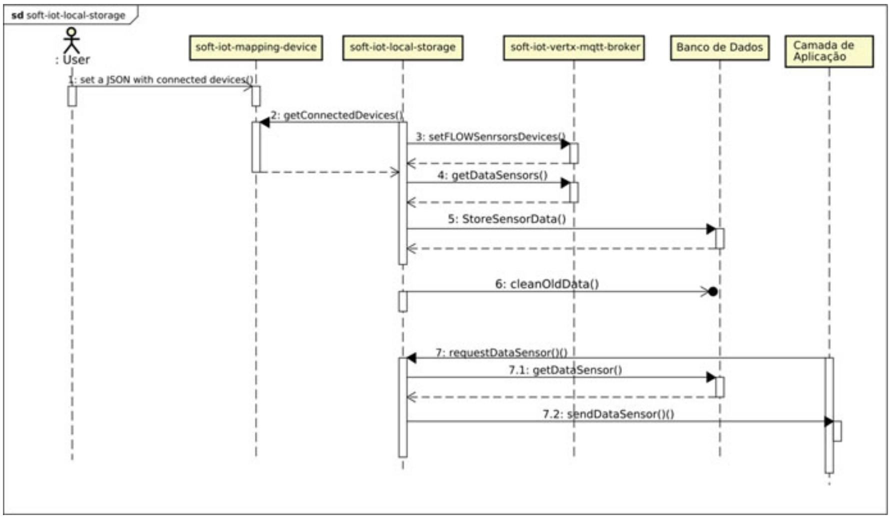


Fig. 2.12 Sequence diagram with soft-iot-local-storage operations and relationships

addition to the advantages of using a modular architecture, using the Vert.x MQTT Server API makes it possible to scale the reactive MQTT agent according to, for example, the number of cores in the system and thus enables horizontal scalability.

The **soft-iot-local-storage**[11] module is responsible for promoting the collection, storage, and internal access of data produced by FoT-Device sensors. This module concentrates the functionalities related to the internal storage of the sensors data, as well as the access to this data by the other SOFT-IoT platform modules.

With soft-iot-mapping-devices support for translation of TATU messages and identification of platform-connected devices, soft-iot-local-storage requests data flow from each sensor connected to the platform in a user-defined period. The request and response messages are exchanged in the MQTT broker implemented by the soft-iot-vertx-mqtt-broker module to be finally stored in the Apache H2 relational database.

Figure 2.12 presents a sequence diagram of the main operations of soft-iot-local-storage with their relationships. With the devices connected to the platform, through user configuration (1. *set a JSON with connected devices*, Fig. 2.12), the soft-iot-local-storage obtains from the module soft-iot-mapping-devices the connected sensors and their respective collected data flow (2. *getConnectedDevices()*, Fig. 2.12) in order to publish in the soft-iot-vertx-mqtt-broker the sensor data requests (3. *setFLOWSenrsorsDevices()*, Fig. 2.12). With the data request, the sensors will publish in the soft-iot-vertx-mqtt-broker information collected at the configured period. The soft-iot-local-storage will in turn collect this data

[11] https://github.com/WiserUFBA/soft-iot-local-storage.

(4. *getDataSensors()*, Fig. 2.12) and store them in the Apache H2 database (5. *StoreSensorData()*, Fig. 2.12). Soft-iot-local-storage also implements a procedure for removing old data, whose period can be configured by the user (6. *cleanOldData()*, Fig. 2.12) and application layer modules can request data from the sensors through soft-iot-local-storage (7. *requestDataSensor()*, Fig. 2.12).

The soft-iot-web-service module[12] exposes the sensor data of the IoT system through a RESTful Web service. It accesses the data stored in the local database, managed by the module soft-iot-local-storage, allowing users to obtain JSON data and information about the sensors.

2.4.3 SOFT-IoT Server

The SOFT-IoT platform implements three types of FoT-Servers, management server, storage server, and server security provider. The first is a server that acts as a special type of gateway and implements all services related to the dynamic self-organization of the SOFT-IoT platform. The latter two are servers that act as a special type of resource and implement services related to storage and security aspects, respectively. The SOFT-IoT platform implements self-organizing features that must be deployed on servers, for example, the management server must implement all services related to the SOFT-IoT platform self-organization.

The SOFT-IoT platform, based on services, is a sufficiently flexible platform for dynamic deployment of new types of FoT-Servers that can offer their services for specific purposes. For example, IoT services for large data analysis (Big Data Analytics) can be developed and implemented on servers for data analysis locally (at the edge of the network).

The SOFT-IoT platform also proposes self-organizing features that must be deployed on fog servers. For example, management servers must implement all services related to the SOFT-IoT platform self-organization. The main services that can be implemented in the management servers are [22, 26]: self-organized monitoring service; gateway deployment service; disaster recovery service; profile management and balancing service.

2.4.4 SOFT-IoT Applications

The SOFT-IoT architecture implements the following modules (see Fig. 2.3, Application Layer). The **soft-iot-data-aggregation** module is responsible for aggregating data in FoT-Gateways. The **soft-iot-semantic-enrichment** module enriches sensor data, in the FoT-Gateway, with Semantic Web descriptions. The **soft-iot-semantic-**

[12] https://github.com/WiserUFBA/soft-iot-iot-service.

data-aggregation module provides data aggregation from semantic data. The **soft-iot-web-service** application provides services such as RESTful API for accessing IoT devices. These modules enable the Data Interplay which provides the definition and deployment of data operations in regards to the life cycle of data: collection, processing, storage, and access between edge, fog, and cloud infrastructures.

In addition, other modules with the SOFT-IoT platform (see Fig. 2.3, Application Layer) for the construction of rules aiming to create Smart Spaces through FoT were implemented. The **soft-iot-semantic-model** module obtains information semantically described by RDF triples deployed in the FoT-Server. The **soft-iot-semantic-reasoner** module performs semantic reasoning over the RDF data. When the Semantic Reasoner executes the rule and infers that a condition was satisfied, it performs an update on the model. The **soft-iot-semantic-observer** module is responsible for observing changes that are made in the model and notifies the **soft-iot-rules** module which is responsible for informing the actuator and activating the device (for instance, turn on an air conditioner).

Finally, in the Business Layer, the SOFT-IoT architecture implements a Data Visualization model. The aim is to enable the creation of different data visualizations that take into account the hierarchical levels in an IoT architecture involving fog and cloud.

2.5 SOFT-IoT Related Research Topics

In this section, some works related to the SOFT-IoT architecture are presented. The aim is to introduce practical examples that serve as a basis for learning the concepts and inspiration for new projects.

2.5.1 Reactive Microservices

Santana, Alencar, and Prazeres proposed an architecture and platform [11] to enable the implementation of reactive applications in IoT scenarios by the implementation of Reactive Microservices whose objective was to enable the construction of applications that are performative, resilient, and scalable.

According to Escoffier [12] Reactive Microservices have four features:

- **Autonomy**, this characteristic enables a system to adapt to the availability of the services surrounding them.
- **Resilience**, this characteristic applies to systems that require high availability. A resilient system responds even in the presence of failures.
- **Elasticity**, this characteristic refers to the ability of a system to react appropriately to load variations. Therefore, increasing or decreasing resources will depend on the number of requests to the system.

- **Asynchronous**, to have these characteristics (i.e., Autonomy, Resilience, Elasticity), the system must use asynchronous messages in order to guarantee low coupling, isolation, and location transparency communication pattern.

Thus, Santana, Alencar, and Prazeres compared the architecture and platform proposal with a FOG-based platform [21, 22]. The results showed that the use of a reactive approach resulted in better performance when compared to the same scenarios without a reactive approach.

2.5.2 IoT Stream Analytics in Fog Computing

IoT systems, in general, use Fog Computing, which helps solve some issues and improve some aspects of cloud-centric applications such as: achieving low latency; improving quality of service; enabling devices to operate in Fog Computing even without Internet connectivity; Data processing and applications within the boundaries of the local network. In addition, a new area of research has emerged, its biggest challenge is to process and analyze the large amount of data generated by IoT environments in Fog Computing, aiming to take advantage of the benefits of an architecture in fog. This area is described in the literature as part of the more general concept called IoT Stream Analytics [25].

IoT Stream Analytics aims to process data from various sources and domains produced by IoT, such as temperature, humidity, air pressure, luminance, gas, electricity, air quality, and motion sensor data. Common data streams generally follow a statistical distribution over a long period. However, IoT data is produced in large quantities and in a short time. They may also exhibit a variety of sporadic distributions over time [23]. Therefore, IoT Stream Analytics is a new area of research with some issues that should be further investigated.

2.5.3 Blockchain-Based Distributed Fog Solutions

As discussed in Sect. 2.3.4, the lack of intrinsic security measures makes IoT systems vulnerable to different privacy and security threats. Also, the Cloud Computing model has drawbacks in terms of high delays, which harm the IoT tasks that require a real-time response. Together, Blockchain and Fog Computing paradigms can help in addressing significant security and performance requirements in IoT platforms [14].

Blockchain capabilities such as immutability, transparency, auditability, data encryption, and operational resilience can solve most architectural shortcomings of IoT. Also, it provides a decentralized way to share information between IoT applications.

However, nontrivial challenges associated with integrating Blockchain technologies to IoT networks need be overcome [14]. One of the critical challenges is the scalability and latency of IoT Blockchain networks since each entry on the Blockchain requires consensus among all network nodes.

Fog and Edge Computing are complementary and useful paradigms to develop IoT solutions with low latency and QoS guarantees. In Edge Computing, data is processed directly by the device itself (data source) or by a local node rather than being transmitted to the cloud data center. Fog Computing spreads the concept of the edge in a scalable and integrated way with network devices such as switches, routers, and IoT gateways. The fog and edge distributed approach reduces the amount of data going through the core network and reduces latency issues.

The integration of Blockchain and Fog Computing technologies allows the development of responsive, secure, interoperable IoT solutions. The SOFT-IoT project has been investigating a strategy for managing these aspects by exploiting the eventual consistency and security guarantees of distributed ledger technologies.

The development of a Blockchain-as-a-Service (BaaS) model, following a fog architecture, can offer better managed Blockchain environments, resulting in higher uptake, better deployment times, and low latency services. Also, a Blockchain-based distributed fog architecture may lead to improvements in the QoS guarantees for clients, in particular fairness.

2.6 Final Remarks

This Chapter presented the Fog of Things paradigm, which proposes a definition of IoT infrastructure based on Fog Computing. It also relates the FoT paradigm with IoT architecture, describing the components of FoT in each layer of IoT architecture. Furthermore, the Chapter presented the SOFT-IoT platform that is an implementation of the Fog of Things model. The SOFT-IoT is a distributed IoT platform that uses Fog of Things infrastructure to deploy modules on FoT-Gateways, FoT-Servers, and cloud servers. Furthermore, the SOFT-IoT is an agnostic IoT platform and provides a flexible and configurable infrastructure. Lastly, some related research topics that use the SOFT-IoT as an environment to perform and to develop studies were presented.

References

1. Abadi, F.A., Ellul, J., Azzopardi, G.: The blockchain of things, beyond bitcoin: a systematic review. In: 2018 IEEE International Conference on 2018 IEEE Cybermatics, pp. 1666–1672. IEEE, Piscataway (2018)
2. Abdelshkour, M.: IoT, from cloud to fog computing. http://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing (2015). Accessed 20 July 2017

3. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutorials **17**(4), 2347–2376 (2015). https://doi.org/10.1109/COMST.2015.2444095
4. Andrade, L., Serrano, M., Prazeres, C.: The data interplay for the Fog of Things: a transition to edge computing with IoT. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–7. IEEE, Piscataway (2018). https://doi.org/10.1109/ICC.2018.8423006
5. Bassi, A., Bauer, M., Fiedler, M., Kramp, T., van Kranenburg, R., Lange, S., Meissner, S. (eds.): Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model, 1st edn. Springer, Berlin (2013)
6. Batista, E., Andrade, L., Dias, R., Andrade, A., Figueiredo, G., Prazeres, C.: Characterization and modeling of IoT data traffic in the Fog of Things paradigm. In: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), pp. 1–8 (2018). https://doi.org/10.1109/NCA.2018.8548340
7. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: 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, pp. 13–16. ACM, New York (2012). https://doi.org/10.1145/2342509.2342513
8. Carrez, F., Bauer, M., Boussad, M., Bui, N., Jardak, C., De Loof, J., Magerkurth, C., Meissner, S., Nettsträter, A., Olivereau, A., et al.: Internet of Things—architecture IoT-a, deliverable d1. 5—final architectural reference model for the IoT v3.0. European Union, 7th Framework Programme (2013)
9. Chappell, D.: Enterprise Service Bus. O'Reilly Media, Inc., Sebastopol (2004)
10. de Santana, C.J.L., de Mello Alencar, B., Prazeres, C.V.S.: Microservices: a mapping study for Internet of Things solutions. In: 2018 IEEE International Symposium on Network Computing and Applications (NCA), pp. 1–4 (2018)
11. de Santana, C.J.L., de Mello Alencar, B., Prazeres, C.V.S.: Reactive microservices for the Internet of Things: a case study in fog computing. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19, pp. 1243–1251. ACM, New York (2019). https://doi.org/10.1145/3297280.3297402
12. Escoffier, C.: Building Reactive Microservices in Java. O'Reilly, Sebastopol (2017)
13. Fowler, M., Lewis, J.: Microservices, 2014. http://martinfowler.com/articles/microservices.html (2014)
14. Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., Sikdar, B.: A survey on IoT security: application areas, security threats, and solution architectures. IEEE Access **7**, 82721–82743 (2019)
15. Khan, R., Khan, S.U., Zaheer, R., Khan, S.: Future internet: the internet of things architecture, possible applications and key challenges. In: 2012 10th International Conference on Frontiers of Information Technology, pp. 257–260 (2012). https://doi.org/10.1109/FIT.2012.53
16. Kumar, S., Sahoo, S., Mahapatra, A., Swain, A.K., Mahapatra, K.: Security enhancements to system on chip devices for IoT perception layer. In: 2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS), pp. 151–156. IEEE, Piscataway (2017)
17. Leiba, B.: Oauth web authorization protocol. IEEE Internet Comput. **16**(1), 74–77 (2012)
18. Lin, S., Crawford, M., Mellor, S.: The industrial Internet of Things-volume G1: reference architecture. Industrial internet consortium. IIC:PUBG1
19. Newman, S.: Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, Inc., Sebastopol (2015)
20. Pinto, G.P., Prazeres, C.V.S.: Web of things data visualization: from devices to web via fog and cloud computing. In: IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 1–6. IEEE, Piscataway (2019)
21. Prazeres, C., Serrano, M.: SOFT-IoT: self-organizing FOG of Things. In: 2016 30th International Conference on Advanced Information Networking and Applications Workshops, pp. 803–808 (2016). https://doi.org/10.1109/WAINA.2016.153

22. Prazeres, C., Barbosa, J., Andrade, L., Serrano, M.: Design and implementation of a message-service oriented middleware for Fog of Things platforms. In: Proceedings of the Symposium on Applied Computing, SAC '17, pp. 1814–1819. ACM, New York, (2017). https://doi.org/10.1145/3019612.3019820
23. Puschmann, D., Barnaghi, P., Tafazolli, R.: Adaptive clustering for dynamic IoT data streams. IEEE Internet of Things J. **4**(1), 64–74 (2017)
24. Sciancalepore, S., Piro, G., Caldarola, D., Boggia, G., Bianchi, G.: Oauth-IoT: an access control framework for the Internet of Things based on open standards. In: 2017 IEEE Symposium on Computers and Communications (ISCC), pp. 676–681. IEEE, Piscataway (2017)
25. Soldatos, J.: Building Blocks for IoT Analytics: River Publishers Series in Signal, Image and Speech Processing. River Publishers, Gistrup (2016). https://books.google.com.br/books?id=svQRMQAACAAJ
26. Sousa, N.R., Prazeres, C.V.S.: M2-fot: a proposal for monitoring and management of Fog of Things platforms. In: 2018 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6. IEEE, Piscataway (2018)
27. Zhu, X., Badr, Y.: A survey on blockchain-based identity management systems for the Internet of Things. In: 2018 IEEE International Conference on 2018 IEEE Cybermatics, pp. 1568–1573. IEEE, Piscataway (2018)

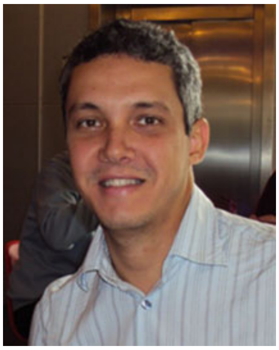
Leandro Andrade, Msc., is a Ph.D. candidate at Federal University of Bahia (UFBA) and received MS (2014) and BS (2012) in Computer Science at UFBA. He is researcher of the Web, Internet and Intelligent Systems Research Group (WISER) where he works in projects related to Internet of Things, Fog Computing, and Big Data. In 2017, Andrade has done a Ph.D. internship at Insight Centre for Data Analytics (former DERI) at the National University of Ireland, Galway (NUIG). Leandro is a member of the Brazilian Computer Society (SBC) and IEEE Communications Society, and he has publications in international and national conferences. He is a substitute lecturer at the Department of Computer Science in UFBA and his research interests are Web Semantic, Web Services, Internet/Web of Things, Fog Computing, Machine Learning, Education, and Free Software.

Cleber Lira, Msc., is a Ph.D. candidate at Federal University of Bahia (UFBA) and received MS (2015) in Computer Systems at UNIFACS. He is a researcher of the Web, Internet and Intelligent Systems Research Group (WISER) and Nucleus of Mathematics in Computational Environment (NUMAC), where he works in projects related to Microservices, Internet of Things, Semantic Web Services, and Education. Santana is a member of IEEE Communications Society and he has publications in international and national conference. Since 2013, he has been a professor of the Federal Institute of Bahia (IFBA) and his research interests are Web Semantic, Web Services, Internet of Things, Fog Computing, Artificial intelligence, and Education.

Brenno de Mello is a MSc. candidate at Federal University of Bahia (UFBA) and received a degree in Systems Analysis and Development (2016) from the Federal Institute of Bahia (IFBA). He has experience in the area of Computer Science, with emphasis on Information Systems and Software Engineer. Currently, Mello is participant in the Web, Internet and Intelligent Systems Research Group (WISER) and his research interests are in Internet of Things, Data Stream Mining, Fog Computing, and Smart Water.

Andressa Andrade received BS (2018) in Computer Engineering from Federal University of Bahia (UFBA). She has acted as a monitor in the subject of Robotics Intelligent at UFBA and currently works with database management. Since 2015, she has been a member of the Web, Internet and Intelligent Systems Research Group (WISER). Her research interests are in the Perception Layer technologies, working mainly on the development of physical devices based on IoT architecture.

Antonio Coutinho, MSc., is a Ph.D. candidate at UFBA and received a MS (2000) and BS (1998) in Computer Science from Federal University of Campina Grande (UFCG). Since 2004, he has been assistant professor of the Department of Technology (DTEC) at the State University of Feira de Santana (UEFS). Since 2016, he has been a member of the WISER and GAUDI research groups at UFBA. His research interests are in Fog Computing, working mainly on its integration with distributed ledger technologies. Particularly in the SBRC 2016, he authored and taught the minicourse “Fog Computing: Concepts, Applications and Challenges”. At SBRC 2018, he authored and taught the minicourse “Blockchain and the Revolution of Consensus on Demand”.



Cássio Prazeres, DSc., has been an assistant professor at Federal University of Bahia (UFBA) since 2010, where he leads research activities and projects in the Web/Internet area, teaches undergraduate and postgraduate courses, and is advisor for Ph.D., MSc, and BSc students. Also at UFBA, Prazeres is a co-founder and leader of Web, Internet and Intelligent Systems Research Group (WISER). He received a Ph.D. (2009) in Computer Science from University of Sao Paulo (USP). He has several publications in international conferences and journals. Prazeres is member of: Brazilian Computer Society (SBC); IEEE Communications Society; IEEE Computer Society Technical Committee on Services Computing; IEEE Smart Cities Technical Community; IEEE Internet of Things Technical Community; ACM SIGWEB (Special Interest Group on Hypertext the Web); W3C Web of Things Community Group. He is interested in research involving topics of Internet of Things, Web of Things, Web Services, Semantic Web, Microservices, Fog Computing, Fog of Things, Web of Data, and Linked Data. Prazeres has coordinated projects in the Internet/Web of Things themes in the last years and has participated in other related projects such as Digital TV, crowdsourcing, and e-learning. Recently, Dr. Prazeres had a sabbatical year as a Postdoctoral Researcher at Insight Centre for Data Analytics (former DERI) at the National University of Ireland, Galway (NUIG).