INVESTIGACIÓN

Análisis de la Plataforma FOGFLOW y su integración con FIWARE en entornos IoT

Analysis of the FOGFLOW platform and its integration with FIWARE in IoT environments

MSc. Yoandi Guzmán Quintero [1*], Dra.C. Tatiana Delgado Fernández [2]

Recibido: 11/2021 | Aceptado: 01/2022

Palabras clave

Computación en la niebla/borde

Computación en la nube

Internet de las Cosas o IoT

FIWARE, NGSI (Interfaz de *software* de Próxima Generación)

FogFlow

Resumen

Actualmente, los proveedores de soluciones para ciudades inteligentes afrontan una situación compleja y costosa que crece exponencialmente, a la hora de gestionar sus infraestructuras distribuidas geográficamente y soportar los distintos servicios IoT ofrecidos, especialmente aquellos que requieren una baja latencia. FogFlow es un *framework* de ejecución distribuido que dinámicamente ensambla los diferentes servicios IoT en la nube y los bordes (Edge) con el fin de reducir el consumo de ancho de banda y ser capaz de ofrecer una baja latencia, ofreciendo una calidad de servicio (QoS) optimizada. Edge-Computing es un modelo de computación que centra y sitúa las capacidades de análisis y procesamiento donde se generan los datos. *Fog Computing* extiende este concepto ubicando la capacidad de procesamiento próximo a la red local donde el flujo de datos es enviado hacia nodos niebla o dispositivos como *routers* o *gateways* de IoT que procesan y enrutan el tráfico hacia donde se necesite. Para un correcto análisis resulta necesario utilizar, un *framework* que sea capaz de simular este escenario de computación en contextos reales, que será presentado en el presente artículo mostrando su funcionamiento, arquitectura y potencialidades de procesamiento e implementación, así como su integración con FiWARE.

Keywords

Fog/ Edge Computing

Cloud-Computing

Internet of Things or IoT

FIWARE, NGSI (Next Generation Software Interface)

FogFlow

Abstract

Currently, smart city solution providers face a complex and costly situation that grows exponentially, when managing their geographically distributed infrastructures and supporting the different IoT services offered, especially those that require low latency. FogFlow is a distributed execution framework that dynamically assembles the different IoT services in the cloud and the edges (Edge) in order to reduce bandwidth consumption and be able to offer low latency, offering an optimized quality of service (QoS). Edge-Computing is a computing model that makes the data analysis and processing capacities be focused on and located where the data is generated. Fog Computing extends this concept by locating the processing capacity close to the

1* ETECSA-DVLH. yoandi.guzman@etecsa.cu
2 Unión de Informáticos de Cuba. tatiana.delgado@uic.cu

local network where the data flow is sent to fog nodes or devices such as routers or IoT gateways that process and route the traffic to where it is needed. For a correct analysis, it is necessary to use a framework to simulate this computing scenario in real contexts. This framework will be provided in this article as well as its operation, architecture and processing and implementation potentialities, including also its integration with FiWARE.

Introducción

Internet de las Cosas (IoT) habilita una sociedad hiperconectada en la que cada vez hay más presencia de objetos como autos, drones o edificios, convirtiéndose en elementos inteligentes, capaces de reaccionar a situaciones en tiempo real utilizando la información generada para un determinado contexto por sensores o actuadores y fuentes de datos. Estos objetos inteligentes son gestionados normalmente por servicios que se ejecutan en infraestructuras de *backend*.

Uno de los grandes retos hoy en día es habilitar métodos fáciles, rápidos y eficientes, que permitan orquestar los servicios, garantizando reaccionen de manera acelerada utilizando el máximo de información posible. La integración de las tecnologías del IoT, el *Cloud Computing* y el *Big Data*, junto al apoyo de las políticas de datos abiertos, ha permitido crear las condiciones para la transformación de las ciudades.

La premisa inicial de una *Smart City* —Ciudad Inteligente— es su condición de ciudad conectada. Para ello, uno de los pilares centrales sobre los que se basa su construcción es la digitalización, en general el uso de las TIC —Teconologías de la Información y las Comunicaciones—. Su objetivo último es permitir la disponibilidad de información sobre la ciudad y la integración de distintos servicios.

A diferencia de las analíticas de datos tradicionales, la orquestación de servicios IoT debe tener en consideración los siguientes puntos medulares (Cheng *et al*., 2017).

Generación de nuevos datos constantemente por los sensores, no siendo sostenible enviar todos los datos en bruto (*raw*) a servicios *cloud* centralizados para su procesamiento, debido fundamentalmente a las restricciones de ancho de banda y latencia existentes. El procesamiento de datos debe descargarse dinámicamente sobre el *edge*, cerca de la fuente de datos. Los datos y el conocimiento existentes deben poder compartirse e intercambiarse entre dispositivos, servicios, aplicaciones y plataformas.

Las cargas de trabajo del sistema son dinámicas. Los dispositivos, servicios y aplicaciones se mueven, reconectan o desaparecen. Esto lleva a cargas y flujos de datos en constante cambio.

Algunos servicios IoT requieren baja latencia y un tiempo de respuesta muy rápido.

Las infraestructuras de *backend* necesitan administrar recursos y dispositivos muy heterogéneos y geográficamente distribuidos.

Todos estos aspectos suponen nuevos desafíos y añaden un alto grado de complejidad tecnológica a los proveedores de infraestructura para administrar los diferentes servicios IoT ofrecidos. En las ciudades inteligentes existe una amplia heterogeneidad y movilidad de dispositivos IoT, y se requieren altos niveles de escalabilidad e interoperabilidad entre las soluciones desplegadas. Los datos recopilados de los dispositivos IoT son procesados y almacenados utilizando las posibilidades que brinda la computación en la nube. Sin embargo, a pesar de su poder, el modelo de la nube no es aplicable, por sí solo, a entornos en los cuales las operaciones son sensibles a la latencia, como la telemedicina, entre otros, donde los milisegundos pueden tener importantes consecuencias. Tener cada dispositivo conectado a la nube enviando datos, a través de Internet, puede traer implicaciones legales, de seguridad y de privacidad. La complejidad de la gestión de grandes cantidades de datos, debido al constante aumento de dispositivos IoT, también puede constituir una dificultad si se realiza de forma centralizada pues se ocupa un gran ancho de banda en la transmisión de dichos datos. Por otra parte, en la actualidad, la mayoría de los dispositivos de IoT no cuentan con los recursos de cómputo y de almacenamiento requeridos para realizar tareas de análisis y aprendizaje automático; los

ISSN: 1813-5056, RNPS: 0514, Vol. 18, No. 1, enero - julio, 2022, pp. 46 - 57 fono

servidores de la nube cuentan con los recursos necesarios, pero están demasiado lejos para procesar los datos y responder a tiempo a las peticiones que se generan más cerca del borde de la red (Pérez *et al*., 2021).

La computación en la niebla —*Fog Computing*— surge como un paradigma novedoso para dar solución a lo antes planteado. Con el empleo de recursos de computación, almacenamiento y redes imita las capacidades de la nube, respalda la ingestión local de datos y el procesamiento de las aplicaciones sensibles a la latencia en el borde de la red; mientras que otras, con alta tolerancia al retardo y que requieren de un uso intensivo de recursos de computación, pueden procesarse en la nube. Esto significa que la computación en la niebla permite el procesamiento más cerca del borde de la red sin dejar de ofrecer la posibilidad de interactuar con la nube (Pérez *et al*., 2021).

Simular una red o infraestructura es entonces una herramienta útil de análisis y diseño que permite conformar una idea del comportamiento de un sistema real. En concreto, un simulador es un *software* capaz de modelar y ejecutar las operaciones de dispositivos de redes reales y sus interacciones. Brinda análisis detallado del proceso y datos de utilidad para su despliegue real (Mahmud y Buyya, 2017).

El trabajo con simuladores proporciona un grupo de ventajas y desventajas. Entre las desventajas, la reproducción que proporciona un simulador no es del todo exacta, desde la complejidad de ciertos escenarios reales hasta la ausencia de eventos no controlados, atenta contra la fiabilidad de muchas herramientas de simulación (Mahmud y Buyya, 2017).

Materiales y Métodos

Para el desarrollo del presente artículo se han suscitado 4 fases:

La primera fase consiste en la recopilación de la información, principalmente acerca de las diferencias y funcionamiento de los *framework* para la simulación de la computación en la niebla y el borde, así como de las tecnologías y protocolos que lo hacen posible. Para ello se han consultado múltiples recursos bibliográficos, contrastando la información obtenida de las fuentes.

La segunda fase consiste en la presentación del funcionamiento, Arquitectura, Componentes y Modelos de Programación del *framework*, asociando una familiarización con el mismo, tomando como referencia los manuales y características. Experimentando con las distintas posibilidades que ofrece.

En la tercera fase se representa el escenario de integración de FogFlow con *FIWARE* desplegando comandos y códigos necesarios para ello.

En la cuarta y última fase se exponen dos casos de uso aplicando los modelos de programación utilizados por el *framework*.

Resultados y discusión

La integración de IoT, el borde y la nube ha incentivado muchos trabajos de investigación. Muchos artículos han presentado los desafíos y requisitos para dicha integración. Recientemente, se han presentado muchas herramientas que admiten la simulación y emulación de la computación en la niebla y de borde. Aunque los estudios empíricos exhaustivos aportan conocimientos valiosos, los nuevos diseños de bordes no pueden hacer uso de estudios empíricos. Por lo tanto, para nuevos diseños de bordes, los investigadores han utilizado intensamente la programación de tareas, los recursos informáticos y las estructuras de red, las simulaciones y los métodos de estimación para validar experimentos. Hasta la fecha, no faltan artículos que describan o analicen escenarios de IoT/Cloud y el número aumenta continuamente.

La mayoría de los marcos de simulación de borde/niebla son simuladores de eventos discretos, muchos se construyen a partir de simuladores de red y nube. La mayoría de estos *frameworks* de simulación de computación de borde/niebla están dedicados al estudio de infraestructuras, especialmente la gestión de recursos. MyiFogSim simplemente simula la migración de una máquina virtual. EdgeCloudSim se concentra en la simulación de rendimiento para la ejecución de tareas de borde en arquitecturas de borde. iFogSim es una herramienta de código abierto altamente escalable, que mide parámetros como ocupación de la red, latencia, consumo de procesamiento y de energía, para diferentes políticas de gestión de recursos y aplicaciones en ambientes de computación en niebla. Se desarrolla sobre el marco fundamental de CloudSim (Gámez *et al*., 2020). CloudSim es uno de los simuladores ampliamente adoptados para modelar el entorno de computación en la nube, pero solo para eventos discretos. IFogSim además permite definir la ubicación de los dispositivos que reciben un servicio de los servidores

de la niebla, información que resulta estática y no actualizada por ningún modelo de movilidad, por lo que esta versión no permite movilidad, además de limitada escalabilidad (Calheiros y Ranjan, 2010).

EmuFog permite definir una topología de infraestructuras y realizar simulaciones de topología de red/niebla utilizando simulaciones de eventos discretos. FogNetSim ++ se enfoca en redes de borde/niebla y nuevamente es una simulación basada en eventos puramente discretos. En otras investigaciones se analiza la combinación de diferentes simulaciones para crear una simulación híbrida en escenarios de IoT /Edge. Sin embargo, los métodos de simulación no se basan en códigos y sistemas reales. Desde el punto de vista de la implementación, se basan en herramientas y código de simulación puro, por ejemplo, utilizando OMNeT ++ y MATLAB. Tampoco se centra en los servicios y los aspectos de interoperabilidad (Hong, 2021).

En general, la mayoría de los trabajos relacionados se centran en métricas comunes, como el rendimiento y la escalabilidad a través de simulaciones teóricas. Tampoco permiten el estudio de las relaciones con las partes interesadas, la integración con servicios del mundo real y el uso de infraestructuras del mundo real para ejecutar la simulación. Además, el soporte se debe basar en escenarios, más que en problemas específicos, como la gestión de recursos, el consumo de energía o el rendimiento.

FogFlow es diferente porque se enfoca en artefactos de *software* reales que se ejecutan en bancos de pruebas reales que emulan sistemas de borde para interacciones entre aplicaciones. En principio, es posible estimar los costos del trabajo mediante la medición del uso del servicio. ***FogFlow*** utiliza el monitoreo integrado con los servicios, pudiendo obtener la información para determinar costos.

En este sentido, ***FogFlow*** está diseñado para encargarse de estos problemas complejos y ayudar a administrar estos servicios IoT de manera automática y eficiente en un entorno compartido y distribuido. ***FogFlow*** provee un modelo de programación estándar y centrado en el dato para que los proveedores de servicios puedan de manera sencilla y rápida adaptarse a las demandas.

Diferenciación

Respecto a otros *frameworks* de computación en el borde existentes como *EdgeX*, *Azure IoT Edge y Amazon Greengrass*, ***FogFlow*** presenta las siguientes características únicas como se ilustra en la figura 1 (Carpintero, 2019):

Programación centrada en los datos: Facilita el diseño y uso de los servicios IoT al proporcionar un modelo de programación centrado en los datos para diferentes roles o puntos de vista, de cara a expresar los objetivos en diferentes niveles de una manera más intuitiva. A nivel de operador, los proveedores solo tienen que anotar qué tipo de datos pueden manejar, qué funcionalidad se proporciona y qué tipo de resultados se producen. A nivel de servicio, los diseñadores pueden componer fácilmente diferentes operadores para configurar las plantillas del servicio en pocos minutos. Por último, los usuarios del servicio pueden definir a alto nivel el uso que va a tener el dato, de cara a que, durante el procesamiento de datos en tiempo de ejecución, ***FogFlow*** pueda organizar automáticamente los flujos en función de esta información.

Los usuarios del servicio pueden ser tanto productores de datos como consumidores de los resultados. Desde la perspectiva del productor, pueden establecer qué tipo de lógica se aplicará a sus datos, mientras que desde la perspectiva del consumidor pueden fijar el tipo de resultado que se generará. Lo que hace ***FogFlow*** es "traducir" estos objetivos de uso de los datos definidos a alto nivel en flujos de procesamiento concretos, para después desplegarlos y mantenerlos sin interrupciones en los nodos *cloud/edge* disponibles de la manera más eficiente posible. Más relevante aún resulta que, con este modelo de programación centrado en el dato y basado en un modelo de datos estándar, los flujos de procesamiento subyacentes pueden compartirse y optimizarse entre múltiples servicios y usuarios.

Gestión Autónoma: ***FogFlow*** puede llevar a cabo decisiones de orquestación de servicios de IoT de manera descentralizada y autónoma. Esto significa que cada nodo *edge* puede tomar sus propias decisiones teniendo en cuenta solo una vista de contexto local. De esta manera, la mayoría de las cargas de trabajo se pueden manejar directamente en los bordes sin depender de una nube central. Este enfoque "sin nubes" permite no solo poder proporcionar un tiempo de respuesta rápido, sino que también permite obtener una gran escalabilidad y fiabilidad.

Implementación optimizada: La configuración de las tareas y la implementación de los flujos de procesamiento se optimizan para que el entorno comple-

ISSN: 1813-5056, RNPS: 0514, Vol. 18, No. 1, enero - julio, 2022, pp. 46 - 57 tono

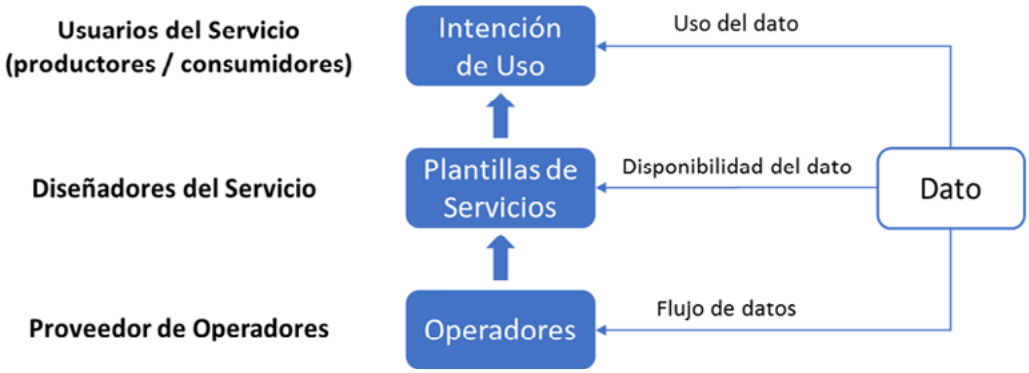

Figura 1. Roles en *FogFlow*

to *cloud/edge* cumpla determinados objetivos, por ejemplo, minimizar el tráfico interno de datos entre los nodos *cloud/edge* o minimizar la latencia a la hora de producir los resultados. Cabe destacar que la optimización de flujos de procesamiento de datos no solo ocurre al inicio del servicio, sino que continúa durante toda la vida útil del servicio adaptándose el mismo de forma dinámica. Uno de estos comportamientos de optimización es la migración dinámica de tareas de un borde a otro para mantener el tiempo de respuesta mínimo para objetos móviles, como pueden ser *smartphones*, coches conectados o drones.

La orquestación del servicio en FogFlow está impulsada por el contexto, en lugar de eventos o textos en bruto. Esta característica es habilitada por la introducción en el diseño de una nueva capa, a saber, *IoT Discovery* como se muestra en la figura 2 (Carpintero, 2019), que proporciona una actualización resumida de los datos de entidad disponibles sobre todos los intermediarios. En comparación con la orquestación basada en eventos o temas, la orquestación basada en el contexto en ***FogFlow*** es más flexible y más liviana. Esto se debe a que, en la orquestación, las decisiones se pueden tomar en función de un contexto agregado, sin leer todos los flujos de datos involucrados. Por otro lado, ***FogFlow*** tiene en cuenta las intenciones de alto nivel definidas por usuarios para tomar decisiones de orquestación optimizadas para una mejor QoS (Hong, 2021).

Los servicios y aplicaciones de ***FogFlow*** están diseñados con una vista global de todos los nodos de la nube y los nodos de borde, en este sentido ***FogFlow*** puede admitir aplicaciones bien distribuidas que podrían ejecutarse en todos los nodos de la nube y nodos de borde sin problemas, sin conocer los detalles de cómo están planificadas y deben llevarse a cabo las tareas entre la nube y el borde o entre los diferentes nodos de borde. Sin embargo, para la mayoría de los demás *frameworks* de *IoT/Edge* (Hong, 2021), los servicios o aplicaciones están diseñados para cada borde y no son realmente servicios o aplicaciones distribuidos, porque esos servicios o aplicaciones pueden ejecutarse en la nube o en algún perímetro, pero no pueden ejecutarse sobre nodos de la nube y de borde en una fusión distribuida.

Las diferenciaciones más detalladas se resumen en la tabla 1:

Sistemas	FogFlow	Otros(EdgeX, Azure IoT, Edge/ AWS Greengrass)
Mecanismo-Desencadenante	Basado contexto	Basado tema
Vista para Orquestación	Global (Todo Borde + Nube)	Cada Borde + Respaldo Broker en la nube
Modelos Programación	Funciones guiadas por el contexto	Funciones Guiadas por Tema
Soporte de Movilidad	Si	No

Tabla 1. Diferenciación *Frameworks*

Por lo que el *framework* idóneo para la simulación del paradigma de computación en la NIEBLA y el BORDE, para escenarios reales es ***FogFlow***, a

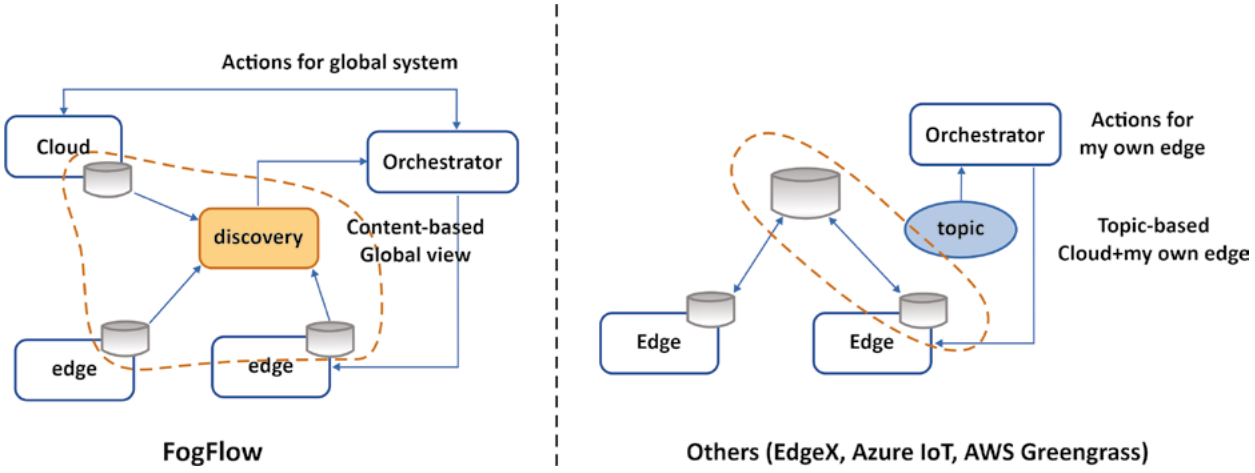

Figura 2. Características ***FogFlow*** vs Otros *frameworks*

continuación serán descritos el funcionamiento, la arquitectura, los modelos de programación y los casos de usos.

Funcionamiento *FogFlow*

En ***FogFlow***, un servicio IoT es definido como un flujo de procesamiento de datos representado por operadores vinculados. Un operador toma información directamente de los dispositivos IoT o de partes anteriores del flujo de procesamiento, realiza una cierta lógica de negocio del servicio IoT y pasa el resultado al siguiente operador. Las tareas se vinculan entre sí durante el tiempo de ejecución en función de la dependencia de datos entre entradas y salidas. **(**Plataforma FIWARE FOGFLOW, 2021)

Como se muestra en la figura 3 (Carpintero, 2019), los servicios se organizan como flujos de procesamiento dinámicos entre productores (por ejemplo, sensores) y consumidores (como actuadores o aplicaciones) para realizar el procesado de datos necesario.

En primer lugar, los desarrolladores de servicios deben crear una plantilla del servicio para definir la lógica del mismo. ***FogFlow*** proporciona un editor de flujos gráfico basado en la web para el diseño de las diferentes tareas. La plantilla del servicio representa la lógica abstracta y estática de procesamiento de datos del servicio IoT, incluidos los detalles sobre qué operador se utiliza para tomar qué entrada y producir qué salida, y también cuándo

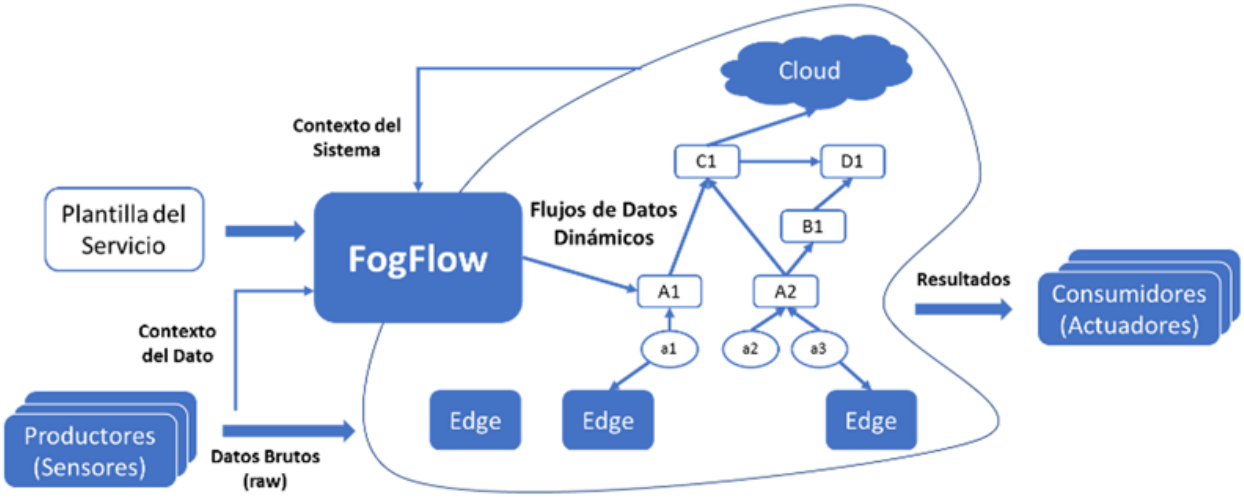

Figura 3. Modelo de alto nivel de FogFlow

ISSN: 1813-5056, RNPS: 0514, Vol. 18, No. 1, enero - julio, 2022, pp. 46 - 57 tono

y cómo debe activarse el operador. Los proveedores de servicios pueden así mismo reutilizar operadores ya registrados o implementar sus propios operadores (Plataforma FIWARE FOGFLOW, 2021).

Una vez definida la plantilla, el sistema monitorea la información de contexto de los datos disponibles en el sistema en tiempo de ejecución para determinar cuándo y cómo se debe instanciar el servicio. Así mismo, ***FogFlow*** también determina cuántas instancias de tarea se requieren para cada operador y también las configuraciones detalladas de cada instancia.

La estructura de datos de todos los flujos de datos se describe según un mismo modelo de datos normalizado denominado NGSI. ***FogFlow*** puede aprender qué tipo de datos se crean en cada nodo del borde —*edge*— y lanzar flujos dinámicos de procesamiento de datos para cada nodo en función de la disponibilidad de la metainformación de contexto registrada. Aplicando algoritmos de optimización se determina automáticamente, en qué lugar implementar las instancias de la tarea de acuerdo con el contexto del sistema en tiempo real, como puede ser la cantidad de recursos disponibles en cada nodo edge o la carga de trabajo prevista.

Arquitectura del Sistema

El marco de ***FogFlow*** opera en una infraestructura de TIC geodistribuida, jerárquica y heterogénea que incluye nodos de nube, nodos de borde y dispositivos de IoT. La siguiente figura 4 ilustra la arquitectura del sistema de ***FogFlow*** y sus componentes principales en tres capas lógicas (Plataforma FIWARE FOGFLOW, 2021):

gestión de servicios: convierte los requisitos del servicio en un plan de ejecución concreto y luego implementa los planes de ejecución sobre la nube y los bordes.

gestión de contexto: gestiona toda la información de contexto y la hace visible y accesible a través de interfaces de consulta y suscripción.

procesamiento de datos: inicia tareas de procesamiento de datos y establece flujos de datos entre tareas

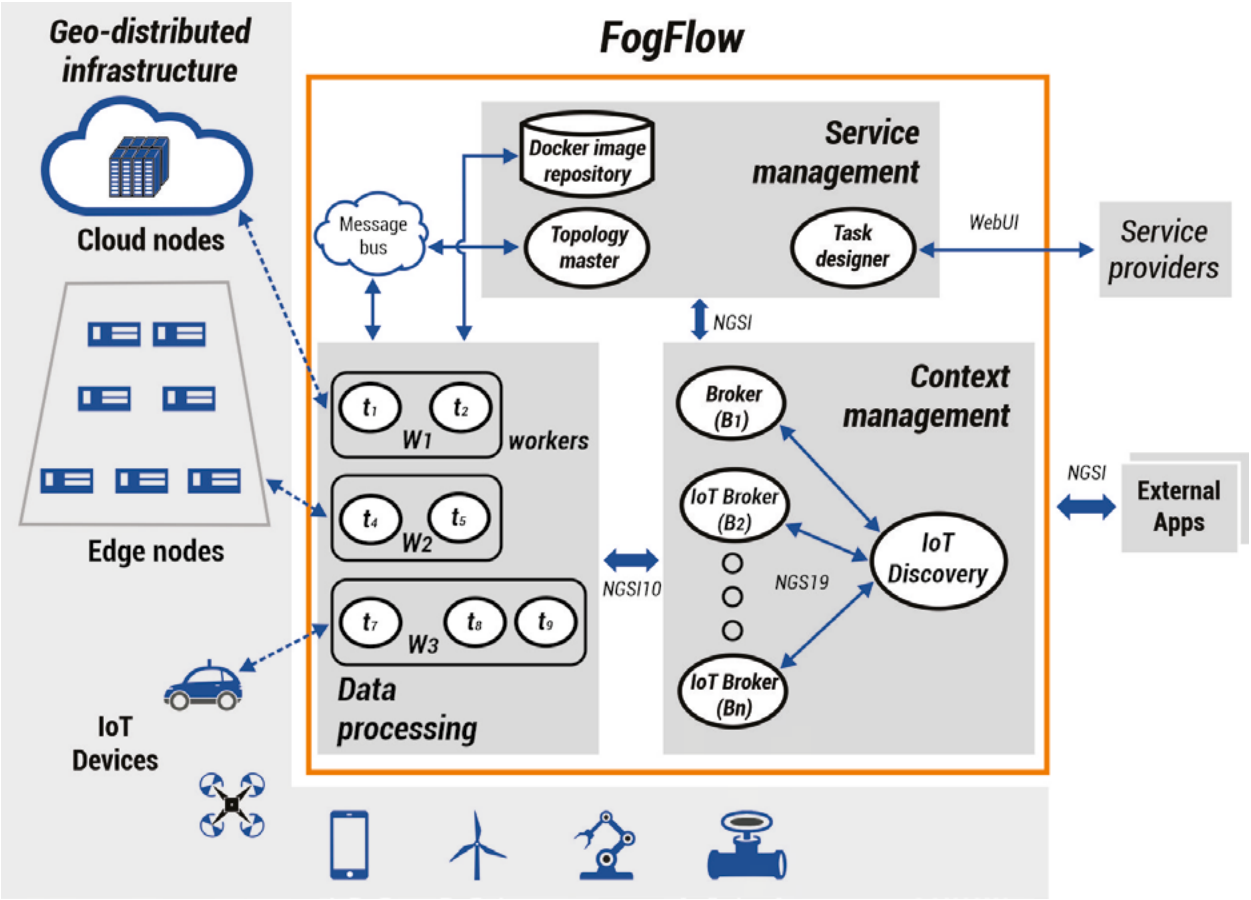

Figura 4. Arquitectura Sistema FogFlow

a través de las interfaces pub / sub proporcionada por la capa de gestión de contexto.

Componentes del sistema

Dispone además de los siguientes componentes de servicio centralizados para ser implementados en la nube (Plataforma FIWARE FOGFLOW, 2021):

Diseñador de tareas: proporciona la interfaz basada en web para que los proveedores de servicios especifiquen, registren y administren sus tareas y topologías de servicios;

Topology Master: descubre cuándo y qué tarea debe instanciarse, configuración dinámicamente y también decide dónde implementarlos en la nube y los bordes;

Descubrimiento de IoT: administra toda la información de disponibilidad de contexto registrada, incluida su ID, tipo de entidad, lista de atributos, y metadatos; permite que otros componentes consulten y suscriban su información de disponibilidad de contexto interesado a través de NGSI9;

Componentes distribuidos para ser implementados tanto en la nube como en los bordes;

Worker: según la asignación del maestro de topología, cada *worker* lanzará su tarea programada instanciadas en contenedores docker en su host local; configura sus entradas/salidas y administra todas las instancias de tareas localmente según la prioridad de la tarea;

IoT Broker: cada agente gestiona una parte de las entidades de contexto publicadas por dispositivos de IoT cercanos y también proporciona una vista única de todas las entidades de contexto para que los dispositivos de *IoT* consulten y suscriban las entidades que necesitan.

Componentes de servicio externos que se utilizarán en ***FogFlow***:

Dock Registry: administra todas las imágenes de la ventana acoplable proporcionadas por los desarrolladores;

RabbitMQ: la comunicación interna entre el maestro de topología y los workers

PostgreSQL: la base de datos *backend* para guardar la información de disponibilidad de contexto registrada.

MODELOS DE PROGRAMACIÓN

En la actualidad ***FogFlow*** proporciona dos modelos de programación diferenciados para admitir diferentes tipos de patrones de carga de trabajo.

Topología de servicio

El primer patrón es activar los flujos de procesamiento necesarios para producir algunos datos de salida solo cuando los consumidores solicitan los datos de salida.

Para definir un servicio IoT basado en este patrón, el proveedor de servicios necesita definir una topología de servicio, que consiste en un conjunto de operadores vinculados en el que cada operador está definido con una granularidad específica. ***FogFlow*** tomará en cuenta la granularidad del operador para decidir cuántas instancias de tarea de dicho operador se deben instanciar en función de los datos disponibles (Plataforma FIWARE FOGFLOW, 2021).

Una topología de servicio debe ser activada explícitamente por una petición emitida por un consumidor o aplicación. La petición definirá qué parte de la lógica de procesamiento en la topología de servicio debe activarse y también puede definir opcionalmente un alcance geográfico específico, de cara a filtrar las fuentes de datos para aplicar la lógica de procesamiento activada.

Topología de niebla

El segundo patrón está diseñado para un escenario en el que los diseñadores del servicio no conocen a priori la secuencia exacta de pasos de procesamiento de la secuencia. En su lugar, pueden definir una función de niebla o función *fog* para incluir un operador específico que manejará un tipo de información. ***FogFlow*** puede crear la gráfica de flujos de procesamiento en base a esta descripción de todas las funciones *fog*.

A diferencia de la topología de servicio, esta topología de niebla es muy simple, con un solo operador que se activa cuando sus datos de entrada están disponibles. Dado que se pueden encadenar automáticamente diferentes funciones de niebla y permitir que más de una función de niebla maneje los nuevos datos, durante el tiempo de ejecución se pueden activar y administrar automáticamente nuevas instancias en función de la carga de datos.

Desde la perspectiva del diseño, la función de niebla es más flexible que la topología de servicio, ya que la lógica de procesamiento general de un servicio IoT se puede modificar fácilmente con el tiempo, agregando o eliminando funciones de niebla cuando la lógica de procesamiento del servicio deba modificarse para dar respuesta a nuevos requisitos. Con este

ISSN: 1813-5056, RNPS: 0514, Vol. 18, No. 1, enero - julio, 2022, pp. 46 - 57 tono

modelo de programación basado en funciones de niebla, FogFlow soporta arquitecturas de computación sin servidor —*serverless computing*— en entornos *cloud/edge* (Plataforma FIWARE FOGFLOW, 2021).

INTEGRACIÓN CON *FIWARE*

El módulo ***FogFlow*** como *Generic Enabler (GE)* en el ecosistema *FIWARE* ocupa una posición ideal como orquestador *cloud/edge* para procesar y administrar flujos de datos dinámicos sobre la nube y los bordes de la red, así como la inserción o transformación de datos y el análisis avanzado. Como muestra la figura 5 (Carpintero, 2019) puede interactuar coordinadamente con otros módulos de *FIWARE* de manera nativa y proveer servicios avanzados a distintas plataformas inteligentes.

FogFlow puede trabajar, por un lado, conjuntamente con el *Context Broker Orion*, pieza central de *FIWARE* y la mayoría de plataformas inteligentes basadas en esta iniciativa, mediante notificaciones de interfaces NGSI estandarizadas, y a través de esta con cualquier otro *Generic Enabler* o aplicación *FIWARE* en la capa superior. En la inferior, al trabajar de manera acoplada y reutilizar los conocidos Adaptadores (Agentes IoT) basados en el modelo de programación de función niebla, permite la integración de cualquiera de los principales protocolos No-NGSI soportados por *FIWARE* nativamente, como *MQTT, COAP, OneM2M, OPC-UA o LoRaWAN*. *FogFlow* puede dinámicamente desplegar los adaptadores necesarios para la integración directa de los dispositivos en cualquier nodo *edge* que lo requiera. Posibilitando que cualquier nodo ***FogFlow*** sea capaz de comunicarse con una amplia gama de dispositivos IoT cubriendo prácticamente la totalidad de casos posibles. (Plataforma FIWARE FOGFLOW, 2021)

Para configurar *Orion* se necesita un requisito previo, tener el *DOCKER* instalado, además el contenedor de *Orion* depende de la base de datos *MongoDB*, contenedor que debe ser iniciado utilizando el siguiente comando:

sudo docker run --name mongodb -d mongo:3.4

y luego se ejecuta *Orion* con este comando:

sudo docker run -d --name orion1 --link mongodb:mongodb -p 1026:1026 fiware/orion -dbhost mongodb

finalmente se comprueba que todo funciona con:

curl http://<Orion IP>:1026/version

Se debe permitir el puerto 1026 en el *firewall* para acceso público.

Para emitir una suscripción reenviando el resultado generado a *Orion Context Broker* se debe utilizar la siguiente solicitud *curl* para suscribir *Fogflow Broker* a *FIWARE Orion*:

```
curl -iX POST \
 'http://coreservice_ip/ngsi10/subscribeContext' \
 -H 'Content-Type: application/json' \
 -H 'Destination: orion-broker' \
 -d '
{
 "entities": [
  {
   "id": ".*",
   "type": "Result",
   "isPattern": true
  }
 ],
   "reference": "http://<Orion IP>:1026/v2/op/notify"
}'
```

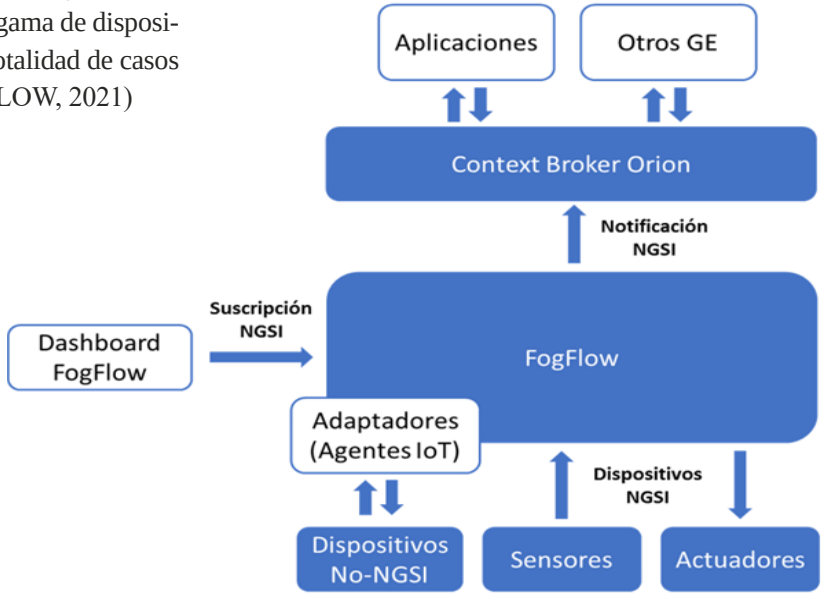

Figura 5. Integración con otros GE's de FIWARE

Tenga en cuenta que esta solicitud de suscripción no utiliza restricciones ni atributos, es una solicitud de suscripción general basada en el tipo de entidad.

Para consultar el resultado de *Orion Context Broker*, la primera forma es considerar a *Orion Broker* como el destino de cualquier información de contexto generada por un servicio *FogFlow/IoT*. En este caso, una aplicación externa o el panel de control de *FogFlow* debe emitir una suscripción a NGSI para solicitar a ***FogFlow*** que reenvíe las actualizaciones de contexto solicitadas a un agente de *Orion* específico.

Se proporcionan dos formas de decirle a ***FogFlow*** qué entidad debe enviarse al corredor de *Orion*. La primera forma es emitir una suscripción sin procesar a ***FogFlow*** *Broker*. La segunda forma es escribir un pequeño programa JavaScript para hacer esto. La integración utiliza la interfaz *NGSI V2* de *Orion Broker*.

```
Function subscribeFogFlow(entityType, orionBroker)
{
   var subscribeCtxReq = {};
    subscribeCtxReq.entities = [{type: entityType, isPattern: true}];
    subscribeCtxReq.reference =  'http://' + orionBroker + '/v2/op/notify';

      client.subscribeContext4Orion(subscribeCtxReq).then( function(subscriptionId) {
        console.log(subscriptionId);
        ngsiproxy.reportSubID(subscriptionId);
      }).catch(function(error) {
        console.log('failed to subscribe context');
      });
}

// client to interact with IoT Broker
var client = new NGSI10Client(config.brokerURL);

subscribeFogFlow('PowerPanel', 'cpaasio-fogflow.inf.um.es:1026');
```

CASOS DE USO

A continuación, se explican dos casos de uso en los que ***FogFlow*** puede emplearse para el diseño de servicios de Smart. El primero de ellos está basado en la topología de servicio, mientras que el segundo está basado en funciones de niebla.

Buscador de niños perdidos

Este caso de uso pretende localizar a un niño perdido lo antes posible aprovechando la computación *edge* habilitada por ***FogFlow*** (Figura 6).

Un servicio fácilmente implementable basado en ***FogFlow*** para la localización de un niño usando la topología de servicio podría ser el siguiente, contando el mismo de tres operadores diferentes:

Extracción de rostros, encargado de reconocer y extraer imágenes de rostros de secuencias de video de las cámaras existentes.

Generación de rasgos, que, en base a una imagen de rostro, calcula el vector de características únicas para cada rostro detectado.

Coincidencia de rostros, que compara los rostros detectados con una imagen de referencia (la cara del niño perdido) en términos de sus vectores de características y rasgos.

Cabe destacar que se define una granularidad diferente para cada operador en la topología de servicio, por ejemplo, la coincidencia de rostros se instancia por cada persona a detectar, mientras que los otros dos operadores se instancian por cada cámara.

Para disparar esta topología de servicio, una aplicación externa debe realizar la petición y suscribirse a la salida del operador de coincidencia de rostros para obtener el resultado. Al cambiar el alcance geográfico definido para la petición, la aplicación externa puede controlar ***FogFlow*** para orquestar la topología de servicio con un alcance geográfico cambiante. En nuestro caso de uso, la búsqueda comenzaría en un ámbito pequeño y se expandiría el radio de búsqueda progresivamente paso a paso si el niño no es localizado (Carpintero, 2019).

En este ejemplo la aplicación externa, que actúa como consumidor, resulta muy simple, ya que es ***FogFlow*** quien maneja toda la complejidad de cómo organizar dinámicamente los flujos de procesamiento de datos para un alcance geográfico variable.

De acuerdo a experimentos realizados, se consigue reducir el ancho de banda consumido en hasta un 95% respecto a una aproximación tradicional basada plenamente en *cloud* (Carpintero, 2019).

Estacionamiento inteligente

Se distinguen dos tipos de estacionamientos. Por un lado, zonas de estacionamiento regulado y que pueden proporcionar información histórica sobre cómo se

ISSN: 1813-5056, RNPS: 0514, Vol. 18, No. 1, enero - julio, 2022, pp. 46 - 57 fono

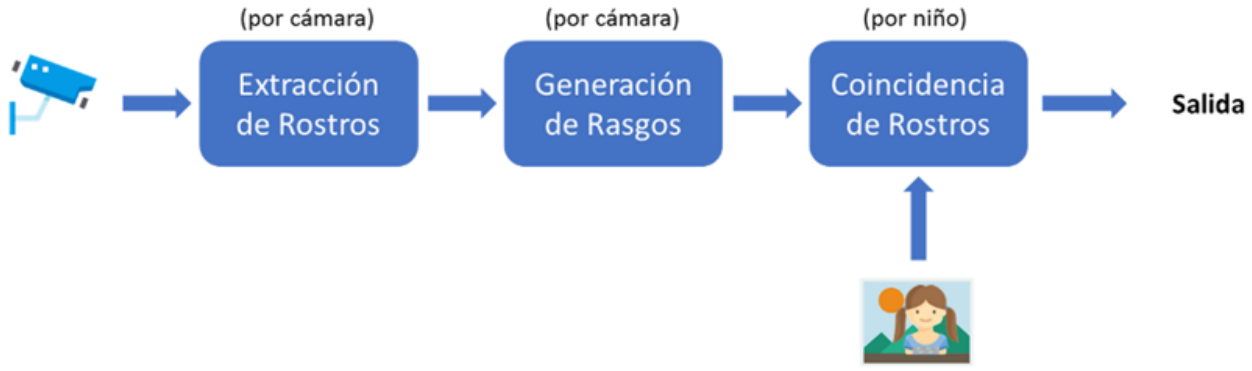

Figura 6. Caso de Uso de Buscador de niños perdidos

utilizan, y por otro lado parqueos privados, operados por empresas privadas, y que pueden proveer información en tiempo real sobre la disponibilidad de plazas. Utilizando estos dos tipos de información y otra información pública disponible de transporte, el servicio de estacionamiento inteligente planteado puede proveer información en tiempo real y recomendaciones de estacionamiento personalizadas para cada conductor (Carpintero, 2019).

En este caso, no resulta fácil aplicar una topología de servicio, pero haciendo uso de funciones *fog* la lógica de procesamiento de datos resulta sencilla. Como se observa en la figura 7, solo es necesario diseñar e implementar funciones *fog* dedicadas para cada objeto o variable involucrada en el caso de uso.

En este ejemplo se implementaría una función de niebla por cada zona de estacionamiento público regulada para realizar la predicción de cuantas plazas estarán disponibles en base a la información histórica de uso, así como otra para realizar las estimaciones de ocupación en los estacionamientos privados. Por su parte, los vehículos conectados por una parte aportan información sobre su ruta y hora estimada de llegada al destino, y por otra parte esperan recibir una recomendación personalizada sobre el mejor estacionamiento disponible a su llegada, por lo que por cada vehículo se contará con dos funciones de niebla (Carpintero, 2019).

Estas instancias se encuentran desplegadas en nodos *edge* cercanos al origen de cada uno de los datos de entrada, consiguiendo una reducción de ancho de banda considerable frente a enfoques tradicionales, lo que logra proveer información actualizada en tiempo real y personalizada para cada vehículo.

Conclusiones

El presente artículo pretende aportar una visión general de ***FogFlow*** como *framework* de desarrollo y ejecución distribuido para organizar servicios IoT mediante la gestión dinámica de los flujos de procesamiento de datos entre la nube y los bordes de una manera transparente y escalable, presentando sus objetivos de diseño, características tecnológicas clave y propuesta de valor, así como su arquitectura.

También se explica brevemente sus modelos de programación, incluida la topología de servicio para el procesamiento de datos bajo demanda y la función de niebla (*fog*) para la computación *edge* sin servidor (*serverless*).

Se presentan dos posibles casos de uso para mostrar cómo estos dos modelos de programación

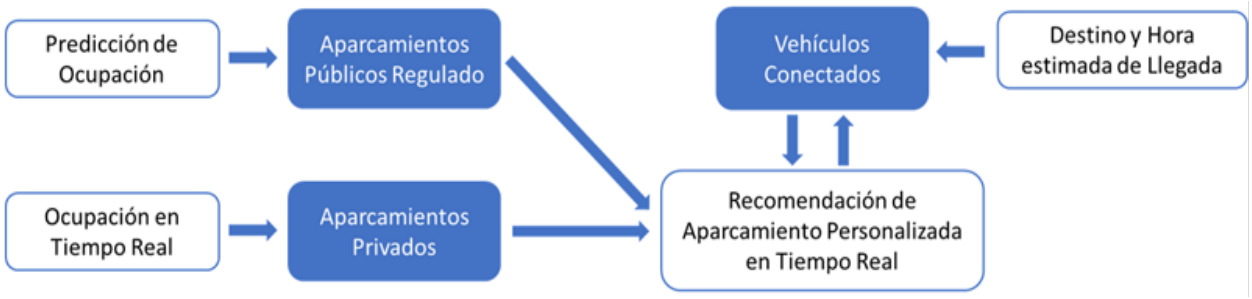

Figura 7. Caso de Uso de Estacionamiento Inteligente

pueden usarse para realizar servicios de IoT a escala de ciudad. Los dos casos de uso propuestos ayudan a ilustrar cómo los dos modelos de programación (topología de servicio y topología de niebla) pueden usarse para implementar servicios IoT reales, contribuir a la optimización de anchos de bandas y a simplificar toda la problemática y complejidad asociada a la gestión y diseño de este tipo de servicios.

De forma general se expone y muestra porqué es el *framework* mejor posicionado para desplegar un escenario de simulación para *Smart City* utilizando el paradigma de la computación en la niebla y el borde. Vale la pena destacar que en este apartado no se trató el tema de la dependencia tecnológica, aunque si se presentó su integración con *FIWARE*, hoy en día el *Docker* necesario para su ejecución no se encuentra libre para descarga.

Referencias bibliográficas

Calheiros R. N., y Ranjan, R. (2010). *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*. Wiley Online Library.

Carpintero Ordóñez, J. (2019). *Cambiando las nubes por niebla: Orquestando servicios IoT entre cloud y edge.* Smartcity. https://www.smartcity.es/comunicaciones/comunicacion-cambiando-nubes-por-niebla-orquestando-servicios-iot-entre-cloud-edge

Cheng, B., Solmaz, G., Cirillo, F., Kovacs, E. Terasawa, K. y Kitazawa, A. (2017). FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities. *IEEE Internet of Things Journal*, *5*(2), 696-707. https://doi.org/10.1109/JIOT.2017.2747214

Gámez Picó, L., Rollón Rius, Y. D., Anías Calderón, C. y Frómeta Fonseca, D. (2020). Procedimiento de simulación de redes de computación en la niebla. *Revista Científico Técnica Tono*, *16*(2), 4-16.

Hong Linh, T. (2021). *Using IoTCloudSamples as a Software Framework for Simulations of Edge Computing Scenarios.* https://doi.org/10.1016/j.iot.2021.100383

Mahmud R., y Buyya R. (2017). *Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. Fog and Edge Computing: Principles and Paradigms.*

Pérez Colomé, A. L., Anías Calderón, C. y Delgado Fernández, T. (2021). Procedimiento para la implementación de la computación en la niebla en ciudades inteligentes. *Ingeniería Electrónica, Automática y Comunicaciones*, *42*(1), 45-57.

Plataforma FIWARE FOGFLOW. (2021). *FogFlow v3.2.6 documentation*. FogFlow. https://fogflow.readthedocs.io/en/latest/