A microservice-based platform for IoT application development

Guido Porruvecchio, Alessandro Romanino, Carlino Casari, Raffaella Sanna
CRS4 - Center for Advanced Studies, Research and Development in Sardinia
Loc. Piscina Manna, Edificio 1, 09050 Pula (CA) Italy
Email: {guido, aromanino, casari, raffa}@crs4.it

***Abstract*—Modern Internet of Things ecosystems and applications must be able to cope with an increasing demand for flexibility, efficiency, scalability, security. This paper presents CMC-IoT (CRS4 Microservice Core - IoT), an Internet of Things platform built upon a microservice-based architecture, providing tools for developing scalable and robust IoT applications. The platform aims to act as a generic middleware to connect a wide variety of devices, using a connector-driven and vendor-agnostic approach, allowing the development of applications in many domains, as shown in the provided examples. The focus is on the most prominent architectural features of CMC-IoT, that are compared with a reference architecture available in literature.**

***Index Terms*—Internet of Things, Microservices, IoT middleware**

I. Introduction

Over the past years, software architectures has shifted away from the traditional monolithic approach. An urgent need for systems to be more scalable, resilient, efficient and cloud-ready, paved the way for modular architectures, which are nowadays largely adopted in many domains and drive the development of modern software systems. In this context, Microservices represent a state-of-the-art architectural model [1] [2], especially in domains where distributed and pervasive computing is of paramount importance, as in Internet of Things [3]. IoT systems are built upon a large number of connected devices communicating through the Internet, with strict requirements for device coordination and for management of a large amount of device generated data.
This paper is structured as follows. Section II analyzes the state of the art of IoT platforms and middleware, with a focus on their architectural peculiarities. Section III introduces CRS4 Microservice Core (CMC), a microservice-based framework for the development of modular and scalable applications, created at CRS4 research center. Section IV presents CMC-IoT, an IoT platform developed as a service of CMC framework, and compares its software architecture with an IoT reference architecture. Further developments and final considerations are discussed in Section V.

This work was partially supported by Italian Ministry of University and Research (MIUR), within the Smart Cities framework (Project Cagliari 2020, ID: PON04a2-00381)

978-1-6654-0690-1/21$31.00 ©2021 IEEE

II. Background and related work

As soon as a variety of heterogeneous connected devices became the backbone of pervasive and ubiquitous networks, emerged the necessity for middleware systems that could act as an abstraction layer between applications and Things, integrating additional services [4] and supporting interoperability among these applications and services [5]. As a consequence, many IoT platforms and middleware were developed, and industry and academia made an effort to survey these systems, classifying them with respect to their architecture [6] as well as functional/non-functional features [5] [4]; furthermore, objective criteria were generally defined for their evaluation and adoption in different scenarios. An insight on cloud IoT platforms is given in [7], [8] focuses on middleware particularly suited for application development, while a comparative analysis on Open Source IoT middleware platforms is performed by [9]. [4] underlines the centrality, in IoT platforms and middleware, of the different aspects lying under the notion of interoperability [10], which is one of the requirements for implementing the abstraction functionalities. Network interoperability acts as a bridge between the different communication protocols used by Things; syntactic interoperability deals with the format of data exchanged among Things; semantic interoperability defines the semantic domain-specific model, which establishes the rules for understanding the meaning of information content. The semantic model will be then translated into a data model used by the concrete software system.
[11] proposes a classification of the different types of architecture for IoT middleware, which are identified as SOA, cloud-based and actor-based. Service Oriented Architectures represent a well established approach in software design [12], and in IoT systems as well [13] [14] [15] [16]. Microservices further developed and refined SOA concepts, focusing on the subdivision of large monolithic applications in small, highly decoupled, independently deployable and scalable services, performing a single, well defined and domain-oriented task, and communicating among themselves by means of a simple protocol, e.g. HTTP [1] [2].
A comprehensive review on the adoption of microservices for IoT systems can be found in [17]; previous studies already highlighted how well microservices could help IoT platforms to overcome limitations of SOA approach, such as scalability

and maintainability [18] [19], providing the design principles and technological tools to build distributed and large-scale applications deployable in the cloud [18] [20] [21] [22]. Consequently, several IoT platforms in various domains applied microservice architectural style, gaining benefits in terms of resilience, scaling, modularity, heterogeneity and independence of technology [23] [24] [25] [26] [27] [28] [29].
In the next sections we propose our contribution to the above cited challenges, represented by a microservice architecture acting as a framework for building scalable and cloud-ready services and applications, and an IoT platform deployed as a microservice on top of that architecture.

III. CMC - CRS4 Microservice Core

CRS4 Microservice Core (CMC)[1] is a high-level and general purpose framework, built upon a microservice architecture and conceived for supporting the development of vertical services and applications. It provides cross-cutting functionalities, each of them deployed as a microservice, that allow the developers to focus on the business logic of their application. These basic microservices, called "Core" services, are the following:

- **CMC-Auth** is the central service of CMC. It takes care of service registration, token management and authorization to all services and vertical applications. Three categories of JWT-Base64 tokens are generated by CMC-Auth:
  - *Microservice token*, authorizing communication among internal microservices
  - *User token*, granting access to authenticated CMC users on registered services
  - *Application token*, granting access to third-party applications on registered services

  CMC-Auth also implements a rule engine to manage access to every single HTTP resource exposed by services. Each token belonging to one of the three categories can also have a type defining authorizations with a finer level of granularity.
- **CMC-User** exposes a number of features for user management. It takes care of user registration and authentication, providing a user token (generated by CMC-Auth) required to access a protected resource (if the access to that resource has been enabled by Auth service)
- **CMC-App**, similarly to CMC-User, allows the complete management of third-party applications, and in general any system calling CMC services

This architecture allows a seamless extension with further microservices, which can be registered in the platform and communicate with other CMC services or the external world (Fig. 1). In order to be integrated in CMC, a service must comply with few specifications:

- It must expose an authenticated API, using CMC-Auth
- It must be registered in CMC using CMC-Auth
- It must communicate with the other CMC microservices by means of the token generated by CMC-Auth on service registration
- Service authorization rules must be configured using CMC-Auth, via API or UI

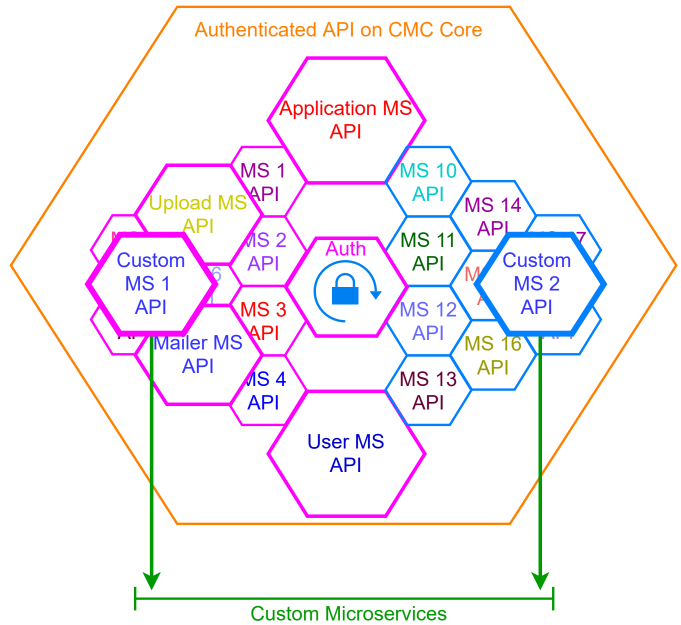

Fig. 1. CMC services.

Once registered in the platform, any service can talk to other CMC services via REST API, if allowed by authorization rules managed by CMC-Auth. It can also independently scale according to user's needs.

IV. CMC-IoT

CMC-IoT[2] is a general purpose IoT platform and middleware, developed as a custom service compliant with CMC specifications. In literature there is already a platform named CMC-IoT proposed by CRS4 [30]; however, this paper presents a complete rework of that system, from both architecture and features perspective.
CMC-IoT is basically a middleware supporting a wide class of devices, since it implements an abstraction between the Things and the applications interacting with them. Therefore, it provides a uniform interface to Things, which can be native compatible (e.g. a sensor developed in-house) or need a specific driver or *connector* to be integrated in the platform. In the system architecture (Fig. 2), Connector Middleware depicts the logical block encompassing any kind of connector available.

Fig. 2 in its entirety describes a four layer architecture:

- The base layer is composed by *Things*, the physical objects hosting one or more *Devices*, where a Device is a sensor capable of performing a single physical measurement (e.g. temperature, humidity, voltage) or an actuator

[1] https://github.com/smartenv-crs4/cmc

[2] https://github.com/smartenv-crs4/cmc-iot

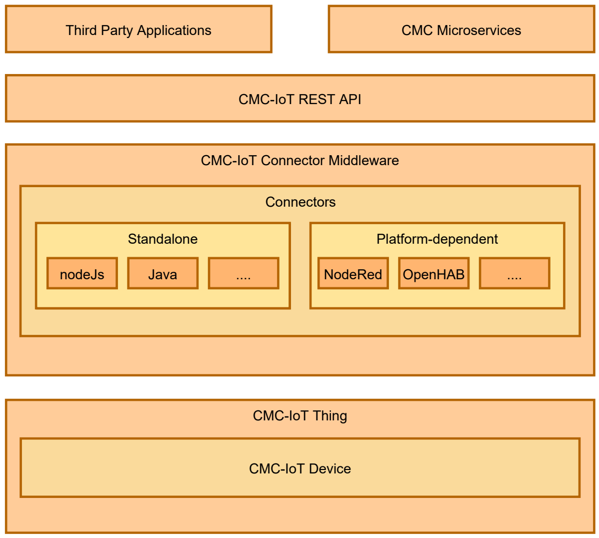

Fig. 2. CMC-IoT architecture.

able to accept commands for acting on the environment (e.g. a switch).

- The CMC-IoT *Connector Middleware*, previously introduced, is composed by any piece of software needed to integrate heterogeneous Things in the platform; in other words, the layer responsible for the communication between CMC platform and each Thing's exposed interface. It can be summarized as a composition of a communication interface, an interpreter of the device protocol, and a driver for heterogeneous access by the platform. From the technological point of view, connectors can be implemented using any technology suited to eventually talk to the platform's REST API; *Standalone* connectors comprehend a complete stack, while *Platform-dependent* connectors rely on a third-party platform offering a framework to ease the connector implementation.
- The CMC-IoT *REST API* logical block encompasses all the IoT features of the platform, which are exposed through a HTTP REST interface. It is responsible for Things management (discovery, device data processing and exposing to applications, command sending and so on), data storage, context detection, system configuration. Ultimately, all these services provide a coherent interface to the functionalities offered by Things, abstracting their heterogeneity and complexity.
- The top layer is constituted by any client calling the REST API, which can be an external application consuming CMC-IoT services, or a microservice registered in the CMC platform.

As a general purpose IoT platform, CMC-IoT can be adopted in a broad range of scenarios; its customizable data model is based on the concept of *Device Type*, allowing users to map their own devices onto custom categories with custom properties. As an example, CMC-IoT could manage a network of hygrometers placed in a humidity controlled environment; a *Hygrometer* device type will be defined, associated with a *Relative Humidity* observed property and a *Percentage* unit of measurement. In an analogue manner, a smart mobility application could acquire traffic information from sensors registered in CMC-IoT as belonging to *Traffic Sensor* device type, observing the *Number of Vehicles* property.

A. *Reference architecture*

At the beginning of Section IV, the CMC-IoT architecture was described in order to give an insight into the functional decomposition of the system, and how the modules interact and exchange data.

Several IoT reference architectures, as well as models and taxonomies, have been previously proposed with the aim of providing frameworks and conceptual models that could be generic and flexible enough to be adopted and supported by concrete software platforms [31] [32] [33].

In this paper the reference architecture proposed by [6] has been chosen to perform a mapping onto CMC-IoT architecture. By adopting abstract concepts and a common terminology, the platform will be better described, capturing it in a framework that, as [6] stated, could help in comparing the existing platforms in terms of features and further developments.

Every concept of the Reference Architecture (RA) will be mapped onto each logical block of CMC-IoT architecture, evaluating the similarities and differences of "overlapping" elements (Fig. 3)

Starting from the bottom layer, a *Sensor* in RA represents a hardware component that captures information from the physical environment, while an *Actuator* manipulates that environment. So they fit almost perfectly with the CMC-IoT concept of Device, which performs a single measure or a single command inside a CMC-IoT Thing. In both architectures, this lower level kind of component communicates with the outer world only by means of another piece of hardware encapsulating it, namely the RA *Device* and the CMC-IoT Thing, which are for this reason functionally equivalent.

The RA *Driver* is a software running on the RA Device enabling uniform access to its Sensors and Actuators; in CMC-IoT architecture, its purpose can be accomplished by a software layer allowing a Thing to be directly invoked without further interoperability layers either by the platform (i.e. a CMC native compatible Thing directly talking to the REST API component), or by an external client/application (e.g. a Thing with an on-board HTTP layer able to communicate with a Web UI).

The RA *Gateway* takes care of communication between external systems and those RA Devices that are not capable of managing that kind of connection. This task perfectly fits with CMC-IoT Connector Middleware; the only difference lies in the possibility, for a CMC-IoT connector, of running inside a Thing or in an external system. In the former case, that Thing would be CMC native compatible, and the connector would be acting more as a RA Driver rather than a middleware.

The *IoT Integration Middleware* in the Reference Architecture models an integration layer for Sensors, Actuators, Devices

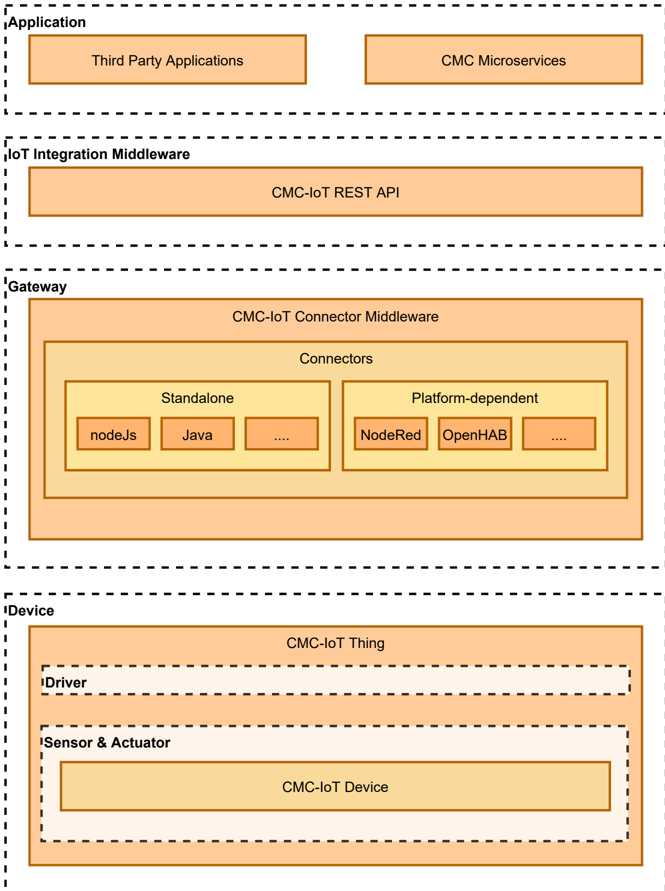

Fig. 3. CMC-IoT architecture vs Reference Architecture.

and Applications, thus enabling a common interface to them, abstracted from their actual implementation. With respect to this concept, the CMC-IoT REST API performs the same task by means of HTTP protocol, while other protocols may be adopted in the RA. The exposed API is the interface to some common IoT functionalities provided by both architectures, mainly the management of Devices/Things and sensor data. Further features are made available, but in CMC-IoT some of them are provided by means of the CMC environment, with microservices each exposing a narrow set of features, for example the core functionalities of authentication and user management. Other higher level features would need ad-hoc microservices, such as graphical dashboards or business intelligence on sensor data.

Finally, the RA *Application* represents any software system which calls the IoT Integration Middleware to use the connected Devices, so any third-party application (whose access is managed by CMC-App) or CMC service invoking its REST API is fully equivalent.

V. Conclusion

CMC-IoT as a service satisfies the main functional requirements of an IoT platform, such as abstraction, context awareness and resource management; at the same time, it is deployed in the context of CMC, a general-purpose and microservice-based framework addressing non functional requirements in terms of availability, scalability, security [5]. From this perspective, the overall CMC-IoT architecture can be defined as a two-layer framework: the high level provides vertical features to ease the development of modern IoT applications and services; these features have their solid foundations in the lower level, a microservice architecture that takes care of all the cross-cutting concerns that every modern application has to address, allowing developers to focus on the implementation of domain functionalities.

The purpose of mapping the CMC-IoT architecture onto a generic and more abstract reference is to describe it in a uniform and consistent way, by using a common basis of concepts and terms that ease the understanding of its main features and the comparison with other IoT platforms.

Future work will expose a more in-depth review of the IoT platform technological features, as well as the implications of a microservice approach in terms of performance and resilience of the system. Furthermore, a real-world use case of CMC-IoT will be presented in order to better clarify how an application can benefit from its adoption.

References

[1] M. Fowler and J. Lewis, "Microservices," 2014. [Online]. Available: http://martinfowler.com/articles/microservices.html

[2] S. Newman, *Building Microservices*, 1st ed. O'Reilly Media, Inc., 2015.

[3] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870512000674

[4] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta, "Role of middleware for internet of things: A study," *International Journal of Computer Science and Engineering Survey*, vol. 2, August 2011.

[5] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[6] J. Guth, U. Breitenbücher, M. Falkenthal, P. Fremantle, O. Kopp, F. Leymann, and L. Reinfurt, "A detailed analysis of iot platform architectures: concepts, similarities, and differences," in *Internet of everything*. Springer, 2018, pp. 81–101.

[7] P. P. Ray, "A survey of iot cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1-2, pp. 35–46, 2016.

[8] P. Agarwal and M. Alam, "Investigating iot middleware platforms for smart application development," *Smart Cities—Opportunities and Challenges*, pp. 231–244, 2020.

[9] R. Scott and D. Östberg, "A comparative study of open-source iot middleware platforms," Ph.D. dissertation, KTH Skolan för kemi, bioteknologi och hälsa, 2018. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-223980

[10] C. Jovellanos, "Semantic and syntactic interoperability in transactional systems," in *Proceedings of the 4th ACM conference on Electronic commerce*, 2003, pp. 266–267.

[11] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[12] M. P. Papazoglou and D. Georgakopoulos, "Introduction: Service-oriented computing," *Communications of the ACM*, vol. 46, no. 10, pp. 24–28, 2003.

[13] E. Avilés-López and J. García-Macías, "Tinysoa: A service-oriented architecture for wireless sensor networks," *Service Oriented Computing and Applications*, vol. 3, pp. 99–108, June 2009.

[14] W. Zhiliang, Y. Yi, W. Lu, and W. Wei, "A soa based iot communication middleware," in *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, 2011, pp. 2555–2558.

[15] Z. D. Patel, “A review on service oriented architectures for internet of things (iot),” in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 466–470.

[16] Y. Zhang, L. Duan, and J. L. Chen, “Event-driven soa for iot services,” in *2014 IEEE International Conference on Services Computing*, 2014, pp. 629–636.

[17] A. Razzaq, “A systematic review on software architectures for iot systems and future direction to the adoption of microservices architecture,” *SN Computer Science*, vol. 1, no. 6, pp. 1–30, 2020.

[18] B. Butzin, F. Golatowski, and D. Timmermann, “Microservices approach for the internet of things,” in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–6.

[19] B. el Khalyly, A. Belangour, M. Banane, and A. Erraissi, “A comparative study of microservices-based iot platforms,” *International Journal of Advanced Computer Science and Applications*, vol. 11, January 2020.

[20] S. K. Datta and C. Bonnet, “Next-generation, data centric and end-to-end iot architecture based on microservices,” in *2018 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, 2018, pp. 206–212.

[21] C. Santana, B. Alencar, and C. Prazeres, “Microservices: A mapping study for internet of things solutions,” in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018, pp. 1–4.

[22] L. Banica, C. Stefan, and A. Hagiu, “Leveraging The Microservice Architecture For Next-Generation Iot Applications,” *Scientific Bulletin - Economic Sciences*, vol. 16, no. 2, pp. 26–32, 2017. [Online]. Available: https://ideas.repec.org/a/pts/journl/y2017i2p26-32.html

[23] A. Krylovskiy, M. Jahn, and E. Patti, “Designing a smart city internet of things platform with microservice architecture,” in *2015 3rd International Conference on Future Internet of Things and Cloud*, 2015, pp. 25–30.

[24] T. Vresk and I. Čavrak, “Architecture of an interoperable iot platform based on microservices,” in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 1196–1201.

[25] K. Thramboulidis, D. C. Vachtsevanou, and A. Solanos, “Cyber-physical microservices: An iot-based framework for manufacturing systems,” in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 232–239.

[26] K. Khanda, D. Salikhov, K. Gusmanov, M. Mazzara, and N. Mavridis, “Microservice-based iot for smart buildings,” in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2017, pp. 302–308.

[27] D. Lu, D. Huang, A. Walenstein, and D. Medhi, “A secure microservice framework for iot,” in *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 2017, pp. 9–18.

[28] S. Pallewatta, V. Kostakos, and R. Buyya, “Microservices-based iot application placement within heterogeneous and resource constrained fog computing environments,” in *2019 12th IEEE/ACM International Conference on Utility and Cloud Computing*, ser. UCC’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 71–81. [Online]. Available: https://doi.org/10.1145/3344341.3368800

[29] M. Taneja, N. Jalodia, J. Byabazaire, A. Davy, and C. Olariu, “Smartherd management: A microservices-based fog computing–assisted iot platform towards data-driven smart dairy farming,” *Software: Practice and Experience*, vol. 49, May 2019.

[30] C. Lai, F. Boi, A. Buschettu, and R. Caboni, “Iot and microservice architecture for multimobility in a smart city,” in *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2019, pp. 238–242.

[31] M. Bauer, M. Boussard, N. Bui, J. Loof, C. Magerkurth, S. Meissner, A. Nettsträter, J. Stefa, M. Thoma, and W. Joachim, *IoT Reference Architecture*. Springer Berlin Heidelberg, December 2013.

[32] P. Fremantle, “A reference architecture for the internet of things,” October 2015. [Online]. Available: http://wso2.com/wso2_resources/wso2_whitepaper_a-reference-architecture-for-the-internet-of-things.pdf

[33] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, July 2012.