# MorArch: A Software Architecture for Interoperability to improve the communication in the Edge layer of a smart IoT ecosystem

Juan Moreno-Motta[1], Felipe Moreno-Vera[2], and Frank A. Moreno[3]
juan.moreno2@unmsm.edu.pe, felipe.moreno@ucsp.edu.pe, fmorenov@uni.pe

[1] Universidad Nacional Mayor de San Marcos, Lima, Perú
[2] Universidad Católica San Pablo, Arequipa, Perú
[3] Universidad Nacional de Ingeniería, Lima, Perú

**Abstract.** Currently, IoT has evolved to such an extent to extend to all corners of each place through devices that are connected to a network and generate information. In most cases, to be processed for a specific purpose or storage as historical data; an IoT ecosystem is implemented to manage those tasks between different devices, frameworks, or applications. Besides, more complex IoT ecosystems require more complex architecture to manage the information flow, at this level, we found a problem called interoperability. This problem is not limited to the compatibility of adding/removing devices to an ecosystem, it is also expected that the information generated by devices and processed comply with a standard optimizing the data transmission. In this work, we present a new software architecture pattern to avoid the problem of interoperability through the process of exchange information between devices and prevent store heterogeneous information coming from different layers of an IoT ecosystem.

**Keywords:** Interoperability · IoT · data encode · data decode · protocol buffer · Edge Computing · REST API · Software Architecture · IoT Ecosystem · Architecture · Fog Computing · Edge Computing · SOAP · REST · web services.

## 1 Introduction

The Internet of Things (IoT) is a technology that has been emerging and converging of many technologies for all-purpose solutions such as health, environment, manufacturing, energy-saving, citizen security, etc. According to Guinard [3], IoT is a system of physical objects that can be organized, controlled, or connected with electronic devices that communicate through various network interfaces and can finally be connected to the Internet.

One study about the 7 main challenges in IoT data provenance [7], mentions that interoperability is one of the challenges related to the origin of the data making it more difficult to deploy heterogeneous systems. Despite the current

techniques, they remain very challenging in their implementation and optimization. Then, in two research about those challenges, Pace et al. [8] and Madaan et al. [9] said that is complex to integrate all information generated by all systems with different devices, brands, hardware design, protocols, body message encode-decode, different programming languages, different data structures, etc.

Interoperability is defined as the extent to which different systems and devices can exchange data, and interpret that shared data. For two systems to be interoperable, they must be able to exchange data and subsequently present that data such that it can be understood by a user. It's the form of computing where data is stored on multiple servers and can be accessed online from any device.

In 2014, Oliver Kleine [1] estimated that IoT devices will be increased at least 1.7 billions at 2017 and Utkarshani Jaimini [2] said the storage for those devices surpass 150 Exabytes in 2020. Some concepts in IoT used in Interoperability are Cloud computing, Fog Computing, and Edge Computing. The first one is commonly the standard of IoT data storage [21]. For the Internet of Things, this means securely storing and managing a lot of data and having immediate access to it from multiple devices, anytime, anywhere.

Fog computing refers to computing infrastructure close to data sources, where many connections occur with low bandwidth and minimal data transmission and the data processing is performed at local networks, although servers themselves are decentralized [22]. Edge computing refers to the data processing information directly on it, without being sent to the centralized servers like Fog or Clouds. In this way, any device connected to the Internet is capable of processing the received data -through a short-term analysis- and bouncing part of this information to the cloud, which simplifies the communication chain and reduces potential points of error.

## 2 Present context and previous works

Interoperability as we defined above, is the way to communicate different systems or devices, in most cases it can be very complex at the moment of design one architecture capable to support the processing and amount of data.

### 2.1 Interopreability architectures

This problem appears in IoT solutions proposals with a specific task like AAL-IoTSys (Active and Assisted Living) [10] which is a platform oriented to Ambient Assisted Living (AAL). AAL-IoTSys is a prototype based on Wireless Sensor Network (WSN) with heterogeneous devices using a Binary encoder to transfer information between low power devices. Another solution based on AAL is INTER-IoT [8], based on AAL, implements an interoperable medical application between BodyCloud and universalAAL platform, to save and analyze medical historical data about their patients.

In 2017, Talavera et al. [4] study 720 IoT solution in topics like Agribusiness and Environment of which they select around 72 projects to study in deep.

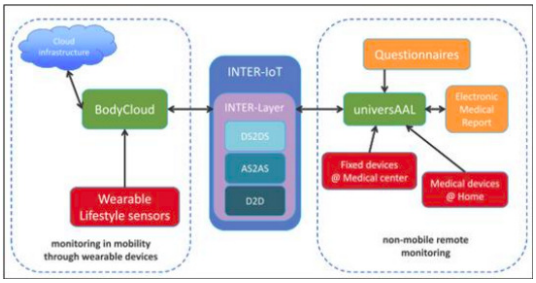
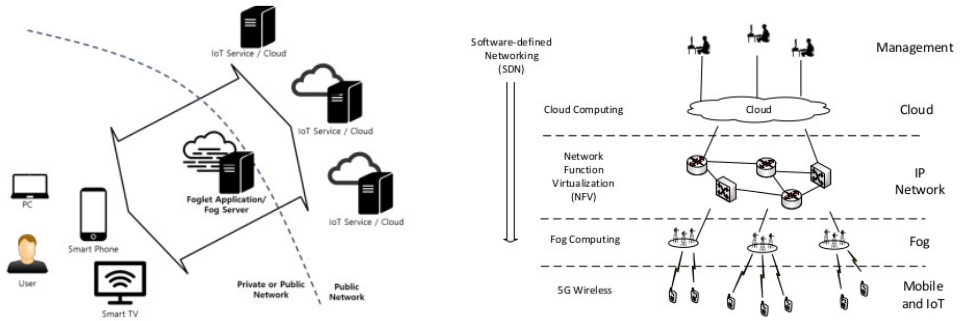



Fig. 1: Pace et al. INTER-IoT Architecture, a merge of BodyCloud and universalAAL [8].

From this study, they propose a general architecture for Agro applications, but they consider as challenging the way to implement a standard, compatible, and security guarantee interoperability service between devices in the edges and cloud services.

Also, in 2017 Woznowski et al. [5] develop SPHERE (A Sensor Platform for Healthcare in a Residential Environment), in this project they have determined that there are 9 requirements that the IoT needs to cover to implement a smart city, but the most difficult requirement to guarantee and implement was Interoperability. Based on SPHERE, an improvement was proposed Elsts et al. [6], this work considers that IoT systems must comply with existing low-power IoT standards and protocols to (1) be susceptible to future extensions with third-party components; (2) reduce learning time for new staff.




(a) (b)

Fig. 2: (a) Architecture proposed by Kum. [15] and (b) Architecture proposed by Luan. [16].

Futhermore, another solution to allow IoT interoperability was based on web semantic [11] using the JSON-LD method (JavaScript Object Notation for

Linked Data) [20]. Using this method they can create a description of data information and link objects and properties in a JSON file, then they have a piece of information about their components and can identify if some information corresponds to a specific component.

Other approach based on REST API for interoprability in Web of Things was proposed by Sun et al. [12], this work using the JSON format compares Microservices architecture against Monolithic architecture. They use REST API to communicate all devices including the IoT ecosystem, the control of the environment through the central service is dynamic.

Based on some previous works [12,11,4], Kum et al. [15] propose an architecture for Fog Computing applications, in this case, Fog Computing allows to manage with more efficiency the information flow and reduce the latency between devices and the central server, in other words, they found the best performance doing information exchange between edge layer and cloud services trough middle servers (see Fig.2 (a)).

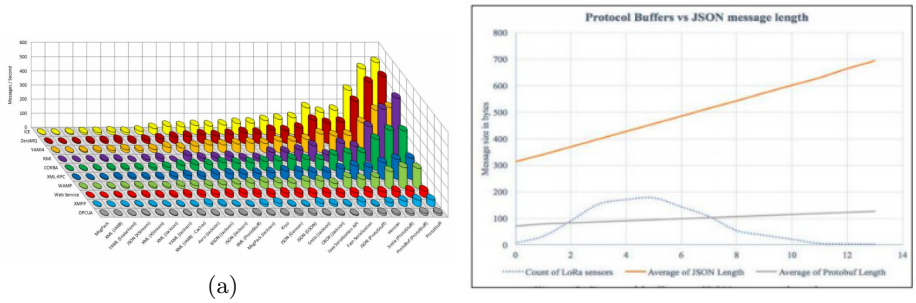
Another approach proposed is the implementation of a routing gateway, this work was proposed by Luan et al. [16]. In this work, they define the gateway as the manager of all information and communication between devices (edges), fog nodes, and cloud servers. They design a network topology showed in Figure 2 (b) that improve the processing time and reduce the time to transfer data. The aim was to communicate all devices with all servers and nodes was mobiles, the architecture transmits data through mobiles (5G infrastructure) as end-users or edge layer.

### 2.2 Interoperability protocols

Another study divides the concept of interpretability into three parts, Lim et al. [13] define three different tiers: (i) Basic connectivity: provides a common standard or path for data exchange between two sub-systems and established a communication link; (ii) network: enables message exchange between heterogeneous systems across multiple communication links; and (iii) syntactic interoperability: provides a mechanism to understand the data structure in messages exchanged between two entities. To solve this, the propose a framework with 3 components: provider, requester, and registry. They are communicate using a SOAP web service and XML protocol.

One study about a comparison between the most common used protocols in IoT interoperability was described by Nitin Naik [18], In this document, the author write a complete document about protocols, applications, operating systems, and other metric for different situations in IoT environments. They present the protocols: HTTP, AMQP, MQTT and CoAP protocols (commonly used in IoT solutions) and make some comparison based on their characteristics. According authors, the user can decide their relevant usage in IoT systems based on their requirements and suitability.

In 2018, Petersen et al. [19] make a demonstration of the performance of the different formats, arriving at the conclusion that the binary format generated with Protobuff developed by Google is the one of better performance and




(a) (b)

Fig. 3: (a) petersen et al. performance chart[19], and (b) lysogor et al. performance chart[17].

less memory use (see Figure 3 (a)), It is observed that Protocol-Buffer for any communication protocol can serialize many more messages per second, being one of these kinds of protocols ZeroMQ (an asynchronous messaging library) performance. Additional, Lysogor et al. [17] conduct a study on the transfer and exchange of data in heterogeneous networks where network infrastructure is absent. In their research, they focus on satellite networks and how to transfer information between them, but they found a limitation on the size of the data transmitted. They show that the binary format generated by Protocol Buffer allows transferring more bytes than the JSON format (See Fig.3(b)).

## 3 Proposal

Due to the analysis of all previous works mentioned above, we note that the problem was centered on how to implement an IoT system that can manage interoperability without troubles using a simple architecture. These works range from microservices and go through semantic web such as JSON-LD, in both cases, it requires processing capacity and memory that the servers have, but not restricted devices.

Our focus in this study is on low-throughput and memory-capable devices. The provenance of data is part of the interoperability challenge. We consider that there are hostile communication environments, so the serialized data frame must be very compact. The proposal is applied at the layer on the edge of an IoT ecosystem. in figure 4 we show our architecture based on Cloud-Fog-Edge computing. In the Cloud tier, we have the main and central server where is store the main data about our application, the connection only can be accessed online.

We define our Fog tier as an ecosystem and is treated as a process to encode-decode data, based on Micro-services architecture. The data exchange between devices is very important when you need to get information or analyze it by sector. For that, we divide all interactions with the aim to achieve better management of the complete system.

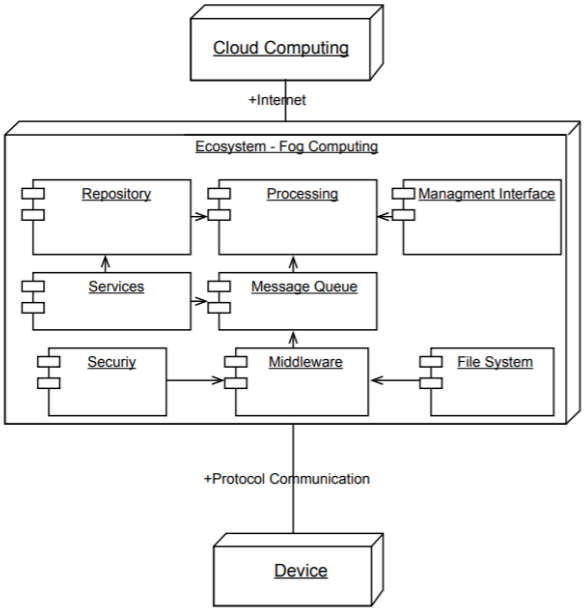



Fig. 4: MorAch: Architecture proposed.

Then, the main tier in charge to manage all communications, connections, and exchange of information between cloud and devices is the fog tier. For this purpose, we divide our fog tier eight important components:

- Services: It is the layer of services that fog computing can deliver so that other external devices can receive information (from sensors) or send an action to be executed (actuators).
- Repository: It is of persistence where the data frames will end up being stored. In addition, we will have historical information about the data processed and stored, this is going to keep the processing line of all information.
- Processing: It is in charge of processing the binary data formats that are received from the device component. The processing of the binary information is to decode the encoded data using Protocol Buffer.
- Management interface: This component is only for maintenance purposes of our IoT ecosystem. This component can make a diagnostic by itself. It checks the historical status stored in our repository component. Besides, this component can fix issues founded in the processes. Additionally, it can be accessed from a terminal that can be a PC, a laptop, or a cell phone. This manager registers the types of devices, devices, middleware, and the ecosystem.
- Security: This component is a service consumed by the middleware. Each device has a unique ID using a certificate to be validated in this component.

Besides, this component ensures that the information received is undangerous for the Ecosystem.
- MessageQueue: It is responsible for providing high availability and scalability for all incoming data from several devices in real-time. This component ensures that all the information provided by the devices and processing by the middleware component will always be saved in the message queue.
- Middleware: It is responsible for receiving the binary format coming from the devices. This component uses the processing component that has greater processing capacity since it behaves like a server. This component also depends on the file system component as an ecosystem configuration file.
- File system: This component store the description and properties of our ecosystem. Store the security key generated by the historical data in the Repository component in case of backups. Another purpose of this component is to save the certificates of the IDs of the encrypted devices and decoded data.

Additionally, we have 2 main tier: Cloud and devices with protocols like internet (for Cloud services) and Protocol buffer to connect with devices.

- Devices: Represents all devices that connect to the ecosystem through the middleware component. The encode process serialization is in binary format performed by the protocol buffer method. In general, they can be sensors, actuators, or any other devices which generate and send information to the Ecosystem.
- Cloud: Is the final node that collect and show all pre-processed data from Fog Ecosystem.

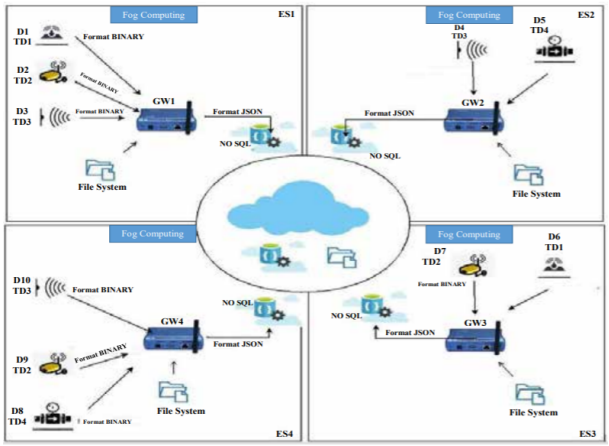



Fig. 5: MorAch: Experimental Test implemented.

## 4 Experiments

For our experiments, we deploy a set of tiers (fog nodes, devices, cloud) and make a connection between them. As we show in Figure 5 our experimental IoT ecosystem is composed of a central server denominated as the cloud, four fog ecosystems (named as the gateway) each of them has the eight components defined above.

### 4.1 File system

We implement our file system using a NoSQL database, this kind of database allows us to have a fast and good performance for multiple queries in a short time (like real-time). As we show in the Figure 6

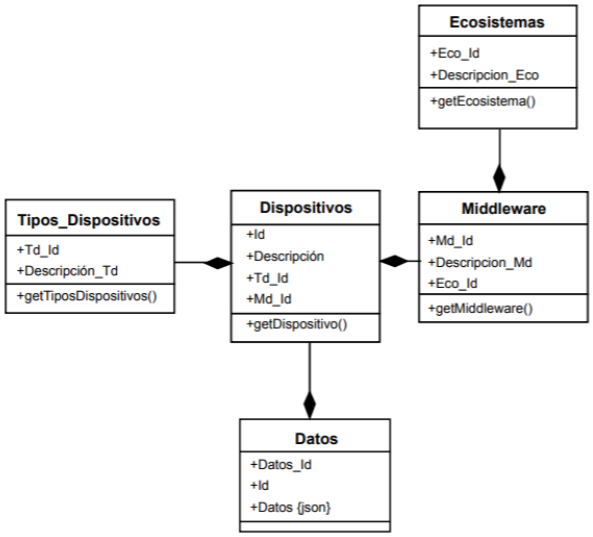



Fig. 6: MorAch: File system implemented.

### 4.2 Data flow inside our Fog Ecosystem

To describe the process, we will use only one of the four Fog Ecosystem (see Figure 7), the first step is to register the devices. In this case, we have five devices that we will have D1, D2, D3, D4, and D5. Each of them has an additional identifier (the type of device) as a category.

Each device needs to define the data block (to be encoded), the header has the category of the device called ID and the identifier of the type of device called

TID. The body of the message has the binary information of the encoded data obtained by the device. This data frame is sent to the middleware.

In the middleware, the first step is to send the data frame to the Message-Queue component, then using the component Processing we get the decoded data. After this, the decoded data is validated by the Security Component (ensures that the data is not a broken file). Once is validated, this data is store in the File System component verifying if exist the type, is not it will be registered as a new type. If the amount of information is correct, the Repository component will be updated, with this newly stored data, generating a new Fog Ecosystem status.

In the Processing component, the decoded modules should be installed as a micro-service (to be automatized). The Processing component will organize by type of device all information received. In other words, the decoded method for a new type of device will be prepared in this module and this method will be stored in the File system component. All these steps allow scalability and do not depend on the modules.

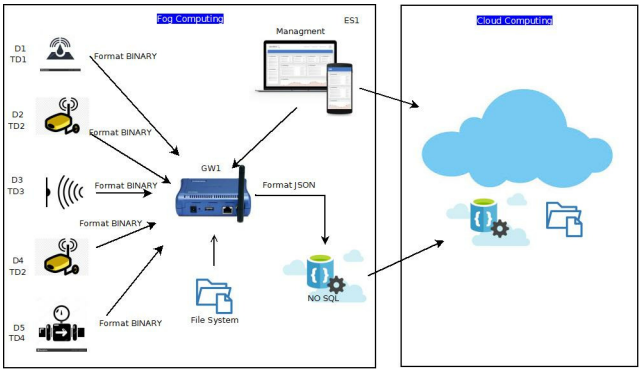



Fig. 7: Architecture proposed for multiple solutions.

### 4.3 Results

We are implemented this architecture in real time applications to get our proof of concept for this architecture, our test is a real time system monitors of data measure from environment, we use Temperature, Humidity, Sound and Monoxide sensors, Raspberry Pi 3 Model B (1.4GH, Quad Core) board, Arduino board, SQLite, protocol buffer to encode-decode data and Cloud services like Firebase to storage data and analyze, compare, measure latency, compatibility and processing or organizing data speed between each layer of the ecosystem (See Fig.7).

For the performance test, we simulate parallel programs from different computers that send messages in real-time (using RabbitMQ library) to our edge nodes devices. This process reaches five million messages per second. Besides,

we send information in different formats, and in some cases, we sent broken files to test the security component, for this test when is a broken file, the encode-decode process has a different behavior than the other data and that is how our Fog ecosystem identifies possibles vulnerabilities.

## 5 Conclusions

In this paper, we propose a new architecture for interoperability using Fog and Edge layers to manage and improve smart communication and transfer information between devices. The IoT architecture proposed can be integrated with others, making it scalable once identifying the origin of the data, keeping them all ordered and communicated among themselves.

Furthermore, we use Protocol-Buffer over our data format for all devices (or at least most of them) to get a better performance and get less latency transferring data and a high flexible fog architecture to manage all dynamic changes in devices.

## Acknowledgment

This work was supported by grant 234-2015-FONDECYT (Master Program) from CienciActiva of the National Council for Science,Technology and Technological Innovation (CONCYTEC-PERU). Also, thanks to SCOTIABANK MASTER PROGRAM GRANT for the financial support to Juan Moreno.

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

## References

1. Oliver Kleine, CoAP Endpoint Identification - A Protocol Extension for Crowd Sensing in the mobile Internet, 2014 IEEE International Conference on Internet of Things (iThings 2014), Green Computing and Communications (GreenCom 2014), and Cyber-Physical-Social Computing (CPSCom 2014).
2. Jaimini, U. (2017). PhD Forum: Multimodal IoT and EMR based Smart Health Application for Asthma Management in Children.
3. Guinard, D. D. and Trifa, V. M. (2016). Building the Web of Things. NY 11964: MANNING.
4. Talavera, J. M., Tobón, L. E., Gómez, J. A., Alejandro, M. A., Aranda, J. M., Parra, D. T., Garreta, L. E. (2017). Review of IoT applications in agro-industrial and environmental fields.

5. Przemyslaw Woznowski, Alison Burrows, Tom Diethe, Xenofon Fafoutis, Jake Hall, Sion Hannuna, Massimo Camplani, Niall Twomey, Michal Kozlowski, Bo Tan, Ni Zhu, Atis Elsts, Antonis Vafeas, Adeline Paiement, Lili Tao, Majid Mirmehdi, Tilo Burghardt, Dima Damen, Peter Flach, Robert Piechocki, Ian Craddock, George Oikonomou, SPHERE: A Sensor Platform for Healthcare in a Residential Environment. https://link.springer.com/chapter/10.1007/978-3-319-44924-1_14. Last visited dec 2020.
6. Elsts, A., Oikonomou, G., Fafoutis, X. and Piechocki, R. (2017). Internet of Things for Smart Homes: Lessons Learned from the SPHERE Case Study.
7. Alkhalil, A. and Ramadan, R. A. (2017). IoT Data Provenance Implementation Challenges.
8. Pace, P., Gravina, R., Aloi, G., Fortino, G., Fides-Valero, A., Ibañez-Sanchez, G., Yacchirema, D. (2017). IoT platforms interoperability for Active and Assisted Living Healthcare services support.
9. Madaan, N., Ahad, M. A. and Sastry, S. M. (2017). Data integration in IoT ecosystem: Information linkage as a privacy threat.
10. Yacchirema, D. C., Palau, C. E. and Esteve, M. (2017). Enable IoT Interoperability in Ambient Assisted Living: Active and Healthy Aging Scenarios.
11. Andročec, D., Tomaš, ,. B. and Kišasondi, T. (2017). Interoperability and Lightweight Security for Simple IoT Devices.
12. Sun, L., Li, Y. and Memon, R. A. (2017). An Open IoT Framework Based on Microservices Architecture.
13. Lim, N., Majumdar, S. and Nandy, B. (2010). Providing Interoperability for Resource Access Using Web Services.
14. Malik, S. and Kim, D.-H. (2017). A Comparison of RESTful vs. SOAP Web Services in Actuator Networks.
15. Kum, S. W., Moon, J. and Lim, T.-B. (2017). Design of Fog Computing based IoT Application Architecture.
16. Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi We and Limin Sun (2016). Fog Computing: Focusing on Mobile Users at the Edge.
17. Lysogor, I., Voskov, L. and Efremov, S. (2018). Survey of Data Exchange Formats for Heterogeneous LPWAN-Satellite IoT Networks.
18. Nitin Naik. Choice of effective messaging protocols for IoT systems: Mqtt,coap, amqp and http.In2017 IEEE International Systems EngineeringSymposium (ISSE), pages 1–7, 2017
19. Petersen, B., Bindner, H., You, S. and Poulsen, B. (2017). Smart Grid Serialization Comparison.
20. JSON-LD. https://json-ld.org/. Last visited dec 2020.
21. Mengistu, T., Alahmadi, A., Albuali, A., Alsenani, Y., y Che, D. (2018). A N̈o data centeršolution to cloud computing. doi:10.1109/CLOUD.2017.99
22. Paharia, B., y Bhushan, K. (2018). Fog computing as a defensive approach against distributed denial of service (DDoS): a proposed architecture. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-7. doi:10.1109/ICCCNT.2018.8494060