Propuesta de un patrón de arquitecturas de *software* para la interoperabilidad en dispositivos en la capa al borde de un ecosistema IoT

Juan Moreno-Motta
juan.moreno2@unmsm.edu.pe / Escuela de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima, Perú

Felipe Moreno-Vera
felipe.moreno@ucsp.edu.pe / Departamento de Ciencia de la Computación, Universidad Católica San Pablo, Arequipa, Perú

Frank Moreno-Vera
fmorenov@uni.pe / Escuela de Ingeniería Electrónica, Universidad Nacional de Ingeniería, Lima, Perú

Recepción: 31-5-2019 / Aceptación: 9-7-2019

Resumen. En los últimos años el Internet de las cosas ha generado una disrupción en el ecosistema de soluciones para usuarios finales con aplicaciones en la salud, agroindustria, medio ambiente y otras soluciones heterogéneas. Cada una de estas tienen su propio formato para verbalizar los datos proporcionados por los diferentes servicios SOAP, RESTful, REST-LD en formato JSON o XML que pueden ser entendidos por personas. Estos dispositivos se denominan "heterogéneos" porque provienen de diferentes proveedores de manufactura, diferentes lenguajes de programación como C, C++, Lua, Python, JavaScript, etc. Este documento se enfoca sobre dispositivos restringidos, es decir, dispositivos con poca capacidad de procesamiento y memoria con diferentes cadenas de datos sensados. Esta investigación propone un patrón de arquitectura de *software* para la interoperabilidad entre los dispositivos al borde de un ecosistema IoT y entre ecosistemas entre sí.

Palabras clave: interoperabilidad, IoT, dispositivos restringidos, microservicios, semántica web, *encoder*, *decoder*, *fog computing*, *edge computing*, REST API, *software architecture*, *IoT Ecosystem*

Proposal for a Software Architecture Pattern for the Interoperability Between Devices on the Edge of an IoT Ecosystem Layer

Abstract. In recent years, the Internet of Things has generated a disruption in the solution ecosystem for end users with applications in health, agro-industry, environment and other heterogeneous solutions. Each of these solutions has its own format to verbalize the data provided by different web services, including SOAP, RESTful and REST-LD, in JSON or XML format, so that said data can be understood by people. These devices are called "heterogeneous" because they come from different manufacturers and use different programming languages such as C, C++, Lua, Python, Javascript, etc. This document focuses on constrained devices, that is, devices with little processing capacity and memory, and different chains of acquired data. This research proposes a software architecture pattern for interoperability between devices on the edge of an IoT ecosystem and between ecosystems.

Keywords: interoperability, IoT, constrained devices, microservices, semantic web, encoder, decoder, fog computing, edge computing, REST API, software architecture, IoT ecosystem

1. INTRODUCCIÓN

1.1 Revisión previa

El *Internet of Things* (de ahora en adelante, IoT) no es una tecnología nueva, es la convergencia de muchas tecnologías para soluciones como la salud, medio ambiente, manufactura, ahorro de energía, seguridad ciudadana. El IoT es un sistema de objetos físicos que se pueden descubrir, controlar o interactuar con dispositivos electrónicos que se comunican a través de varias interfaces de red y que, finalmente, se pueden conectar a Internet (Guinard y Trifa, 2016). Hasta hace poco, los proyectos de IoT se centraban principalmente en la creación de implementaciones a pequeña escala, cerradas y aisladas en las que los dispositivos no estaban diseñados para ser fácilmente accesibles o reprogramables (Guinard y Trifa, 2016).

En el año 2014, según en el estudio realizado por Oliver Klein (2017), se informó que en el año 2013 existían 1 billón de unidades de Smartphone y que para el año 2017 se esperaba 1.7 billones. En el mismo año, Jaimini (2017) indicó que en el año 2014 National Health Interview Survey realizó un estudio sobre el asma que afecta a los niños en EE. UU. lo cual conlleva a la pérdida de días de escuela y otros costos sociales. En ese año se estimó que para el año 2017 se incrementarían los dispositivos IoT a 15 mil millones y que el almacenamiento de los datos superaría 150 *exabytes*.

1.2 Actualidad

Talavera *et al.* (2017) realizan un trabajo de estado del arte sobre el número de implementaciones de IoT en los sectores de la agroindustria y el medio ambiente, tomando como referencia los resultados de otros, los clasificaron en identificación (3578), en selección (2652), en elegibles (720) y se tomaron finalmente 72 proyectos. Como resultado presentaron una arquitectura que trata de consolidar el ancho espectro de implementación (véase la figura 1). Uno de los retos que consideran en su trabajo de investigación es que una estandarización más sólida ayudaría a mejorar la compatibilidad entre diferentes proveedores y garantizar las medidas de seguridad más sólidas en todas las capas de IoT, desde los dispositivos de campo hasta los proveedores de la nube y las interfaces de usuario final. Este punto se refiere a la interoperabilidad entre las distintas capas de comunicación.

También en el año 2017, se desarrolló una plataforma para el control y monitoreo de la salud de personas dentro de sus hogares. Esta plataforma fue denominada SPHERE (Sensor Platform for HEalthcare in Residential Environment). Este proyecto determinó que existen nueve requerimientos que el IoT necesita cubrir y uno de ellos es la interoperabilidad. En el caso de dispositivos restringidos, es decir, con baja capacidad de memoria y procesamiento, Elsts, Oikonomou, Fafoutis y Piechocki (2017) consideran que los sistemas deben cumplir con los estándares y protocolos de IoT existentes de baja potencia para 1) ser susceptible de futuras extensiones con componentes de terceros; 2) reducir el tiempo de aprendizaje para el nuevo personal.

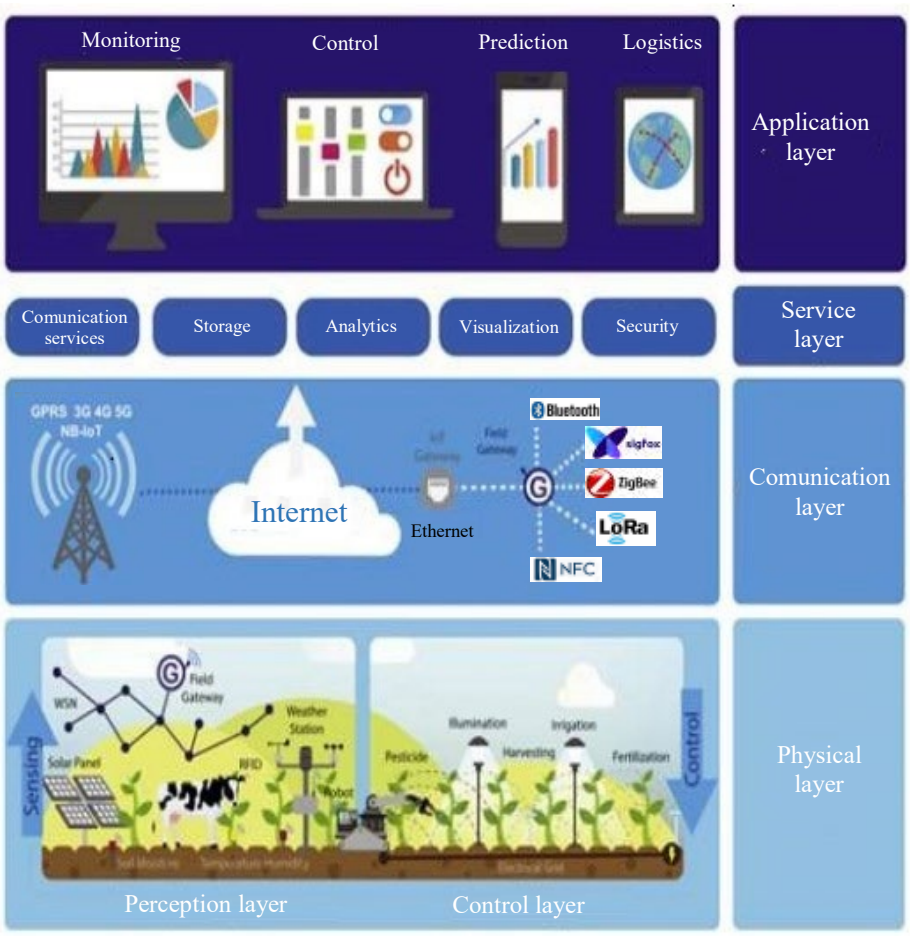

Figura 1. Propuesta de arquitectura IoT para aplicaciones agroindustriales y ambientales

Fuente: Datos tomados de Talavera *et al.* (2017)

Existen muchas soluciones de IoT. Cada implementación de IoT presenta retos de requerimientos no funcionales como son seguridad, almacenamiento, escalabilidad, interoperabilidad y energía. Alkhalil y Ramadan (2017) mencionan que los retos relativos a la procedencia de los datos, a pesar de las técnicas actuales, sigue siendo un desafío en su implementación y optimización. Por lo cual el reto de procedencia de datos se enlaza directamente con la interoperabilidad.

En la actualidad, la interoperabilidad es uno de los desafíos en las implementaciones de soluciones de IoT. Se realiza demasiado esfuerzo al integrar la información generada por los diferentes dispositivos en el dominio de salud, agroindustria y medio ambiente. Dentro de cada dominio existen diferentes sensores, para cada sensor existen diferentes marcas y fabricantes

(Pace *et al.*, 2017). Cada marca y fabricante trae consigo herramientas de desarrollo, con su propio lenguaje de programación. Los datos generados tienen diferentes estructuras tanto en la cabecera como en el cuerpo del mensaje transmitido, generando un riesgo de privacidad aumentando este desafío de interoperabilidad (Madaan, Ahad y Sastry, 2017).

1.3 Propuesta de solución

Después de analizar los trabajos previos, se identificó el reto de la interoperabilidad y la procedencia de los datos como parte de ella en las soluciones de IoT.

Pace *et al.* (2017) presentan su trabajo para apoyar la interoperabilidad de dos aplicaciones de asistencia médica Active and Assisted Living (AAL) mediante la integración de dos plataformas heterogéneas y no interoperables ya existentes como BodyCloud y universalAAL. Este trabajo concluye con la propuesta del marco de trabajo INTER-IoT. Este proyecto tiene un horizonte hasta el 2020 y es un proyecto de la Comunidad Europea. Un caso de uso específico es INTERHealth que se despliega para validar el INTER-IoT propuesto. El proyecto estuvo compuesto por tres bloques de construcción principales: infraestructuras orientadas a capas (INTER-LAYER) para adaptar capas de pares heterogéneas (dispositivo a dispositivo, conexión de red a red, *middleware* a *middleware*, servicios de aplicaciones a servicios de aplicación, datos y semántica a datos y semántica (véase la figura 2).

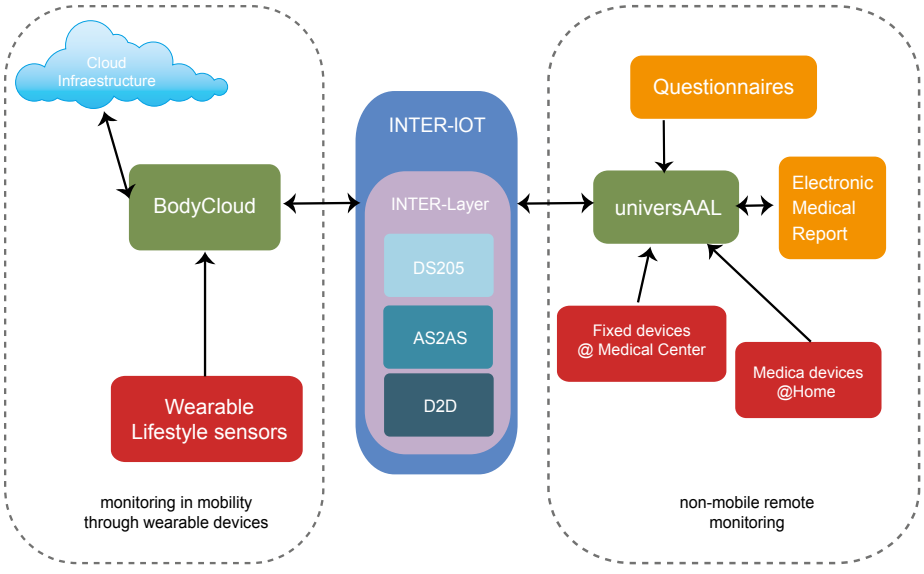

Figura 2. INTER-Health: integración BodyCloud y universAAL

Fuente: Datos tomados de Pace *et al.* (2017)

Di Martino, Esposito, Augusto Maisto, y Nacchia (2017) mencionan que existen varios actores que están en proceso de crear plataformas interoperables como M2M, AllJoyn, OIC y INTER-IoT, revisado líneas arriba. Presentan una propuesta de un marco de trabajo que permite una solución de interoperabilidad. Realizan una investigación sobre los diferentes IDL (*interface description language*) como los APIs WADL, RAML, BluePrint y OpenAPI para RESTful.

Di Martino *et al.*, (2017) presentan un *framework* como propuesta, que se aprecia en la figura 3, en el cual se muestra una arquitectura de alto nivel. Los componentes principales se destacan junto con los flujos de comunicación e interacción entre ellos. El *AgnosticEndPoint* es la interfaz que el usuario puede usar para realizar la solicitud. El *middleware* es el componente que es responsable de comprender si una entrada proporcionada por el usuario es útil para uno de los *endpoints* del proveedor registrado en el sistema; transformar la entrada proporcionada por el usuario en algo soportado por un "*endpoint*" del proveedor específico o múltiple; hacer la solicitud actual a *VendorEndPoint*; transformar la salida del *VendorEndPoint* en el formato estándar proporcionado por el *AgnosticEndPoint*.

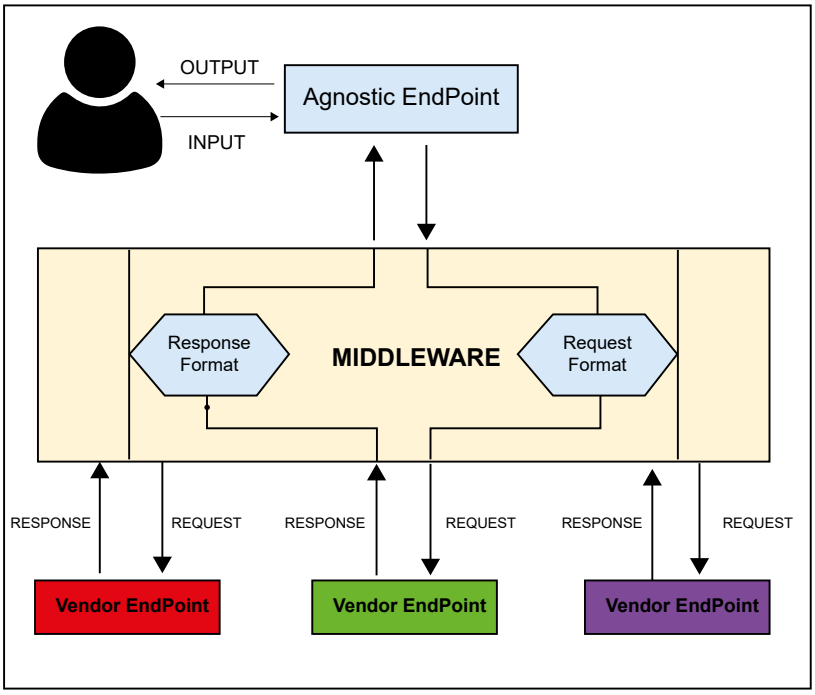

Figura 3. El flujo de proceso del *framework* presentado

Fuente: Di Martino *et al.* (2017)

Yacchirema, Palau y Esteve (2017) proponen una arquitectura con dos capas: capa lógica de aplicación y capa de módulos comunes. La primera capa expone servicios; la segunda, cinco módulos básicos que habilitan la interoperabilidad de IoT para todos los servicios que se quieran supervisar, se la denomina Smart IoT Gateway y es la que aumenta la flexibilidad y escalabilidad del AAL-IoTSys propuesto. Esta arquitectura se desarrolló considerando que los dispositivos IoT, generalmente, son de recursos limitados o restringidos en términos de procesamiento, memoria y almacenamiento. La figura 4 muestra la arquitectura propuesta.

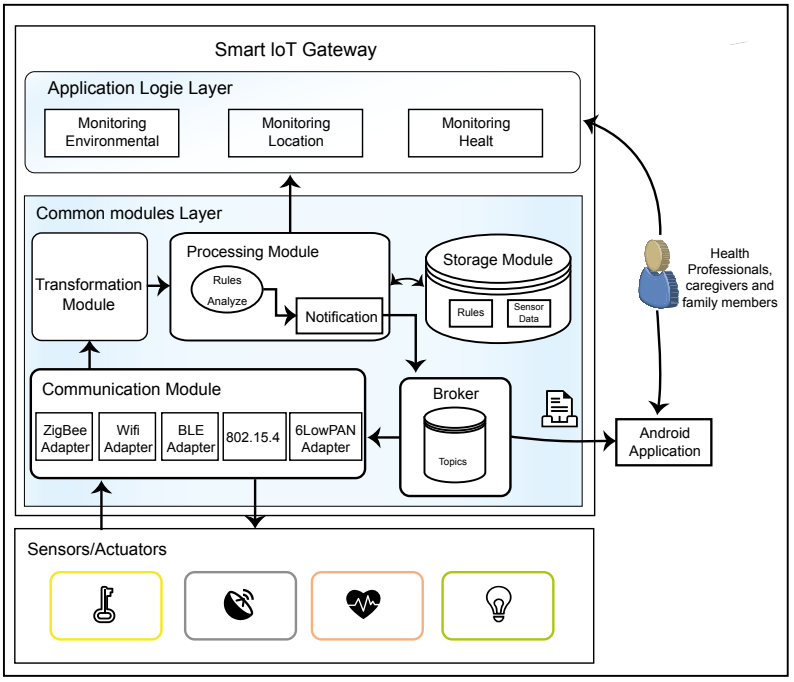

Figura 4. Descripción general de la arquitectura ALL-IoTSys
Fuente: Yacchirema *et al.* (2017)

Andročec, Tomaš y Kišasondi (2017) plantean el uso de la web semántica para permitir la interoperabilidad en IoT utilizando JSON-LD. La idea básica de JSON-LD es crear una descripción de los datos en forma de un llamado contexto. También exploran algunas posibilidades de interoperabilidad y comunicación entre dispositivos IoT económicos.

Sun, Li y Memon (2017) formulan una REST API y una comparación entre la arquitectura monolítica y de microservicios. Utiliza REST API para la comunicación desde los dispositivos, incluyendo el registro del mismo en el ecosistema. La asignación de microservicios a dispositivos funciona dinámicamente al interactuar con el servicio central, proporciona un buen soporte para la (WoT) y tiene una mayor adaptabilidad, escalabilidad e interoperabilidad.

Lim, Majumdar y Nandy (2010) y Malik y Kim (2017) proponen y comparan arquitecturas basadas en servicios SOAP y REST API como propuesta de solución a la interoperabilidad.

Kum, Moon y Lim (2017) exponen una arquitectura novel para construir aplicaciones para *fog computing*, usando el procesamiento de datos en el borde en vez de hacerlo en la nube.

Petersen, Bindner, Poulsen y You (2017a) y Wendt, Faschang, Leber, Pollhammer y Deutsch (2013) hacen una demostración del desempeño de los diferentes formatos llegando a la conclusión de que el formato binario generado con Protobuff desarrollado por Google es el de mejor rendimiento y menor uso de memoria (véase la figura 5).

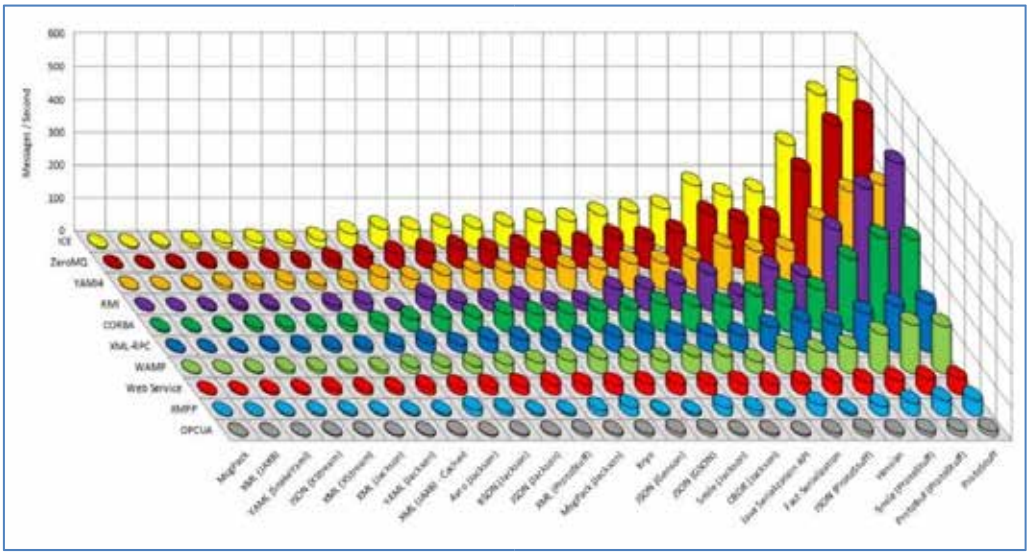

Figura 5. Prueba de rendimiento de *request-response* de formatos con diferentes protocolos

Fuente: Petersen *et al.* (2017a)

2. PROPUESTA

Hemos revisado varias propuestas de arquitectura de *software* para superar la interoperabilidad, así como de formatos de datos SOAP y REST API. Estas van desde microservicios y pasan por web semántica como JSON-LD, en ambos casos se requiere de capacidad de procesamiento y memoria que poseen los servidores, pero no dispositivos restringidos. Nuestro enfoque en este estudio es para dispositivos de bajo procesamiento y capacidad de memoria. La procedencia de los datos forma parte del reto de la interoperabilidad. Consideramos que existen ambientes hostiles de comunicación por lo que la trama de datos serializada debe ser muy compacta. La propuesta va dirigida a la capa al borde de un ecosistema de IoT (véase la figura 6).

Para nuestra propuesta tomamos como base el estudio comparativo de Petersen *et al.* (2017a) donde se demuestra un gran rendimiento utilizando el formato binario para la transferencia de datos desde los dispositivos.

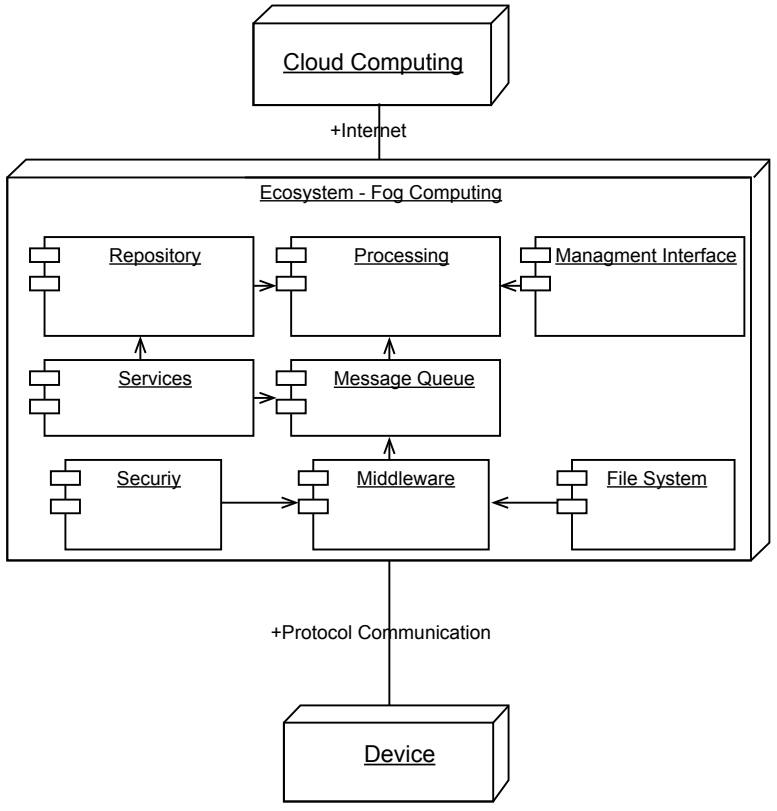

Figura 6. Propuesta de un patrón de arquitectura de *software* de IoT para superar la interoperabilidad en la capa del borde de un ecosistema IoT

Elaboración propia

2.1 Componentes que conforman la arquitectura

Describimos los componentes que conforman la arquitectura propuesta:

Device. Representa todos los dispositivos que se conectan al ecosistema a través del componente *middleware*. Aquí se debe implementar la serialización en formato binario. En general pueden ser sensores y actuadores.

- Sensor: detecta magnitudes físicas o químicas.
- Actuador: transforma la energía para causar un efecto en un elemento externo.

Middleware. Se encarga de recibir el formato binario. Este componente hace uso del componente *processing* que tiene mayor capacidad de procesamiento ya que se comporta como un servidor. Este componente también hace uso del componente *file system* como archivo de configuración del ecosistema.

MessageQueue. Se encarga de proporcionar alta disponibilidad y escalabilidad. Asegura que, ante la presencia de alta concurrencia, la trama de datos de los dispositivos se guarden siempre en la cola de mensajes para que de esa forma gestionar el encolamiento (Rostanski, Grochla y Seman, 2014).

Security. Este componente es un servicio consumido por el *middleware*. A cada *device* se le asigna un ID único mediante un certificado que luego será validado por este servicio.

Processing. Se encarga de procesar los formatos de datos binarios que son recibidos desde el componente *device*.

Repository. Es de persistencia donde se terminarán alojando las tramas de datos.

Managment interface. Interfaz de administración a la cual se puede acceder desde un terminal que puede ser un PC, una laptop o un celular. Mediante este administrador se registran los tipos de dispositivos, los dispositivos, *el middleware* y el ecosistema.

File system. Describe las propiedades del ecosistema y guarda los certificados de los ID de los *device* encriptados.

Services. Es la capa de servicios que el *fog computing* puede entregar para que otros dispositivos externos puedan recibir información (de sensores) o enviar una acción a ejecutar (actuadores).

Fog computing. Alberga componentes descritos. *Fog computing* tiene todas las ventajas del *cloud computing* (Paharia y Bhushan, 2018).

Cloud computing. Este componente, que está fuera del ecosistema de IoT, es el repositorio final y es donde se deben realizar el análisis y los cálculos de gran volumen de información. De tipo propietario de ciento de miles de servidores dedicados. De alta disponibilidad, permite que se puedan generar múltiples análisis de datos para diferentes fines (Mengistu, Alahmadi, Albuali, Alsenani y Che, 2018).

2.2 Descripción de interacción de los componentes

Describimos la interacción entre los componentes, lo primero que se tiene que hacer es identificar cada uno de los dispositivos, se observan los dispositivos D1, D2, D3, D4 y D5, cada uno de ellos tiene una identificación adicional que es el tipo de dispositivo (TD), vemos que los dispositivos D2 y D4 tienen el mismo tipo de dispositivo TD2, esto vendría a ser la categoría. En el dispositivo se deben definir dos bloques de datos, la cabecera de los datos donde se graba

el identificador del dispositivo que llamaremos ID y el identificador del tipo de dispositivo que llamaremos TID. En el cuerpo de los datos de la trama del dispositivo se registran los datos censados. Una vez enviada la trama de datos en formato binario, este llega al *middleware*. En el *middleware* lo primero que se hace es enviar la trama de datos al componente MessageQueue y luego se deserializa utilizando el componente *processing* y se valida que los datos de cabecera estén registrados en el componente *file system.* Previamente en el *file system* se ha registrado el tipo de dispositivo del cual pueden recibir la trama de datos y los dispositivos que le pueden enviar datos. En el componente *processing* se deben instalar los módulos de deserialización. Cuando se recibe un tipo de dispositivo el componente *processing* lo atenderá con el módulo que le corresponde, de esta forma si se registra un nuevo tipo de dispositivo se instalará solo el módulo de deserialización y se registrará en el *file system.* Esto permitirá realizar pruebas de concepto sin afectar a los otros módulos que ya están en operación y permitirá escalabilidad.

2.3 Diseño del *file system*

En nuestra base de datos NOSQL se crea la colección identificando el ecosistema al cual corresponde (véase la figura 9). Dentro de un ecosistema puede existir más de un *middleware.* Un *middleware* puede recibir tramas de varios tipos de dispositivos TD y puede haber varios dispositivos por tipo de dispositivo.

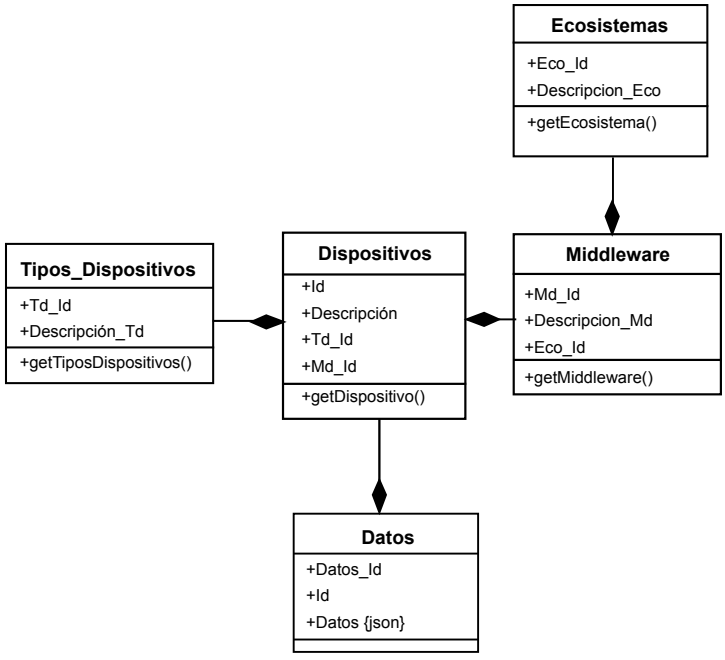

Figura 9. Diseño de datos del *file system*

Elaboración propia

Finalmente, si tuviéramos varios ecosistemas, cada uno tendría su identificación que luego podrían ser consolidados en la nube de tal manera que se aseguraría el escalamiento desde un *fog computing* hacia el *cloud computing.*

Por ejemplo:

ES1 / GW1 / TD1 / D1 / Trama-JSON

ES1 / GW1 / TD2 / D2 / Trama-JSON

ES1 / GW1 / TD2 / D4 / Trama-JSON

2.4 Escalando múltiples ecosistemas IoT mediante interoperabilidad

La figura 10 muestra la capacidad de interoperar entre los *device* con el ecosistema (*fog computing*), respetando el patrón propuesto tipificando los tipos de dispositivos, asignándole un comportamiento único en el proceso de serialización y deserialización. Esto facilita la integración de un nuevo dispositivo en el ecosistema IoT.

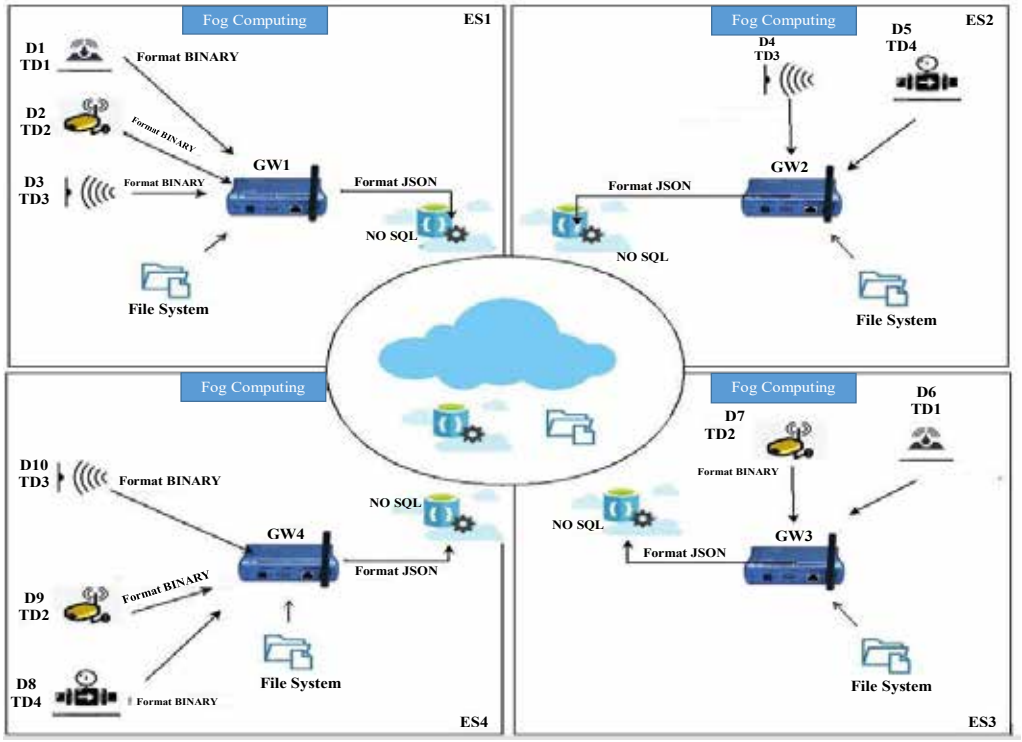

Figura 10. Integrando múltiples ecosistemas IoT
Elaboración propia

3. CONCLUSIONES

En este *paper* proponemos una nueva arquitectura de *software* para superar el reto de la interoperabilidad en soluciones IoT, iniciando con el diseño de un ecosistema (*fog computing*). Identificamos todos los componentes y sus interacciones. Diseñamos la estructura de datos que permitirá determinar la procedencia de los datos haciéndolos únicos dentro del ecosistema. Creamos ID únicos de los dispositivos a través de certificados encriptados para seguridad. Creamos la trama de formato binario para mejorar la velocidad de comunicación. Con este patrón logramos escalabilidad y alta disponibilidad integrándolos a través de un *cloud computing*.

4. TRABAJO FUTURO

Se ha implementado esta arquitectura en una aplicación en tiempo real como prueba de concepto. La solución implementa sensores para medir parámetros de medio ambiente tales como temperatura, humedad, sonido y monóxido, una minicomputadora Raspberri Pi 3 Model B (1.4 GH QuadCore) como *middleware*, una tarjeta Arduino, SQLite para el *file system*, Protocol Buffer para el *encode-decode* de los datos y el servicio de Firebase para hacer *cloud computing* de análisis de datos y almacenamiento.

Está pendiente desarrollar los otros componentes, como también el implementar dos o más ecosistemas con este patrón para demostrar la escalabilidad e identificar la procedencia de los datos mediante este diseño. El diseño de los datos de la prueba de concepto se puede ver en la figura 11.

Además, se debe realizar una prueba de estrés provocando concurrencia de los datos para la implementación del uso del componente MessageQueue (MQ) y asegurar la alta disponibilidad. Una primera implementación del uso de MQ se realizaría con RabbitMQ, sin embargo, luego estudiaremos la implementación de NATS y Kafka para manejo de MQ.

Descripción de las columnas

Los datos que se aprecian en la figura 11 se obtuvieron de un proceso de descarga que formatea en modo fila-columnas, pues los datos están guardados en formato JSON.

Detalle de los nombres de las columnas:

- *gwName*: nombre del *gateway*.
- *gwid*: identificador del *gateway*.
- *devName*: nombre del dispositivo.
- *devid*: identificador del dispositivo.

date	*Oil temperature*	*Water concentration*	*Hidrogen*	*Coil temperature*	*Carbon monoxide*	*gwName*	*devName*	*gwId*	*devid*
1/02/2019 5:10	*27*	*6.4*	*35.1*	*67.8*	*273.6*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 5.34	*27*	*6.4*	*35.1*	*67.9*	*273.6*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 5:46	*26.8*	*6.4*	*35.1*	*67.8*	*273.6*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 5:58	*26.8*	*6.4*	*35.1*	*67.8*	*273.6*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 6:10	*26.7*	*6.4*	*35.1*	*67.8*	*273.6*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 6.22	*26.7*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 6:34	*26.7*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 6:46	*26.7*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 6:58	*26.7*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 7:10	*26.7*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 7:22	*26.7*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 7:34	*26.7*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 7:46	*26.5*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*
1/02/2019 7:58	*26.5*	*6.4*	*35.1*	*67.8*	*280.8*	*SS.EE. Carapongo*	*CARP5000TR114-AUTOT-S*	*LisaGw*	*Lisa1_03*

Figura 11. Ejemplo de datos de dispositivos sensados
Elaboración propia

En nuestra prueba de concepto desarrollamos un dispositivo con múltiples sensores, esto permitió ahorrar en *hardware*.

REFERENCIAS

Alaya, M. B., Drira, K., y Gharbi, G. (2017). Semantic-aware Iot platforms. *2017 IEEE International Conference on AI & Mobile Services (AIMS)*. doi:10.1109/AIMS.2017.15

Alkhalil, A., y Ramadan, R. A. (2017). IoT data provenance implementation challenges. doi:10.1016/j.procs.2017.05.436

Andročec, D., Tomaš, B., y Kišasondi, T. (2017). Interoperability and lightweight security for simple IoT devices. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. doi:10.23919/MIPRO.2017.7973621

Bloebaum, T. H., y Johnsen, F. T. (2015). Exploring SOAP and REST communication on the Android platform. MILCOM 2015 - *2015 IEEE Military Communications Conference*. doi:10.1109/MILCOM.2015.7357509

Di Martino, B., Esposito, A., Augusto Maisto, S., y Nacchia, S. (2017). A semantic IoT framework to support RESTful devices' API interoperability. doi:10.1109/ICNSC.2017.8000071

Dragoni, N., Giallorenzo, S., Lluch Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., y Safina, L. (2017). Microservices: yesterday, today, and tomorrow. doi:10.1007/978-3-319-67425-4_12

Elsts, A., Oikonomou, G., Fafoutis, X., y Piechocki, R. (2017). Internet of things for smart homes: lessons learned from the SPHERE case study. Global Internet of Things Summit (GIoTS), 2017 Institute of Electrical and Electronics Engineers (IEEE). doi:10.1109/GIOTS.2017.8016226

Gligorić, N., Dejanović, I., y Krčo, S. (2011). Performance evaluation of compact binary XML representation for constrained Devices. doi:10.1109/DCOSS.2011.5982183

Guinard, D. D., y Trifa, V. M. (2016). *Building the web of things*. NY: Manning Publications.

Jaimini, U. (2017). PhD Forum: Multimodal IoT and EMR based smart health application for asthma management in children. *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. doi:10.1109/SMARTCOMP.2017.7947025

Jarwar, M. A., Ali, S., Kibria, G. M., Kibria, G., Kumar, S., y Chong, I. (2017). Exploiting interoperable microservices in web objects enabled internet of things. doi:10.1109/ICUFN.2017.7993746

Kum, S. W., Moon, J., y Lim, T.-B. (2017). Design of fog computing based IoT application architecture. doi:10.1109/ICCE-Berlin.2017.8210598

Lim, N., Majumdar, S., y Nandy, B. (2010). Providing interoperability for resource access using web services. doi:10.1109/CNSR.2010.23

Lysogor, I., Voskov, L., y Efremov, S. (2018). Survey of data exchange formats for heterogeneous LPWAN-Satellite IoT networks. doi:10.1109/MWENT.2018.8337257

Madaan, N., Ahad, M. A., y Sastry, S. M. (2017). Data integration in IoT ecosystem: Information linkage as a privacy threat. doi:10.1016/j.clsr.2017.06.007

Malik, S., y Kim, D.-H. (2017). A comparison of RESTful vs. SOAP web services in actuator networks. *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 753-755. doi:10.1109/ICUFN.2017.7993893

Mengistu, T., Alahmadi, A., Albuali, A., Alsenani, Y., y Che, D. (2018). A "No data center" solution to cloud computing. doi:10.1109/CLOUD.2017.99

Nakayama, M., Yamazaki, K., Tanaka, S., y Kasahara, H. (2014). Parallelization of Tree-to-TLV serialization. doi:10.1109/PCCC.2014.7017059

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. doi:10.1109/GIOTS.2017.8016250

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

Petersen, B., Bindner, H., Poulsen, B., y You, S. (2017a). Smart grid communication comparison. Distributed control middleware and serialization comparison for the internet of things. *Proceedings of 7th IEEE International Conference on Innovative Smart Grid Technologies IEEE.* doi:10.1109/ISGTEurope.2017.8260268

Petersen, B., Bindner, H., You, S., y Poulsen, B. (2017b). Smart grid serialization comparison. Comparision of serialization for distributed control in the context of the Internet of Things. *2017 Computing Conference*, 1339-1346. doi:10.1109/sai.2017.8252264

Ray, P. P., Mukherjee, M., y Shu, L. (2017). Internet of things for disaster management: State-of-the-art and prospects. doi:10.1109/ICWS.2017.26

Rostanski, M., Grochla, K., y Seman, A. (2014). Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ. *2014 Federated Conference on Computer Science and Information Systems*, 879-884.

Serrano, D., Stroulia, E., Lau, D., y Ng, T. (2017). Linked REST APIs: A middleware for semantic REST API integration. *2017 IEEE International Conference on Web Services (ICWS)*, 138-145. doi:10.1109/ICWS.2017.26

Silverajan, B., Ocak, M., Jimenez, J., y Kolehmainen, A. (2016). Enhancing lightweight M2M operations for managing IoT gateways. *9th IEEE International Conference on Internet of Things (iThings 2016)* (pp. 187-192). IEEE. doi:10.1109/iThings-GreenCom-CPSCom-SmartData.2016.55

Sun, L., Li, Y., y Memon, R. A. (2017). An open IoT framework based on microservices architecture. *China Communications*, *14*, 154-162. doi:10.1109/CC.2017.7868163

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. *Computers and Electronics in Agriculture*, *142*, 283-297. doi:10.1016/j.compag.2017.09.015

Wendt, A., Faschang, M., Leber, T., Pollhammer, K., y Deutsch, T. (2013). Software architecture for a smart grids test facility. *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 7062-7067. doi:10.1109/IECON.2013.6700304

Yacchirema, D. C., Palau, C. E., y Esteve, M. (2017). Enable IoT interoperability in ambient assisted living: active and healthy aging scenarios. *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 53-58. doi:10.1109/CCNC.2017.7983081

Yigitoglu, E., Liu, L., Looper, M., y Pu, C. (2017). Distributed orchestration in large-scale IoT systems. *2017 IEEE International Congress on Internet of Things (ICIOT)*, 58-65. doi:10.1109/IEEE.ICIOT.2017.16