XV Jornadas de Ingeniería Telemática.
JITEL 2021.
Universidad de A Coruña.

Actas de las XV Jornadas de Ingeniería Telemática (JITEL 2021), A Coruña (España), 27-29 de octubre de 2021.

Arquitectura para redes IoT orientada a la sostenibilidad medioambiental

Jorge Navarro-Ortiz, Natalia Chinchilla-Romero, Félix Delgado-Ferro, Juan J. Ramos-Munoz
Departamento de Teoría de la Señal, Telemática y Comunicaciones,
Universidad de Granada
C/ Periodista Daniel Saucedo Aranda, s/n. ETSI Informática y de Telecomunicación.
jorgenavarro@ugr.es, nataliachr@ugr.es, felixdelgado@correo.ugr.es, jjramos@ugr.es

En este trabajo se presenta una arquitectura para redes IoT orientada a la sostenibilidad medioambiental. Debido a la adecuación de sus características en términos de cobertura, potencia y soporte de un gran número de dispositivos, se ha elegido una red LoRaWAN mejorada como base de la presente propuesta. La arquitectura se completa con la virtualización mediante contenedores de las diferentes entidades de red LoRaWAN y el uso de una red definida por software para su interconexión. La publicación y suscripción a los datos medioambientales se realiza mediante el protocolo MQTT, que ha sido optimizado gracias al uso de la red SDN y al uso de recursos de *edge computing*. La arquitectura propuesta ha sido implementada mediante una red prototipo a modo de prueba de concepto.

***Palabras Clave*- IoT, LoRaWAN, MQTT, SDN**

I. INTRODUCCIÓN

El objetivo del presente trabajo es el diseño de una arquitectura de red IoT (*Internet of Things*) para la recopilación, procesado y distribución de información medioambiental. Se pretende que sea lo suficientemente flexible y potente para poder integrar, en el futuro, las diferentes soluciones desarrolladas en este ámbito.

Esta red se compone de varias partes. Por un lado, una red de acceso radio de tipo LPWAN (*Low Power Wide Area Network*) adecuada para comunicaciones de sensores masivos. Se ha elegido una red de tipo LoRaWAN (*Long Range Wide Area Network*) [1] debido a sus características adecuadas de bajo consumo, alta cobertura, fácil escalabilidad y a que utiliza una banda de frecuencias sin licencia, lo que facilita su desarrollo y reduce costes.

Por otro lado, una red troncal que contará con las diferentes entidades necesarias para este tipo de redes y para el procesado y distribución de los datos. Al usarse una red LoRaWAN para la parte radio, la red troncal incluirá un servidor de red y un servidor de aplicación, elementos necesarios en este tipo de redes. Este tipo de servidores utiliza habitualmente el protocolo MQTT (*Message Queuing Telemetry Transport*) [2] para el intercambio de información con entidades tanto internas como externas. Por ello, como se explicará en las siguientes secciones, se introducirán elementos que permitirán reducir tanto el tráfico generado como la latencia en MQTT. También se incluirá una plataforma de Inteligencia Artificial (IA) para el procesado de los datos. Todas las entidades se implementarán como funciones de red virtuales (NVFs, *Network Virtualization Functions*) de forma que se facilite su orquestación, despliegue y ejecución en nubes locales.

Por último, se utilizará una red definida por software (SDN, *Software Defined Networking*) para la comunicación entre la red de acceso radio y la red troncal. Este tipo de redes proporciona una gran flexibilidad y facilidad de desarrollo para incluir e.g. nuevos protocolos u optimizaciones sobre los existentes.

Este artículo se organiza en las siguientes secciones. La sección actual introduce los objetivos del trabajo y su contexto. La Sección II realiza una revisión del Estado del Arte. La Sección III describe la arquitectura de red IoT diseñada, exponiéndose la red prototipo realizada a modo de prueba de concepto en la Sección IV. A continuación, la Sección V presenta algunos resultados preliminares, concluyendo el artículo en la Sección VI.

II. ESTADO DEL ARTE

En este apartado se hará un breve resumen de las distintas posibilidades para desplegar una red LoRaWAN. Otros aspectos como plataformas para IoT, más centradas en el almacenamiento, distribución, procesado y visualización de datos (e.g. Google Cloud Platform, IBM Watson IoT, Amazon AWS IoT Core, Microsoft Azure, entre otros) se quedan fuera del ámbito de este trabajo.

Como ejemplo de arquitectura de red LoRaWAN de gran tamaño, cabe destacar *The Things Networks*, una red colaborativa y abierta con más de 20.000 gateways en todo el mundo. Su arquitectura [3] se compone de *bridges* que interactúan con los *gateways*, conectados a *routers* que encaminan hacia los *brokers* correspondientes (dependientes de la ubicación geográfica). Estos

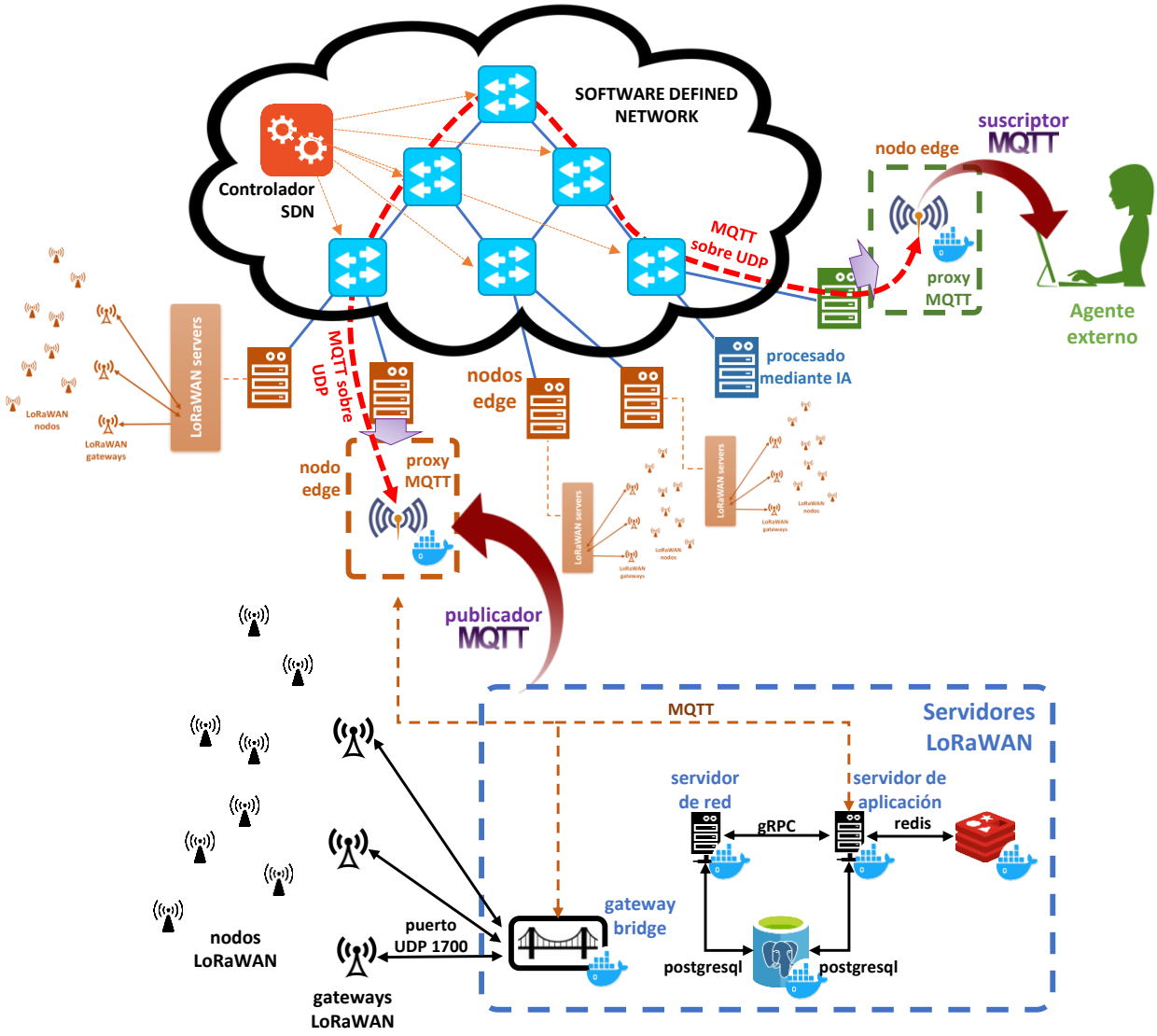

Fig. 1. Propuesta de arquitectura de red IoT para la recolección, procesado y uso de variables medioambientales.

interactúan con servidores de red y *handlers* que gestionan la comunicación con servidores de aplicación.

Respecto a plataformas LoRaWAN caben destacar Chirpstack [4], gratuita, de código abierto y que permite desplegar una red propia, y LORIOT [5], gratuita para redes pequeñas (hasta 30 nodos) pero sin posibilidad de montar una red privada propia.

III. Diseño de Arquitectura de Red IoT

La Fig. 1 muestra la arquitectura propuesta, que consiste en una red SDN que conecta *nodos edge*. Estos nodos implementarán la funcionalidad de *proxy* MQTT y/o la de procesamiento mediante IA. Estos *nodos edge* se conectarán con un nodo o *cluster* que implementará las diferentes entidades de una red LoRaWAN, es decir, el servidor de red, el servidor de aplicación y un *bridge* que permite la conexión entre el servidor de red y el *gateway*, así como las bases de datos necesarias. El *gateway* permite la conexión radio con las motas LoRaWAN, que serán los sensores y actuadores que envíen información medioambiental. En una red LoRaWAN típica también se incluiría un *broker* MQTT, que en este caso se ve reemplazado por el *proxy* MQTT.

Los *proxies* MQTT permiten la reducción del tráfico MQTT dentro de la red SDN, así como la disminución de su latencia. Para ello, actúan como *brokers* MQTT respecto a los clientes (implementando los mismos mecanismos de seguridad, e.g. usando MQTT sobre TLS/SSL), tanto suscriptores como publicadores, directamente conectados. En el caso de esta arquitectura, varias entidades de la red LoRaWAN lo utilizarán en sustitución del *broker*: *gateway bridge*, servidor de red y servidor de aplicación. Además, los datos publicados mediante MQTT podrán ser consultados por entidades externas ("*agente externo*" en la figura). Para ello, estos suscriptores MQTT se conectarán a su vez a otro *proxy* MQTT que actuará como su *broker* e intercambiará la información necesaria con el otro *proxy* usando UDP. La ventaja de usar UDP es que se reduce la latencia (varios *Round-Trip Times* (RTT) en caso de usar TCP) y nos permitirá en un futuro enviarlo por multidifusión (véanse los trabajos futuros en las conclusiones). Además, si bien no se ha incluido en la prueba de concepto, se puede utilizar la solución dada en [6] para garantizar la fiabilidad de UDP sobre SDN.

Concretamente, los *proxies* reenviarán los mensajes de suscripción (*Subscribe*) –cuando aparezca un nuevo tópico que no tuviese suscriptores previos en ese *proxy*–, publicación (*Publish*) –para todos los mensajes publicados en tópicos que tengan algún suscriptor en el *proxy* destino– y desconexión (*Disconnect*) –cuando se desconecte el último suscriptor a ese tópico en ese proxy–. De esta manera, los *proxies* tendrán la información necesaria para reenviarse los mensajes de tópicos activos.

IV. Prueba de Concepto

Como prueba de concepto, se ha implementado una red SDN utilizando *mininet* [7] siguiendo una topología en árbol con 3 *switches*, i.e. un *switch* raíz (*s1*) y dos *switches* hoja (*s2* y *s3*), que conectan 4 nodos *edge* (*h1* y *h2* conectados a *s2*, y *h3 y h4* conectados a *s3*). Los nodos *h1* y *h4* ejecutan sendos *proxies* MQTT que han sido programados utilizando *Scapy* [8]. El nodo *h1* utiliza un interfaz de red real del PC en el que se ejecuta, de manera que permite conectar directamente con la red troncal LoRaWAN. En este caso, por sencillez, se ha optado por utilizar contenedores, orquestados usando Kubernetes [9], que ejecutan las diferentes entidades de la plataforma Chirpstack (*gateway bridge* y servidores de red y aplicación LoRaWAN) en el propio *gateway* LoRaWAN, de forma similar a [10]. Dicho *gateway* es un Lite Gateway [11] de IMST, que utiliza una Raspberry Pi junto con un concentrador LoRaWAN iC880A. Se dispone de motas FiPy [12] con placas de expansión PySense. Como controlador SDN se ha utilizado RYU [13], que ejecuta un *learning switch* para OpenFlow v1.3 [14]. Los datos enviados por MQTT son almacenados en una base de datos InfluxDB por un *script* Python. Estos datos finalmente son visualizados utilizando para ello Grafana [15]. Como demostración de su funcionamiento, la Fig. 2 muestra las trazas de todos los equipos cuando *h2* se conecta a *h1* como suscriptor (usando `mosquitto_sub`) y *h4* manda un mensaje "*message1*" como publicador al tópico "topic1" (usando `mosquitto_pub`). Tal como se muestra en la traza *Wireshark*, tomada en *h1*, *h1* responde a la petición de suscripción de *h2* comportándose como haría un *broker*. Se pueden observar los mensajes MQTT enviados (*Connect*, *ConnAck*, *Subscribe*, *SubAck*). Concretamente, se puede ver un mensaje UDP después del mensaje *Subscribe Request*, que es el reenvío de dicho mensaje al otro *proxy* usando UDP en vez de TCP. Posteriormente, *h3* manda el mensaje MQTT *Publish Request* a *h4*, que lo reenvía a *h1* usando UDP (también visible en la traza *Wireshark*) y este lo publica de forma que lo recibe *h2*, que lo muestra en su consola.

V. Resultados Preliminares

A modo de resultados preliminares, la Fig. 3 muestra parte de los datos visualizados en Grafana. Se muestran, a modo de ejemplo, valores de temperatura, humedad y luz instantáneos y gráficas con los valores del último día. En los *dashboards* implementados también se visualizan gráficas de la última hora y con los valores mínimo, medio y máximo de estas métricas para el último día por horas, y para la última semana y mes por días.

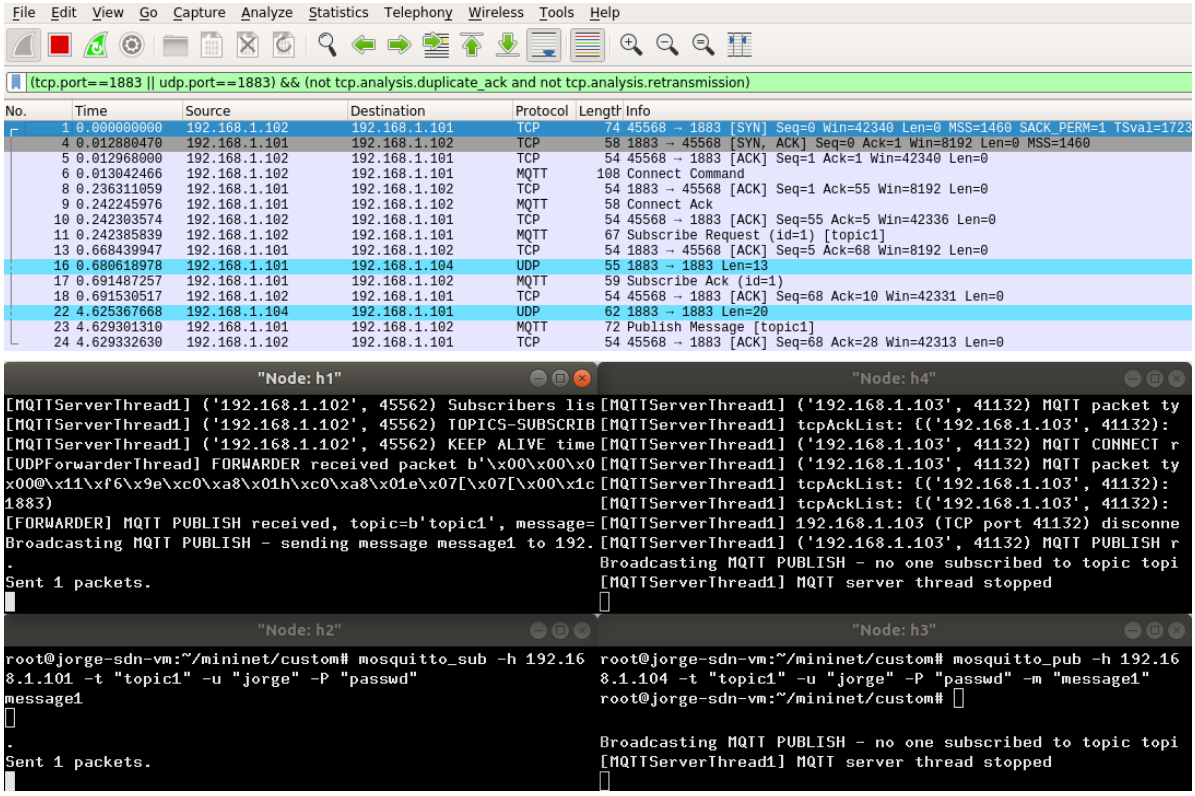

Fig. 2. Funcionamiento de proxies MQTT en los *edge switches*.

Fig. 3. Ejemplo de visualización de variables medioambientales.

También se ha comprobado la reducción de latencia y tráfico MQTT gracias al uso de UDP en la red SDN. Respecto al tráfico, una publicación MQTT normal requiere 12 mensajes (establecimiento de conexión TCP, Connect/ConnAck, Publish, Disconnect, finalización de conexión, ACKs de TCP), i.e. unos 550 bytes suponiendo cabeceras IP y TCP sin opciones y para tópicos y mensajes de tamaño pequeño (12 bytes entre ambos). Al utilizar nuestra solución, solo el mensaje *Publish* atravesaría la red SDN con unos 52 bytes más las longitudes del tópico y el mensaje (64 bytes en el ejemplo), lo que implica una reducción de tráfico de un 88%. Esta reducción en mensajes implica una menor latencia (desde que se publica el dato hasta que se recibe), efecto más apreciable para valores de RTT elevados, tal como muestra la Fig. 4. Con el objetivo de que la comparativa sea justa, el *broker* será una modificación de nuestro *proxy* realizado con Scapy. En media, la latencia se reduce aprox. un 80%.

VI. Conclusiones

En este artículo se ha presentado el diseño de una arquitectura de red IoT orientada a la recopilación, procesado y visualización de variables medioambientales. Además de realizar una prueba de concepto, el prototipo incluye una modificación de MQTT que divide la conexión TCP en partes, usando UDP para el intercambio de mensajes de MQTT entre *proxies* a través de la red SDN. Este prototipo ha permitido mostrar algunos resultados preliminares como la reducción de latencia y tráfico MQTT y la visualización de datos medioambientales.

Para mejorar la arquitectura de red propuesta, los autores prevén los siguientes trabajos:

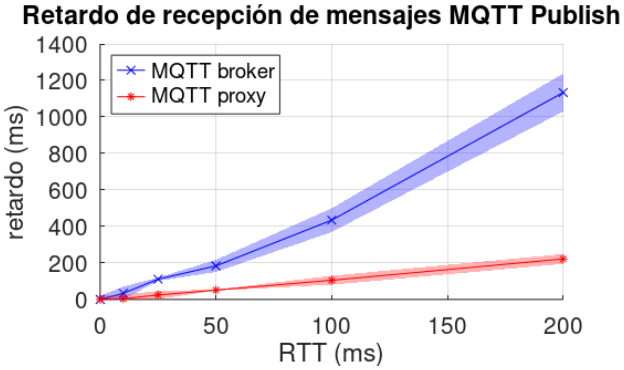

Fig. 4. Resultados de latencia y tráfico generado MQTT.

- Mejora de la capacidad de la red LoRaWAN. Este trabajo ya ha sido realizado, consiguiéndose aumentar la capacidad un 95% con condiciones radio ideales y un 40% con modelos de propagación realistas [16].
- Mejora de la tasa de datos de los dispositivos LoRaWAN. Esta mejora permitiría utilizar el mismo dispositivo para enviar fotografías y vídeo de baja resolución en tiempo real que podrían usarse para e.g. confirmar situaciones de alarma.
- Uso de una plataforma de Inteligencia Artificial o *Machine Learning* para el procesado de datos para, por ejemplo, predecir series temporales o eventos futuros.
- Dado el gran tamaño de una red IoT de este tipo, se podría optimizar el envío de tráfico MQTT utilizando un protocolo de multidifusión dentro de la red SDN.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Agencia Andaluza del Conocimiento (proyecto A-TIC-241-UGR18), el Ministerio de Economía y Competitividad (proyecto TEC2016-76795-C6-4-R) y el proyecto H2020 5G-CLARITY (Grant No. 871428).

Referencias

[1] LoRaWAN Specification v1.1, LoRa Alliance, 2017. Disponible en https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/ (visitado el 30/5/2021).

[2] MQTT v3.1 Protocol Specification, IBM and Eurotech, 2010. Dispoinble en https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html (visitado el 30/5/2021).

[3] The Things Network – Network Architecture. Disponible en https://www.thethingsnetwork.org/docs/network/architecture/ (visitada el 30/5/2021).

[4] Chirpstack, Open-Source LoRaWAN Network Server Stack. Disponible en https://www.chirpstack.io/ (visitada el 30/5/2021).

[5] LORIOT – Connecting the Internet of Things. Disponible en https://www.loriot.io/ (visitada el 30/5/2021).

[6] M. Wang, L. Chen, P. Chi and C. Lei, "SDUDP: A Reliable UDP-Based Transmission Protocol Over SDN," in IEEE Access, vol. 5, pp. 5904-5916, 2017, doi: 10.1109/ACCESS.2017.2693376.

[7] Mininet – An Instant Virtual Network on your Laptop. Disponible en http://mininet.org/ (visitado el 30/5/2021).

[8] Scapy – Packet Crafting for Python2 and Python3. Disponible en https://scapy.net/ (visitado el 30/5/2021).

[9] Kubernetes – Production-Grade Container Orchestration. Disponible en https://kubernetes.io/ (visitado el 30/5/2021).

[10] J. Navarro-Ortiz, J. J. Ramos-Munoz, J. M. Lopez-Soler, C. Cervello-Pastor, M. Catalan, "A LoRaWAN Testbed Design for Supporting Critical Situations: Prototype and Evaluation", Wireless Communications and Mobile Computing, vol. 2019, DOI: 10.1155/2019/1684906

[11] Lite Gateway – Demonstration Platform for LoRa Technology, IMST. Disponible en https://wireless-solutions.de/products/lora-solutions-by-imst/development-tools/lite-gateway/ (visitado el 30/5/2021).

[12] FiPy Development Board, Pycom. Disponible en https://pycom.io/product/fipy/ (visitado el 30/5/2021).

[13] RYU – Component-Based Software Defined Networking Framework. Disponible en https://ryu-sdn.org/ (visitado el 30/5/2021).

[14] OpenFlow Switch Specification, Version 1.3.0. Open Networking Foundation, 2014. Disponible en https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf (visitado el 30/5/2021).

[15] Grafana – Your Observability Wherever You Need It. GrafanaLabs. Disponible en https://grafana.com/ (visitado el 30/5/2021).

[16] Natalia Chinchilla-Romero, Jorge Navarro-Ortiz, Pablo Muñoz, Pablo Ameigeiras, "Collision Avoidance Resource Allocation for LoRaWAN", Sensors, 21 (4), 2021. DOI: 10.3390/s21041218.