UNIVERSIDAD NACIONAL DEL SUR

Tesis Doctor en Ingeniería

Internet de las Cosas, Redes Oportunistas y Sistemas Distribuidos

Jose Mariano Finochietto

BAHIA BLANCA ARGENTINA

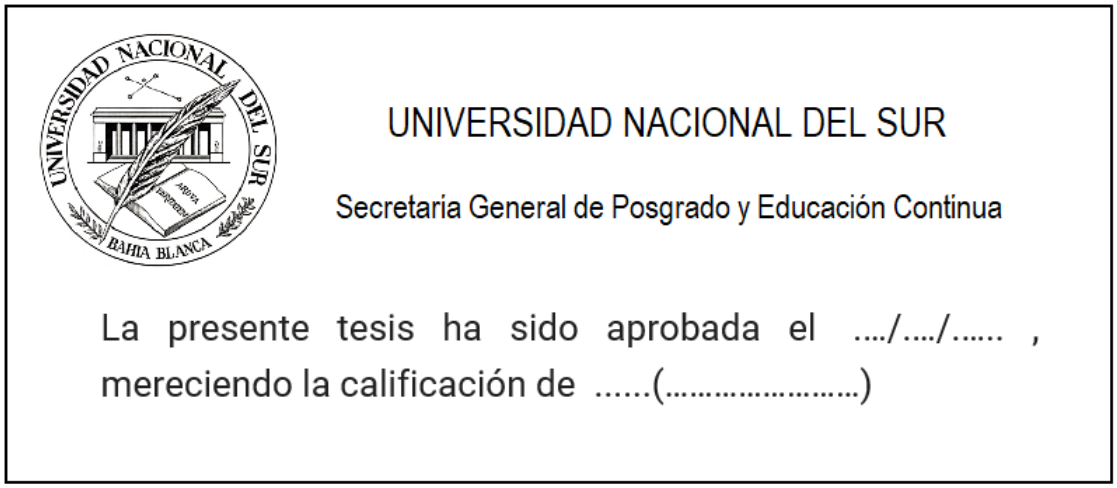
2021

Prefacio

Esta tesis se presenta como parte de los requisitos para optar al grado Académico de Doctor en Ingeniería, de la Universidad Nacional del Sur y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en Laboratorio de Sistemas Digitales dependiente del Departamento de Ingeniería Eléctrica y de Computadoras entre el 22 de Agosto de 2017 y el 21 de Diciembre de 2021, bajo la dirección del profesor Dr. Rodrigo M. Santos y la co-dirección del Dr. Jérémie Bellec.

_____

Mg. Ing. Jose Mariano Finochietto



Agradecimientos

A la Universidad Nacional del Sur y al Departamento de Ingeniería Eléctrica y de Computadoras, por haberme acompañado y estado a disposición en cada paso de la tesis.

A la Universidad Nacional de Mar del Plata, especialmente a los docentes y directivos del Departamento de Informática de la Facultad de Ingenería, por haberme alentado desde el primer momento a realizar este doctorado y por brindarme los medios que me permitieron desenvolverme con comodidad durante este periodo doctoral.

A mi director, el Dr. Rodrigo Santos, que sin conocerme previamente y yo viniendo de otra casa de estudios y otra ciudad, me trató como uno más, brindandome toda su confianza y dedicandome mucho tiempo, siempre con paciencia y un trato afectuoso.

Finalmente, a toda mi familia. De manera muy especial a mis padres, que desde chico me inculcaron la importancia de la formación y la superación personal. Y a mi mujer, con la que tuvimos 2 hijos durante este período, y siempre me apoyó en este logro que es parte de nuestro proyecto de familia.

Resumen

Los sistemas distribuidos han evolucionado a lo largo de los años a través de distintas vertientes. Dentro de ellas, uno de los paradigmas que más se desarrolló en el último tiempo es el Internet de las Cosas, que puede definirse como la conexión a Internet de objetos físicos, con el fin de interactuar con los mismos de manera remota e integrarlos a otros procesos informáticos. Son varios y multidisciplinarios los desafíos que esto implica: desde la gestión de energía de los dispositivos, hasta la lectura de los datos por parte de usuarios finales. Dentro del amplio rango de objetos de estudio que involucra este paradigma, esta tesis intenta contribuir con soluciones concretas a problemas específicos, haciendo énfasis principalmente en cuestiones relacionadas con las capas lógicas de las redes de comunicación.

En los diferentes capítulos que componen este trabajo, se presentan, entre otras cosas, arquitecturas, diseños, modelos formales, extensiones de protocolos de comunicación existentes e implementaciones en escenarios reales. Todas las propuestas tienen como principal motivación aportar al desarrollo del Internet de las Cosas en casos de uso reales que buscan el bienestar social.

Abstract

Distributed systems have evolved over the years in different ways. Among them, one of the most developed paradigms in recent times is the Internet of Things, which can be defined as connecting physical objects to the Internet, in order to interact with them remotely and integrate them with other computer processes. The challenges that this implies are several and multidisciplinary: from energy management, to data consumption by end users. Within the wide range of study subjects involved in this paradigm, this thesis attempts to contribute with concrete solutions to specific problems, focusing mainly on issues related to the logical layers that compose communication networks.

The different chapters that compose this work present architectures, designs, formal models, extensions of existing communication protocols and implementations in real scenarios, among other things. The main motivation of all these proposals is to contribute to the development of the Internet of Things in real use cases that aim social wellfare.

Índice general

Parte I

Introducción

Capítulo 1

Marco de referencia

1.1. Motivación

La cantidad de objetos conectados a Internet ha estado creciendo sustancialmente en los últimos años. Estudios presentados en Noviembre de 2020, graficados en la Figura 1.1, estimaron que, en el mundo, alrededor de 21,7 mil millones de objetos enviaron y/o recibieron información a través de internet, lo que implica un aumento del 8.5 % con respecto al año anterior [1]. De esa cantidad 9,9 mil millones fueron teléfonos móviles, laptops o computadoras, mientras que 11,7 mil millones fueron otros dispositivos, como autos, electrodomésticos y maquinaria industrial. Esta tendencia ha dado lugar a que se defina el término "Internet de las Cosas" - IoT (Internet of Things) - como un nuevo paradigma en el que se embeben capacidades computacionales y de comunicación en cualquier objeto concebible [2].

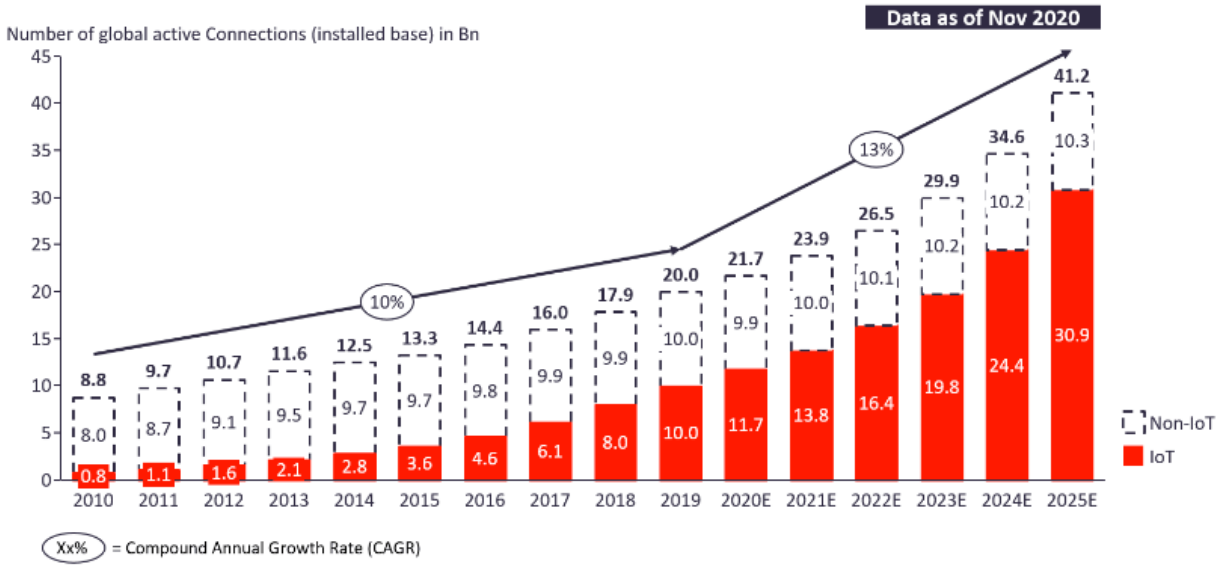


Figura 1.1: Cantidad de objetos conectados a internet a lo largo de los años.

IoT encontró en las redes de sensores inalámbricas (WSN, por sus siglas en inglés), una base importante teórica y práctica para su desarrollo [3]. Estas redes buscaban diferentes

propósitos asociados al monitoreo ambiental, la vigilancia y aplicaciones militares entre otras. Basadas en microprocesadores y sistemas embebidos de bajo consumo, los nodos permitían medir distintas variables y transmitir la información. Sin embargo, la comunicación utilizada no contaba con suficiente alcance ni escalabilidad para que pueda implementarse en grandes áreas. Las tecnologías actuales de comunicación han superado muchas de las limitaciones que tenían éstas propuestas como el rango de cobertura de las comunicaciones y el consumo de energía. Esto permite que IoT crezca mucho más allá que las redes de sensores, que si bien siguen siendo un importante campo de desarrollo, no es el único.

En efecto, IoT se encuentra en la salud, la educación, el gobierno, la industria, el transporte y las ciudades en general, por mencionar algunas áreas. Las posibilidades de intercambiar datos, procesarlos en la nube con una capacidad escalable y accesible, integrarlos con otras fuentes, y generar información que será consumida a través de distintos canales y en casi cualquier lugar del mundo, dando lugar a nuevos conocimientos, convierte a la IoT en un paradigma que no encuentra prácticamente límites para su crecimiento. Todo esto se basa en redes de comunicaciones que cada día alcanzan mayor distancia con más autonomía y confiabilidad, integrando redes públicas, privadas y telefónicas. Por esta razón fundamental, la generación de nuevos protocolos o la adaptación de los ya existentes, pensados para un red en principio estática y de un número mucho menor de nodos, constituyen por cierto un área de investigación en constante desarrollo.

Dentro del gran grupo de tecnologías de comunicación asociadas a esta nueva rama de la computación, se encuentran protocolos que abarcan desde las capas inferiores del modelo OSI como la física, hasta las más superiores como la de aplicación. Algunos fueron creados incluso antes de que se bautice con el término "IoT", y su utilización fue adaptada para actualizarse a las nuevas necesidades. Tales son los casos de MQTT y HTTP. El primero, es un protocolo de publicación-suscripción, donde los nodos están desacoplados al haber un agente que actúa de intermediario en el paso de mensajes. Fue creado en 1999 con el objetivo de estandarizar la comunicación en las aplicaciones de telemetría. El segundo, es un protocolo desarrollado originalmente para la web, y si bien popularmente se utiliza allí, con el tiempo fue ganando espacio para ser una alternativa de intercambio de datos en IoT. Por otro lado, hay protocolos que fueron diseñados específicamente para satisfacer los requerimientos que aplicaciones típicas de IoT pueden tener. Dos ejemplos son CoAP y LoRaWAN. Mientras el primero está basado en la arquitectura REST utilizada comúnmente en HTTP, pero con paquetes mucho más comprimidos y funcionales a este nuevo paradigma, el segundo busca ser una solución de comunicaciones integral para redes de largo alcance.

Son numerosos los desafíos aún en discusión relacionados a esta nueva realidad, que incluye una diversidad de protocolos y tecnologías, y cada vez más objetos conectados. Algunos de los identificados son la calidad de servicio - QoS (Quality of Service) - y la escalabilidad [4]. El primero trata de brindar la capacidad de controlar el tráfico para ofrecer distintos tipos de calidad de conexión a los usuarios. En IoT, debido a limitaciones de ancho de banda en algunas regiones u otras limitaciones de conectividad, asegurar QoS es un problema que sigue abierto. Por otro lado, al ser cada vez mayor el volumen de información enviada y almacenada, cuestiones como la escalabilidad toman relevancia ante las limitaciones físicas que la red impone. En muchos casos, se vuelve necesario implementar mecanismos que permitan filtrar y extraer los datos más relevantes, tanto en las bases de datos como en las redes de comunicaciones.

A medida que crece esta tendencia, y el internet invade nuevos espacios y objetos cotidianos, estos desafíos se vuelven aún más evidentes. El poder resolverlos permitirá, entonces, continuar con la integración de las computadoras en el entorno de los humanos, concepto conocido como computación ubicua y estrechamente ligado a IoT. Para que las interacciones entre máquina-máquina y máquina-humano sean cada vez más frecuentes y menos notorias, se necesita seguir desarrollando las tecnologías de comunicación que subyacen dichas interacciones.

Como ocurrió a fines del siglo XX y comienzos del XI con Internet, IoT es un paradigma que cambia el concepto de la comunicación y la interacción entre la tecnología y las personas. La ubicuidad entendida como aquella tecnología disponible en todo lugar y en todo momento y la "omnipresencia" como aquella que enmascara su existencia al punto de dejar de notarla como tal, llevan a IoT a un lugar desde el cual los seres humanos quedan inmersos en el ciclo de comunicación como un eslabón más. Son productores, consumidores, interfaces hacia otras personas y/o dispositivos, una cadena en la cual cada eslabón indistintamente sea un dispositivo o un ser vivo, se transforma en necesario para la transferencia de la información. Este sistema gigante interconectado de diferentes maneras y por distintos caminos es el soporte sobre el que infinidad de aplicaciones se podrán construir y desarrollar. Estas aplicaciones podrán incidir como se describe más adelante en esta tesis, sobre distintos aspectos que abarcan desde la organización social en ciudades (educación, salud, gobierno, tráfico, ambiente), producción agrícola o industrial, manejo de situaciones críticas en regiones de riesgo como aquellas sometidas a desastres naturales, entre otras.

1.2. Organización de la tesis

Este trabajo de tesis busca proponer redes que se adapten a distintas necesidades, requerimientos y oportunidades, partiendo de protocolos existentes utilizados en Internet de las Cosas. Para ello, se tuvieron en cuenta distintos escenarios, como la evacuación de una ciudad ante un desastre natural y redes de sensores en lugares remotos, como puede ocurrir en el sector agrícola.

Se organiza del siguiente modo. El trabajo está divido en Partes. Cada Parte está compuesta por Capítulos que están relacionados bajo una misma idea. En cada Capítulo se presenta un trabajo, su desarrollo y sus conclusiones. Al final, se encuentra una conclusión general de la tesis, que integra y relaciona los distintos Capítulos y Partes. La Parte I introduce la tesis, su motivación y los fundamentos tecnológicos y sociales que sirven como contexto de la misma. La Parte II contiene dos capitulos que se centran en aspectos relacionados al diseño e implementación de sistemas IoT. Por un lado, se describe la arquitectura de un sistema de monitoreo de riego agrícola, que fue desarrollado y probado en maquinaria real. Por otro lado, se propone un diseño de una infraestructura para un sistema de auto-evacuación, que fue evaluado formalmente y mediante experimentos realizados sobre una implementación prototipo. La Parte III, contiene otros dos capítulos que describen problemáticas encontradas en la utilización de protocolos IoT y propone cómo resolverlas. El primer capitulo presenta una extensión de MQTT para agregarle soporte de comunicación en tiempo real suave. El segundo describe una modificación para que gateways LoRaWAN puedan modelar el tráfico basandose en distintas políticas. Finalmente, la Parte IV aborda

las conclusiones y el trabajo futuro.

1.3. Contribuciones

Durante el trabajo doctoral se realizaron publicaciones científicas que enmarcan la propuesta de esta tesis. Cada una de ellas hace foco en alguna problemática de IoT. A continuación, se presenta una lista de las mismas.

1.3.1. Artículos de revista de alcance internacional

REV01 Finochietto, M., Eggly, G. M., Santos, R., Orozco, J., Ochoa, S. F., & Meseguer, R. (2019). A Role-Based Software Architecture to Support Mobile Service Computing in IoT Scenarios. Sensors, 19(21), 4801.

REV02 Finochietto, J. M., Micheletto, M., Eggly, G. M., Pueyo Centelles, R., Santos, R., Ochoa, S. F., Meseguer, R., & Orozco, J. (2021). An IoT-based infrastructure to enhance self-evacuations in natural hazardous events. Personal and Ubiquitous Computing.

REV03 Finochietto, M., Santos, R., Ochoa, S. F., & Meseguer, R. Dealing with the Backhaul Traffic in IoRT Monitoring Systems using LoRaWAN. En revisión.

1.3.2. Artículos en conferencia

CONF01 Eggly, G., Finochietto, M., Dimogerontakis, E., Santos, R., Orozco, J., Meseguer, R. (2018). Real-Time Primitives for CoAP: Extending the Use of IoT for Time Constraint Applications for Social Good. In UCAmI 2018. The International Conference on Ubiquitous Computing and Ambient Intelligence. MDPI.

CONF02 Eggly, G. M., Finochietto, J. M., Micheletto, M., Centelles, R. P., Santos, R., Ochoa, S. F., Meseguer, R., Orozco, J. (2019). Evacuation Supporting System Based on IoT Components †. In 13th International Conference on Ubiquitous Computing and Ambient Intelligence UCAmI 2019. UCAml 2019. MDPI

CONF03 Albisu, I., Araneta, J. P., Cisneros, F., Finochietto, J. M., Garin, J. M., & Robetto, J. (2019). Sistema de monitoreo remoto versátil para riego por pivote central. In I Taller Argentino de Internet de las Cosas (TAIC 2019)-JAIIO 48 (Salta).

CONF04 Eggly, G. M., Finochietto, J. M., Micheletto, M., & Santos, R. (2019). Internet de las cosas como bien social. In I Taller Argentino de Internet de las Cosas (TAIC 2019)-JAIIO 48 (Salta).

1.3.3. Resumen de detalles

En [CONF01] se presenta una extensión del protocolo petición-respuesta CoAP para que este soporte comunicación en tiempo real suave. Se describe el modelo general con una

máquina de estados y las modificaciones de diseño necesarias. Como ejemplo de aplicación, se propone una que persigue fines sociales, ya que busca eficientizar el transporte en combis, popularmente utilizado en Perú. Mediante el uso del protocolo modificado, los usuarios pueden conocer qué combi les conviene tomar para llegar a su destino más rápidamente. Con la misma idea de agregar capacidad en tiempo real suave, se llevo a cabo el trabajo de [REV01], pero tomando como base un protocolo publicación-suscripción. Se describió el modelo nuevamente con una máquina de estados, y se realizó un diseño más detallado para MQTT. Además, se implementaron todos los componentes necesarios en código abierto, y se publicó en la web un simulador para que dicha extensión pueda ser probada. Como caso de uso, se describió un sistema de agricultura de precisión que registra las distintas variables del suelo y el clima en tiempo real. Otro trabajo relacionado a la agricultura es [CONF03]. En el mismo, se describe la arquitectura de un sistema IoT para monitorear sistemas de riego por pivot central. Asimismo, una implementación se llevó a cabo y se evaluó su funcionamiento en campo. Este tipo de aplicaciones, como las presentadas hasta el momento con fines sociales y ambientales, representan la idea de [CONF04]. En este trabajo, se introduce el paradigma de Internet de las Cosas como una herramienta de bien social, que busca mejorar la calidad de vida de la sociedad y el medio ambiente. Se describen brevemente sus tecnologías, sus desafíos y algunas aplicaciones.

En los trabajos [CONF02] y [REV02] se presenta un modelo de auto-evacuación masiva para ciudades donde ocurren desastres naturales. El contexto de estos eventos implica que la comunicación puede ser inestable y la infraestructura en general puede verse afectada (transporte, caminos, electricidad, etc). Por lo tanto, se describe un sistema de IoT donde los mismos ciudadanos pueden evacuar, dependiendo lo menos posible de terceros. Particularmente en [REV02], se hace foco en las interacciones entre los componentes de este sistema, incluyendo los usuarios humanos. Además, el modelo descrito es evaluado formalmente mediante un dervidado de calculo de procesos, y se realizan pruebas empíricas en una implementación a nivel prototipo.

Finalmente, en [REV03] se describe un modelo de gateway para LoRaWAN que tiene como objetivo disminuir el tráfico de la red de backhaul del sistema. En las aplicaciones IoT instaladas lejos de centros urbanos, en lugares donde no se cuenta con una infraestructura de comunicación estable, se debe contemplar que las redes alternativas son caras o con un ancho de banda limitado. Es necesario entonces filtrar el tráfico a nivel gateway para disminuir el throughput de estas redes. Se presentan en este trabajo 4 políticas de modelado de tráfico, y se desarrollo un simulador código abierto basado en NS3 para llevar a cabo una serie de simulaciones que buscan validar la propuesta.

Capítulo 2

Fundamentos de IoT desde una perspectiva social

2.1. Motivación

IoT se instaló con fuerza a partir del surgimiento de la versión 6 del protocolo de Internet (IPv6). La enorme disponibilidad de direcciones IP, (2^{128}), permitió imaginar un mundo en el que todas las cosas pudieran ser enumeradas y eventualmente identificadas por medio de ese número. Así descrito, IoT serviría para poder seguir a una vaca desde su nacimiento hasta que es vendida en la góndola de un supermercado trazando toda su vida. Sin embargo, en este caso, la vaca identificada no tendría la inteligencia y no brindaría más información que la que se pudo recolectar en su camino. Si el sistema fuera inteligente, se podría interactuar con ella para determinar otra serie de información. El concepto de IoT fue cambiando con el transcurrir de los años a medida que se acrecentaba la disponibilidad de comunicación y la capacidad de los sensores para procesar datos y brindar información.

La disponibilidad de datos provenientes de múltiples tipos de sensores requiere de dos etapas adicionales imprescindibles para que puedan ser usados. La primera etapa consiste en darle contexto al dato para que pueda ser evaluado como información. Para esto es importante conocer la localización del sensor, la calidad del mismo, el instante en que el dato fue obtenido, entre otras. Luego esa información se transforma en conocimiento al ser correlacionada con otra información. Por ejemplo, un sensor de temperatura brinda un dato 20 grados centígrados, este dato en sí carece de un valor práctico si no se indica dónde fue adquirido, en qué momento y cómo. Supongamos que el dato proviene del exterior de una base antártica en el mes de junio. Se tiene la información asociada al dato pero el conocimiento dice que ese dato no es válido porque en el mes de junio no puede haber esa temperatura en la Antártida.

Hoy en día las ciudades cuentan con diversos tipos de sensores que permiten monitorear distintos aspectos de la vida urbana. Cámaras de seguridad, identificadores de patentes de autos, sistemas de transporte con medio único de pago, sistemas de parquímetros, seguimiento de unidades de transporte público como taxis, colectivos, trenes, subterráneos. Es común que al ingresar por una autopista se indique la demora estimada para llegar a diferentes lugares y el estado del tráfico. Sin embargo, estos sistemas actúan de manera independiente

y no hay un mecanismo integral que permita detectar inconvenientes y modificar las condiciones para disminuir los tiempos necesarios para trasladarse por la ciudad. Aplicaciones como Waze o el navegador de Google utilizan la información que obtienen de los dispositivos móviles para determinar las rutas más rápidas de manera dinámica. Si además estos datos se fusionaran con otros provenientes de otras fuentes sería posible por ejemplo modificar los tiempos de los semáforos o incluso el modo en que se habilita la circulación en cruces de avenidas de doble mano y sendas peatonales. La frecuencia y eficacia del transporte público también podrían ser ajustados de manera inteligente en función de las variaciones constantes que se manifiestan.

Si bien el tránsito es un problema presente en todas las ciudades en la actualidad, hay otros inconvenientes de órden público que son también importantes como la recolección de la basura, el tratamiento de efluentes urbanos, sistemas inteligentes de iluminación, atención primaria de salud, seguridad, etc. En cada uno de los casos mencionados, la disponibilidad de información en tiempo real convenientemente procesada puede brindar beneficios importantes.

2.2. IoT como herramienta de mejora en la calidad de vida

Con el inicio del siglo XXI y la explosión de la Web como un medio de comunicación, negocios, publicidad y relaciones sociales, Internet se transformó en un aglutinador de cuestiones bien diversas. Con la incorporación de IPv6 las limitaciones en el número de dispositivos identificables prácticamente desaparecieron y es posible entonces hablar de IoT sin mayores inconvenientes.

En ciudades grandes, las personas parten de sus hogares temprano para regresar entrada la tarde. El tiempo de viaje es considerable y muchas veces una jornada de 12 horas o más tiene incluidas de 3 a 5 horas de transporte diario por diferentes medios. Poder disminuir estos tiempos constituye un incremento en la calidad de vida de miles de habitantes. Para lograr mejorar estos problemas se requiere avanzar en varios aspectos. El principal es el transporte público. Contar con el adecuado número de unidades en los horarios de mayor demanda y con recorridos que permitan optimizar el viaje de los pasajeros. En segundo lugar se debe brindar la infraestructura necesaria para que se pueda garantizar la circulación del transporte público libre de congestiones. Para esto se introducen zonas exclusivas para circulación de colectivos, se eliminan cruces a nivel entre autos y trenes quitando barreras entre otras acciones. A pesar de esto, en ciudades grandes del mundo desarrollado donde se han implementado las mejoras de infraestructura y de transporte público, las congestiones continúan. IoT tiene la posibilidad de mejorar considerablemente esto. Cada unidad de transporte, vagón de tren o subterráneo, colectivo, taxi o remise tendría un nodo que indique en todo momento la posición actual, hacia donde se dirige y la cantidad de pasajeros que lleva. Las cámaras de seguridad ya dispuestas o en caso de ser necesario se deben agregar nuevas cámaras monitorean el flujo de autos y colectivos en las diferentes calles. Finalmente, la información proveniente de los propios usuarios que pueden con sus dispositivos móviles indicar también su posición y velocidad de desplazamiento. Todos estos elementos se utilizan para decidir

la frecuencia de los semáforos y priorizar el flujo en las arterias con mayor congestión. El problema del tránsito es complejo y debe ser analizado con algoritmos de ruteo evitando transformar una calle sin congestión en una congestionada al derivarle un caudal de autos importante.

El estacionamiento es otro aspecto que influye en la vida de las ciudades. Habitualmente restringido en las zonas céntricas, se desarrollan lugares para dejar los automóviles. Aquí nuevamente IoT puede brindar una herramienta para ayudar a los conductores a estacionar sin mayores problemas. Para esto los sitios de estacionamiento deben estar sensorizados, sea por cámaras o por detectores de ocupación, de modo de poder contabilizar los espacios libres. Esta información debe ser puesta en línea de manera que el conductor pueda saber hacia qué lugar dirigirse. Debido a que muchos buscan los mismos lugares, se debe considerar nuevamente la densidad de autos en una determinada zona o dirigiéndose a la misma para computar los espacios disponibles. De este modo un automovilista que inicia su viaje con 30 minutos de anticipación al llegar a la zona de estacionamiento encontrá una situación diferente a la presente al partir. Por este motivo el algoritmo de asignación de estacionamiento debe funcionar con características de tiempo real.

IoT puede impactar además en el sistema de salud. Las aplicaciones de e-health son comunes en la actualidad y varían su complejidad de acuerdo al nivel de desarrollo. Desde una historia clínica digital como elemento más sencillo a tratamientos ambulatorios con monitoreo remoto desde el centro de salud, hay una gran cantidad de aplicaciones que pueden mejorar la calidad de vida de muchos pacientes y personas con un bajo costo. La utilización de sensores de caídas en adultos mayores como de apneas en niños menores a un año de vida, son algunas de las posibilidades con las que ya se cuenta en la actualidad. Sin embargo es posible también pensar en detectores de accidentes viales para la pronta atención de los accidentados, y el seguimientos de pacientes cardíacos y de personas con enfermedades neurológicas como Alzheimer o Parkinson, entre otras. Enmarcado en un contexto más amplio, con una adecuada sensorización del hogar, un adulto mayor podría ser monitoreado en sus variables vitales en tiempo real y recurrir a la asistencia médica cuando las condiciones fueran desfavorables. Esto incluye además la administración de la climatización de los ambientes, la asistencia lumínica, el control del gas y de la electricidad. Los protocolos de comunicación para los sensores en este caso se podrían conectar utilizando redes pensadas para distancias cortas como ZigBee o Bluetooth [5][6]. Los adultos mayores sufren con los cambios de sus lugares habituales, por este motivo las internaciones en hogares geriátricos o, peor las internaciones hospitalarias, son muchas veces causas de depresiones y de agravamiento de los cuadros. Sin embargo, muchas veces por su cuadro clínico están incapacitados para vivir solos sin asistencia. Este tipo de sistemas contribuyen a prolongar la calidad de vida manteniendo a los mismos dentro del espacio que les es habitual.

El campo educativo es otra área en la cual el IoT puede mejorar la calidad de vida. Si bien es cierto que en este caso hay ya muchas aplicaciones relacionadas en Internet por medio de las cuales los alumnos pueden estudiar siguiendo cursos virtuales y evitando tener que transportarse hasta los lugares en los que se dictan las clases, también es cierto que se pueden desarrollar nuevas herramientas y actividades en relación a esto utilizando para dispositivos móviles, cámaras de video, cuadernos digitales, libros digitales, etc.

Los servicios públicos como gas, agua y electricidad pueden ser monitoreados y administrados digitalmente también por medio de medidores inteligentes que permitan detectar fallas

cuando las hubiera y ayuden a controlar el consumo mejorando el rendimiento energético y por lo tanto globalmente el consumo masivo.

En la descripción anterior nos referimos a aplicaciones que mejoran la calidad de vida de las personas en zonas urbanas y en particular con una alta densidad poblacional. Sin embargo, podemos extender el modelo a la zona rural. En este caso las aplicaciones no tendrían que ver con los sitios de estacionamiento, las congestiones de tránsito o cualquiera de los problemas urbanos mencionados. Por ejemplo, los factores meteorológicos son importantes debido a que los caminos son, por lo general, de tierra y cuando llueve se anegan, las crecidas de río pueden causar daños en animales y cultivos, la utilización de sistemas de apoyo productivo para la agricultura y ganadería de precisión aseguran procesos de riego controladom, entre otros. Hay un sinnúmero de oportunidades además en documentación y trámites administrativos a nivel nación, provincia y municipio. Los automóviles podrían estar identificados digitalmente y la propiedad de los mismos evitando todos los trámites burocráticos asociados. La forma más básica de IoT está basada en la identificación de las cosas y en este campo podrían encontrarse también los documentos personales, laborales, etc.

2.3. Protocolos

En IoT es usual que la comunicación sea entre máquinas (M2M). En muchos casos, los dispositivos que conforman los nodos tienen baja capacidad de cómputo y memoria por lo que implementan protocolos que no les resultan complejos. En la capa de transporte se utilizan tanto UDP [7] como TCP [8] y por sobre ellos se usan tanto MQTT [9] como CoAP [10]. El primero de ellos utiliza un paradigma publicador/suscriptor sobre TCP y el segundo una petición/respuesta sobre UDP. En el caso de nodos más sencillos, CoAP aparece como una opción más natural dado que el conjunto UDP/CoAP requiere de menos recursos computacionales que TCP/MQTT.

El mayor desafío en cualquiera de los dos casos es lograr que los mensajes lleguen a destino con la menor demora posible pero además que los datos que transportan puedan ser interpretados por diferentes aplicaciones en un modo transparente a fin de poder obtener el mayor provecho de los mismos. Este punto es muy importante dado que las fuentes de información se multiplican rápidamente con el avance de IoT. Es fácil encontrar nuevas estaciones meteorológicas, cámaras de seguridad, lectores de patentes, sensores de estacionamiento, entre las más comunes. Dada la naturaleza de mejor esfuerzo presente en Internet, es necesario para dar coherencia temporal a los datos incorporar una marca de tiempo. Con la misma se puede computar la demora con que la información fue obtenida y descartarla cuando una más actualizada se obtenga.

En la capa de enlace y física hay una variedad muy amplia de posibilidades para conectar los eventuales nodos. Entre las más usadas están IEEE 802.11 [11] y sus derivados cuando es inalámbrica e IEEE 802.3 cuando es cableada. Estas dos constituyen las normas básicas para redes locales en la actualidad pero hay otras que permiten la conexión a distancia con bajo costo como son los casos de LoRA/LoRaWAN y SigFox [12] [13] [14] [15]. En zonas urbanas cada vez es mayor la presencia de las redes 4G. A diferencia de todas las anteriores, estas tienen un costo directo sobre cada sensor que se incorpora a la red.

En zonas rurales en donde es difícil conseguir acceso a las redes de datos, la utilización

de LoRa/LoRaWAN es una alternativa de bajo costo que permite transmitir mensajes a distancias grandes con una baja densidad de nodos intermedios. La implementación de sistemas de apoyo al mejoramiento en la calidad de vida podría entonces extenderse también a las zonas rurales con baja densidad poblacional.

De igual modo, poder acceder de manera simple a centros de atención primaria sin largas esperas brindan a los ciudadanos una mejora que se manifiesta claramente. Todos estos son solamente algunos puntos en los cuales se observa que IoT en un correcto uso de los datos sin violación de la privacidad o la seguridad de las personas puede significar una mejora que se traduce en un bien social.

2.4. Consideraciones finales

En este capítulo se introdujeron algunos conceptos básicos de IoT y el potencial que tiene. Actualmente ya irrumpió en la vida cotidiana de tal manera que está cambiando muchos de sus aspectos. En algunos casos, las aplicaciones que se desarrollen sobre IoT podrían implicar una mejora significativa en la calidad de vida de las personas. Muchas de las áreas en las cuales se podría influir tienen como objetivo mejorar sustancialmente la movilidad, el acceso a la salud, la educación, la seguridad, la documentación, el monitoreo medioambiental, y la industria en general, entre otras. Cuestiones que sin ser primordiales como el acceso a los servicios públicos elementales (agua, gas, electricidad, cloacas) impactan directamente en la vida cotidiana y en última instancia en la productividad económica.

Parte II

Diseño e implementación de sistemas IoT

Capítulo 3

Diseño e implementación de un sistema para monitoreo de riego

3.1. Motivación

La Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), ha establecido como desafío para la agricultura del 2050 [16] suplir un incremento en la demanda mundial de alimentos de manera más sustentable, contemplando, entre otros factores, una futura crisis del agua. Alineado con este objetivo, desde el Estado Argentino se están llevando a cabo diversos proyectos orientados a aumentar y hacer más eficiente la superficie regada para explotación agrícola. Entre ellos se encuentran el Plan Nacional de Riego [17], cuya meta es duplicar la superficie regada para el 2030, y el Plan Nacional de Agua [18], con la meta de incorporar nuevas áreas a la producción agrícola mediante riego.

Uno de los sistemas de riego que más creció en superficie regada en la última década en Argentina, es el sistema por pivote central [19]. Sólo en la Provincia de Buenos Aires, en el año 2015, se hallaron 2.500 círculos regando una superficie de 147.001,73 ha. [20].

El sistema de riego por pivote central es un método de riego por aspersión conformado por una pirámide en el centro, y una serie de tramos enlazados que giran alrededor de la misma formando un círculo. La pirámide está compuesta generalmente por cuatro patas, y contiene un tubo con un codo giratorio por donde sube el agua desde una bomba y se traslada a los tramos. Cada tramo cuenta con un motor que le permite avanzar por el campo y una tubería principal, desde donde cuelgan las boquillas por donde cae el agua al suelo. La Figura 3.1 ilustra un esquema del sistema. Estos equipos operan de manera automática, por lo que no requieren la presencia de una persona en el lote que los opere.

A pesar de su eficiencia de riego, y su autonomía, estos equipos presentan fallas. Una falla implica que el equipo deja de funcionar de la manera esperada. Es decir, que estando el mismo encendido, no avanza y/o no esparce agua. Las causas de las mismas pueden ser varias, como falta de energía (sea eléctrica o combustible) o paradas de seguridad de los motores. Sin embargo, las mismas no son consideradas en este trabajo ya que lo relevante es que la falla ocurre, sin importar el por qué. Existen dos tipos de fallas frecuentes, que pueden resumirse a los siguientes dos escenarios:

1. La bomba de agua deja de suministrar agua, pero los tramos del pivote siguen avan-

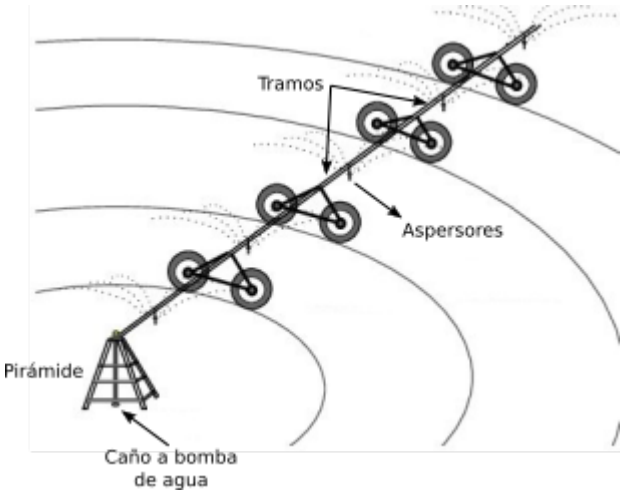


Figura 3.1: Componentes de un pivot.

zando sin regar la superficie debajo de ellos.

2. Los tramos del pivote se frenan, pero la bomba de agua sigue funcionando lo cuál hace que se inunde una zona y se deje de regar otras.

Cualquiera sea el escenario que se presente, el cultivo se verá afectado de manera negativa, ya sea por no recibir agua, o por recibir demasiada. En aquellos con una sensible demanda hídrica, como la papa, esto puede derivar en lo que se conoce como estrés hídrico y disminuir su rendimiento [21]. Además, desde un punto de vista ambiental y económico, también pueden ser muy perjudiciales, ya que estos equipos manejan un caudal aproximado de 200.000 litros / hora, y, cuando están alimentados por generadores, consumen aproximadamente 20 litros / hora de combustible. Como agraviante, en zonas como el sudeste de la Provincia de Buenos Aires, el agua utilizada para riego se obtiene de las napas, por lo que cada hora que un equipo está esparciendo agua sin regar, se están malgastando 200.000 litros de agua potable.

Por estas razones, los responsables de riego (muchas veces los mismos productores agrícolas), supervisan de manera personal y diaria su correcto funcionamiento. Sin embargo, una gran desventaja de este método, es que si se presenta una falla al poco tiempo de finalizada una ronda de supervisión, se debe esperar a la siguiente ronda para que esta sea detectada. Esto se traduce a un desperdicio de combustible y agua directamente proporcional a las horas en las que el responsable está ausente sin supervisar.

En este capítulo se propone un sistema que disminuye el desperdicio en combustible y agua derivados de las fallas detectadas de manera tardía en equipos de riego de pivote central. Existen productos similares en el mercado, pero tienen dos características que dificultan la adopción por parte de los productores. La primera es que dependen de una infraestructura de comunicación que en muchas zonas de Argentina no está disponible [22]. Estas necesitan de cobertura celular o WiFi en el lote o en las proximidades del mismo, ya sea para enviar alertas SMS, para comunicarse desde el lote a un servidor en Internet, o para comunicarse con un nodo cercano mediante RF y utilizarlo como gateway a Internet. La segunda es que muchos de estos productos son provistos por los mismos fabricantes de equipos de riego y sólo funcionan en algunos modelos.

3.2. Materiales y métodos

El sistema puede organizarse mediante las cinco capas arquitectónicas [23] comúnmente utilizadas para estructurar un sistema de Internet de las Cosas: percepción, transporte, procesamiento, aplicación y negocio. Para simplificar el análisis arquitectónico del sistema, no se incluye la capa de negocios y algunos detalles de diseño de servicios (como las notificaciones SMS) y los algoritmos de detección de fallas. La Figura 3.2 muestra los componentes principales, y a continuación se describen las capas.

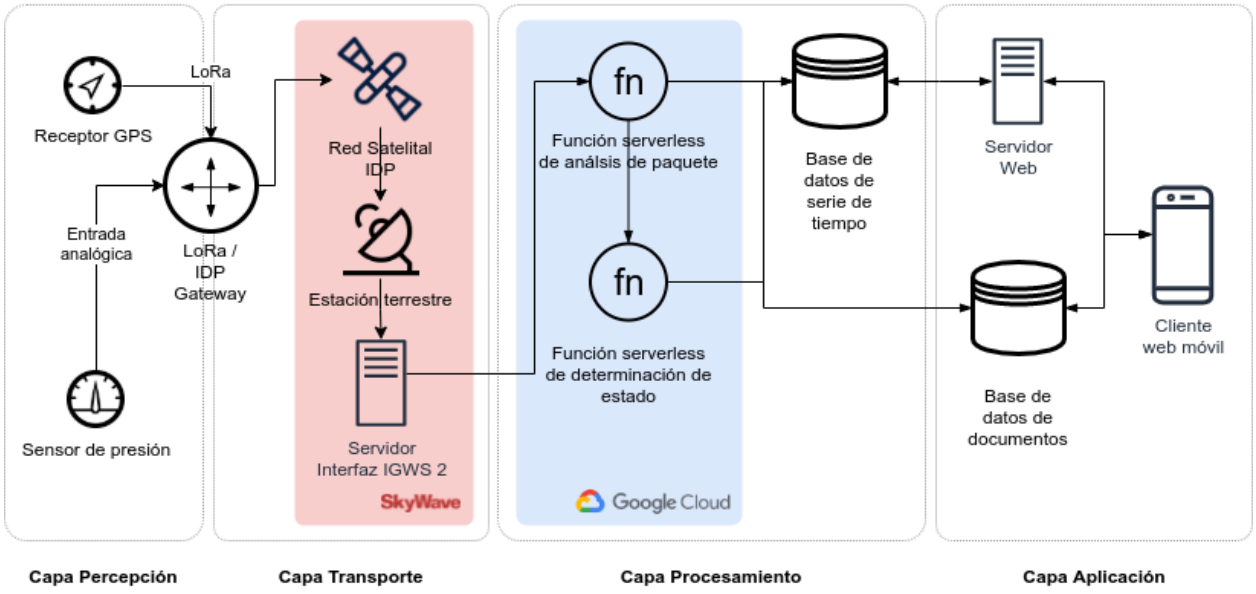


Figura 3.2: Diagrama de capas de la solución con orientación horizontal.

3.2.1. Capa percepción

El desafío principal en esta capa es el de diseñar un sistema de sensores que pueda ser utilizado en cualquier pivote central (interoperable). Luego de analizar productos similares [24], se determinó utilizar un receptor GPS en la punta de los tramos y un sensor de presión en el caño principal de agua del equipo. El receptor GPS es la placa open source SODAQ ONE v3 [25] alimentado con un panel solar de 1 WATT, el cuál envía la posición periódicamente a un gateway diseñado especialmente para este proyecto. El gateway se alimenta del tablero principal del pivote, y tiene conectado el sensor de presión a una entrada analogica de su microcontrolador ATMEGA2560. Con la posición se determina si se está moviendo (mediante diferencia de ubicaciones), y con el sensor de presión si hay agua fluyendo. Estas 2 variables son la salida principal de esta capa.

3.2.2. Capa transporte

Dos puntos importantes a resolver en esta capa incluyen cómo comunicar el receptor GPS con el gateway, teniendo en cuenta que puede haber cientos de metros de distancia

entre ambos, y cómo enviar los datos del campo a un servicio remoto, considerando que el lote puede no contar con señal celular ni servicio de Internet. Para el primer caso, se resolvió utilizar LoRa, un protocolo RF que tiene un alcance teórico mayor a 30 km [26]. La configuración del mismo fue de Spreading Factor 12 y un ancho de banda de 125 KHz. Para el segundo caso, se escogió la comunicación satelital ya que esta garantiza conectividad en cualquier ubicación a cielo abierto sin necesidad de montar una infraestructura aparte. El proveedor elegido fue SkyWave que cuenta con una red de satélites geoestacionarios llamada IsatData Pro (IDP) con cobertura en la región. El gateway, entonces, arma un paquete con datos de presión y posición, y los envía al satélite cada 15 minutos, para que luego sean consumidos por una interfaz REST. El período de envío se eligió considerando los costos del servicio satelital y la necesidad de negocio.

3.2.3. Capa procesamiento

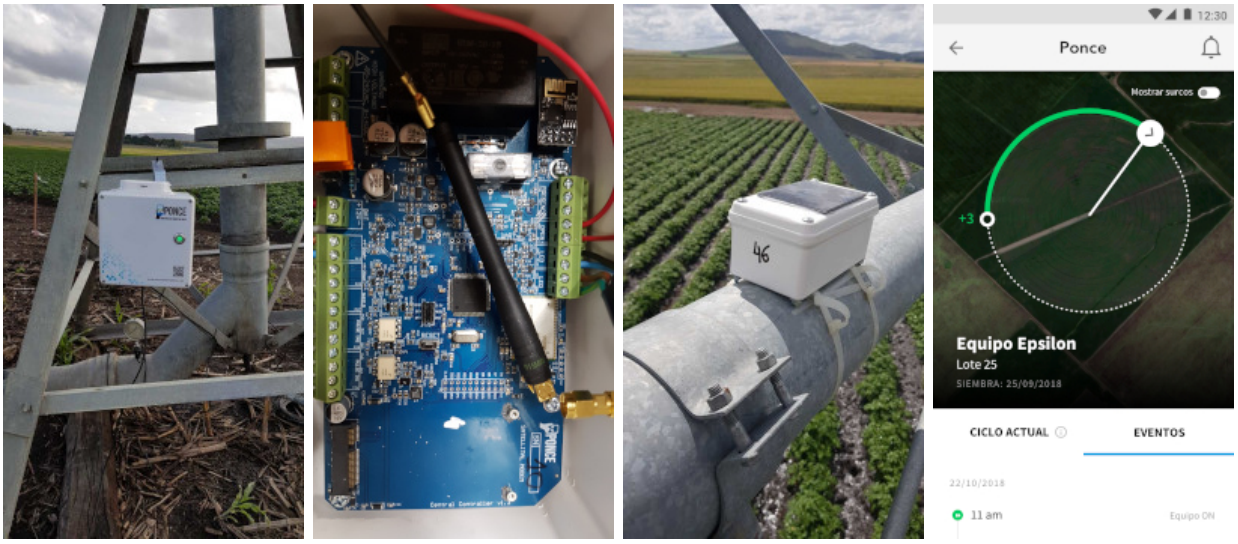
El objetivo de esta capa es procesar y almacenar los datos que llegan desde el campo, principalmente si se está moviendo el equipo y si hay presencia de agua. Con estas variables se puede determinar en cuál de los 3 estados básicos se encuentra el equipo: regando (moviéndose con agua), parado (quieto sin agua) o falla (moviéndose sin agua o parado con agua). Al estar en una fase experimental, es importante poder aumentar o disminuir fácilmente los recursos necesarios para el procesamiento. Se optó por montar un ambiente serverless, en el cuál el proveedor asigna los recursos necesarios para ejecutar el código de manera dinámica [27]. Puntualmente se eligió un modelo FaaS (Function-as-a-Service), donde el código que se envía al servidor es una función asociada con un activador. Como proveedor, se optó por Google Cloud Platform, debido a que ya se contaba con experiencia en su uso, y los productos utilizados fueron Cloud Functions como herramienta FaaS y Cloud PubSub como sistema de mensajería y activadores. Para almacenar tanto los paquetes sin procesar (a modo de copia de seguridad) y los estados ya procesados, se eligió Keen.io que es un servicio robusto de almacenamiento y consulta de eventos, el cual permite asociar dichos datos con una fecha y hora para optimizar consultas con parámetros temporales.

3.2.4. Capa aplicación

Se relevaron cuatro requerimientos que se destacan. El primero es que se debe poder acceder desde un teléfono celular ya que el productor agrícola no siempre está en su oficina. El segundo involucra la interoperabilidad por lo que debe funcionar en gran variedad de modelos y marcas de celular (smartphones). El tercero es que la interfaz gráfica debe reflejar rápidamente los cambios de estado del riego de manera automática. Y el cuarto es que se debe poder consultar la información aún cuando no haya conexión a Internet, por ejemplo en el campo. Para resolver las primeras dos cuestiones se desarrolló una aplicación web adaptable a distintos tamaños de pantalla, lo que permite utilizar cualquier celular moderno como cliente. Para las otras dos, se utilizó la base de datos NoSQL Firestore, que mediante WebSockets actualiza a los clientes cuando detecta un cambio en sus documentos, y se optó por desarrollar la aplicación como PWA (Progressive Web Applications) con caché en el cliente.

3.3. Resultados

La Figura 3.3 muestra imágenes de los resultados del diseño e implementación del sistema propuesto, que comenzó en mayo de 2018 y culminó en marzo de 2019. En total, se monitorearon 59 equipos de 5 marcas distintas: Valley, Lindsay, T-L, Montenegro y Rockink. La distancia más larga entre el receptor GPS y el gateway fue de 770 m mientras que la más corta fue de 282 m. Los mismos estuvieron distribuidos en la Provincia de Buenos Aires, procesando más de 8.035.200 bytes provenientes del gateway y detectando 144 fallas verificadas por los usuarios.



(a) Foto del gateway en campo. (b) Foto de la placa del gateway. (c) Foto del receptor GPS en campo. (d) Captura de la aplicación.

Figura 3.3: Imágenes del sistema implementado.

3.4. Discusiones

Los resultados obtenidos indican que el sistema propuesto cumple con el objetivo de detectar fallas. Sin embargo, se debe continuar probando en más equipos y durante más tiempo para seguir evaluando su rendimiento y confiabilidad a lo largo del tiempo.

La capa percepción no presentó grandes problemas, a excepción de que a algunos sensores presentaron fallas producto de la condensación y hubo que reponerlos. En la capa de transporte, hubo algunas pérdidas de mensajes, que si bien no se registraron formalmente, fueron aisladas y no afectaron el funcionameinto del sistema. La comunicación satelital tuvo también algunas caídas informadas por el proveedor, pero fueron períodos cortos y aislados, y tampoco provocaron grandes interrupciones en el monitoreo. En lo que respecta a la capa de procesamiento y aplicación todo funcionó correctamente y los bugs que se encontraron fueron menores y rápidamente corregidos.

3.5. Consideraciones finales

En este capítulo se presentó la descripción de una arquitectura de un sistema de Internet de las Cosas y algunos detalles de su implementación, aplicada a una problemática del agro. Esta es una primera versión, que será tomada como base para continuar su desarrollo. A continuación, se describen brevemente algunas de las características y desafíos en las próximas tareas de investigación y desarrollo.

En primer lugar, el registro histórico de distintas variables, como la presión, las horas de riego, y las precipitaciones, puede resultar interesante como un sistema de soporte de toma de decisiones para el responsable de riego. En este área se pueden incorporar técnicas de inteligencia artificial para el procesamiento de datos.

También se busca realizar una estadística de la cantidad de veces que un sistema de riego por pivote central puede fallar en una misma campaña de riego. Con esta información, se podrá estimar la cantidad total de fallas en un grupo de pivotes, y, por lo tanto, de recursos que podrían ahorrarse con un sistema como el propuesto en este trabajo.

Una característica a considerar, es cortar la bomba de agua cuando los tramos se frenan y viceversa. Una gran diferencia con respecto al sistema propuesto, es que ejercer control de esta manera resulta poco viable de aplicar en todos los casos. Como primer ejemplo, se puede considerar el caso en que una bomba suministre varios equipos de riego. Si uno de ellos se frena y el sistema apaga la bomba, el resto de los lotes se verán perjudicados por falta de agua. Otro ejemplo, es aquellos sistemas con motor a combustión, donde resulta muy complejo automatizar el apagado de la bomba, y cualquier inconveniente no atendido a tiempo, puede poner en riesgo gran parte de la producción. Se necesitaría de personal capacitado y disponible de manera constante para acudir ante cualquier desperfecto que se presente.

Además, se buscará evaluar la factibilidad de adaptar este sistema a otros equipos de riego utilizados en la producción agrícola. De esta manera, será mayor la posibilidad de mitigar el desperdicio de agua producto de fallas, ayudando a enfrentar la crisis de agua que se espera.

Por último, se trabajará para migrar la comunicación a un estándar como LoRaWAN. Esto traerá el beneficio de utilizar un protocolo probado por varias industrias e incorporar nuevos sensores de terceros. El principal desafío es adaptar LoRaWAN a una red con capacidad limitada como es la satelital. Un primer paso se da en otro capítulo de esta tesis.

Capítulo 4

Diseño de una infraestructura para un sistema de auto-evacuación ante desastres naturales

4.1. Motivación

Según las Naciones Unidas, durante el 2019 el 55,3 % de la población mundial vivía en asentamientos urbanos. Este número aumentará a un 60 % para 2030, donde una de cada tres personas vivirá en ciudades con al menos medio millón de habitantes [28]. El crecimiento de la población urbana hace que las ciudades sean más vulnerables a los desastres naturales y a los provocados por el hombre [29]; por lo tanto, se requieren nuevos sistemas para hacer que las ciudades sean más resilientes.

La estrategia más aceptada para alcanzar tal objetivo es diseñar sistemas robustos e inteligentes que permitan reducir riesgos y mitigar el impacto de eventos extremos en la población civil. Por lo general, el diseño y la implementación de estos sistemas se realizan durante la etapa de "preparación" para desastres (es decir, durante la planificación de cómo responder a un evento extremo), y los sistemas resultantes se utilizan durante la etapa de "respuesta" [29].

Aunque todas las actividades de socorro en casos de desastre son importantes, este trabajo se centra en apoyar la evacuación de la población civil durante los desastres naturales. El objetivo del proceso de evacuación es llevar a las personas a un lugar seguro lo antes posible, considerando el estado actual y futuro del evento extremo (por ejemplo, un tsunami, huracán, deslizamiento de tierra, inundación o incendios forestales).

Mientras que en las emergencias urbanas, como incendios en edificios, los primeros en dar soporte son las organizaciones de respuesta temprana (por ejemplo bomberos, policías y paramédicos), en el caso de desastres naturales el proceso se realiza principalmente a través de auto-evacuaciones, ya que son masivas y se realizan en paralelo. Por lo tanto, los civiles y la infraestructura pública deben estar preparados para abordar ese proceso. Contar con planes de evacuación y sistemas de apoyo adecuados reduce los riesgos y también el impacto de un evento extremo.

La idoneidad de un sistema para soportar una evacuación es consecuencia de su diseño.

Aunque la implementación del sistema también juega un papel clave en la efectividad de la solución, su diseño es mucho más relevante ya que un diseño deficiente generalmente no se puede solucionar con una implementación adecuada.

El diseño de un sistema de evacuación debe abordarse en la etapa de preparación, y su efectividad potencial debe evaluarse teórica y empíricamente para asegurar su idoneidad cuando se utilice en escenarios reales. En este sentido, la evaluación del diseño del sistema antes de implementarlo, es tan importante como su modelado.

El diseño de estos sistemas debe considerar como usuarios no sólo a los socorristas y gestores de emergencias, sino también a los evacuados, ya que son productores y consumidores directos de información sobre el proceso de evacuación y del evento extremo que están cursando. Por lo tanto, estos sistemas deben ser interactivos y, con suerte, proporcionar inteligencia ambiental (Ambient Intelligence o AmI) a través de soluciones distribuidas y autónomas [30].

Dado que la infraestructura de comunicación regular en el área afectada suele caer o colapsar durante un desastre [31], el sistema de evacuación debe contar con su propia infraestructura de comunicación, que debe ser lo suficientemente resistente como para operar en condiciones extremas. Teniendo en cuenta la gran cantidad de civiles que participan en estas evacuaciones, el flujo de información permitido entre los participantes debe diseñarse adecuadamente para evitar la sobrecarga de comunicaciones y el colapso de la infraestructura.

En este capítulo se propone una infraestructura de comunicación basada en IoT que permite la participación activa de civiles en auto-evacuaciones a gran escala. Describimos su estructura, comportamiento, la forma de desplegarlo en áreas urbanas, y también la forma en que varios tipos de usuarios lo emplean para proporcionar y consumir información. La infraestructura proporciona inteligencia ambiental a los evacuados en la ruta hacia los refugios.

El trabajo también presenta una representación formal de dicha infraestructura utilizando IoT- Calculus [32]. Esta formalización facilita la evaluación temprana del diseño del sistema de evacuación y permite su mejora durante el proceso de preparación para emergencias.

Este trabajo amplía la presentación de una conferencia previa en la que los autores introdujeron los conceptos básicos de esta propuesta [30]. Esta extensión incluye una definición completa de la infraestructura basada en IoT, su validación, una estrategia simplificada para desplegarla, una implementación real de los nodos de la infraestructura y simulaciones de medidas reales que muestran el desempeño potencial de un sistema de evacuación en particular.

4.2. Estado del arte

Diseñar e implementar sistemas de evacuación de emergencia basados en TICs representa un desafío para cualquier organización; particularmente, cuando estos sistemas tienen que soportar auto-evacuaciones. Este desafío aumenta con el tamaño y la complejidad del área a abordar, y también con el número y tipo de participantes involucrados en el proceso de evacuación. Por lo tanto, modelar e implementar estos sistemas tiende a ser más factible cuando tienen que respaldar las auto-evacuaciones de los edificios que abordar procesos masivos (por ejemplo, después de una alerta de tsunami).

Los avances en la tecnología de Internet de las cosas (IoT) nos han permitido concebir e implementar inteligencia ambiental en espacios reducidos (por ejemplo, edificios, centros comerciales y aeropuertos), y así dar un mejor apoyo a los procesos de auto-evacuación [33]. En [34, 35] se presentan ejemplos de estos sistemas de evacuación basados en IoT. Por lo general, estos sistemas determinan las rutas de evacuación en tiempo real, considerando las características del peligro, el comportamiento de los evacuados y las condiciones ambientales, y luego guían a los evacuados a zonas seguras. Aunque son útiles, la mayoría de los sistemas de evacuación basados en IoT implican una importante instrumentación de infraestructura física, lo que limita su capacidad para abordar evacuaciones a gran escala.

En las evacuaciones masivas, los sistemas de alerta temprana (Emergency Warning Systems o EWS) suelen utilizar la infraestructura de comunicación de transmisión (por ejemplo, transmisión de radio y televisión, y sistemas de sirena) para iniciar el proceso de evacuación y dar indicaciones generales a los evacuados. La transmisión de información a través de estas infraestructuras es unidireccional, lo que restringe la participación de los evacuados como potenciales proveedores de información [36].

También hay EWS y sistemas de gestión de emergencias que permiten la comunicación digital full-duplex entre organizaciones de respuesta temprana y agencias gubernamentales durante las evacuaciones; por ejemplo, a través de redes celulares y privadas [37, 38]. Sin embargo, estos sistemas no suelen estar disponibles para civiles [39, 40].

Varios investigadores han planteado la necesidad de incluir a los evacuados como actores formales en los esfuerzos de socorro en casos de desastre, dado que las organizaciones de respuesta temprana generalmente no pueden ocuparse de estas actividades en períodos cortos de tiempo [41, 42]. Por lo tanto, los civiles que actúan como sensores humanos pueden proporcionar información sobre el estado y la evolución tanto del evento como del proceso de evacuación [43, 44, 45, 46]. Esto abre varias oportunidades para concebir nuevos sistemas interactivos, basados en el paradigma de Internet de las Personas [47], que ayuden a incrementar la efectividad de las evacuaciones masivas.

Con respecto a la participación de civiles en situaciones de emergencia, la literatura reporta varias experiencias en las que las personas utilizan los servicios de redes sociales (Social Network Services o SNS) para ayudar a lidiar con estos eventos [48, 49]. Sin embargo, en los desastres naturales el área de incidencia suele verse afectada por un apagón de comunicación debido a daños físicos, sobrecarga de tráfico o falta de electricidad [50, 37, 38]. Por lo tanto, el uso de SNS y la infraestructura de comunicación regularmente utilizada puede no ser factible en estos lugares. Las personas ubicadas fuera del área de incidencia difunden frecuentemente información a través del SNS sobre el proceso de respuesta y el evento; sin embargo, esta información tiende a estar desactualizada, ser inexacta o incompleta [51]. Esto limita la capacidad de usar SNS para monitorear la evolución del proceso de evacuación y el evento.

La literatura también reporta pocas propuestas que permitan a los civiles ubicados en el área afectada reportar y recibir información durante una evacuación; la mayoría de ellos considera sistemas de comunicaciones móviles ad hoc (por ejemplo, basados en Twitter o mensajes de texto ad hoc [52, 53]). Estos sistemas interactivos son útiles para coordinar actividades en áreas pequeñas, pero presentan limitaciones en evacuaciones a gran escala ya que la infraestructura de comunicación que utilizan tiene baja confiabilidad y ancho de banda, y un corto rango de alcance [54]. En este sentido, el trabajo más cercano a esta propuesta es

el sistema de comunicación propuesto en [55], donde los civiles utilizan una aplicación móvil y una infraestructura LoRaWAN para intercambiar mensajes de texto cortos con familiares y organizaciones de respuesta temprana. Por lo tanto, estas personas informan su estado y reciben información de otros. Dicho sistema fue concebido para apoyar a la población civil después de un terremoto, donde las evacuaciones no son frecuentes y la movilidad de las personas es baja o nula. Por lo tanto, esa propuesta debe adaptarse para usarla como soporte de comunicación durante evacuaciones a gran escala.

Los trabajos anteriores destacan la necesidad de que los sistemas de evacuación sean interactivos y permitan la participación ubicua de los evacuados, no solo como sensores humanos, sino también como receptores de información. Las capacidades de estos sistemas deben diseñarse y evaluarse durante la etapa de preparación de estas evacuaciones para tratar de asegurar la efectividad de estos sistemas cuando se utilicen en la práctica. El modelado formal aparece como un enfoque apropiado para representar tanto la estructura y comportamiento de los sistemas como su interacción con los usuarios potenciales. Este modelo formal también nos permite visualizar las capacidades de los sistemas y realizar evaluaciones y ajustes tempranos durante el proceso de preparación.

La literatura reporta varias propuestas para modelar diferentes aspectos de un sistema de evacuación. Por ejemplo, en [56] los autores presentan una ontología llamada SEMA4A (Simple EMergency Alert 4 [for] All) para modelar la accesibilidad a los mensajes de notificación de emergencia a través de distintos sistemas de información de respuesta a emergencias. En [57], los autores amplían el trabajo anterior e introducen una técnica semiautomática para la adquisición de conocimientos y el modelado de rutas de evacuación accesibles. Finalmente, en [45] los autores proponen algunas pautas para implementar aplicaciones móviles que apoyen las notificaciones de emergencia y actividades de respuesta donde los ciudadanos actúan como sensores humanos.

Por otro lado, en [58] los autores informan de una encuesta sobre enfoques para el modelado jerárquico de sistemas complejos de IoT, como los necesarios para soportar grandes evacuaciones. En [59] se introduce un enfoque basado en modelos para modelar a los diferentes usuarios (por ejemplo, evacuados y socorristas) que participan en un sistema AmI. En [60] los autores introducen el cálculo para representar la dinámica de estos sistemas, basándose en el cálculo π propuesto en [61]. Siguiendo un enfoque similar, en [32] los autores amplían el cálculo π para hacer frente a las características especiales de los dispositivos, procesos y servicios de redes de IoT.

En este trabajo, se utiliza este último enfoque para representar formalmente la estructura y el comportamiento de una infraestructura basada en IoT que admite auto-evacuaciones en procesos masivos. Esta formalización utiliza el IoT-Calculus para describir los componentes del sistema y la interacción entre ellos; también se emplea para validar la funcionalidad de un sistema de evacuación particular. Las siguientes secciones describen y formalizan la infraestructura propuesta basada en IoT.

4.3. Propuesta

4.3.1. Arquitectura

Para comprender mejor la infraestructura propuesta, primero se deben identificar los tipos de usuarios que emplean el sistema de soporte de evacuación para realizar sus actividades. Estos usuarios son tomadores de decisiones, socorristas y evacuados como se muestra en la Figura 4.1, con especial énfasis en esta última población porque es la más grande y generalmente no se incluye en otros sistemas de evacuación basados en TIC.

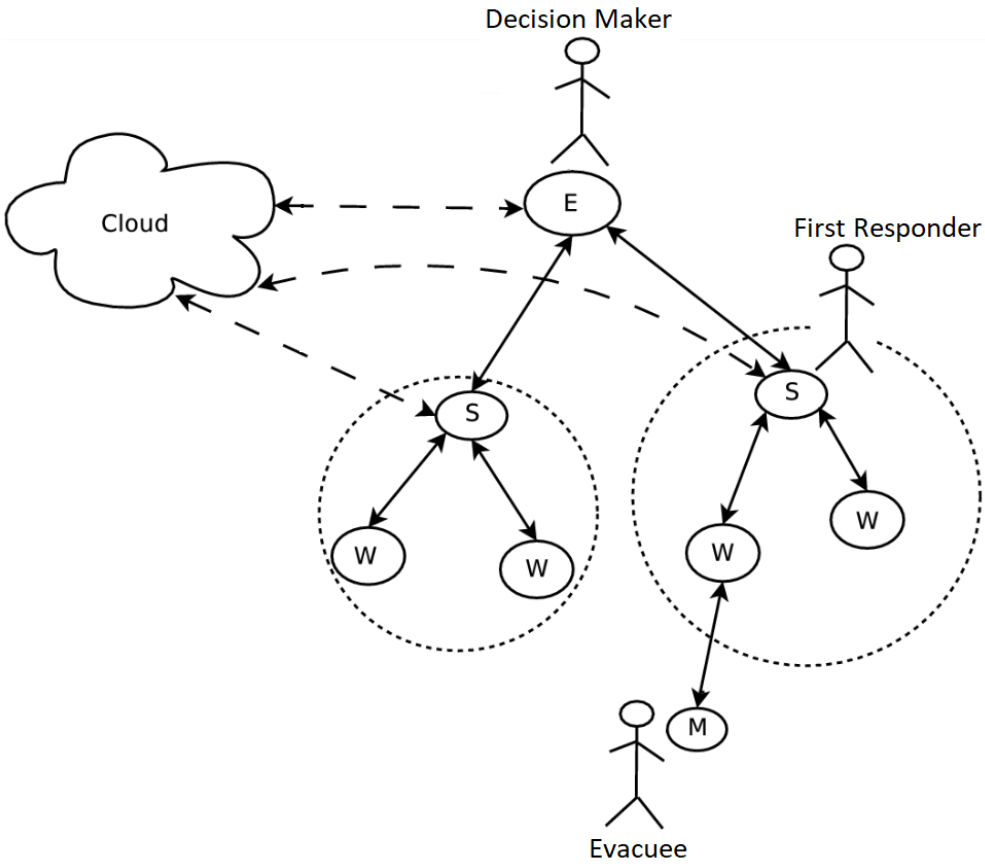


Figura 4.1: Jerarquía de roles considerada en el sistema.

La infraestructura propuesta no considera interacciones directas entre estos usuarios, como una forma de reducir la sobrecarga de información en sus nodos y enlaces de comunicación. En cambio, proporciona inteligencia ambiental a través de una jerarquía de componentes que coopera para brindar servicios a los usuarios. La interacción entre un usuario y el sistema de evacuación, y también entre los componentes de la infraestructura, se basa en un conjunto de servicios predefinidos. Estos servicios utilizan información estructurada para asegurar su transmisión y procesamiento apropiados, aún si la carga en los componentes del sistema es alta.

La participación de los usuarios finales se materializa mediante el uso de aplicaciones de software que son capaces de interactuar con un tipo particular de nodo de la infraestructura

propuesta. Los tipos de nodos se definieron como roles de los componentes del sistema y se organizaron en una jerarquía de acuerdo con la infraestructura propuesta basada en IoT. Estos roles son los siguientes: Centro de Operaciones de Emergencia (E), Refugios (S, del inglés Shelters), Unidades de Testigos (W, del inglés Witness Units) y Unidades Móviles (M). Producen y consumen información según su tipo, y todos son estacionarios, excepto las unidades móviles. La Figura 4.1 presenta un diagrama de la infraestructura que incluye la jerarquía de roles y las interacciones permitidas entre ellos.

El centro de operaciones de emergencia es la raíz del sistema; concentra toda la información disponible y también la mayor parte de su procesamiento. Los administradores de emergencias monitorean y coordinan los esfuerzos de evacuación desde este nodo. Los refugios son administrados por socorristas que reciben, albergan y eventualmente enrutan a los evacuados. Cada refugio informa sobre su estado (es decir, su disponibilidad y capacidad para recibir evacuados) al centro de operaciones de emergencia y a las unidades de testigos ubicadas dentro de su área de influencia. Las unidades de testigos son componentes autónomos que permiten a las unidades móviles (es decir, los evacuados) acceder a la infraestructura de apoyo de evacuación, por ejemplo, para informar sobre su ubicación o para obtener rutas de evacuación sugeridas para llegar al refugio más cercano. Finalmente, las unidades móviles están representadas por los dispositivos móviles utilizados por los evacuados.

Estos nodos pueden producir y consumir diferentes tipos de información de acuerdo con su jerarquía, y estas acciones pueden realizarse por orden explícita de un ser humano (usuario) o como resultado de interacciones no supervisadas entre los agentes autónomos que se encuentran en los nodos. El tipo de interacciones permitidas entre nodos debe definirse en el momento del diseño del sistema, ya que afecta a la escalabilidad del mismo, su capacidad para lidiar con el flujo de información y, en consecuencia, su efectividad potencial para soportar evacuaciones a gran escala.

Esta infraestructura puede operar independientemente de una conexión a Internet, sin embargo, se guarda una copia de su información en la nube como respaldo, para aumentar la disponibilidad de la información en caso de colapso de nodos o enlaces de comunicación, y también para permitir el monitoreo remoto del proceso sin interferir con el funcionamiento del sistema. Aunque los nodos tipo E y S están preparados para trabajar en condiciones extremas, existe una conexión satelital entre estos nodos y la nube (enlaces de puntos en la Figura 4.1).

4.3.1.1. Descripción de roles

Cada tipo de nodo (es decir, el rol considerado en la infraestructura) tiene capacidades particulares para producir y consumir información, y también para interactuar con otros nodos. A continuación, se describen las propiedades y capacidades de estos nodos.

Unidades Móviles Los nodos etiquetados como M representan los dispositivos móviles utilizados por los evacuados (por ejemplo, un teléfono inteligente). Estos dispositivos deben tener una interfaz WiFi y una aplicación de software específica que permita a los usuarios conectarse a nodos W (es decir, unidades testigo) cerca de la ubicación de los usuarios. Una vez iniciada, la aplicación escanea el entorno en busca de un W que sirva como punto de acceso para iniciar sesión en el sistema de evacuación. Esto permite a los nodos M recuperar rutas de

evacuación actualizadas en función de su ubicación utilizando la información intercambiada con la W. Las unidades móviles pueden informar sobre su ubicación actual y también la presencia de obstáculos en el área a través de la unidad testigo a la que están conectadas.

Unidades Testigo Como se mencionó, los nodos W le permiten a los nodos M acceder a la misma información disponible en el sistema; por ejemplo, las rutas de evacuación para llegar al refugio más cercano. Estos nodos pueden verse como puntos de acceso que al mismo tiempo son nodos finales de una LPWAN (Low Power Wide Area Network). La funcionalidad de los W les permite recibir información de Ms sobre nuevos obstáculos, calcular las rutas basándose en la información del área y enviar una ruta actualizada al refugio seguro más cercano. La cantidad de dispositivos conectados a un nodo W permite al sistema visualizar la cantidad de evacuados que requieren un refugio en el área y la cantidad de personas que utilizan los diferentes caminos. La interacción entre los M y los W debe ser ligera (por ejemplo, a través de mensajes JSON codificados) para aumentar tanto como sea posible la capacidad de Ws para actuar como un punto de acceso para los evacuados. La cobertura que brindan los W en el área de evacuación influye fuertemente en la capacidad del sistema AmI para dirigir a los evacuados hacia los refugios. Por lo tanto, el despliegue de estos nodos debe planificarse cuidadosamente para garantizar la mayor cobertura posible en el área de evacuación.

Refugios Los refugios, etiquetados como S, son el destino de los evacuados. Las características de cada refugio (por ejemplo, ubicación, capacidad, disponibilidad, ocupación y área de influencia) son informadas y actualizadas por los equipos de respuesta temprana a cargo de la gestión de estos nodos. Esta información se entrega periódicamente a través del sistema ya que tiene consecuencias directas en la evolución del proceso de evacuación. Normalmente, un refugio se representa en el sistema a través de una o más computadoras con conexión satelital a la nube y conexión inalámbrica tanto a las W en su área de influencia como al centro de operaciones de emergencia. Como se muestra en la Figura 4.1, cada S tiene un conjunto de W que se distribuyen dentro de su área de influencia. Cuando un S ya no está disponible por algún motivo, los W enrutan a las personas a otro nodo S.

Centro de Operaciones de Emergencias Este nodo, etiquetado como E, representa el principal repositorio y unidad de procesamiento de información del sistema de evacuación; también es la raíz de la infraestructura propuesta. Además de recibir todos los mensajes de los S, E tiene que actualizar las rutas de evacuación hacia los S y dar cuenta de su capacidad para recibir nuevos evacuados. La aplicación de software utilizada por los tomadores de decisiones para monitorear el proceso les permite agregar o eliminar obstáculos en el área de evacuación. Con esta información adicional, el sistema de evacuación puede calcular dinámicamente rutas más apropiadas para los evacuados y mantener informados a los nodos.

4.3.1.2. Flujo de la información

La Figura 4.2 muestra el flujo de información entre los roles considerados en la infraestructura propuesta. La información se mueve a lo largo del sistema a través de caminos

jerárquicos. La información sobre obstáculos o bloqueos detectados por los evacuados eventualmente llega al E en una propagación aguas arriba, a través de Ws y Ss. Esta información se propaga posteriormente en sentido descendente para mantener a todos los nodos debidamente informados. La figura también muestra la forma en que W y S toman decisiones locales basadas en datos provenientes de los M.

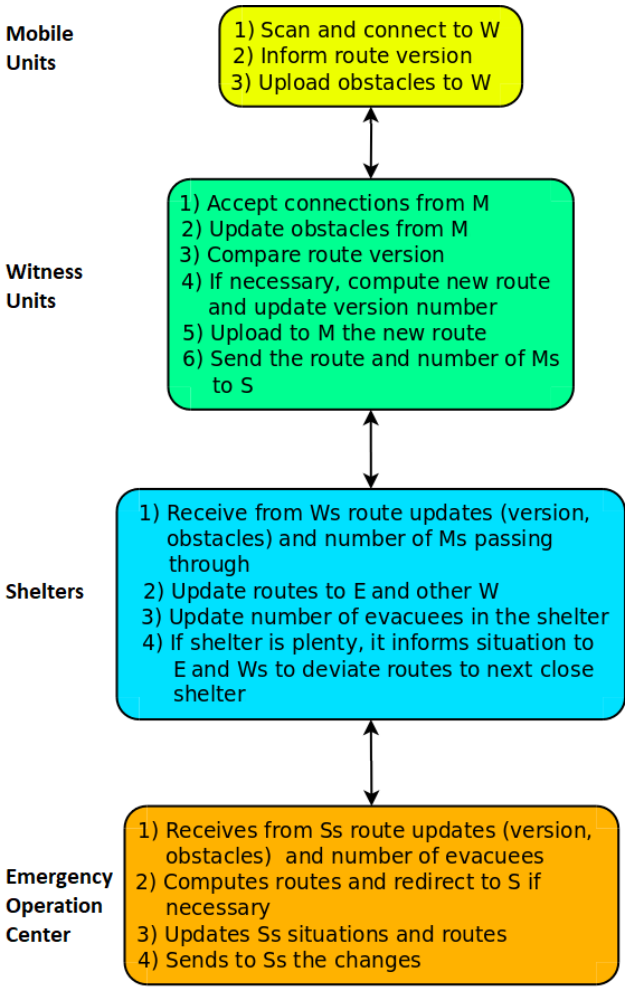


Figura 4.2: Flujo de información entre los roles del sistema.

4.3.1.3. Tolerancia a fallas

El sistema propuesto se basa en una infraestructura distribuida, donde algunos nodos pueden fallar debido a varias razones. Por lo tanto, en el diseño de la infraestructura se consideró su capacidad para recuperarse de fallas parciales. Por ejemplo, los nodos implementan un protocolo "keep-alive" mediante el intercambio periódico de mensajes de estado como una forma de detectar fallas en los nodos del sistema.

Cuando un nodo W está caído, el refugio vinculado a esa unidad asume la función de W e informa la situación a E. Cuando un refugio está caído (o sobrepoblado), los W vinculados

a él reconocen la situación y dirigen a los evacuados al siguiente más cercano. El E también detecta la situación e informa en consecuencia a los demás refugios de la zona.

Cuando el nodo E está caído, la estructura jerárquica del sistema permite que cada refugio asuma la coordinación de los W en su área de influencia. En consecuencia, se crean varias islas de infraestructura que operan de manera independiente una de otra hasta que los E vuelven a estar disponibles para coordinar y unificar el sistema. Si bien esta última situación tiene una baja probabilidad de ocurrir, dado que el E está particularmente preparado para enfrentar eventos extremos, también se consideró en el diseño de la infraestructura.

4.3.2. Modelado de interacciones

En esta sección, describimos formalmente los nodos y sus interacciones utilizando IoT-Calculus [32]. El uso de una representación formal nos permite validar el diseño de un sistema de evacuación particular que utiliza la infraestructura propuesta; particularmente, es posible determinar la capacidad del sistema para soportar interacciones y flujo de información entre los nodos. Por lo tanto, el rendimiento potencial del sistema se puede evaluar a través de simulaciones y su diseño se puede ajustar en el momento de la preparación, considerando los resultados de la simulación.

Siguiendo la semántica propuesta en IoT-Calculus, todo el sistema puede ser descrito por(4.1):

$$n_E \leftrightarrow n_S | n_S \leftrightarrow n_W | n_W \leftrightarrow n_M \vdash \{\text{E}|\text{S}|\text{W}|\text{M}\} \tag{4.1}$$

donde n_x indica un nodo de tipo x, en este caso E, S, W o M. El símbolo $\leftrightarrow$ indica la presencia de una comunicación de enlace entre ambos nodos y | indica que los enlaces, nodos o procesos funcionan en paralelo.

El alcance de este modelo se ha limitado a describir las principales interacciones entre los roles considerados en el sistema. Además, para poner foco en el modelado de las interacciones, la computación local adicional que realiza cada nodo no se describe en el cálculo, sino que se ha encapsulado en las funciones siguientes:

- `setV(`v`)`: almacena el número de versión v de la ruta en el nodo actual.
- `getV()`: devuelve el número de versión de la ruta almacenado en el nodo actual.
- `setO(`o`)`: Almacena los obstáculos o en el nodo actual.
- `getO()`: Devuelve los obstáculos almacenados en el nodo actual desde la última vez que se invocó esta función.
- `setR(`r`)`: almacena los cambios de la ruta r en el nodo actual.
- `getR(`v`)`: Devuelve los cambios de ruta desde el número de versión v de la ruta.
- `updateR(`o`)`: calcula y almacena una nueva ruta y el número de versión de la ruta, en función de la ruta almacenada actual, y los obstáculos agregados o eliminados en o.

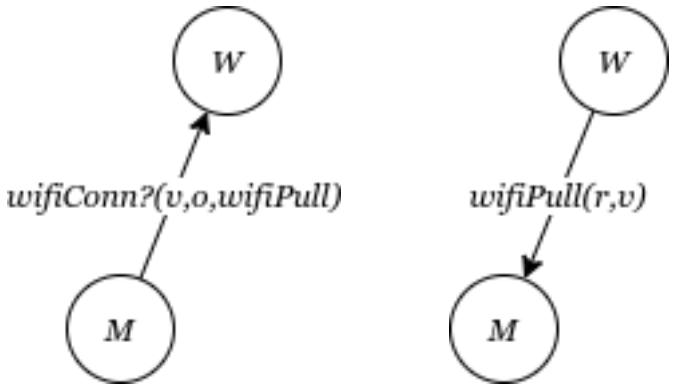
Se asume que cada vez que se recibe una nueva ruta, las funciones anteriormente descritas resuelven la gestión de conflictos y la fusión de rutas.

Una vez que el sistema está formalmente definido, el siguiente paso es identificar los procesos que se ejecutan en cada tipo de nodo, y los canales y variables que utilizan para comunicarse dentro de un nodo y entre diferentes nodos.

4.3.2.1. Interacciones entre unidades Móviles y Testigo

Siempre que los evacuados se muevan hacia los nodos S durante el proceso de evacuación, es necesario representar el hecho de que los M y los W tendrán conexiones transitorias entre ellas. Por lo general, los nodos M escanean el área en busca de la presencia de un nodo W que les permita iniciar sesión en el sistema y, luego, obtener o actualizar su ruta de evacuación. Por esta razón, los M envían automáticamente una solicitud de transmisión $wifiConn$ con el canal de descarga $wifiPull$, y los valores de la ruta almacenada localmente con versión v y los nuevos obstáculos o. Para facilitar la comprensión, se ha dejado fuera del modelo los detalles involucrados en el proceso por el cual ambos roles se conectan.

Una vez que estos nodos establecen una conexión, el W calcula y actualiza la ruta y su versión en base a los obstáculos recibidos de las unidades móviles (si los hay), y utiliza el canal de descarga $wifiPull$ para enviar la última ruta r y el número de versión v. La ruta r podría ser solo la diferencia entre la ruta conocida M y la ruta recién creada en W, mientras que la versión v podría ser un número basado en una marca de tiempo relacionada con la última ruta. La Figura 4.3 muestra los canales y la interacción entre los nodos M y W.



(a) (b)

Figura 4.3: (a) M envía una solicitud de conexión y sus datos locales a W, y (b) W envía la última ruta y versión a M.

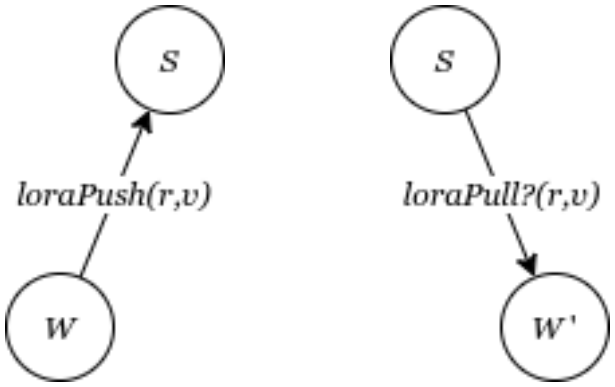
La definición formal de M se puede encontrar en (4.2). Está compuesto por dos procesos que se ejecutan en paralelo. El proceso P se encarga de interactuar con el nodo W y actualizar la pantalla del móvil am, mientras que el proceso Q interactúa con la entrada explícita de un usuario a través de sm para obtener la lista de obstáculos en el área y registrarlos en el sistema.

$$
\begin{aligned}
\mathrm{M} &= n_M[\ !P \mid !Q\ ] \\
P &= \overline{wifiConn}!\langle \texttt{getV()}, \texttt{getO()}, wifiPull\rangle. \\
&\quad\ wifiPull(r, v).\texttt{setV}(v).r \rightarrow a_m \\
Q &= (o) \leftarrow s_m.\texttt{setO}(o)
\end{aligned}
\tag{4.2}
$$

La descripción de los W es más compleja ya que interactúan con los M, pero también con los S. La representación de su interacción se completa con la Figura 4.4.

4.3.2.2. Interacciones entre unidades Testigo y Refugios

Una vez que el W tiene una nueva ruta, además de actualizar el M, transmitirá estos nuevos cambios a S a través del canal *loraPush*. Luego, el S utilizará dicho canal para transmitir actualizaciones a otros W (representados en el diagrama como W') cuyas rutas de evacuación se vean afectadas por estos cambios. En este caso, no es necesario establecer una conexión como se hacía con WiFi, ya que muchas LPWAN (como LoRa, o LoRaWAN en modo ABP) no requieren un proceso de registro.



(a) (b)

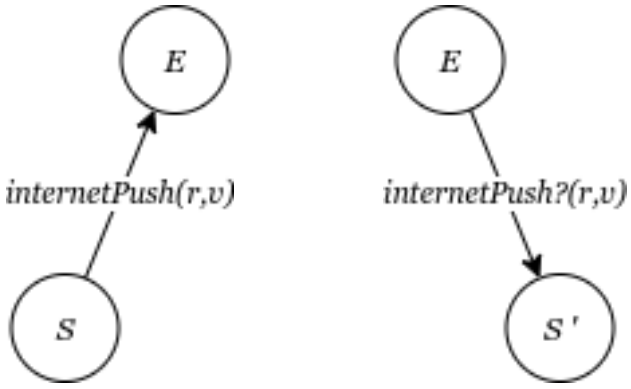
Figura 4.4: (a) Una vez que W calcula una nueva ruta, la envía a S, y (b) S actualiza las otras unidades testigo W' correspondientes cuyas rutas se han visto afectadas.

El nodo W, que se define formalmente en (4.3), ejecuta simultáneamente los procesos R y T. El proceso R acepta la solicitud de conexión del nodo M, calcula la nueva ruta y luego envía los datos de ruta actualizados a los nodos M y S en dos subprocesos paralelos. El proceso T describe la obtención de datos actualizados de la ruta del nodo S.

$$
\begin{aligned}
\mathrm{W} &= n_W[\ !R \mid !T\ ] \\
R &= wifiConn?(v, o, wifiPull).\texttt{updateR}(o). \\
&\quad\ (\overline{wifiPull}\langle \texttt{getR}(v), \texttt{getV()}\rangle \mid \\
&\quad\ \overline{loraPush}\langle \texttt{getR}(v), \texttt{getV()}\rangle) \\
T &= loraPull?(v, r).\texttt{setV}(v).\texttt{setR}(r)
\end{aligned}
\tag{4.3}
$$

4.3.2.3. Interacciones entre Refugios y el Centro de Operaciones de Emergencias

Las S son similares a las W ya que mantienen una doble conexión: una con los W y la otra con el E. La interacción con E se puede realizar con una conexión a Internet por satélite confiable o con un protocolo de RF de largo alcance confiable. Cada nodo S puede pensarse como un gateway LPWAN (por ejemplo, LoRa o LoRaWAN). Cuando recibe nuevos datos de ruta, los envía a los nodos W como se explicó anteriormente, y también al nodo E con un enlace unidireccional llamado *internetPush*. Otros nodos, S', recibirán esta actualización de E a través de un canal de transmisión de difusión llamado *internetPull*. Esto se ilustra en la Figura 4.5.



(a) (b)

Figura 4.5: (a) S envía nuevos datos de ruta a E, y (b) E actualiza los otros refugios S' con los datos recibidos.

S se define formalmente en (4.4). Como en casos anteriores, dos procesos paralelos establecen el comportamiento de este tipo de nodo. El proceso U recibe datos de ruta del W y los envía a E y a otros W. El proceso V recibe datos de E y los envía a los W que se encuentren en el área de influencia de S.

$$
\begin{aligned}
\mathrm{S} &= n_S[\ !U \mid !V\ ] \\
U &= \mathit{loraPush}(r,v).\texttt{setR(}r\texttt{).setV(}v\texttt{).} \\
&\quad (\overline{\mathit{loraPull}}!\langle r,v\rangle \mid \overline{\mathit{internetPush}}\langle r,v\rangle) \\
V &= \mathit{internetPull}?(r,v).\texttt{setR(}r\texttt{).setV(}v\texttt{).} \\
&\quad \overline{\mathit{loraPull}}!\langle r,v\rangle
\end{aligned}
\tag{4.4}
$$

Finalmente, el centro de operaciones de emergencia E se describe en (4.5), en el que están involucrados los procesos X e Y. El primero almacena y transmite a los S' los nuevos datos de ruta recibidos de un S. El segundo recibe información sobre obstáculos de los tomadores de decisiones a través de s_e, luego actualiza los datos de ruta y, finalmente, transmite dicha información a los nodos S'.

$$
\begin{aligned}
\mathrm{E} &= n_E[\ !X \mid !Y\ ] \\
X &= internetPush(r, v).\texttt{setR(}r\texttt{).setV(}v\texttt{).} \\
&\quad\ \overline{internetPull}!\langle r, v\rangle \\
Y &= (o) \leftarrow s_e.\texttt{updateR(}o\texttt{)}.\overline{internetPull}!\langle r, v\rangle
\end{aligned}
\tag{4.5}
$$

4.3.2.4. Formalizando el flujo de información a través del sistema

$$
\begin{aligned}
&\text{(a) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[!trigP().P.\overline{trigP}\langle\rangle \mid !trigQ().Q.\overline{trigQ}\langle\rangle \mid \overline{trigP}\langle\rangle \mid \overline{trigQ}\langle\rangle] \mid \\
&\qquad\qquad n_W[!trigR().R.\overline{trigR}\langle\rangle \mid !trigT().T.\overline{trigT}\langle\rangle \mid \overline{trigR}\langle\rangle \mid \overline{trigT}\langle\rangle] \mid \\
&\qquad\qquad n_S[!trigU().U.\overline{trigU}\langle\rangle \mid !trigV().V.\overline{trigV}\langle\rangle \mid \overline{trigU}\langle\rangle \mid \overline{trigV}\langle\rangle] \mid \\
&\qquad\qquad n_E[!trigX().X.\overline{trigX}\langle\rangle \mid !trigY().Y.\overline{trigY}\langle\rangle \mid \overline{trigX}\langle\rangle \mid \overline{trigY}\langle\rangle] \xrightarrow{\tau} \\
&\text{(b) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[!trigP().P.\overline{trigP}\langle\rangle \mid !trigQ().Q.\overline{trigQ}\langle\rangle \mid P.\overline{trigP}\langle\rangle \mid Q.\overline{trigQ}\langle\rangle] \mid \\
&\qquad\qquad n_W[!trigR().R.\overline{trigR}\langle\rangle \mid !trigT().T.\overline{trigT}\langle\rangle \mid R.\overline{trigR}\langle\rangle \mid T.\overline{trigT}\langle\rangle] \mid \\
&\qquad\qquad n_S[!trigU().U.\overline{trigU}\langle\rangle \mid !trigV().V.\overline{trigV}\langle\rangle \mid U.\overline{trigU}\langle\rangle \mid V.\overline{trigV}\langle\rangle] \mid \\
&\qquad\qquad n_E[!trigX().X.\overline{trigX}\langle\rangle \mid !trigY().Y.\overline{trigY}\langle\rangle \mid X.\overline{trigX}\langle\rangle \mid Y.\overline{trigY}\langle\rangle] \xrightarrow{\tau:o' \leftarrow s_m} \\
&\text{(c) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[!trigP().P.\overline{trigP}\langle\rangle \mid !trigQ().Q.\overline{trigQ}\langle\rangle \mid P.\overline{trigP}\langle\rangle \mid \{o'/o\}Q.\overline{trigQ}\langle\rangle] \mid \\
&\qquad\qquad n_W[*] \mid n_S[*] \mid n_E[*] \xrightarrow{\overline{wifiConn}!\langle v', o', wifiPull\rangle} \\
&\text{(d) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[!trigP().P.\overline{trigP}\langle\rangle \mid !trigQ().Q.\overline{trigQ}\langle\rangle \mid \{v'/\texttt{getV()}\}\{o'/o\}P.\overline{trigP}\langle\rangle \mid \overline{trigQ}\langle\rangle] \mid \\
&\qquad\qquad n_W[!trigR().R.\overline{trigR}\langle\rangle \mid !trigT().T.\overline{trigT}\langle\rangle \mid \{v'/v\}\{o'/o\}R.\overline{trigR}\langle\rangle \mid T.\overline{trigT}\langle\rangle] \mid \\
&\qquad\qquad n_S[*] \mid n_E[*] \xrightarrow{\tau} \\
&\text{(e) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[!trigP().P.\overline{trigP}\langle\rangle \mid !trigQ().Q.\overline{trigQ}\langle\rangle \mid \{v'/\texttt{getV()}\}\{o'/o\}P.\overline{trigP}\langle\rangle \mid \overline{trigQ}\langle\rangle] \mid \\
&\qquad\qquad n_W[!trigR().R.\overline{trigR}\langle\rangle \mid !trigT().T.\overline{trigT}\langle\rangle \mid \{v"/v\}\{o'/o\}R.\overline{trigR}\langle\rangle \mid T.\overline{trigT}\langle\rangle] \mid \\
&\qquad\qquad n_S[*] \mid n_E[*] \xrightarrow{\overline{wifiPull}\langle r', v"\rangle} \\
&\text{(f) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[!trigP().P.\overline{trigP}\langle\rangle \mid !trigQ().Q.\overline{trigQ}\langle\rangle \mid \{v'/\texttt{getV()}\}\{o'/o\}\{r'/r\}P.\overline{trigP}\langle\rangle \mid \\
&\qquad\qquad \overline{trigQ}\langle\rangle] \mid \\
&\qquad\qquad n_W[!trigR().R.\overline{trigR}\langle\rangle \mid !trigT().T.\overline{trigT}\langle\rangle \mid \{v"/v\}\{o'/o\}R.\overline{trigR}\langle\rangle \mid T.\overline{trigT}\langle\rangle] \mid \\
&\qquad\qquad n_S[*] \mid n_E[*] \xrightarrow{\tau:r' \rightarrow a_m} \\
&\text{(g) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[!trigP().P.\overline{trigP}\langle\rangle \mid !trigQ().Q.\overline{trigQ}\langle\rangle \mid \overline{trigP}\langle\rangle \mid \overline{trigQ}\langle\rangle] \mid \\
&\qquad\qquad n_W[!trigR().R.\overline{trigR}\langle\rangle \mid !trigT().T.\overline{trigT}\langle\rangle \mid \{v"/v\}\{o'/o\}R.\overline{trigR}\langle\rangle \mid T.\overline{trigT}\langle\rangle] \mid \\
&\qquad\qquad n_S[*] \mid n_E[*] \xrightarrow{\overline{loraPush}\langle r', v"\rangle}
\end{aligned}
\tag{4.6}
$$

$$
\begin{aligned}
&\text{(h) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[*] \mid \\
&\qquad\qquad n_W[!trigR().R.\overline{trigR}\langle\rangle \mid !trigT().T.\overline{trigT}\langle\rangle \mid \overline{trigR}\langle\rangle \mid T.\overline{trigT}\langle\rangle] \mid \\
&\qquad\qquad n_S[!trigU().U.\overline{trigU}\langle\rangle \mid !trigV().V.\overline{trigV}\langle\rangle \mid \{^{r'}/_{r}\}\{^{v"}/_{v}\}U.\overline{trigU}\langle\rangle \mid V.\overline{trigV}\langle\rangle] \mid \\
&\qquad\qquad n_E[*] \xrightarrow{\overline{internetPush}\langle r', v"\rangle} \\
&\text{(i) } n_E \leftrightarrow\ n_S \mid n_S \leftrightarrow n_W \mid n_W \leftrightarrow n_M \vdash n_M[*] \mid n_W[*] \mid \\
&\qquad\qquad n_S[!trigU().U.\overline{trigU}\langle\rangle \mid !trigV().V.\overline{trigV}\langle\rangle \mid \{^{r'}/_{r}\}\{^{v"}/_{v}\}U.\overline{trigU}\langle\rangle \mid V.\overline{trigV}\langle\rangle] \mid \\
&\qquad\qquad n_E[!trigX().X.\overline{trigX}\langle\rangle \mid !trigY().Y.\overline{trigY}\langle\rangle \mid \{^{r'}/_{r}\}\{^{v"}/_{v}\}X.\overline{trigX}\langle\rangle \mid Y.\overline{trigY}\langle\rangle]
\end{aligned}
\tag{4.7}
$$

En (4.6) y (4.7) se muestra cómo el sistema evoluciona desde un nodo M reportando un obstáculo, hasta que la ruta actualizada llega a E. Para comprender mejor el flujo de información a lo largo del sistema de nodos, se han tenido en cuenta estos aspectos:

- Como los nodos tienen varios procesos ejecutándose simultáneamente, solo se muestran aquellos que tienen actividad, mientras que los otros nodos se escriben como $n_x[*]$, siendo x el nombre del nodo, lo que indica que no están realizando acciones.
- Solo se han utilizado nombres de procesos para abstraer los detalles de la implementación.
- Se han agregado funciones de activación a cada proceso y, por lo tanto, se pueden crear y ejecutar nuevas instancias de los mismos. El nombre de los activadores está compuesto por la prefijo *trig* (del inglés *trigger*) y el nombre del proceso.

La formalización del flujo de información comienza en 4.6(a) con la formulación de los nodos y los activadores. En la siguiente iteración del sistema -4.6 (b)-, todos los activadores se disparan para que cada proceso genere una nueva instancia y espere los datos a través de sus canales de entrada. En 4.6(c) un evacuado (es decir, un nodo M) presenta un obstáculo de o' a s_m. Luego, M encuentra a un W y se conecta por WiFi en 4.6(d). En 4.6(e) W calcula una nueva ruta (y un nuevo número de versión) basándose en la información recibida. El sistema luego itera hasta 4.6(f), donde W envía la nueva ruta y el número de versión a M, y este último actualiza la pantalla de la aplicación móvil a_m en 4.6(g). En este punto, la instancia del proceso P generado en 4.6(b) finaliza su ejecución en M.

El sistema continúa procesando y evoluciona a 4.6(h), donde S recibe de W la nueva ruta y el número de versión; así, la instancia del proceso R en W termina su ejecución. Finalmente, S envía la nueva ruta y el número de versión a E en 4.6(i). Ahora, E tiene una ruta actualizada que se generó considerando la información de un obstáculo presentada por un evacuado. A partir de este punto, el sistema está listo para continuar actualizando la información almacenada en todos los S y W que participan en el sistema de evacuación.

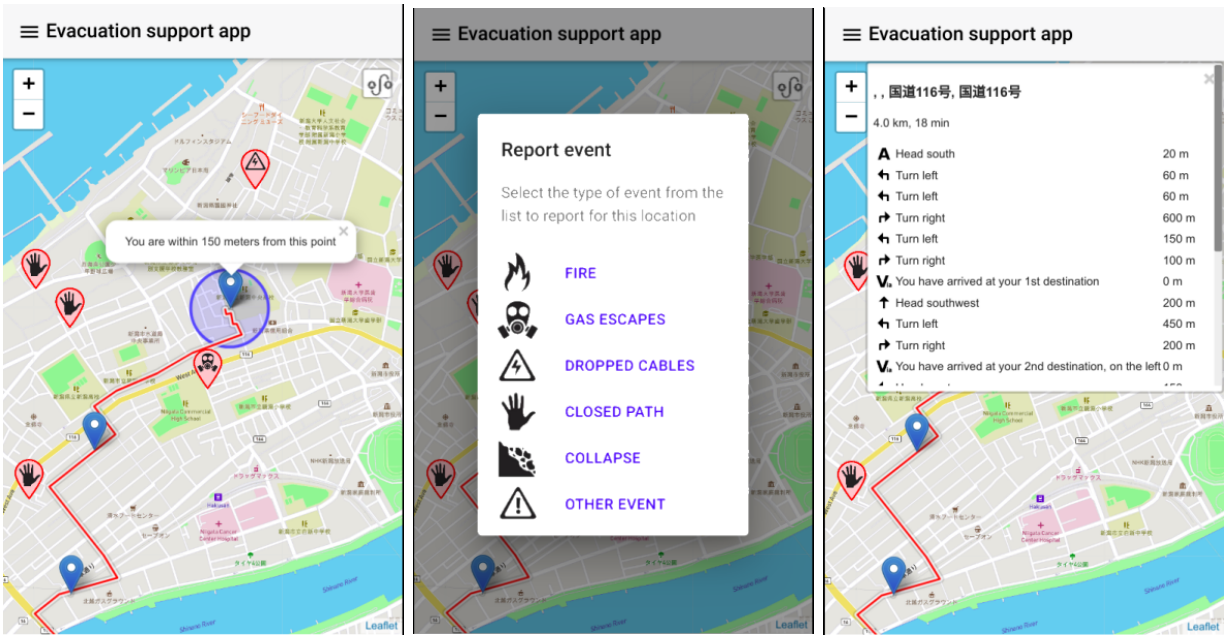
4.3.3. Implementación

El sistema propuesto se implementó como prueba de concepto en el Laboratorio de Sistemas Digitales de la Universidad Nacional del Sur, Argentina. En esta implementación, se utilizaron componentes y placas de desarrollo para realizar prototipado. Básicamente, el sistema requiere que se puedan implementar los cuatro roles ya definidos: unidad móvil, unidad testigo, refugio y el servidor en el centro de operaciones de emergencia. A continuación, presentamos una breve descripción de estos componentes.

4.3.3.1. Unidades Móviles

Estas unidades son los dispositivos habituales que utilizan los evacuados (por ejemplo, su teléfono inteligente) y representan los nodos M del sistema. Estos dispositivos ejecutan una aplicación de software diseñada para interactuar con la infraestructura (particularmente con unidades testigo) y así obtener soporte para el usuario durante la evacuación.

La aplicación muestra un mapa del área donde se encuentra el evacuado, y también varias marcas que incluyen la ubicación del usuario, la ruta de evacuación sugerida para llegar al refugio más cercano y obstáculos en el camino, como por ejemplo, cables eléctricos caídos, incendios y puentes inaccesibles (Figura 4.6 (a)).



(a) (b) (c)

Figura 4.6: Interfaces de usuario del Sistema de Apoyo de Evacuación: (a) ruta de evacuación, (b) interfaz para el informe de obstáculos e (c) indicaciones para llegar al refugio.

Mediante la aplicación, el usuario puede actuar como sensor humano durante el proceso de evacuación, proporcionando información sobre obstáculos aún desconocidos en el sistema. Al tocar en la pantalla del dispositivo la ubicación de un evento a informar, la aplicación muestra un menú con obstáculos comunes (Figura 4.6 (b)). Una vez agregado el obstáculo, el W actualiza la información localmente, descartando las rutas que pasan por esa zona y emitiendo la alerta a su S e indirectamente a E. El sistema también presenta indicaciones (distancia, dirección y tiempo de viaje estimado) para llegar al refugio, considerando la ruta de evacuación sugerida (Figura 4.6 (c)).

En las interfaces de usuario, la ubicación de los nodos W se muestra como marcadores azules, las ubicaciones de las zonas peligrosas se muestran como marcadores rojos con sus respectivos iconos y el camino rojo en la calle marca el camino sugerido que debe seguir un usuario. Cada marcador del mapa tiene un menú emergente con información detallada. En la sección superior derecha, la interfaz de usuario muestra un botón desplegable con instrucciones para el evacuado.

Esta aplicación utiliza las unidades testigo para implementar una guía a los evacuados a los refugios, como si se tratase de un "rastro de migas de pan" donde se debe llegar a un destino para luego verificar el rumbo del siguiente. En tal proceso, los nodos M se conectan alternativamente a varios W según la ubicación de las personas. Las acciones que realizan

los dispositivos móviles para mantener a los usuarios finales conectados a la infraestructura (y también para proporcionarles inteligencia ambiental) son invisibles para los evacuados. Por lo tanto, estos usuarios finales perciben la aplicación móvil como un sistema ubicuo que les ayuda a llegar a un refugio, independientemente de su ubicación actual. En este sentido, el adecuado despliegue de nodos W en la zona de evacuación juega un papel clave en la ubicuidad percibida por los usuarios finales durante el proceso de evacuación. Más adelante se analiza este aspecto y se propone un algoritmo para realizar esta implementación.

La aplicación móvil utiliza tecnologías basadas en la web, como por ejemplo la librería Leaflet para mostrar un mapa interactivo con rutas y marcadores. Los componentes GUI son proporcionados por la biblioteca de estilos de Framework 7 y la comunicación con los W se maneja a través de la biblioteca Axios, que es un cliente HTTP basado en aplicaciones para navegadores. Esta implementación permite distribuir la aplicación de muchas formas, sin modificar el código fuente. Una primera opción de distribución podría ser generar una aplicación independiente para computadoras móviles y de escritorio, que podría compilarse con varios frameworks, como Apache Cordova o Electron. Además, la aplicación también puede alojarse en un servidor web y distribuirse a través de HTTP, como la mayoría de los sitios web. Por último, también se puede distribuir como una aplicación web progresiva (PWA). Una PWA es un sitio web que se puede instalar como una extensión de un navegador y se ejecuta de forma autónoma, como si se tratase de una aplicación nativa.

4.3.3.2. Unidades Testigo

Los nodos W juegan un papel clave ya que los evacuados acceden al sistema de evacuación a través de ellos. Los usuarios móviles interactúan directamente con los W subiendo obstáculos en la ruta y descargando la ruta actualizada al refugio. Este tipo de nodo no solo actúa como un intermediario entre el sistema de apoyo a la evacuación y los evacuados, sino que también tiene la capacidad de calcular rutas alternativas basadas en información local. Hay dos interfaces de comunicación presentes en cada W: (a) un enlace LoRa que se usa para intercambiar información con el S asociado, y (b) un WiFi que se usa como punto de acceso a los M.

Los W se pueden construir usando una computadora Raspberry Pi 3B comercial. En ella, se implementa una interfaz LoRa con la placa TTGO LoRa32 SX1276, a través de la comunicación en serie. La figura 4.7 muestra una alternativa para implementar estas unidades testigo y sus interfaces para interactuar con unidades móviles y refugios.

La placa Raspberry Pi está configurada como un punto de acceso que proporciona una red WiFi abierta bajo el SSID de "WU_ID ". Esta red está configurada con un portal cautivo, por lo tanto, cuando el usuario móvil accede a la red WiFi, el navegador del dispositivo se abre automáticamente con una página web "personalizable" que muestra un mapa fuera de línea centrado en la ubicación actual del usuario. Esto ayuda a aumentar la sensación de ubicuidad percibida por el usuario.

Un servidor de WebSockets que se ejecuta en Raspberry Pi (es decir, en la unidad testigo) está a cargo de administrar el intercambio de información y la interacción con otros nodos. Cada obstáculo informado por un usuario se almacena en un archivo de formato JSON en la base de datos del dispositivo móvil. Cada vez que una unidad móvil se conecta a la red WiFi, un script del servidor fusiona las bases de datos locales y remotas a través de un

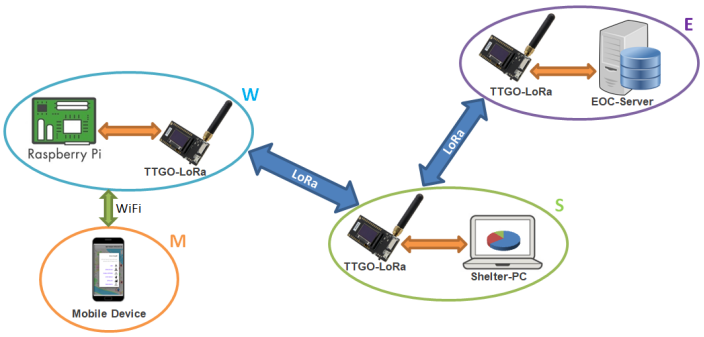


Figura 4.7: Esquemático del sistema.

proceso de sincronización de información. Un proceso de este tipo no es exigente en términos de computación y uso de memoria, ya que la información a sincronizar está estructurada, principalmente en base a representaciones de texto, y los algoritmos de sincronización pueden ser simples.

Todos los nuevos informes provenientes de los nodos móviles se ponen en una cola del W para transmitirlos al S correspondiente. Por lo tanto, los W actúan como puente entre los M y el S. Además, todos los mensajes de los W cercanos, que llegan a través de la comunicación LoRa, se transmiten al servidor Raspberry Pi a través de mensajes en serie y se almacenan en la base de datos local de la unidad testigo.

4.3.3.3. Refugios

El nodo que representa un refugio (S) se implementa mediante una computadora de escritorio con capacidad de procesar y almacenar la información del proceso de evacuación en su área de influencia. Dado que estas unidades están ubicadas en áreas seguras, están preparadas para enfrentar eventos extremos; por lo tanto, podemos suponer que los nodos S contarán con energía y servicios de comunicación durante el proceso de evacuación (o la mayor parte del mismo).

En la implementación actual, los S reciben información de los W en su área de influencia y cargan dicha información en el Centro de Operaciones de Emergencia (E) a través de un enlace LoRa. Los refugios cuentan también con acceso a Internet confiable (por ejemplo, Internet satelital) para respaldar la información en la nube mientras el servicio está disponible y comunicarse con el E.

Este tipo de nodo informa periódicamente su disponibilidad para recibir evacuados. Cuando se alcanza su límite de capacidad, informa de la situación a los W y E para desviar a más evacuados al refugio disponible más cercano. Los S también implementan un protocolo de persistencia en la comunicación (keep-alive) para detectar caídas en los enlaces con W y E.

4.3.3.4. Centro de Operaciones de Emergencias

Al igual que los S, el E está ubicado en un área segura, por lo tanto, se puede implementar utilizando un servidor con múltiples capacidades de comunicación. Aunque no es obligatorio, este servidor debería contar con un enlace satelital a Internet que le permita respaldar la información de evacuación en la nube.

Este nodo cuenta con un sistema de software que permite a los gestores de emergencias establecer y modificar rutas cuando sea necesario, y transmitir dicha información a los S y los W. El sistema permite también a estas personas monitorear el proceso de evacuación y tomar decisiones que se transmiten a los nodos en función del rol que desempeñan en la infraestructura propuesta.

4.3.3.5. Disponibilidad de la información

Las estrategias utilizadas para representar y compartir información a través de la infraestructura basada en IoT afectan la disponibilidad de la información y, por lo tanto, las actividades de toma de decisiones y la efectividad de los procesos de evacuación. Para mejorar la disponibilidad de la información, la infraestructura propuesta utiliza mensajes codificados (es decir, tipos de mensajes predefinidos) estructurados en formato JSON. Esto ayuda a reducir el tamaño del mensaje y también a comprender y sincronizar su contenido cuando llega a un nodo de destino. En consecuencia, reduce el consumo de energía y ancho de banda involucrado en la difusión de información.

La interacción entre nodos sigue la jerarquía de roles que se muestra en la Figura 4.1, y se realiza usando un paradigma maestro/esclavo, donde solo un nodo puede transmitir mensajes en un cierto intervalo de tiempo. Esta estrategia ayuda a evitar colisiones.

Para garantizar la equidad en las oportunidades de transmitir mensajes, E inicia la encuesta utilizando el algoritmo Round-Robin, dando control a el primer S (ver Figura 4.8). Este nodo reconoce la solicitud e indica en el mensaje cuántos W están trabajando en su área de influencia, y también el número de evacuados alojados en el refugio. Luego, el S inicia el sondeo de los W también de forma rotativa, y cada W responde con uno o más mensajes dependiendo de la cantidad de nuevos eventos u obstáculos a reportar.

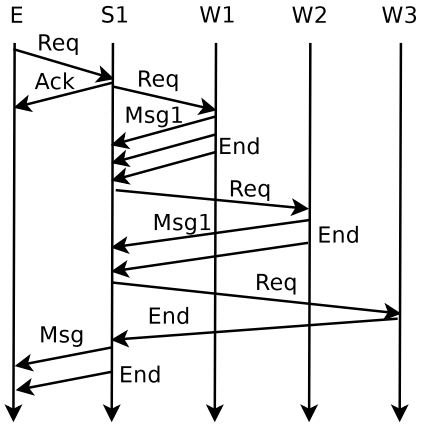


Figura 4.8: Proceso de intercambio de información.

La respuesta de un W es un objeto estructurado JSON que contiene tres campos: la identificación del nodo, los datos del evento y el mensaje de terminación (verdadero/falso). El campo de datos de eventos es una matriz que puede contener hasta 5 eventos (una carga útil de 220 bytes) y la cantidad de nodos M que atraviesan ese W. Cada evento está representado por su ubicación (latitud y longitud), un identificador que representa el tipo de evento u

obstáculo, una clave para guardarlo en la base de datos y una marca de tiempo que indica el momento en el que se registró. El W filtra eventos duplicados cargados por los nodos M.

El último mensaje entre un W y un S incluye un campo de terminación que cierra la interacción entre estos nodos. Una vez que el S recibe este mensaje de finalización, utiliza la operación por turnos para determinar el próximo W para sondear. Cuando se han sondeado todos los W, el S envía la información actualizada al E y cierra la comunicación. Después de esto, la E inicia la interacción con el siguiente S y se repite el proceso.

En escenarios de evacuación, la disponibilidad de información también está relacionada con la capacidad del sistema de apoyo para cubrir el área física donde se encuentran los evacuados. Por lo tanto, la tecnología de comunicación utilizada para implementar los enlaces entre nodos, y también el algoritmo utilizado para implementar los W, juegan un papel clave. Las siguientes secciones abordan estos dos aspectos.

4.3.3.6. Tecnologías de comunicación

En cuanto a los enlaces de comunicación entre los nodos E, S y W, se implementaron mediante tecnología LoRa, y la comunicación entre los nodos W y M se implementó mediante WiFi. Los transceptores LoRa se configuraron para tener el mayor rango de comunicación posible en entornos urbanos y, por lo tanto, permitir a los evacuados un fácil acceso a la información en la ruta de evacuación. En particular, la red LoRa se configuró con los siguientes parámetros: el Spreading Factor (SF) se establece en 12, el ancho de banda (BW) en 125 KHz y la tasa de codificación (CR) en 5.

La implementación de la infraestructura propuesta implica el uso de componentes simples que están disponibles en el mercado local, son confiables y su costo es asequible. Estas características hacen que la propuesta sea factible de ser implementada en países subdesarrollados o en desarrollo. Además, la decisión de utilizar dispositivos móviles (en particular, teléfonos inteligentes) para iniciar sesión en la infraestructura de soporte permite a los evacuados tener un acceso masivo y ubicuo a la información relevante durante las evacuaciones, y contribuye a la adopción del sistema por parte de los civiles.

Con respecto al despliegue del sistema, y considerando que este es un problema NP-Hard, la siguiente sección describe el algoritmo para desplegar los nodos de la infraestructura propuesta.

4.3.3.7. Despliegue de la infraestructura

La infraestructura fue diseñada para ayudar a proporcionar inteligencia ambiental al entorno de evacuación. Por lo tanto, debe brindar un apoyo efectivo para que los evacuados cuenten con una gran cantidad de nodos W, convenientemente desplegados, para orientar y encaminar el proceso de evacuación hacia los S. Para esto, la ubicación de los nodos W debe permitir que el sistema implemente senderos como “migajas de pan” que puedan ser seguidos fácilmente por los evacuados mientras se mueven hacia áreas seguras. Cuántos W implementar y dónde ubicarlos es un problema de cobertura típico que ha demostrado ser NP-Hard [62].

En [63] se introdujo un algoritmo voraz (greedy en inglés) para abordar este problema, y en [30] se propuso un enfoque metaheurístico basado en un algoritmo genético. La idea

general para desplegar los W es colocar las unidades de tal manera que un evacuado (es decir, una unidad móvil) tenga una distancia de cobertura máxima de D entre dos de ellos. Así, cuando un M se mueve de un W a el siguiente en su ruta de evacuación, el tiempo transcurrido sin el apoyo del sistema es mínimo.

La distancia se puede medir en metros o en el tiempo promedio que necesita un evacuado para moverse entre los dos W. La forma adecuada de medir la distancia depende del tipo de evacuación que se apoye, ya que no es lo mismo si el M está caminando o usando un vehículo (scooters, autos, camiones), y también si está subiendo en una colina o en un territorio llano. Como la distancia es una variable que depende de las condiciones que rodean la ciudad en la que se desplegará el sistema AmI, simplemente nos referimos a ella de forma abstracta (sin unidad de medida).

Para determinar un número adecuado de unidades de testigos y sus ubicaciones para maximizar la cobertura del sistema AmI, la ciudad con un área A se divide en regiones $\rho = R_1, \ldots, R_n$, de modo que $\forall i \neq j \ R_i \cap R_j = \phi$ y $\cup_{\forall i} R_i = A$. Por lo general, las ciudades tienen una gran red de calles que ayuda a reducir estos problemas a regiones más pequeñas. En cada región, se debe construir un grafo $G(\nu, \xi)$, donde cada nodo $x_i \in \nu$ representa una intersección de calles y las aristas ξ_{ij} indican la unión entre x_i y x_j. El peso $w(\xi_{ij})$ representa la distancia, $\xi_{ij} = \xi_{ji}$. Finalmente, se debe definir una distancia máxima deseable entre dos W, D.

El algoritmo de ubicación de nodos W es bastante sencillo. La distancia máxima D debe configurarse como parámetro. Teniendo en cuenta este valor y el área de la región de despliegue, se puede calcular una cuadrícula regular para lograr una distribución equidistante de nodos W. Los nodos en el mapa que están cerca de cada punto de la cuadrícula regular se seleccionan como ubicaciones para nodos W y, por lo tanto, es posible obtener una distribución tentativa en la región.

Esta solución se puede mejorar si se utiliza esta distribución como condición inicial de un método de optimización, como el propuesto en [30]. Sin embargo, la simplicidad de este algoritmo de implementación representa una fortaleza.

El algoritmo se ejecutó para determinar el despliegue de infraestructura en la ciudad de Bahía Blanca, en Argentina. La forma del terreno hace que esta ciudad sea propensa a las inundaciones. Además, la ciudad alberga una gran zona industrial petroquímica y un puerto donde funciona una estación de transferencia de GLP (gas licuado de petróleo) con transportistas. Por lo tanto, se requiere un sistema de apoyo a la evacuación de la ciudad.

El mapa de la ciudad de Bahía Blanca se descargó de OpenStreetMap[1], y el modelo de la calle se extrajo de los datos de la misma plataforma y se convirtió en un gráfico completamente conectado utilizando las herramientas de Octave[64]. La Figura 4.9 muestra el mapa original de la ciudad a la izquierda y su representación gráfica a la derecha. D se estableció en 500 metros.

La figura 4.10 muestra las ubicaciones donde se deben implementar los W de acuerdo con el algoritmo propuesto. Los puntos negros indican la ubicación de los W. Los círculos rojos muestran una posible agrupación de W, y la ubicación de S se indica con un punto verde. Aproximadamente 40 W están en el área de influencia de un S. Como puede verse en algunos casos, los círculos (es decir, el área de influencia de S) se superponen; sin embargo,

[1](https://www.openstreetmap.org/)

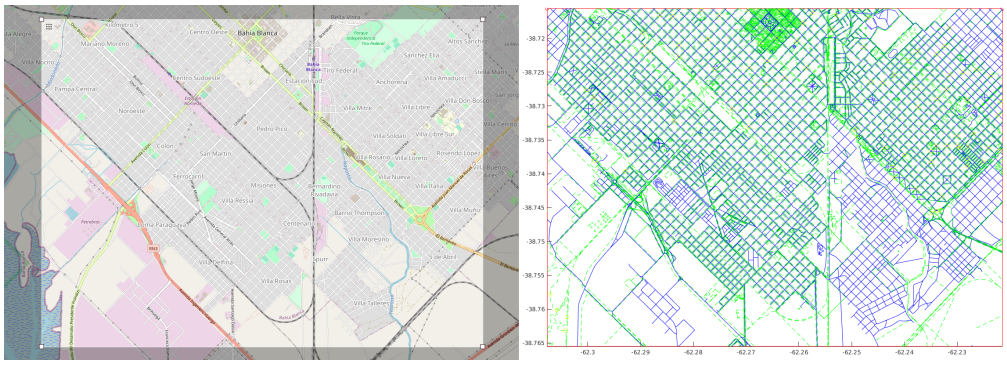


Figura 4.9: Mapa de la ciudad de Bahía Blanca.

en esa situación, los W están vinculados a un solo S.

La distribución de W sobre el mapa de la ciudad no representa una cuadrícula perfecta, ya que el patrón de calles en algunas áreas no es regular. En estos casos, la distribución de distancias mínimas entre nodos W vecinos da como resultado una distribución normal, como se muestra en la Figura 4.11.

4.3.4. Evaluación

La evaluación del sistema se dividió en dos pasos: la recolección de datos experimentales y la simulación del sistema. Las siguientes secciones explican las actividades llevadas a cabo en estos pasos y los resultados obtenidos.

4.3.4.1. Recolección de datos experimentales

La recopilación de datos experimentales implicó la instalación de un nodo que representa un refugio, que se ubicó en un lugar a 30 m de altura. Dicho nodo estaba vinculado a una unidad testigo mediante un enlace LoRa. Los transceptores LoRa se configuraron como se indicó previamente, es decir, con SF=12 y BW=125KHz.

La unidad testigo se colocó en seis puntos diferentes de la ciudad, mezclando áreas residenciales y comerciales, y también considerando diferentes distancias entre el refugio y la unidad testigo. En todos los casos, la unidad testigo se instaló en una ubicación de 1,6 m de altura. El Cuadro 4.1 presenta los resultados obtenidos para las variables RSSI (indicador de fuerza de la señal recibida) y SNR (relación señal/ruido).

Distancia [m]	RSSI [dBm]	SNR [dB]
2000	-113	-12.95
1800	-119	-10.50
1300	-116	-14.80
1000	-111	-10.53
700	-118	-9.55
500	-115	-16.50

Cuadro 4.1: Valores de RSSI y SNR a diferentes distancias entre el refugio y la unidad testigo.

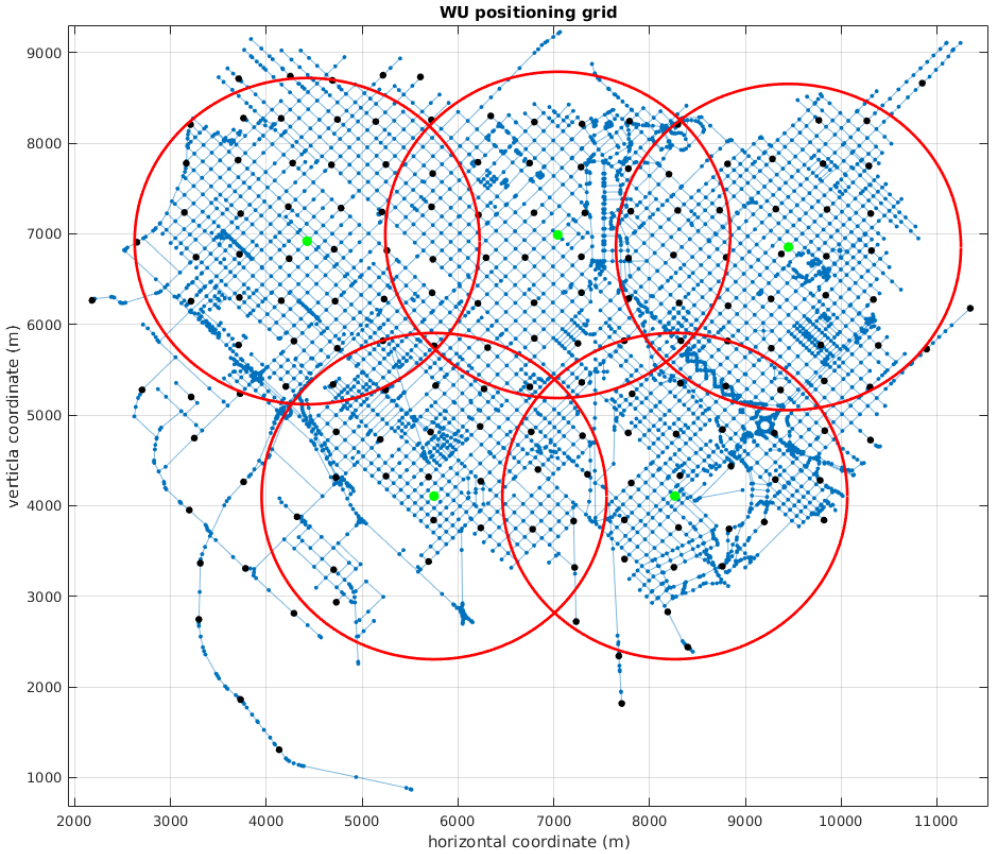


Figura 4.10: Mapa para el despliegue de nodos W en la ciudad de Bahía Blanca, considerando $D = 500$m

Para cada ubicación de la unidad testigo, se realizaron cinco experimentos para informar 1, 5, 10, 15 y 20 eventos, lo que representa 11 mensajes en total. La última solicitud del refugio fue respondida con 4 mensajes para brindar los 20 eventos. En todos los casos, el retardo de transmisión del enlace fue de 33,01 segundos, con un rendimiento efectivo de 26,9 [Bytes de carga útil]/[s]. La tasa total de fallos de las solicitudes fue del 9 %. Estos números ilustran la capacidad de los enlaces de red para entregar mensajes con una baja tasa de errores, considerando las interferencias propias de una ciudad.

Con el fin de analizar el desempeño potencial de un sistema de apoyo a la evacuación basado en la infraestructura propuesta, simulamos el despliegue de 100 unidades de testigos en el área urbana de la ciudad de Bahía Blanca. Los valores obtenidos a través de la recolección de datos experimentales se utilizaron para modelar la infraestructura de la red, y las simulaciones se implementaron usando Octave.

Con respecto a la idoneidad de este despliegue para soportar una evacuación masiva real, es importante notar que cada unidad testigo puede proporcionar acceso simultáneo a 30-40 personas. Esto significa que, a través del despliegue simulado, entre 3000-4000 evacuados pueden obtener simultáneamente información sobre sus rutas de evacuación en pocos segun-

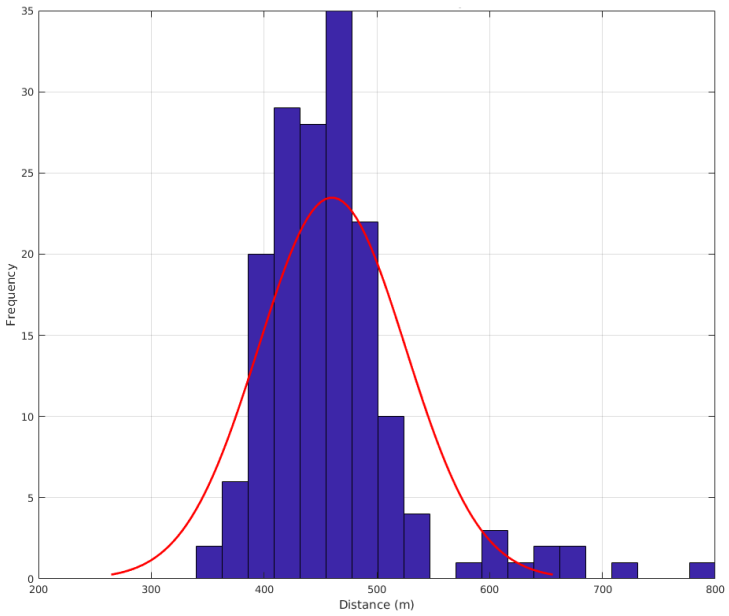


Figura 4.11: Distribución de distancias entre nodos vecinos usando $D = 500$m

dos. Como se mencionó, la información de las rutas de evacuación es liviana, fácil de fusionar y se gestiona localmente en el dispositivo móvil de los evacuados, por lo que las unidades testigo pueden informar las rutas de evacuación a cientos de miles de personas en unos pocos minutos. A continuación, se explica el proceso de simulación y los resultados obtenidos.

4.3.4.2. Simulación del sistema

Para cada unidad testigo, se simuló el número de eventos que se reportarán (es decir, mensajes que se enviarán al refugio) con un generador de Poisson aleatorio. Se eligió esta distribución porque modela eventos aleatorios con cierta probabilidad de ocurrir en un determinado intervalo de tiempo o área, siguiendo una distribución exponencial que no tiene memoria. La tasa de la distribución de Poisson varió entre 1 y 10, lo que equivale a informar entre 1 y 50 eventos por unidad testigo.

El sistema se simuló 100 veces para cada tasa posible y se promediaron los resultados. Esto representa un escenario de simulación en el que la red está estresada. Más mensajes en la red implica más tiempo de espera para ser seleccionados para ser enviados, hasta que el incremento en el tráfico satura la capacidad de la red de comunicaciones.

La figura 4.12 muestra la latencia en el acceso a la red. Como puede verse, el retraso es proporcional a la velocidad a la que se generan los mensajes en las unidades testigo. Como la política de programación propuesta es una simple operación por turnos, el retraso promedio aumenta cuando se deben enviar más mensajes.

La figura 4.13 ilustra el rendimiento (throughput) de la red medido en [bytes de carga útil]/[s]. La variación absoluta no es significativa en función del índice de generación de eventos, menor al 1 %. Esto se debe a que cuando la tasa es baja, la cantidad de información a enviar también es baja; y cuando se incrementa la tasa, también se incrementa el retardo

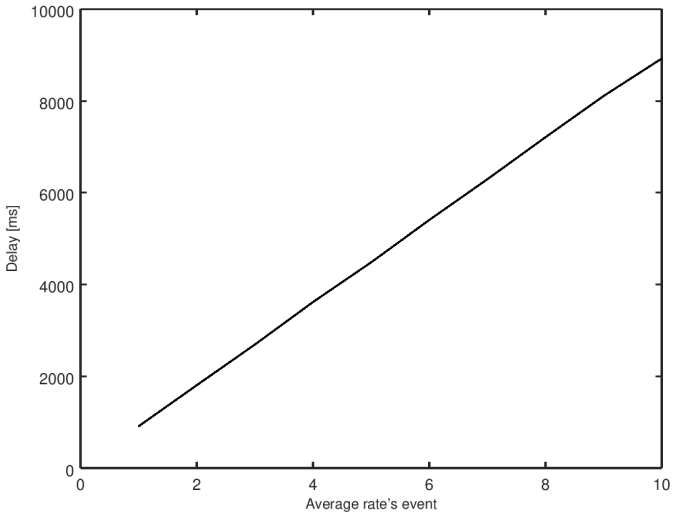


Figura 4.12: Retraso de transmisión en una infraestructura con 100 unidades testigo.

en la red, por lo tanto, la relación final es similar. Finalmente, cuando la red está saturada, el rendimiento es independiente del número de eventos, ya que no hay más ancho de banda disponible.

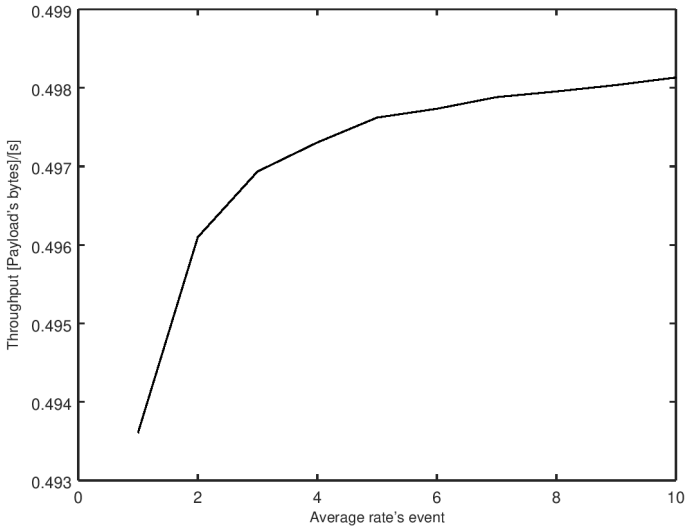


Figura 4.13: Rendimiento (throughput) de una infraestructura con 100 unidades testigo

Teniendo esto en cuenta, es posible predecir el comportamiento del sistema en función de la cantidad de unidades testigo necesarias para respaldar las evacuaciones masivas en una ciudad o área en particular. Claramente, existe una compensación entre el número de estas unidades desplegadas y el rendimiento final de la red.

Se pueden implementar varias alternativas para mantener el rendimiento en niveles razonables y, al mismo tiempo, proporcionar buenos senderos para los evacuados. Esto se puede

lograr permitiendo que algunas unidades testigo reporten eventos a los refugios utilizando políticas de prioridad dinámica para sondear las unidades testigos. Además, el sistema debería permitir el sondeo simultáneo de unidades testigo que no se encuentran dentro del rango de comunicación, evitando colisiones. Se puede hacer en casos como el de la Figura 4.10, donde hay cinco refugios y aproximadamente 170 unidades de testigos que pueden emparejarse para mejorar el tiempo de respuesta. De esta manera, la región superior izquierda se puede planificar simultáneamente con la inferior derecha, mientras que la zona superior derecha se puede planificar junto con la inferior izquierda. Finalmente, la región central se planifica para realizarse sola, ya que tiene regiones superpuestas con las otras cuatro. De esta manera, el retardo promedio se reduce significativamente y el rendimiento se incrementa ya que dos refugios reciben información actualizada de manera simultánea.

Las capacidades que muestra el sistema simulado son consecuencia del uso de la infraestructura propuesta, pero también del conjunto de decisiones de diseño que ayudan a incrementar los límites y la cobertura de dicha infraestructura. En particular, la tecnología utilizada para vincular los nodos, la densidad y ubicación de las unidades testigo desplegadas en el campo, el tipo de información intercambiada entre los componentes y el tipo de interacción permitida entre los nodos, son aspectos de diseño que deben abordarse durante la etapa de diseño, ya que tienen consecuencias directas sobre el rendimiento del sistema.

4.4. Consideraciones finales

En este cuarto capítulo se propuso una infraestructura basada en IoT para brindar apoyo a civiles y organizaciones de respuesta temprana durante procesos de evacuación a gran escala. La misma las interacciones implícitas y explícitas entre el sistema AmI y los evacuados.

La formalización de esta infraestructura se realizó mediante una adaptación de IoT-Calculus, que permite modelar estos sistemas y también evaluar su desempeño en tiempo de preparación. Por lo tanto, el diseño de los sistemas de apoyo a la evacuación puede ajustarse en función de los resultados de la simulación e implementarse solo después de que hayan demostrado ser adecuados desde una perspectiva teórica.

También se propuso un algoritmo de implementación simple para determinar las ubicaciones adecuadas de W en un área urbana y, por lo tanto, proporcionar inteligencia ambiental a los evacuados mientras son guiados a los refugios. El camino a los refugios requiere que las unidades móviles utilizadas por los evacuados se conecten a diferentes W mientras las personas cambian de un lugar a otro. Estas interacciones entre los componentes del sistema son invisibles para los usuarios finales, quienes perciben la aplicación móvil como un sistema ubicuo que les permite acceder a información sobre su ruta de evacuación en cualquier momento y lugar. En este sentido, la densidad de nodos W, la tecnología utilizada para conectar los componentes del sistema y el tipo de información intercambiada entre estos nodos juegan un papel clave. Esto se debe a que la ubicuidad percibida por los usuarios finales y el límite de colapso del sistema debido a la sobrecarga de información, dependen de estos aspectos.

La infraestructura propuesta implementa un protocolo de persistencia de conexión que detecta fallas en los componentes del sistema, y mecanismos para cambiar dinámicamente el comportamiento de los nodos en caso de que ocurran. Este cambio de comportamiento también considera situaciones en las que un nodo vuelve a funcionar normalmente después

de un colapso temporal.

Se implementó una prueba de concepto del sistema basado en placas de desarrollo. Utilizando datos recopilados de un escenario real, se modeló una infraestructura particular para soportar evacuaciones masivas en el área urbana de la ciudad de Bahía Blanca, en Argentina. La capacidad de ese sistema se evaluó mediante simulaciones y los resultados muestran no sólo la viabilidad de la propuesta, sino también su capacidad para afrontar el proceso de evacuación incluso cuando la red está sobrecargada.

Aunque la infraestructura propuesta fue diseñada particularmente para apoyar la inteligencia ambiental en áreas de evacuación urbana, podría usarse con otros propósitos si los sistemas consideran restricciones de implementación particulares; por ejemplo, el despliegue de W debe garantizar la cobertura del servicio en toda el área de destino, y la información intercambiada por los nodos de la infraestructura debe minimizar los colapsos debidos a la sobrecarga de información. En este sentido, se requiere una validación más experimental de la infraestructura propuesta para determinar sus límites, no sólo para soportar procesos de evacuación a gran escala, sino también para ser aplicada en otros dominios de aplicación.

Parte III

Propuestas de extensión de protocolos IoT

Capítulo 5

MQTT con capacidad de Tiempo Real Blando

5.1. Motivación

Mobile Service Computing (MSC o informática de servicios móviles en español) es un nuevo paradigma que fusiona los paradigmas de la informática de servicios y la informática móvil. Deng y col. [65] identifican tres macroescenarios (o patrones de implementación) para proveer y consumir servicios de software: nube a móvil (cloud to mobile o C2M), móvil a móvil (mobile to mobile o M2M) y una combinación de los anteriores (híbrido). En el primer escenario, los servicios se implementan en la nube o en servidores y son consumidos por usuarios a través de un dispositivo móvil. La nube actúa como un intermediario que almacena y procesa datos y también proporciona servicios que permiten a los usuarios de móviles actualizar y consumir dichos datos. Ejemplos de aplicaciones que utilizan este patrón de implementación son Waze y Foursquare. Este patrón es ampliamente utilizado hoy en día ya que tiene varios beneficios, entre ellos su simplicidad para estructurar aplicaciones, pero requiere de un enlace de comunicación estable entre la nube y las unidades móviles.

En el segundo escenario (M2M), cada dispositivo móvil realiza interacciones punto a punto con otros dispositivos sin utilizar unidades intermediarias obligatorias como servidores o la nube. En este escenario, los dispositivos consumen y brindan datos y servicios de manera ad hoc, generalmente utilizando redes oportunistas. Ejemplos de aplicaciones que utilizan el patrón de implementación M2M son aquellas que interactúan con asistentes digitales, como Amazon Echo o Google Home, donde la comunicación entre los dispositivos es ad-hoc y peer-to-peer. Siempre que, en escenarios M2M, el soporte de comunicación no dependa de tener una infraestructura de comunicación disponible (por ejemplo, redes 4G o Wi-Fi), las aplicaciones que utilizan este patrón de implementación tienden a ser más robustas en términos de capacidad de interacción ad-hoc. Este patrón aprovecha los enlaces de comunicación estables, pero no necesariamente les exige que presten o consuman servicios. Por esa razón, los enfoques M2M se utilizan con frecuencia en aplicaciones que respaldan procesos de respuesta a emergencias o actividades de colaboración en áreas sin comunicación basada en infraestructura.

Finalmente, los escenarios híbridos son aquellos en los que partes del sistema usan in-

teracciones C2M y otras partes usan configuraciones M2M. Este escenario heterogéneo es el más representativo en Internet de las cosas. Por ejemplo, en un sistema de vigilancia del hogar, los sensores y actuadores utilizan regularmente un enfoque M2M para interactuar con la unidad de control que detecta intentos de entrada no autorizados. Normalmente, esta unidad de control utiliza un escenario C2M para enviar alertas a la policía, las empresas de seguridad y los habitantes de esa casa.

Se han identificado varios desafíos en la implementación de MSC en entornos híbridos, por ejemplo, seguridad, comunicación, eficiencia y conciencia del contexto o "context-awareness" [65]. En particular, para una amplia variedad de aplicaciones basadas en IoT, es necesario analizar datos y dar soporte a interacciones en tiempo real, ya que generalmente implican monitoreo, toma de decisiones autónomas y entrega de notificaciones tempranas que representan actividades con restricciones de tiempo [66] [66] [67] [68] [69]. Si el sistema no reacciona a tiempo, su eficacia está en riesgo. Este requisito se hace evidente en los sistemas de IoT que detectan caídas de adultos mayores que viven solos o en aquellos que entregan alertas tempranas de peligros naturales.

Dar soporte a interacciones en tiempo real ha sido abordado por varios trabajos de investigación, pero las propuestas actuales asumen la disponibilidad y estabilidad del vínculo entre los componentes del sistema. Aunque eso está bien para muchos dominios de aplicaciones, no lo es para otros donde el soporte de comunicación es incierto, por ejemplo, en sistemas de alerta temprana de peligros naturales que generalmente involucran varias redes públicas y privadas para detectar y notificar eventos extremos a la población (por ejemplo inundaciones, incendios forestales, deslizamientos de tierra, tsunamis y erupciones volcánicas). Siempre que estas acciones se realicen lo más rápido posible, se requiere un soporte de interacción blando en tiempo real en una red con estabilidad incierta [70] [71].

Este capítulo aborda el soporte de interacción en tiempo real para dispositivos de un sistema de IoT orientado a servicios, que utiliza enlaces de comunicación inestables. Este desafío de comunicación no ha sido abordado lo suficiente en la literatura, particularmente cuando el proveedor de servicios y el consumidor interactúan a través de un enlace de comunicación con estabilidad incierta. En entornos de IoT, este tipo de esquema de interacción ocurre frecuentemente debido a diferentes razones, como la gran variedad de interfaces de red utilizadas o la cantidad de soluciones autónomas que coexisten e interoperan en un mismo entorno. Después de una extensa revisión de la literatura, no se ha encontrado ningún modelo de comunicación capaz de soportar interacciones en tiempo real entre componentes de un escenario de IoT que considere enlaces de comunicación inestables.

La principal contribución de este capítulo es la introducción de una arquitectura MSC, denominada Software Real-Time Interaction (SRTI), que involucra los roles que juegan los diferentes dispositivos participantes (publicadores, suscriptores y brokers) y también una dinámica entre roles para proporcionar y consumir estos servicios de software dentro de las restricciones temporales. Esta arquitectura facilita la inclusión de marcas de tiempo para manejar el tráfico en tiempo real blando, incluso en condiciones de redes inestables. Para probar el concepto, se presenta una librería que extiende el proyecto Mosca [72], y cuenta también con una interfaz gráfica para definir las funciones de agregación y procesamiento de los publicadores. Esta propuesta fue evaluada mediante simulaciones y los resultados obtenidos son muy prometedores.

5.2. Estado del arte

Como se mencionó anteriormente, la mayoría de las soluciones IoT implican restricciones de tiempo para recopilar y procesar información, tomar decisiones y enviar acciones que los componentes del sistema deben realizar. Cuando existen restricciones de tiempo, se dice que el sistema es de tiempo real si al menos una de las tareas a realizar debe ejecutarse antes de una fecha límite determinada. Esta restricción es difícil de cumplir cuando los enlaces de comunicación son inestables o si están sujetos a grandes variaciones de rendimiento debido a la carga de tráfico.

A continuación, se presenta una revisión de trabajos relacionados que consideran limitaciones de tiempo en los tres aspectos involucrados en esta propuesta: modelos estructurales de IoT, aplicaciones de IoT y tecnologías y modelos de procesamiento de datos para soportar las aplicaciones.

5.2.1. Modelos estructurales de IoT

En su artículo fundamental, Deng et al. [65] presentan y discuten las oportunidades y desafíos para proporcionar MSC en tres escenarios, pero particularmente en aquellos que se adhieren a patrones de implementación híbridos, que tienen una estructura similar a los escenarios de IoT. Entre estos desafíos, los autores plantean la necesidad de estudiar los aspectos de la comunicación que inciden en la prestación y el consumo del servicio, particularmente cuando el enlace de comunicación es inestable o incierto, ya que impacta en la disponibilidad de estos servicios. En dicho artículo, los autores presentan tres escenarios de interacción: C2M, M2M y una mezcla de ellos denominada híbrida. La necesidad de mantener la conexión, incluso cuando algunos de los dispositivos participantes están en movimiento, es un problema común en estos entornos de interacción, ya que determina la capacidad real del sistema para admitir interacciones en tiempo real.

La mayoría de los modelos de interacción de IoT se basan en dos enfoques. Uno de ellos involucra un mecanismo de publicación-suscripción en el que algunos dispositivos actúan como proveedores de datos publicando tópicos específicos en nodos especiales llamados brokers, mientras que otros actúan como consumidores de esos tópicos suscribiéndose a los servicios proporcionados por los brokers. Este enfoque se implementa mediante protocolos como Message Queue Server Telemetry Transport (MQTT) [9]. El otro enfoque está relacionado con una estructura cliente-servidor más clásica en la que ciertos nodos solicitan datos, mientras que otros responden a la solicitud, utilizando un protocolo muy simple y ligero como el Constrained Application Protocol (CoAP) [10]. Ninguna de estas estrategias admite interacciones en tiempo real.

En cuanto a arquitecturas o infraestructuras orientadas a soportar interacciones IoT, se presenta una situación similar a los trabajos anteriores. Por ejemplo, una arquitectura distribuida escalable para IoT se informa en [73], donde los autores proponen un modelo de comunicación integral que no contempla restricciones de tiempo. De manera similar, en [74], los autores introducen un enfoque de cálculo de procesos para formalizar una infraestructura de comunicación de IoT. Aunque esta propuesta aborda varios aspectos del intercambio de mensajes, no se tienen en cuenta las consideraciones en tiempo real.

En [75] se presenta una comunicación cognitiva M2M para sistemas de IoT utilizando

una perspectiva de pila; sin embargo, el tiempo no se incluye como variable crítica dentro del análisis de la propuesta. En [76], los autores utilizan el marco de componentes BIP (del inglés comportamiento, interacción y prioridad) [77] para modelar y reforzar la correctitud de las aplicaciones de IoT con recursos limitados. Al igual que en los trabajos anteriores, las limitaciones de tiempo no se consideran como un valor crítico dentro de la evaluación.

Siguiendo un enfoque diferente, Aziz [78] propone un método formal para representar y analizar una infraestructura de comunicación de IoT. La exactitud de la representación del modelo de comunicación se evalúa mediante métodos formales, y la implementación del sistema utiliza un paradigma de publicación-suscripción analizado con base en el protocolo MQTT [9]. Aunque en esta propuesta las limitaciones de tiempo se consideran una variable, el autor no proporciona un mecanismo para validar los requisitos de tiempo del sistema.

En [79], los autores propusieron una extensión de MQTT para ejecutar sobre redes oportunistas o redes intermitentes para aplicaciones de IoT. En esta situación, los nodos utilizan el mecanismo de almacenamiento y reenvío, ya que la transmisión sólo es posible en el momento en que hay una conexión entre los nodos. Los autores no analizan el comportamiento del sistema desde un punto de vista de tiempo real.

5.2.2. Aplicaciones basadas en IoT

Varios investigadores destacan la necesidad de ampliar el conocimiento en redes de sensores inalámbricos y computación en la nube, como instrumentos que den soporte a las aplicaciones de software de IoT, y comprender el papel que desempeña el big data, el intercambio de información y la colaboración para la prestación de servicios basados en IoT [65] [80]. En particular, el intercambio de información y la colaboración en estas infraestructuras imponen varios desafíos a los diseñadores de sistemas. Dependiendo del servicio que será provisto, se puede necesitar dar soporte a interacciones en tiempo real.

En [81], los autores presentan varias aplicaciones robóticas asistidas por IoT que operan con restricciones temporales. Sin embargo, estos sistemas utilizan redes locales (LAN) o dedicadas; por lo tanto, su propuesta de brindar soporte de interacción en tiempo real entre los componentes del sistema no considera la inestabilidad potencial de los enlaces de comunicación. En este caso, los retrasos en los enlaces de comunicación se analizan con instrumentos tradicionales en tiempo real, ya que la red está dedicada a la aplicación. Un trabajo similar se informa en [82], donde los autores proponen un módulo de clasificación inalámbrico de bajo costo para entornos industriales, también basado en redes dedicadas.

En [83], los autores introducen el concepto de nube de detección, donde Internet se utiliza como la principal red de comunicación y la nube representa el lugar donde los datos se pueden almacenar y recuperar para su procesamiento y uso. Aunque esta propuesta presenta una arquitectura genérica y muestra una implementación de la misma, no hay una consideración en tiempo real en el modelo; por lo tanto, no se puede utilizar para apoyar la interacción en el escenario estudiado.

Una propuesta similar se describe en [83], donde los autores presentan un método de monitoreo ubicuo para respaldar los servicios médicos de emergencia basados en IoT. En este escenario, la información recuperada de los dispositivos de detección se comparte a través de Internet, pero la propuesta no considera garantías en tiempo real en el proceso de interacción. Para el mismo dominio de aplicación, Koley y Ghosal proponen un sistema de

seguimiento de ubicación y comunicación de emergencia que ayuda a reducir los daños en emergencias vehiculares [84]. Aunque este sistema aborda interacciones en tiempo real, no considera formalmente la posibilidad de tener enlaces de comunicación inestables entre sus componentes.

La agricultura de precisión (o agronómica) también es un área en la que se requiere una comunicación en tiempo real blando. En [85] [86], los autores describen el uso de tecnologías de la información para mejorar la productividad de las granjas utilizando sistemas de soporte de decisión que se alimentan con datos provenientes de sensores.

La introducción de nuevas funcionalidades en la tecnología celular, como el protocolo LTE, promueven el desarrollo de nuevas aplicaciones en el campo de las comunicaciones de máquina a máquina. En este sentido, en [87] [88], los autores analizan el desempeño de una red Long-Term Evolution Advanced (LTE-A) para aplicaciones en la comunicación Vehículo-Infraestructura (V2I) y Vehículo-Vehículo (V2V). Además, proponen un mecanismo para que vehículos con baja relación señal / interferencia más ruido (SINR) puedan ser atendidos por otros vehículos, que tienen un enlace de calidad mucho mayor.

En [57], los autores implementaron un simulador llamado LTEV2Vsim, que permite gestionar la movilidad de los vehículos y realiza la asignación de paquetes de comunicación, siguiendo diferentes algoritmos controlados por red y autónomos. La herramienta resulta interesante para representar escenarios de IoT con alta movilidad pero considera enlaces de comunicación estables entre los nodos.

Con respecto a la forma en que los dispositivos de IoT se conectan a Internet, se puede hacer de varias maneras. Muchas veces, estos dispositivos forman parte de redes privadas que tienen acceso a redes abiertas a través de una puerta de enlace. En estos casos, hay varias opciones en el protocolo de la capa de enlace. Por ejemplo, para 802.11, es posible usar LoRa [89], Sigfox [12] y el diseñado para redes vehiculares como 802.11p [90]. Otras veces, los dispositivos pueden estar conectados a Internet mediante una red 4G. En un futuro próximo, existe una discusión abierta sobre el papel que tendrá el próximo protocolo 5G en relación con las redes vehiculares [91] y cómo interactuarán con otros dispositivos [92]. La discusión se centra en la latencia en la comunicación y la forma en que los mensajes se envían realmente entre los nodos que participan en la red [57] [93].

Todas estas propuestas son representativas de muchas otras que comparten fortalezas similares pero, en particular, debilidades para soportar interacciones en tiempo real en redes inestables. Por lo tanto, una primera pregunta que se plantea es ¿cómo están abordando los sistemas actuales estos escenarios de interacción? La respuesta es simple: relajando una de estas condiciones. Si la comunicación en tiempo real es obligatoria, entonces la red debe ser estable (por ejemplo, dedicada). En otros casos, las propuestas actuales no pueden garantizar una QoS en tiempo real; por lo tanto, no se debe proporcionar ningún servicio crítico a través de dicha infraestructura. Si necesitamos abordar ambas condiciones simultáneamente (como en muchos sistemas de monitoreo remoto), se debe definir un nuevo modelo de comunicación para respaldar las interacciones entre los componentes del sistema.

5.2.3. Comunicación en tiempo real en escenarios basados en IoT

Después de realizar una extensa encuesta sobre tecnologías y modelos de procesamiento de datos en tiempo real para respaldar las aplicaciones de IoT, Yasumoto et al. [94] no encontró

ningún protocolo o implementación capaz de proporcionar QoS en tiempo real cuando se trabaja con Internet abierta (o redes inestables). Sin embargo, la literatura reporta varios trabajos interesantes que pueden utilizarse como referencia para definir o analizar propuestas de comunicación en tal dominio de estudio. Por ejemplo, en [95] [96], los autores introducen un análisis temporal en CoAP [10], que permite medir la latencia o retraso en la transmisión de datos. Esto es útil para determinar el rendimiento de un protocolo o enlace en particular; sin embargo, no es suficiente para garantizar restricciones en tiempo real o QoS. Relacionado con CoAP, en [97], los autores analizan una extensión de este protocolo mediante la introducción de nuevas primitivas en tiempo real que se validan mediante máquinas de estados finitos.

Por otro lado, Kolozali et al. [98] introducen un procedimiento para tratar con flujos de datos en tiempo real dentro de aplicaciones de IoT, pero este trabajo de investigación no indica cómo el procedimiento contempla plazos asociados y cómo estos se verifican con pruebas de planificación. El análisis informado se limita a una baja latencia / uso retardado de la red.

Object Management Group (OMG) propone el servicio de distribución de datos (DDS) para la especificación de sistemas en tiempo real [99] que se implementa como un middleware basado en Common Object Request Broker Architecture (CORBA). Esta propuesta trata el intercambio de mensajes entre diferentes componentes de la red. Las interacciones en DDS se adhieren a un mecanismo de publicador / suscriptor, y el modelo de interacción contempla diferentes QoS, incluido el tiempo real [100]. Para hacer eso, DDS introduce varios atributos como fecha límite, latencia y periodicidad. Además, cuenta con varias implementaciones como RTI Connext DDS (propietaria) https://www.rti.com/products/dds y OpenDDS (código abierto) http://opendds.org . Aunque esta propuesta es muy completa, bien especificada y validada, el soporte en tiempo real que proporciona requiere de un enlace de comunicación estable en el tiempo. Por lo tanto, solo los conceptos detrás de la formulación de esta propuesta se pueden reutilizar si queremos proporcionar QoS en tiempo real en el escenario de estudio.

En [70], los autores propusieron el uso de vehículos aéreos no tripulados (UAV) para proporcionar conectividad en escenarios de socorro en casos de desastre estableciendo una red ad-hoc de vuelo. Hay dos aspectos principales considerados en esa propuesta: la planificación de mensajes en tiempo real y el despliegue de los nodos. Aunque la comunicación se realiza mediante dispositivos móviles, la red es dedicada y no está abierta a Internet. En [71], se proporciona un análisis en tiempo real de redes intermitentes cuando se utilizan mulas para vincular nodos desconectados. El análisis de planificación en este caso contempla no solo la periodicidad de los mensajes y el ancho de banda disponible, sino también el orden de visita de la mula y la velocidad real con la que se mueve. Como en el caso anterior, no se considera una aplicación de Internet abierta, sino una aplicación especial ad-hoc creada para apoyar la acción de los bomberos durante los incendios forestales.

5.3. Propuesta

5.3.1. Arquitectura

Cualquier modelo de interacción propuesto para hacer frente al desafío planteado debe definir dos aspectos básicos de la solución: la estructura del escenario de interacción y el comportamiento del modelo de interacción. El primero establece los componentes que estarán presentes en el escenario de IoT y el papel que tendrán dichos componentes. El segundo aspecto describe la dinámica del modelo de interacción y la forma en que los servicios de interacción son provistos considerando las restricciones preestablecidas. A continuación, se presenta el diseño de estos dos aspectos en el modelo SRTI y se discute el fundamento detrás de las principales decisiones de diseño.

5.3.1.1. Estructura del escenario de interacción

Existen varias alternativas para estructurar el escenario de interacción de IoT; sin embargo, la mayoría implementa una arquitectura de tres niveles que involucra las capas de detección, red y aplicación [101] (Figura 5.1). Esta arquitectura está diseñada para coordinar la funcionalidad de los nodos con el fin de obtener un comportamiento integral del sistema, mediante la integración y coordinación de sus partes. Esta estructura no está relacionada con el modelo tradicional de Internet de cinco capas; de hecho, cualquier sistema de IoT debe contar con protocolos anidados de capa de enlace, red y transporte para operar en la red abierta de Internet.

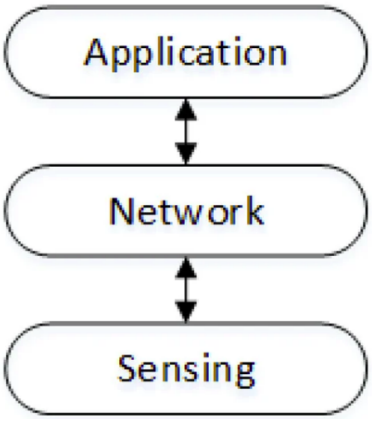


Figura 5.1: Modelo de capas de un escenario de interacción de Internet de las Cosas (IoT).

La capa inferior involucra la colección de dispositivos físicos que participan en el sistema, por ejemplo, sensores y actuadores que interactúan con el entorno físico u objetos inteligentes que actúan como intermediarios de interacción o repositorios de datos temporales. La capa de red se encarga de conectar todos los componentes para permitir interacciones y coordinar el comportamiento individual de los dispositivos para generar un comportamiento colectivo del sistema (o subsistema). Por lo tanto, esta capa transporta la información entre los componentes del sistema e implementa mecanismos para abordar varios aspectos de la comunicación como modulación, control de acceso al medio, enrutamiento, transporte

y control de flujo. Finalmente, la capa de aplicación proporciona los servicios del sistema a los usuarios finales; incluye usuarios humanos, otros sistemas y también componentes de su propio sistema (por ejemplo, actuadores). Los comportamientos complejos del sistema, por ejemplo la toma de decisiones contextualizadas, generalmente se implementan en este nivel.

En este modelo arquitectónico, un nodo puede actuar sobre las tres capas, ya que puede ser una unidad sensora que transmite los datos recolectados luego de procesarlos y transformarlos en información. Algunos nodos pueden recopilar información y luego organizarla siguiendo ciertos criterios o filtros para reenviarla a otros nodos. Finalmente, algunos nodos toman decisiones en función de la información que reciben.

El modelo SRTI se adhiere a dicha estructura considerando tres tipos de nodos: terminales, brokers y procesadores. Hay diferentes tipos de nodos en el sistema y algunos de ellos deben implementar diferentes servicios en cada capa. Las terminales están representadas por los sensores y actuadores presentes en el sistema. Claramente, estos son parte de la capa de detección y brindan servicios simples a los brokers (Figura 5.2). Una terminal puede estar vinculada a más de un broker, y dicho vínculo está débilmente acoplado. En el caso de los actuadores vinculados a más de un broker, cada cambio en el estado del dispositivo se propaga a todos los brokers involucrados, siguiendo un mecanismo de propagación de cambios similar al propuesto para el patrón modelo-vista-controlador [102].

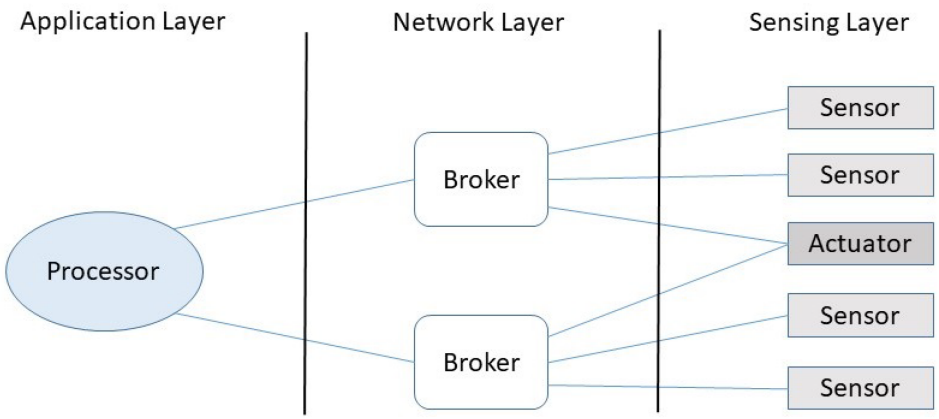


Figura 5.2: Jerarquía de las interacciones entre nodos.

Los brokers son nodos intermediarios que interactúan con un subconjunto de terminales relacionados con el fin de proporcionar cierta inteligencia básica en el nivel medio. Por ello, implementan varios servicios basados en las terminales vinculadas a ellos. Por ejemplo, se puede diseñar un broker para interactuar con todos los sensores y actuadores relacionados con el aspecto de seguridad de una casa inteligente, mientras que otros brokers pueden abordar otros aspectos del sistema, como el consumo de energía, la provisión de recursos o el bienestar ambiental. En este escenario, los brokers actúan como un interruptor, conectando sensores o actuadores con los consumidores en la capa de aplicación.

Finalmente, los nodos procesadores suelen incrementar la inteligencia del sistema utilizando los servicios proporcionados por los brokers. Como en el caso anterior, un procesador se puede vincular a múltiples brokers y viceversa. Los procesadores se implementan en la capa de aplicación. La gestión de eventos en esta capa generalmente se adhiere al modelo de publi-

cación-suscripción [100] [103]. A diferencia de los modelos de acoplamiento cliente-servidor o solicitud-respuesta, la publicación-suscripción facilita la inclusión y exclusión de dispositivos y también la orquestación dinámica de los servicios proporcionados por los nodos, haciendo que los sistemas sean más flexibles y fáciles de evolucionar [103] [104].

Es importante señalar que los tipos de nodos representan construcciones conceptuales; por lo tanto, pueden alojarse en el mismo dispositivo, en particular los brokers y los procesadores.

5.3.1.2. Gestión inteligente de tópicos en el escenario de comunicación

El modelo de IoT descrito en la sección anterior se ocupa de la funcionalidad de los nodos y no de cómo se intercambia la información entre los nodos de la red. Si se considera que las aplicaciones IoT abiertas que se ejecutan en Internet abierta, son aquellas cuyos dispositivos utilizan enlaces proporcionados por proveedores de servicios de Internet (ISP), es necesario describir la interacción entre los nodos en términos de las capas de red tradicionales. Como se mencionó anteriormente, la capa de red de IoT comprende todos los aspectos de comunicación relacionados con el intercambio de información, y estos están representados por las capas física, de enlace, de red y de transporte en el modelo tradicional de Internet.

Aunque las aplicaciones de IoT asumen que se puede acceder a los sensores y actuadores a través de una Internet abierta en un modo de acceso plano, todavía en la práctica existen varias restricciones, como el descubrimiento dinámico de servicios, los proveedores de información y el enrutamiento. Sin embargo, incluso si estas restricciones se abordan adecuadamente, la red requeriría un gran ancho de banda para evitar el desbordamiento y la congestión del tráfico. El uso de brokers en este escenario reduce el tráfico, ya que recopilan información de los publicadores y la envían a los suscriptores bajo demanda.

Los brokers tienen un agente inteligente en tiempo real – Intelligent Real Time Agent o IRTA – asociado en la capa de aplicación. Este agente está suscrito a todos los tópicos publicados en el broker por un lado, mientras que también recibe todas las suscripciones al broker. Los datos publicados son filtrados por el agente de acuerdo con un conjunto de funciones que puede configurar el usuario final del sistema. Cada publicación tiene metadatos que describen atributos como su período de publicación, precisión, orígen y confiabilidad.

Por otro lado, las suscripciones requieren datos de ciertos orígenes, con una periodicidad, precisión y confiabilidad esperadas. En el caso de una coincidencia directa entre la suscripción y la publicación, el IRTA conecta ambas. Si no hay una coincidencia directa, el IRTA puede trabajar con las publicaciones disponibles para satisfacer al suscriptor con una nueva. Para hacer esto, puede agregar datos de múltiples orígenes para generar una frecuencia de muestreo deseada que no es provista originalmente por los publicadores, o fusionar datos sin procesar para generar nueva información.

El IRTA se encarga de verificar que los datos estén a tiempo y de implementar políticas de programación en caso de conflictos. Las funciones del IRTA no están relacionadas con un servidor o la nube. En cambio, estas funciones solo pre-procesan los datos para conformar las suscripciones y validarlas desde una perspectiva de tiempo real. Si existe una suscripción que solicita cierta periodicidad para un dato en particular o requiere una cierta latencia en la red para tomar un valor como bueno, el IRTA debe evaluar si los publicadores tienen datos que cumplan con estos requisitos. Utilizando los últimos retrasos medidos en la red, el IRTA puede determinar si es posible satisfacer la latencia mínima. La suscripción se rechaza si no

se garantizan ambas cosas. La Figura 5.3 presenta un diagrama que muestra esta estructura.

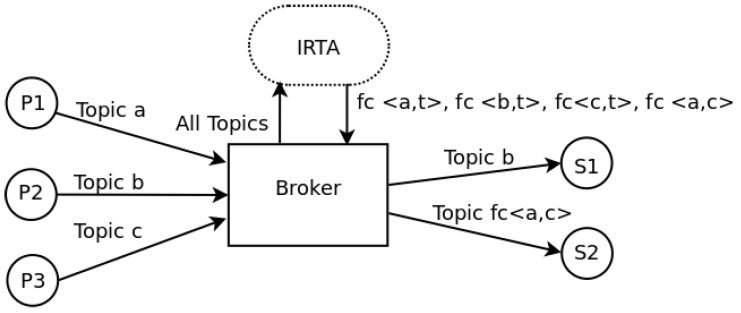


Figura 5.3: Estructura de un broker.

5.3.1.3. Comportamiento del IRTA

Normalmente, cuando el broker recibe una solicitud de suscripción que no tiene una coincidencia directa en las publicaciones, llama al IRTA para analizar si la respuesta se puede construir utilizando los datos almacenados en el repositorio local. Si lo hace, el IRTA genera un nuevo tópico que coincide con la solicitud del suscriptor y sus restricciones de tiempo real. Para ello, el usuario puede implementar un conjunto de funciones para trabajar sobre los metadatos asociados a las publicaciones.

Por otro lado, cuando el IRTA no es capaz de construir la respuesta con la información local, se inicia una segunda etapa de negociación y se busca el tópico solicitado entre los diferentes brokers que participan en el sistema. En esta segunda ronda, los brokers vecinos comparten información para construir la respuesta solicitada.

5.3.1.4. Ejemplo de aplicación

Para ilustrar este punto, supongamos que hay un sensor de temperatura que proporciona la temperatura leída, junto con información adicional. Esta información adicional incluye las unidades en las que se proporcionan los datos (es decir, Celsius, Fahrenheit o Kelvin), la ubicación real del sensor de forma cualitativa y precisa (p. Ej., Buenos Aires, latitud y longitud), la frecuencia de muestreo o período con el que se lee la temperatura, su precisión (8,10,12 o 16 bits), y un último campo que sopesa la fiabilidad del sensor desde la fuente. Los metadatos asociados a este publicador proporcionan al IRTA la información necesaria para agregarlos o combinarlos con otros datos y, de esta manera, obtener nueva información útil para el sistema o los usuarios finales. El Cuadro 5.1 presenta la estructura de datos de la respuesta proporcionada por el sensor.

Cambiar las unidades de las respuestas proporcionadas por el sensor es una operación simple y se puede implementar con solo unos pocos cálculos. Las funciones asociadas a los cambios de unidad se anotan $U(a, b)$, donde a es la unidad utilizada por el sensor y b es la unidad deseada. La frecuencia es más difícil de ajustar. Cuando una suscripción requiere cierta frecuencia de muestreo, hay básicamente dos escenarios a considerar: (1) la tasa de muestreo del publicador está por debajo de la demandada; o (2) es igual o superior. En el primer caso, la única forma de aceptar la suscripción es obtener más datos de otras fuentes.

Variable	Unidad
Temperatura	C, F o K
Ubicación	Lat, Long
Período	s
Precisión	8/10/12/16 bits
Confiabilidad	Alta/Media/Baja

Cuadro 5.1: Metadata para el sensor de temperatura

En el segundo caso, hay dos alternativas: o el suscriptor acepta una actualización de velocidad de datos más alta o el IRTA crea una función de regresión (probablemente de primer orden) para interpolar valores que coincidan con la frecuencia demandada. Sin embargo, para hacer esto, los plazos deben ser lo suficientemente grandes como para admitir la latencia necesaria para construir los datos "calculados", por ejemplo, utilizando la metodología de mínimos cuadrados. El conjunto de valores utilizados para hacer la regresión se puede actualizar dinámicamente.

En el caso de que se requiera una frecuencia de muestreo más alta, el IRTA debe verificar si es posible combinar diferentes publicadores. Para ello, la ubicación, precisión y confiabilidad de los sensores deben ser similares para producir una nueva publicación con la calidad necesaria. En el caso básico, supongamos que hay dos sensores de temperatura en Buenos Aires con parámetros similares e incluso del mismo período. Ambos producen datos cada 30 min, pero el primero lo hace a las horas en punto y pasados los 30 min, mientras que el segundo lo hace a los 15 y 45 min. Combinando ambas lecturas, el IRTA puede proporcionar la temperatura con un período de 15 minutos. Las funciones relacionadas con los ajustes de frecuencia se anotan como $F(a)$.

Cuando una suscripción requiere datos que no están presentes en el broker, el IRTA puede solicitar a los brokers vinculados que determinen si tienen un publicador que satisfaga la suscripción. Si existe un broker de este tipo, el IRTA puede pasar la solicitud a ese broker, actuando como un simple despachador del suscriptor hacia el el nuevo par broker/publicador. La función de buscar el tópico en otro broker se anota como $S(a, b)$, en el que a es el broker original y b es el nuevo broker.

La última funcionalidad del IRTA está relacionada con proporcionar información basada en datos brutos. Para ello, debe ser provista con las funciones adecuadas, que pueden ser tan simples como proporcionar los valores máximos o mínimos de un tópico publicado o tan complejas como calcular una transformada rápida de Fourier en una secuencia de datos para proporcionar el espectro de un tópico. Hay muchas opciones a implementar según el tipo de procesamiento de datos que se requiera. En cualquier caso, estas funciones se anotan como $P(a)$, donde a indica el tópico que se procesa.

Con esta arquitectura, el sistema puede verse como un conjunto jerárquico de endpoints (es decir, sensores o actuadores), intermediarios inteligentes en tiempo real y suscriptores. Los brokers pueden tener dos funciones, ya que además pueden convertirse en publicadores y suscriptores de otros brokers que les permitan obtener información que no está presente en sus entradas o salidas.

La Figura 5.4 ilustra un ejemplo de la estructura general del escenario de interacción,

considerando una arquitectura con cuatro brokers interconectados. Para simplificar, los publicadores están en la parte inferior, mientras que los suscriptores están en la parte superior de la figura. Como se muestra en dicha figura, varios publicadores comparten sus tópicos con diferentes brokers y varios suscriptores reciben tópicos de diferentes brokers. En el medio, los brokers comparten datos como publicadores y suscriptores indistintamente dependiendo de la funcionalidad requerida para cada conexión.

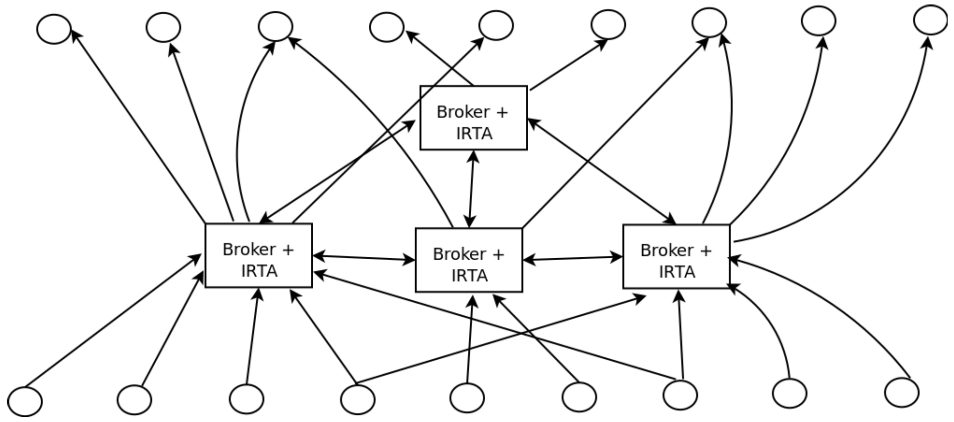


Figura 5.4: Estructura de un escenario de interacción.

5.3.1.5. Modelo de comunicación en tiempo real

Una tarea en tiempo real se define como aquella que debe producir resultados correctos dentro de un plazo determinado denominado vencimiento, y un sistema en tiempo real es aquel en el que hay al menos una tarea en tiempo real [105]. En un escenario de IoT, estas tareas representan subprocesos en los que se basa la interacción entre los componentes del sistema, que son coordinados por una o más aplicaciones de software. Por lo tanto, una tarea en tiempo real, τ_i, se describe como una secuencia de trabajos o instancias.

El intervalo entre estas instancias se denomina período, P_i. Cada instancia tiene un tiempo de ejecución en el peor de los casos denominado C_i y una fecha de vencimiento relativa, D i. Cada instancia tiene un tiempo de activación a_{ij} y una fecha de vencimiento absoluta definida como $d_{ij} = a_{ij} + D_i$.

Existen varias políticas de programación para ordenar la ejecución de tareas, siendo las más utilizadas Rate Monotonic Scheduling (RMS) y Earliest Deadline First (EDF). En RMS, las prioridades de las tareas se asignan en orden inverso a P; por lo tanto, se asigna la prioridad más alta a la tarea con un mínimo P. En EDF, las prioridades son dinámicas y se le da la prioridad más alta a la tarea que tiene el plazo más temprano. En esta propuesta, adoptamos EDF dado que es óptima para sistemas de un solo procesador, como se muestra en [106].

En una interacción de IoT de tiempo real, el tiempo de procesamiento en ambos endpoints (por ejemplo, el productor y el consumidor) y el retraso en la red deben considerarse dentro de la fecha de vencimiento. Como se mencionó anteriormente, el sistema de IoT puede implicar el uso de una red dedicada o una con una estabilidad incierta. En el primer caso, podemos

calcular el retraso de extremo a extremo en el peor de los casos en una interacción, con notación Δ_i, del productor al consumidor de la siguiente manera:

$$\Delta_i = C_{\texttt{con}} + \delta_n + C_{\texttt{prod}} \tag{5.1}$$

donde $C_{\texttt{con}}$ y $C_{\texttt{prod}}$ representan el tiempo de cálculo en el consumidor y el productor, respectivamente, y δ_n es el retraso en la red de comunicación. Con él, se puede implementar una política de planificación en el productor, la red y el consumidor para garantizar los plazos.

Si la red no está dedicada (es decir, su estabilidad es incierta), la situación es mucho más compleja. Aunque la ecuación (5.1) sigue siendo válida, δ_n no siempre es predecible. En este caso, el procesamiento en tiempo real duro no es factible ya que las tareas pueden perder sus plazos mientras se retrasan en la red. En los casos en que se puedan perder algunos plazos sin poner en peligro la correctitud del sistema (tiempo real blando), se puede suponer que el retraso en la red es el peor retraso medido en el proceso de negociación de la conexión. Este se puede actualizar en los próximos intercambios, para así mantener el retraso esperado cerca del retraso del peor caso medido en las últimas transacciones (por ejemplo, en los últimos 10 intercambios).

Para hacer frente a retrasos desconocidos en la red, se ha extendido el protocolo MQTT [9] que, según el análisis realizado en el estado del arte, se acerca más a esta propuesta. La siguiente sección presenta tal extensión.

5.3.2. Modelado del paradigma de publicación-suscripción en el modelo SRTI

Esta sección presenta la implementación del esquema de interacción publicación-suscripción, considerando la estructura del escenario de IoT y las decisiones de diseño presentadas en la sección anterior. Para ello, especificamos el comportamiento de los procesadores (suscriptores), terminales (publicadores) y brokers (publicador / suscriptor) cuando se utiliza el modelo SRTI para soportar interacciones en el escenario de estudio de IoT. Las terminales están asociadas a la capa de detección en el modelo de IoT. En este escenario, son productores de datos (sensores) o consumidores (actuadores). Siempre que la mayoría de los participantes en la capa de detección sean sensores, asociamos terminales a publicadores.

Como se introdujo en la sección anterior, los procesadores transforman los datos provenientes de los sensores en información que queda disponible para todos los componentes del sistema. Para ello, los procesadores se suscriben a tópicos en los diferentes brokers. Para simplificar la descripción, se asume que las terminales en general están asociadas con sensores, aunque también podrían ser actuadores (no son publicadores sino suscriptores que reciben comandos para ejecutar). Similar a MQTT, el modelo SRTI proporciona un conjunto de primitivas que ordenan la comunicación entre procesadores y terminales a través de intermediarios, como se ilustra en la Figura 5.2. Por lo general, las soluciones inteligentes que trabajan en el escenario de IoT interactúan directamente con los procesadores para monitorear las variables relevantes, entregar notificaciones, tomar decisiones y realizar acciones particulares cuando sea necesario. Los procesadores se suscriben a los servicios prestados por los intermediarios (generalmente, suministro de información) dado que estos últimos tienen

un acuerdo de cooperación (contrato) con los terminales vinculados a ellos (es decir, sensores y actuadores).

A continuación, se especifica el comportamiento de las terminales, procesadores e intermediarios utilizando una máquina de estado finito (FSM). Se supone que la conexión entre estos componentes ya está establecida; por tanto, sus especificaciones de comportamiento se centran principalmente en el intercambio de mensajes de datos. En el caso de que un determinado nodo funcione como terminal o procesador (es decir, publicador o suscriptor), puede asumir ambos modos de funcionamiento, conmutando entre ellos según el papel que desempeñe en el funcionamiento del sistema.

5.3.2.1. Terminales: especificación de comportamiento

Como se mencionó anteriormente, las terminales corresponden a los publicadores en el patrón de publicación-suscripción. El comportamiento de estos componentes es simple ya que, mientras está conectado a un broker, principalmente carga datos con una periodicidad determinada. La Figura 5.5 muestra la máquina de estados finitos para la terminal. La única modificación que debe incorporarse dentro de los datos es la marca de tiempo y el período. El broker utiliza esta información para mantener actualizado el retraso entre el publicador y el broker y para evaluar si los datos llegarán potencialmente antes de la fecha límite para el suscriptor.

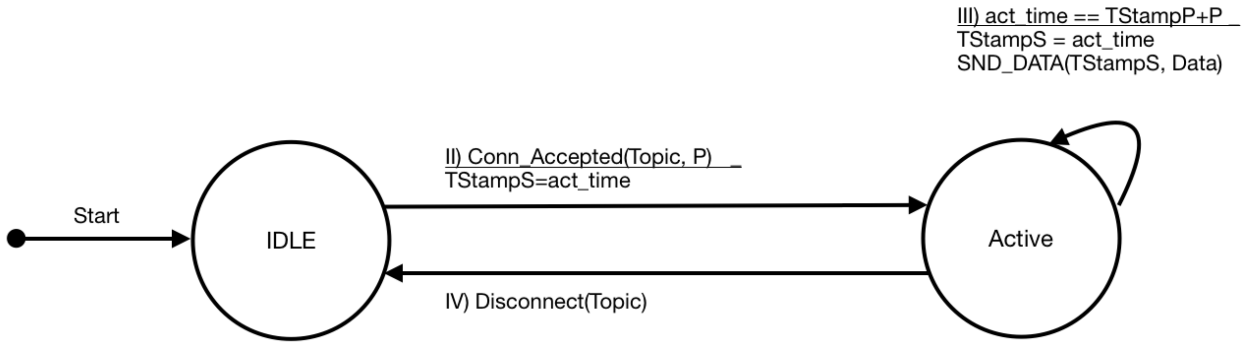


Figura 5.5: Máquina de estados finitos (FSM) que representa el comportamiento de las terminales.

5.3.2.2. Procesador: especificación de comportamiento

En el caso de los procesadores (es decir, los suscriptores), su comportamiento tiene que lidiar con marcas de tiempo y fechas de vencimiento. Como se puede apreciar en la Figura 5.6, el procesador se conecta al broker y permanece conectado mientras este último reenvía los datos recibidos desde las terminales. En el primer estado, el procesador está en `IDLE` (ocioso) esperando una activación de la aplicación de software que controla el escenario de IoT. Una vez que la aplicación solicita datos al procesador, `REQ_DATA`, envía `SEND_S_REQ(Topic,TStampR,D,P)` al broker correspondiente. Los parámetros identifican el tópico, la hora real de la solicitud, la fecha de vencimiento y el período. El *topico*

una variable que se usa para identificar clases de información solicitada (por ejemplo, clima, tráfico o imágenes de cámaras de vigilancia). Cuando el broker acepta la suscripción, `Subs_Ack(Topic,TStampS,D,P)`, los procesadores pasan al estado `WAIT`.

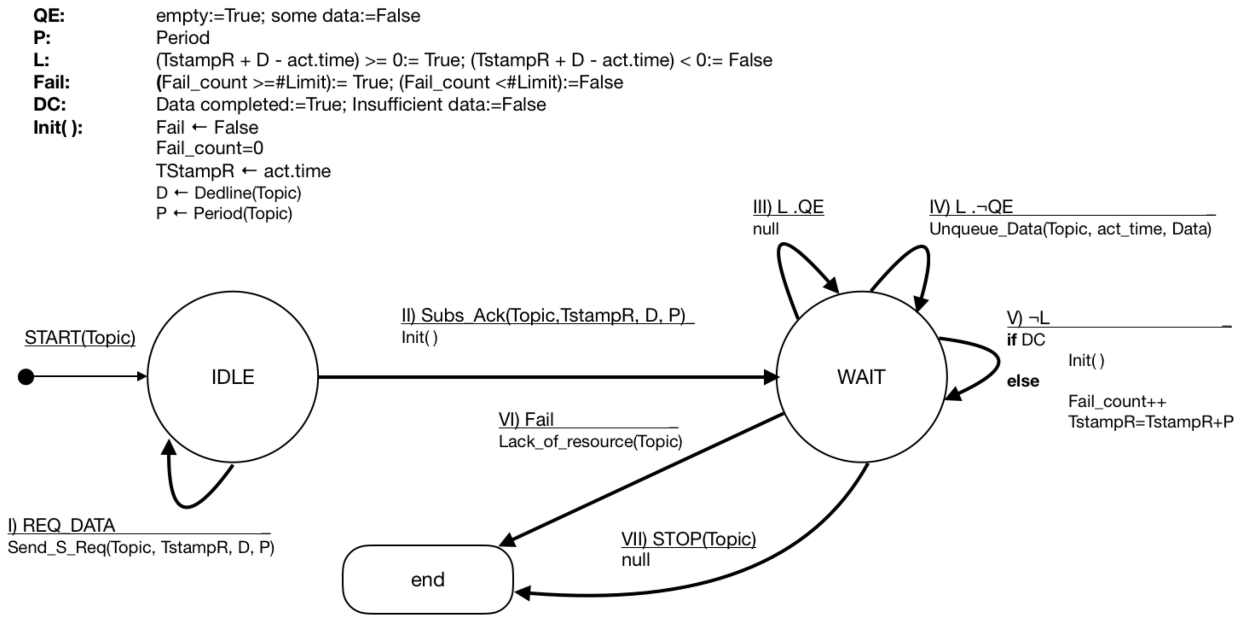


Figura 5.6: FSM que representa el comportamiento del procesador.

Cuando el procesador está en el estado `WAIT`, evalúa cuatro variables booleanas relacionadas con el proceso de suscripción.

- Cola vacía QE es verdadero si la cola de datos de recepción está vacía y es falso cuando se ha recibido al menos un dato válido.
- El tiempo transcurrido L es verdadero mientras no se haya cumplido la fecha de vencimiento. En realidad, representa el tiempo disponible para recibir datos antes de la expiración del plazo. Una vez que pasa la fecha de vencimiento, esta variable se vuelve falsa.
- Falla F es verdadera si la cantidad de fallas es mayor que $Limit$. El procesador registra, en $Fail\_count$, la cantidad de veces que los datos no han llegado dentro de su fecha de vencimiento. Cuando $Fail\_count \geq Limit$, F se vuelve verdadera.
- Datos completados DC se utiliza para identificar si se han recibido suficientes datos para el procesamiento. En este caso, la aplicación que maneja la cola establece o restablece la variable.

Las transiciones asociadas al estado `WAIT` están determinadas por los valores de las variables booleanas. En la tercera transición, hay tiempo ($L = true$) y la cola está vacía; por lo tanto, permanece en `WAIT`. La cuarta transición corresponde al caso en el que hay tiempo y se han recibido algunos datos. En este caso, los datos no se ponen en cola y se entregan

a la aplicación para su uso. En la quinta transición, no hay más tiempo y la máquina de estado decide si debe incrementar el conteo de fallas o restablecer todos los parámetros para el siguiente período. En este caso, el DC se utiliza para evaluar si los datos recibidos son suficientes para la aplicación. Cuando el número de fallos llega a $Limit$, pasa al estado `END`, lo que indica que no hay ningún recurso asociado al tópico solicitado. La aplicación puede finalizar la suscripción con una señal de `STOP`.

Los servicios disponibles para suscripción proporcionan datos únicos o múltiples, que se indican mediante un comodín (como en MQTT). En los servicios de datos únicos, la respuesta tiene una estructura y semántica única, pero en los servicios de datos múltiples, esa estructura y semántica deben ser informadas a los suscriptores para permitirles comprender adecuadamente la información recibida. Los comodines son de uno o varios niveles. En el primer caso, se utiliza para recopilar la información en tópicos jerárquicos. Por ejemplo, al verificar la temperatura en una ciudad utilizando "Ciudad/+/ Temperatura/", no es relevante si la información proviene del centro de la ciudad o de un parque, cualquier valor que esté disponible es bueno para esa suscripción. En el caso de un comodín de varios niveles, toda la información de una ubicación específica es relevante, por ejemplo, "Ciudad/Centro/#" juntaría todos los tópicos asociados al centro de la ciudad.

5.3.2.3. Agente: especificación de comportamiento

En el modelo de publicación-suscripción, el broker es un elemento clave que conecta las acciones de los publicadores y suscriptores. Su objetivo principal es recopilar datos de los publicadores y luego distribuirlos a los suscriptores. El broker interactúa tanto con los publicadores como con los suscriptores, lo que hace que su operación sea la más compleja dentro del protocolo. En este trabajo, además del broker, el IRTA se encarga de agregar datos y producir, si es posible, nuevas publicaciones para satisfacer los requisitos de los suscriptores.

La Figura 5.7 presenta el FSM para el IRTA / broker. Suponemos que las primitivas de conexión, publicación y suscripción siguen el protocolo MQTT estándar. El agente del IRTA recibe todas las publicaciones del broker. Cuando un suscriptor requiere un tópico determinado, el IRTA verifica si tiene el tópico en el directorio de brokers o si es posible agregar información proveniente de otros publicadores para satisfacer la suscripción. Si faltan parte de los datos, verifica con otros brokers para determinar si es posible completar la información necesaria. Extendemos esta funcionalidad incorporando en el proceso de suscripción la identificación del período, la marca de tiempo y la fecha de vencimiento para las transmisiones.

La información de contexto (generalmente metadatos) también se puede utilizar para determinar la validez y relevancia de un dato cuando se debe crear la respuesta a una solicitud. Para ello, es importante contar con la geolocalización del sensor, la calidad de la medida y el momento en el que se han recopilado los datos como se explicó previamente. En este caso, no es necesario conocer la dirección IP del dispositivo.

El broker se vuelve `Active` siempre que se establece una conexión válida con el publicador o el suscriptor (transición I). De la misma forma, se vuelve `Idle` cuando no hay más conexiones (transición V). En la transición II, el broker acepta una suscripción con una determinada marca de tiempov $TStampR$, Fecha de vencimiento D, y el periodo P. Con

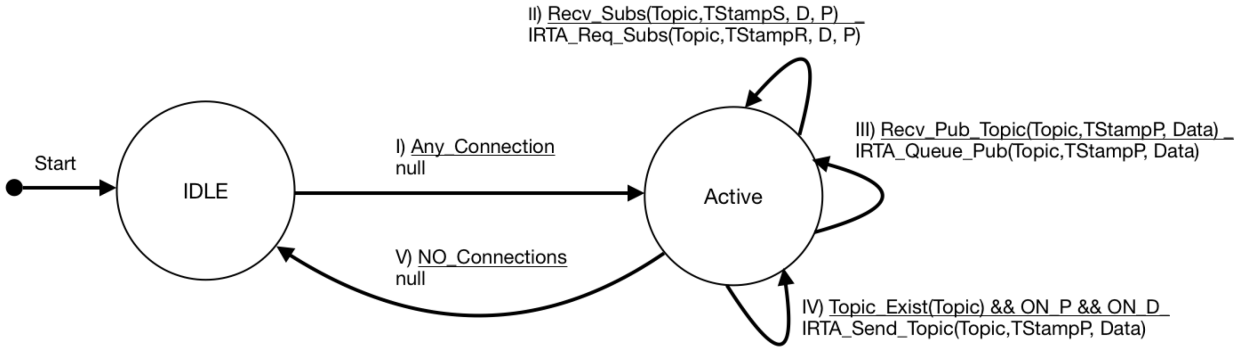


Figura 5.7: FSM que representa el comportamiento del agente.

el primer parámetro, el broker puede evaluar el tiempo restante que tiene para enviar el tópico solicitado al suscriptor. La tercera transición describe el registro del publicador con un tópico y período determinados. También tiene la hora en la que se produjeron los datos. En ambos casos, el IRTA participa en el procesamiento y emparejamiento de publicadores y suscriptores. Finalmente, la cuarta transición describe el instante en el que el broker tiene el tópico. Siempre que esté dentro del plazo y haya transcurrido el plazo, deberá enviar el tópico / datos al suscriptor.

5.3.3. Implementación de la extensión

En esta sección se describen las extensiones propuestas al protocolo MQTT para implementar el modelo SRTI en un sistema de mensajería para el desarrollo de aplicaciones. Estas extensiones consideran la versión 3.1.1 del protocolo.

Hay dos puntos a tener en cuenta para este trabajo. El primero trata sobre cómo un broker de MQTT debe interactuar con el IRTA. El segundo indica cómo adaptar la especificación de esta versión de MQTT, para que los publicadores, suscriptores y brokers puedan enviar la información adicional que el IRTA necesita para brindar las funcionalidades descritas en apartados anteriores. La Figura 5.8 muestra un diagrama de actividad del agente IRTA.

La comunicación entre el broker MQTT y el IRTA es bidireccional; esto significa que ambos componentes pueden enviar y recibir datos. Por un lado, el broker enviará información tanto sobre las suscripciones como sobre las publicaciones que reciba. Por otro lado, el IRTA almacenará la información de las suscripciones (por ejemplo, el tópico, la marca de tiempo, la fecha de vencimiento y el período). Cada vez que recibe una publicación, realiza los siguientes pasos:

1. Si existe una función definida por el usuario, se aplicará tomando como argumento el mensaje de publicación. El resultado de esta función será un nuevo mensaje de publicación que reemplazará al anterior. Por ejemplo, la publicación original podría ser un valor de temperatura con el tópico “Full_Temperature” con una precisión de dos bytes. Sin embargo, la función aplicada lo redondea a un byte porque no se necesita tanta precisión; por lo tanto, para ahorrar ancho de banda, se publica bajo un nuevo tópico llamado “Shrunken_Temperature”.

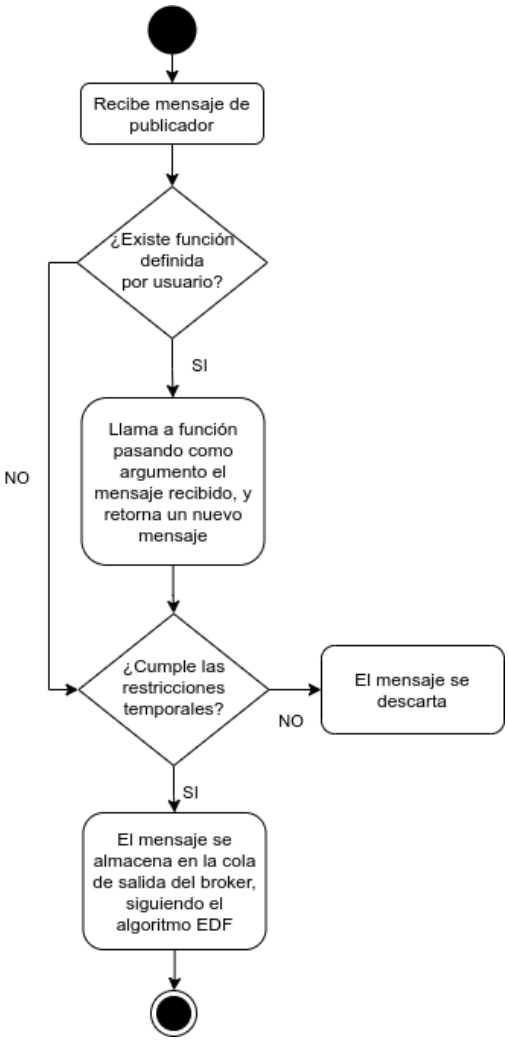


Figura 5.8: Diagrama de actividad de alto nivel de un procesamiento de un mensaje de publicación del agente inteligente en tiempo real (IRTA).

2. El mensaje de publicación generado en el paso 1 se somete a un análisis de restricciones temporales para validar si está dentro de las expectativas del suscriptor o no. Este análisis corresponde a la transición VI de la máquina de estados descrita previamente.

3. Si del paso anterior se decide que la publicación es válida, se pone en la cola del búfer de salida del broker siguiendo el algoritmo de programación EDF.

5.3.3.1. Extensiones a MQTT

Con respecto a la especificación del protocolo, este trabajo propone tres modificaciones para permitir interacciones en tiempo real blando en escenarios de IoT. La primera es agregar un registro de tópico en los paquetes de control de MQTT, similar a MQTT-SN que es una variante del protocolo que introduce una forma de registrar un tópico para reducir el tamaño de los paquetes. El objetivo del registro de tópicos en SRTI no es encoger los paquetes, sino informar los datos temporales asociados al publicador y la periodicidad del tópico. Por lo tanto, su encabezado de variable debe tener la marca de tiempo del publicador, el período en el que enviará los datos correspondientes a ese tópico, y la longitud del tópico junto con su nombre. El identificador del paquete de control de MQTT para el registro podría ser 0 o 15, ambos sin usar actualmente.

La segunda extensión implica la modificación de los encabezados de los paquetes de control de MQTT PUBLISH y SUBSCRIBE, agregando en ambos una marca de tiempo del emisor. Además, en el caso de SUBSCRIBE, se agrega también el período y plazo en el que se espera recibir los datos de ese tópico.

Las estructuras nuevas de los paquetes de registro, publicación y suscripción pueden encontrarse en el Apéndice A.

La tercera extensión considera agregar un nuevo código de retorno en el paquete SUBACK, para que el broker pueda rechazar una suscripción en caso de que no pueda ofrecer datos con los parámetros temporales requeridos por el suscriptor. El valor del código de retorno puede ser cualquiera entre el código de falla 0x80 y el máximo disponible 0xFF.

Se asume que los nodos tienen sus relojes sincronizados. El mecanismo de sincronización, excede este modelo, pero podría ser utilizando NTP o similar.

5.3.4. Evaluación

Con el objetivo de validar de manera empírica la factibilidad de la implementación de esta propuesta, se modificó el broker de código abierto Mosca[72], para que sea compatible con IRTA. Se llevaron a cabo tres simulaciones utilizando dicho broker, cada una con distintas configuraciones. En la primera, el sistema cuenta con dos publicadores, uno envía un caudal de una bomba de riego y otro un registro de tiempo, un suscriptor que requiere una aplicación y un broker con IRTA. El suscriptor requiere la lámina de riego aplicada, cuya fórmula se describe en la fórmula 5.2. Este dato no se encuentra en los tópicos disponibles, pero el IRTA tiene una función definida que, aplicando la fórmula descrita asumiendo una superficie constante, puede permitir la suscripción usando los tópicos provenientes de los dos publicadores.

$$Lámina = \frac{Caudal \times Tiempo\ transcurrido}{Superficie} \tag{5.2}$$

En la segunda simulación, el sistema tiene un publicador, un suscriptor y un broker con IRTA. El publicador envía el caudal de bomba cada 1 segundo, mientras que el suscriptor lo desea consumir cada 3 segundos. Por lo tanto, el suscriptor requiere un tópico con una cierta tasa que no está disponible dentro de las publicaciones. Sin embargo, el tópico está presente pero con mayor frecuencia. El broker acepta la suscripción a través del IRTA y este realiza submuestras de los datos transmitidos en el tópico, reduciendo el tráfico y cumpliendo con los requisitos del suscriptor.

Finalmente, la tercera simulación involucra a un publicador que envía un dato de caudal, un suscriptor y un broker con IRTA. En este caso, el suscriptor requiere un tópico con cierta frecuencia y latencia. Aunque el tópico está presente en el broker con la frecuencia requerida, el IRTA no puede garantizar la latencia y el tópico no se reenvía al suscriptor. Las siguientes figuras muestran los registros obtenidos en cada elemento en los diferentes experimentos.

La Figura 5.9, la Figura 5.10, la Figura 5.11 y la Figura 5.12 muestran el registro del primer experimento en los sensores, el broker y el suscriptor, respectivamente. Como puede verse, el broker espera a los tópicos que provienen de los dos publicadores para fusionarlos antes de reenviar el tópico requerido al suscriptor.

```
[2019/07/26 15:54] Packet sent with topic ‘‘pumpFlow’’ and payload:  146.71
[2019/07/26 15:55] Packet sent with topic ‘‘pumpFlow’’ and payload:  146.04
[2019/07/26 15:56] Packet sent with topic ‘‘pumpFlow’’ and payload:  147.87
[2019/07/26 15:57] Packet sent with topic ‘‘pumpFlow’’ and payload:  149.79
```

Figura 5.9: Registro de la bomba del sensor.

```
[2019/07/26 15:54] Packet sent with topic ‘‘time’’ and payload:  92.43
[2019/07/26 15:55] Packet sent with topic ‘‘time’’ and payload:  99.06
[2019/07/26 15:56] Packet sent with topic ‘‘time’’ and payload:  97.63
[2019/07/26 15:57] Packet sent with topic ‘‘time’’ and payload:  96.78
```

Figura 5.10: Registro de tiempo del sensor.

La Figura 5.13, la Figura 5.14 y la Figura 5.15 muestran los registros para el publicador, el broker y el suscriptor en el segundo experimento, respectivamente. Aquí, se puede ver que el publicador envía información con una frecuencia tres veces mayor a la requerida y que el broker descarta dos de cada tres paquetes antes de reenviar los datos al suscriptor.

La Figura 5.16 y la Figura 5.17 muestran los registros para el publicador y el broker en el tercer experimento, respectivamente. En este caso, la latencia esperada por el publicador es mayor que la requerida por el suscriptor; por lo tanto, el broker descarta el tópico.

El Cuadro 5.2 presenta un análisis cualitativo que compara dos versiones de la solución requerida para abordar los tres escenarios de interacción descritos en esta sección. Una versión de la solución se implementó utilizando el estándar MQTT y la otra utilizó el SRTI propuesto con IRTA (SRTI-IRTA). En los tres experimentos, se puede ver que el uso de SRTI-IRTA proporciona varios beneficios; sin embargo, también se reconoce que esta comparación

```
[2019/07/26 15:54] Message received with topic:  pumpFlow
[2019/07/26 15:54] Message received with topic:  time
[2019/07/26 15:54] Publish new message under topic:  applicationRate
[2019/07/26 15:55] Message received with topic:  pumpFlow
[2019/07/26 15:55] Message received with topic:  time
[2019/07/26 15:55] Publish new message under topic:  applicationRate
[2019/07/26 15:56] Message received with topic:  pumpFlow
[2019/07/26 15:56] Message received with topic:  time
[2019/07/26 15:56] Publish new message under topic:  applicationRate
[2019/07/26 15:57] Message received with topic:  pumpFlow
[2019/07/26 15:57] Message received with topic:  time
[2019/07/26 15:57] Publish new message under topic:  applicationRate
```

Figura 5.11: Registro del broker.

```
[2019/07/26 15:54] Receives packet with topic ‘‘applicationRate’’ and payload:  27.12
[2019/07/26 15:55] Receives packet with topic ‘‘applicationRate’’ and payload:  28.93
[2019/07/26 15:56] Receives packet with topic ‘‘applicationRate’’ and payload:  28.87
[2019/07/26 15:57] Receives packet with topic ‘‘applicationRate’’ and payload:  28.99
```

Figura 5.12: Registro del suscriptor.

no es justa, ya que dicha infraestructura de software implementa funcionalidades que no están presentes en la versión actual de MQTT. Independientemente de tal consideración, esta comparación permite ver que SRTI-IRTA contribuye a alcanzar interacciones en tiempo real blando en escenarios de IoT y se presenta como una mejor alternativa respecto de la expuestas en la revisión del estado del arte reportado previamente.

5.3.5. Escenario de aplicación

Esta sección muestra un escenario de aplicación relacionado con la agricultura de precisión (AP) y describe cómo se puede aplicar el modelo propuesto a dicho dominio. La AP está relacionada con el uso de la tecnología para observar, medir y decidir las acciones para optimizar la productividad de un área de cultivo, preservando los recursos naturales [85] [86]. Como se mencionó al comienzo de este capítulo, hay varias situaciones que requieren tiempos de respuesta predecibles en escenarios de AP. Un caso típico es el riego para prevenir heladas y sus consecuencias en plantaciones de frutales. Cuando se presentan ciertas combinaciones de temperatura, presión atmosférica y humedad, las heladas pueden producir daños severos a las plantaciones de frutas, y el riego de las plantas en un momento determinado evita los efectos de las heladas.

En escenarios de AP, la sincronización entre varios tipos de sensores distribuidos en el campo también es necesaria para obtener una imagen coherente de la situación actual. En países donde la agricultura se desarrolla en grandes áreas, los agricultores no pueden controlar personalmente los cultivos, ni siquiera con varios empleados. Por ello, es importante contar con un sistema automático o semiautomático que supervise y registre, en tiempo real, las diferentes variables necesarias para decidir qué acción realizar.

Cuando estos sistemas de riego operan de manera autónoma, se requiere la interacción en

```
[2019/07/26 16:14] Packet sent with topic ‘‘pumpflow’’ and payload:  146.56
[2019/07/26 16:15] Packet sent with topic ‘‘pumpflow’’ and payload:  146.62
[2019/07/26 16:16] Packet sent with topic ‘‘pumpflow’’ and payload:  147.27
[2019/07/26 16:17] Packet sent with topic ‘‘pumpflow’’ and payload:  145.76
[2019/07/26 16:18] Packet sent with topic ‘‘pumpflow’’ and payload:  145.30
[2019/07/26 16:19] Packet sent with topic ‘‘pumpflow’’ and payload:  149.51
[2019/07/26 16:20] Packet sent with topic ‘‘pumpflow’’ and payload:  148.31
[2019/07/26 16:21] Packet sent with topic ‘‘pumpflow’’ and payload:  147.15
[2019/07/26 16:22] Packet sent with topic ‘‘pumpflow’’ and payload:  146.48
```

Figura 5.13: Registro del sensor.

```
[2019/07/26 16:14] Message received with topic:  pumpFlow
[2019/07/26 16:15] Message received with topic:  pumpFlow
Packet period is not acceptable:  1 (projected) vs 3 (expected)
[2019/07/26 16:16] Message received with topic:  pumpFlow
Packet period is not acceptable:  2 (projected) vs 3 (expected)
[2019/07/26 16:17] Message received with topic:  pumpFlow
[2019/07/26 16:18] Message received with topic:  pumpFlow
Packet period is not acceptable:  1 (projected) vs 3 (expected)
[2019/07/26 16:19] Message received with topic:  pumpFlow)
Packet period is not acceptable:  2 (projected) vs 3 (expected)
[2019/07/26 16:20] Message received with topic:  pumpFlow
[2019/07/26 16:21] Message received with topic:  pumpFlow
Packet period is not acceptable:  1 (projected) vs 3 (expected)
[2019/07/26 16:22] Message received with topic:  pumpFlow
Packet period is not acceptable:  2 (projected) vs 3 (expected)
```

Figura 5.14: Registro del broker.

```
[2019/07/26 16:14] Receives packet with topic ‘‘pumpflow’’ and payload:  146.56
Received pump flow:  146.56 m3/hour
[2019/07/26 16:17] Receives packet with topic ‘‘pumpflow’’ and payload:  145.76
Received pump flow:  145.76 m3/hour
[2019/07/26 16:20] Receives packet with topic ‘‘pumpflow’’ and payload:  148.31
Received pump flow:  148.31 m3/hour
```

Figura 5.15: Registro del suscriptor.

```
[2019/07/26 16:30] Packet sent with topic ‘‘pumpflow’’ and payload:  147.50
[2019/07/26 16:31] Packet sent with topic ‘‘pumpflow’’ and payload:  149.98
[2019/07/26 16:32] Packet sent with topic ‘‘pumpflow’’ and payload:  149.55
[2019/07/26 16:33] Packet sent with topic ‘‘pumpflow’’ and payload:  146.85
[2019/07/26 16:34] Packet sent with topic ‘‘pumpflow’’ and payload:  149.92
```

Figura 5.16: Registro del sensor.

```
[2019/07/26 16:32] Message received with topic:  pumpFlow
Packet latency is not acceptable:  2 (projected) vs 1 (expected)
[2019/07/26 16:33] Message received with topic:  pumpFlow
Packet latency is not acceptable:  2 (projected) vs 1 (expected)
[2019/07/26 16:34] Message received with topic:  pumpFlow
Packet latency is not acceptable:  2 (projected) vs 1 (expected)
```

Figura 5.17: Registro del broker.

Exp.	**Descripción**	**Solución con SRTI-IRTA**	**Solución con MQTT**	**Beneficios de usar SRTI-IRTA**
1	Nuevo tópico basado en los valores de otros tópicos.	El broker utiliza una función para procesar, crear y publicar un mensaje nuevo en un tópico nuevo.	Se necesita un cliente adicional. Debe suscribirse a los tópicos de interés, procesar los valores recibidos y publicar un nuevo mensaje bajo un nuevo tópico.	(1) Menos tráfico en la red; (2) no se necesita ningún cliente adicional, evitando la sobrecarga de tener otro nodo.
2	El publicador envía un mensaje con más frecuencia de lo que necesita el suscriptor.	El broker filtra los mensajes recibidos en función de las necesidades del suscriptor.	El broker enviará todos los mensajes recibidos, y será tarea del suscriptor descartarlos.	(1) Menos tráfico en la red; (2) menos procesamiento en el suscriptor.
3	La latencia del publicador es mayor que la requerida por el suscriptor.	El broker filtra los mensajes recibidos en función de las necesidades del suscriptor.	El broker enviará todos los mensajes recibidos, y será tarea del suscriptor analizarlo y descartarlo.	(1) Menos tráfico en la red; (2) menos procesamiento en el suscriptor.

Cuadro 5.2: Comparación de las soluciones implementadas utilizando Message Queue Telemetry Transport (MQTT) y Software Real-Time Interaction (SRTI) -IRTA.

tiempo real entre los componentes del sistema para detectar rápidamente problemas, como inundaciones en partes del terreno o paradas de las bombas de agua cuando la plantación se encuentra en situaciones críticas (por ejemplo, en riesgo de heladas). El cultivo de papas, cebollas o kiwis es especialmente vulnerable a estas situaciones.

El manejo de las restricciones en tiempo real para mejorar las soluciones de IoT para AP sigue siendo un tema abierto. Aunque MQTT es un protocolo apropiado para soportar las interacciones entre los componentes de un sistema de AP, no proporciona garantías en tiempo real para estas interacciones. Por tanto, la extensión propuesta en este trabajo representa una forma de mejorar la QoS de mensajes críticos en soluciones de IoT que utilizan redes de comunicación inestables.

En los lotes particulares donde está el cultivo, se despliega un conjunto de sensores enterrados que miden las condiciones climáticas en el aire y la tierra. Para ello se pueden medir la temperatura, los rayos ultravioleta y su incidencia, la humedad y muchas otras variables. Estos datos deben muestrearse a lo largo del lote en diferentes lugares, ya que los lotes pueden tener varios miles de metros cuadrados. Estos sensores publican sus datos al broker que mantiene actualizados a los suscriptores. El proceso de recuperación de la información debe realizarse con garantías de tiempo, ya que para tomar una decisión es importante conocer la situación actual de todo el lote. Por ejemplo, si bien el riego podría ser suficiente en una parte del lote, otra parte aún puede requerir riego.

Los sensores desplegados en el campo están equipados con transceptores compatibles con IPv6 y MQTT. Estos sensores publican sus variables medidas en un servidor que implementa un broker MQTT. Los sensores se basan en microcontroladores simples de 8 bits con pequeñas memorias RAM y ROM. Como el escenario propuesto funciona a cielo abierto, se alimentan con células solares que proporcionan energía y cargan las baterías para el funcionamiento nocturno.

Por lo general, contar con una estación meteorológica es suficiente para proporcionar la información meteorológica requerida para un lote completo (por ejemplo, dirección e intensidad del viento, temperatura y humedad). Esta unidad puede proporcionar los datos de forma independiente o integrarlos en un paquete que incluya, por ejemplo, valores de humedad del suelo, temperatura, pH y nitrógeno. Todos los valores muestreados por los sensores en el lote se utilizan para construir mapas de gradiente en las diferentes variables.

Se pueden distribuir otros sensores en el lote para proporcionar información sobre la altura de las plantas y el color y estado de las hojas. Para ello, las cámaras multiespectrales pueden revelar información sobre la salud de los cultivos. Este tipo de información también se puede evaluar periódicamente con fotografías tomadas por drones.

El esquema se repite para cada lote; por lo tanto, puede haber una cantidad importante de información que deba procesarse. Los lotes no están necesariamente cerca unos de otros; de hecho, pueden estar separados por varios kilómetros. Los sistemas IoT desplegados de esta forma facilitan el control y supervisión de los lotes, incluso cuando los responsables de los mismos no están presentes en la zona. Así, el productor agrícola puede tomar decisiones basadas en datos en tiempo real. Por ejemplo, si hay demasiada humedad en ciertas partes del lote pero no lo suficiente en otra, el agricultor puede decidir realizar un riego selectivo.

En este tipo de aplicaciones, aunque los plazos no sean estrictos, el correcto funcionamiento en tiempo real del sistema es un requisito. Este aspecto debe tenerse en cuenta en el momento del diseño del sistema. Los sensores que funcionan con baterías recargadas por

paneles solares deben considerar la demanda de energía en el momento de enviar datos que han quedado desactualizados.

La Figura 5.18 ilustra un escenario de AP en el que se utiliza un sistema de IoT para el control en tiempo real de las condiciones de cultivo. La muestra considera tres lotes; cada uno tiene 800m × 500m. Dentro de cada lote, hay varias líneas de riego que se cruzan en diferentes puntos por líneas de sensores. Un sensor multiparamétrico equipado con un transceptor se despliega en cada intersección de las líneas del sensor. Si bien el lote puede parecer regular, puede haber una diferencia de elevación de 15 m entre una parte y otra, lo que puede provocar que una zona se inunde con más frecuencia o más rápido que otras.

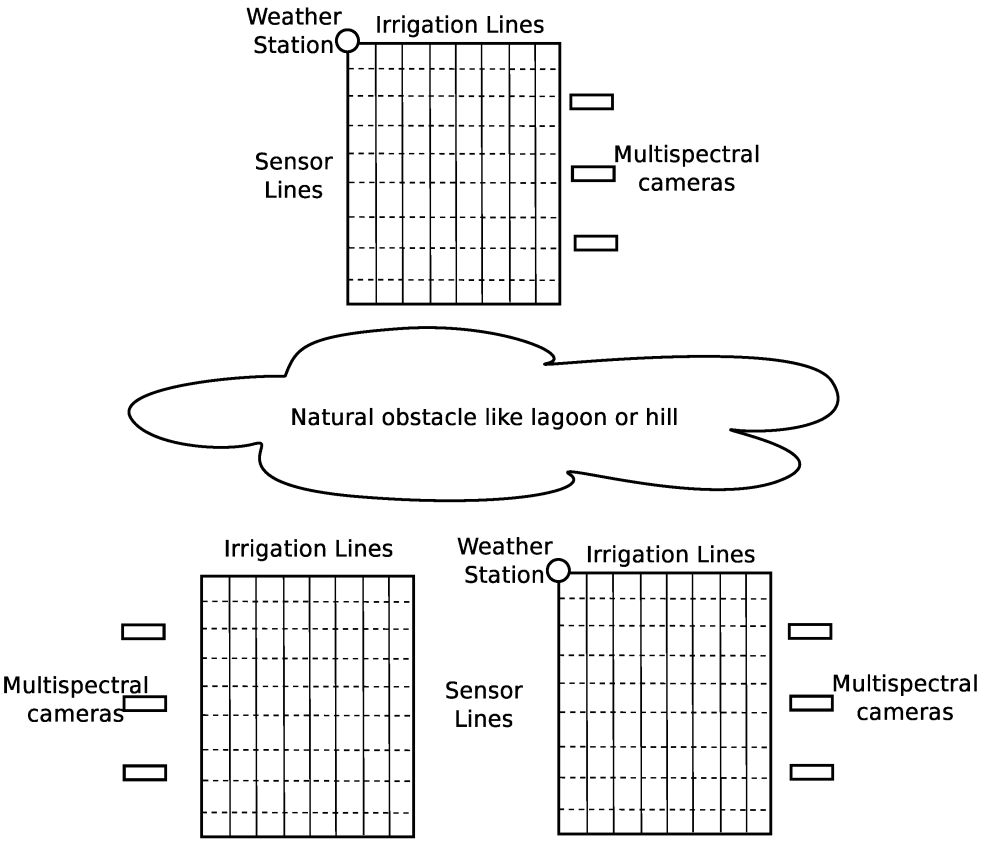


Figura 5.18: Despliegue de sensores en un escenario hipotético de agricultura de precisión (AP).

Otro ejemplo que se puede encontrar en escenarios de AP es tener un lote con dos o más tipos de suelo. Esto significa que los nutrientes en cada parte y la cantidad de agua para dichas áreas deben manejarse de manera diferente. La Figura 5.18 muestra una distribución esquemática de los sensores, que deben ubicarse cuidadosamente para garantizar que la información necesaria se recopile y distribuya a través del sistema. La distribución óptima de sensores no se analiza en este artículo; por lo tanto, podría suceder que dos sensores que lean el mismo valor produzcan reacciones diferentes en el sistema, ya que se colocan en ubicaciones con requisitos distintos, incluso si ambos están dentro del mismo lote.

El transceptor del sistema usa una pequeña antena que opera sin la necesidad de estar

Variable	Tópico
Temperatura	sijk/temperature
pH	sijk/pH
Humedad	sijk/moisture
Nitrogeno	sijk/nitrogen

Cuadro 5.3: Tópicos para los sensores dentro del escenario loT.

Variable	Tópico
Temperatura	wsi/temperatute
Dirección de viento	wsi/direction
Intensidad de viento	wsi/intensity
Humedad	wsi/humidity
Presión	wsi/pressure
Factor UV	wsi/uv

Cuadro 5.4: Tópicos para las estaciones climáticas.

en el sitio del receptor. Las estaciones meteorológicas pueden actuar al mismo tiempo como gateways a otra red, como la de telefonía móvil con tecnología 4G. En este caso, los gateways actúan como brokers MQTT, y toda la información proveniente de los sensores se recopila y procesa en el IRTA. Las suscripciones a los brokers pueden realizarlas otros gateways, servidores de procesamiento o usuarios finales.

Si bien los dos lotes en la parte inferior de la Figura 5.18 pueden usar la misma estación meteorológica, el tercer lote tiene la suya propia. Con esto, destacamos el hecho de que, en primer lugar, los sensores pueden estar en escenarios naturales bastante grandes en los que las condiciones climáticas pueden variar y que, en segundo lugar, la cantidad de información en los dos primeros lotes puede ser excesiva para que un solo broker lo procese en tiempo real. Si el lote tiene una forma diferente, como la de la Figura 5.18, o no hay suficientes sensores disponibles para cubrir todo el lote de manera regular, será crucial analizar la estructura del lote e instalar sensores en diferentes áreas, agrupados por características comunes.

La grilla de sensores que se muestra en la Figura 5.18 se puede mapear dentro de los lotes usando una notación matricial, donde s_{ijk} identifica el sensor dentro del i-ésimo lote, en la j-ésima fila y en la k-ésima columna. Como suponemos que todos son similares dentro de una clase, cada sensor del lote tendrá un nombre general con la estructura mencionada en el Cuadro 5.3. En el caso de las estaciones meteorológicas, siguen la misma idea. Cada estación meteorológica se anota ws_i, y tiene asociado un lote o conjunto de lotes dependiendo de la geografía y ubicación. El Cuadro 5.4 muestra una posible asignación de tópicos.

Finalmente, las cámaras multiespectrales se anotan como msc_{ij}, donde i refleja el lote en el que la cámara está muestreando las plantas y j identifica la cámara individual dentro de ese lote. El tópico en este caso es simplemente "mscij/spectral", y proporciona las intensidades en cada longitud de onda de luz. Usando este esquema de nomenclatura de sensores, es posible cumplir con los requisitos de identificación del protocolo MQTT; en particular, requiere la

identificación de los tópicos informados por cada sensor.

Con la red implementada y la conexión en funcionamiento, el usuario puede requerir cualquier nodo dentro del ecosistema de IoT para obtener la información adecuada. Aunque se puede considerar al usuario, el sistema también podría funcionar de forma autónoma e implementar algún tipo de algoritmo de decisión para determinar si es necesario, por ejemplo, regar todo el lote, regar parte del mismo o no regar. La misma idea se puede aplicar para esparcir herbicidas o pesticidas en el lote o para realizar cualquier otra acción para maximizar la producción de cultivos.

5.3.6. Discusión

El modelo presentado está orientado a la gestión de calidad de servicio en tiempo real blando basado en comunicaciones TCP/IP con MQTT en la Internet abierta. Aunque existe un uso cada vez mayor de sensores y dispositivos finales para proporcionar información real sobre diferentes áreas, la mayoría de las aplicaciones son cerradas y no funcionan en la Internet abierta sino en redes dedicadas. Por el momento, el ancho de banda disponible proporcionado por los proveedores de servicios de Internet no es suficiente para tener una red de topología malla que incluya a todos los objetos conectados. Se están desarrollando varias estrategias para permitir conectar cosas en una red sin la intervención de personas, para que sean realmente M2M. Normalmente, estas estrategias están orientadas a construir redes alternativas que permitan conectar algunos nodos a Internet a través de una puerta de enlace. Como se mencionó antes, la bibliografía informa sobre varios sistemas y arquitecturas para admitir interacciones entre componentes de IoT, pero la mayoría de estas propuestas muestran limitaciones similares para admitir interacciones en tiempo real. Algunos de ellos consideran redes inestables y otros involucran QoS en tiempo real, pero ninguno de ellos admite ambas funciones simultáneamente. Esta situación representa un obstáculo para el desarrollo de sistemas inteligentes que deben garantizar la entrega de mensajes a tiempo pero que funcionan en redes con una inestabilidad incierta. Casi cualquier sistema de monitoreo remoto debe abordar estos dos requisitos simultáneamente.

La extensión propuesta a MQTT brinda la posibilidad de discriminar entre mensajes que pueden cumplir con las restricciones de tiempo de aquellos que no van a cumplir con los plazos. Por el momento, MQTT proporciona una forma de enviar los datos producidos por los publicadores a aquellos suscriptores potencialmente interesados. El broker actúa como un directorio telefónico que proporciona al suscriptor los datos almacenados en la cola de tópicos sin procesarlos. En esta extensión, el broker incorpora un agente que tiene la capacidad de clasificar y agregar datos para generar nueva información. Ambos aspectos, el control en tiempo real y la incorporación del agente inteligente, reducen el tráfico de la red al evitar la transmisión de mensajes desactualizados o no interesantes.

5.4. Contribuciones de código abierto

MQTT se ha implementado utilizando brokers y librerías como Mosquitto [107] y Mosca [108]. Tomando como base el segundo, se han implementado las extensiones que agregan el IRTA. En [72], el código de esta implementación se puede descargar para instalar la extensión.

Además, este trabajo también pone a disposición un ejemplo que muestra la forma en que un suscriptor solicita un valor que se obtiene procesando dos publicaciones en el broker. En [109], se encuentra disponible una interfaz gráfica simple para configurar la aplicación, lo que le permite implementar diferentes funciones IRTA que pueden ser agregadas por el administrador del sistema.

5.5. Consideraciones finales

En este capítulo se presentó el modelo arquitectónico de Simple Real Time Interaction (SRTI), que amplía el esquema de interacción propuesto por el protocolo MQTT para hacer frente a la entrega de mensajes que tienen restricciones temporales. La propuesta fue diseñada particularmente para abordar escenarios de IoT donde el enlace de comunicación es inestable o su estabilidad es incierta.

Si bien IoT está emergiendo como el próximo paradigma de redes, todavía hay varios problemas abiertos relacionados con el descubrimiento de servicios, el consumo de energía, la congestión del tráfico y el enrutamiento de mensajes. Mientras tanto, como en el pasado, hay una carrera para ganar el mercado con la tecnología adecuada en las capas física y de enlace. En América del Sur, en este momento, LoRa es la tecnología que está siendo adoptada por la mayoría de los desarrolladores de aplicaciones de IoT. Se ha implementado el protocolo extendido MQTT (utilizado comúnmente como complemento a LoRa) con primitivas en tiempo real. La propuesta se puede utilizar en diferentes áreas de IoT, pero especialmente en aquellas relacionadas con sistemas de monitoreo que requieren tiempos de respuesta acotados y operan con baterías.

El modelo SRTI incorpora el Intelligent Real-Time Agent (IRTA) para procesar los datos publicados y lo adapta a las demandas de las suscripciones cuando es posible. Dicho modelo se implementó a través de una infraestructura de software denominada SRTI-IRTA, cuyo desempeño se evaluó en tres escenarios de IoT diferentes. Los resultados indican que dicha infraestructura ayuda a abordar la entrega de mensajes en tiempo real blando en redes inestables o en aquellas donde su estabilidad es incierta. Las capacidades de SRTI-IRTA se compararon con las de una implementación reciente del protocolo MQTT, mostrando que el primero supera al segundo en los escenarios de evaluación.

Finalmente, se ha mostrado cómo esta extensión de MQTT puede usarse en agricultura de precisión para monitorear las variables climáticas y las condiciones químicas del suelo, y para controlar el riego y los productos agroquímicos utilizados para maximizar la productividad de los cultivos. En el sistema propuesto para este dominio de aplicación, las cámaras espectrales y las imágenes obtenidas por drones proporcionan información adicional.

Los próximos pasos de esta iniciativa contemplan la realización de varias simulaciones para determinar los límites de esta propuesta. Luego, se implementarán diversas aplicaciones de IoT para ver que los beneficios identificados en las simulaciones también están presentes en los sistemas desplegados en escenarios reales.

Capítulo 6

Políticas de modelado de tráfico para LoRaWAN

6.1. Motivación

Conectar a Internet dispositivos que operan en áreas remotas, no desarrolladas, es una rama de Internet de las Cosas también conocida como Internet de las Cosas Remotas (IoRT por su sigla en inglés) [110]. Algunas aplicaciones de IoRT incluyen agricultura (por ejemplo, se pueden instalar sensores en un campo para monitorear cultivos), monitoreo ambiental (por ejemplo, monitoreo ambiental para detectar avalanchas o erupciones de volcanes) y plataformas petrolíferas (por ejemplo, medir el funcionamiento de la maquinaria para predecir potenciales fallas). Aunque hay muchos casos de uso diferentes, muchas de estas implementaciones consisten en solo unos cientos de dispositivos dispersos en amplias áreas geográficas, lejos de los centros urbanos.

En las aplicaciones de IoRT, la infraestructura de telecomunicaciones terrestre suele ser inestable o inexistente. Esto suele ser más frecuente en los países en vías de desarrollo, donde existe una gran diferencia en la infraestructura de banda ancha entre las zonas más pobladas y el resto del país, que terminan dependiendo de tecnologías satelitales o celulares obsoletas (por ejemplo, 2G) [111]. E incluso si la infraestructura de red está disponible en el sitio, la aplicación de IoRT puede requerir, dependiendo de su criticidad, un backhaul que no se vea afectado por interrupciones del servicio causadas, por ejemplo, por condiciones ambientales adversas. En estos casos, las comunicaciones por satélite facilitan la creación de redes alternativas que puedan solucionar esta falta de conectividad terrestre [110]. Sin embargo, el IoT satelital y otros proveedores de redes de IoT alternativos generalmente ofrecen un servicio de ancho de banda limitado (por ejemplo, mensajes fijos o MB por mes) y/o son más costosos que los servicios comunes utilizados en áreas urbanas, como las tecnologías celulares modernas o la fibra óptica proporcionada por ISP en ciudades [112].

Redes LPWAN como Sigfox, NB-IoT o LoRaWAN parecen ser opciones adecuadas para este tipo de redes. Permiten que dispositivos con bajo consumo (una característica clave ya que es posible que no se disponga de una fuente de energía estable) puedan conectarse a una red con muy poca infraestructura y una amplia cobertura de comunicación (un Gateway puede alcanzar el rango de enlace de 10 km o más) [113]. Dado que LoRaWAN es un protocolo

abierto que permite implementaciones privadas, tiene la ventaja, sobre los otros dos, de que no depende de concentradores o Gateways de terceros. Esto es especialmente importante si se tiene en cuenta el hecho de que, debido a la pequeña cantidad de dispositivos conectados a la red, puede que no sea interesante, desde un punto de vista comercial, que los proveedores de red proporcionen Gateways a estas áreas remotas.

Aunque las especificaciones de LoRaWAN asumen que el backhaul es compatible con IP, potencialmente puede funcionar con otros protocolos. Sin embargo, cuando se trabaja con redes de backhaul limitadas como las que se pueden encontrar en IoRT, presenta algunos desafíos. Por ejemplo, casi todas las decisiones de red tienen lugar en un servicio denominado LoRaWAN Network Server, comúnmente implementado en la nube. Esto significa que la mayor parte del tráfico se reenvía a través del backhaul, incluso paquetes que serán descartados una vez que lleguen al Network Server o a la aplicación. Además, LoRaWAN no contempla que pueda haber algunos paquetes o nodos que deban tener más prioridad que otros, lo que se vuelve significativo cuando el ancho de banda de backhaul se limita a una cantidad fija de mensajes en un período.

En este trabajo, se propone un Gateway LoRaWAN que tiene como objetivo reducir el tráfico enviado al backhaul de la red, mediante el filtrado local de paquetes en base a diferentes políticas de modelado de tráfico definidas por un controlador ubicado junto o dentro del Network Server. Está diseñado en el contexto de pequeñas redes LoRaWAN privadas con backhauls limitados y dispositivos fijos (o con movilidad limitada), como los que se pueden encontrar en IoRT.

6.1.1. Descripción general de LoRaWAN

LoRaWAN es un protocolo de comunicación diseñado para conectar dispositivos IoT a un backend en la nube. Está compuesto por una capa física, que utiliza una modulación de radio llamada LoRa basada en chirp de espectro extendido y patentada por Semtech, y una capa MAC que define un protocolo de red de código abierto.

Una característica clave que afecta el rendimiento de LoRa es el número de chips por símbolo de datos, llamado Spreading Factor en inglés y abreviado como SF. Se configura en el End Device y puede ir de 7 a 12. Cuanto más alto sea, más probable es que un Gateway escuche el End Device, pero también consumirá más batería y el tiempo de transmisión será más lento.

En redes LoRaWAN, generalmente se encuentran 4 elementos claves. Estos se ilustran en la Figura 6.1 y se describen a continuación:

1. *End Devices:* son los dispositivos (sensores o actuadores) que reciben y transmiten a los Gateways usando la modulación LoRa.

2. *Gateways:* son las puertas de enlace, que reenvían paquetes entre los End Devices y el Network Server. Actúan como un puente entre el radio LoRa y el protocolo de backhaul (IP).

3. *Network Server:* a cargo de toda la gestión de la red. Se puede incluir en el mismo nodo el Join Server, que se encarga de registrar los End Devices en la red.

4. *Application Server:* gestiona los datos de la aplicación del sensor/actuador.

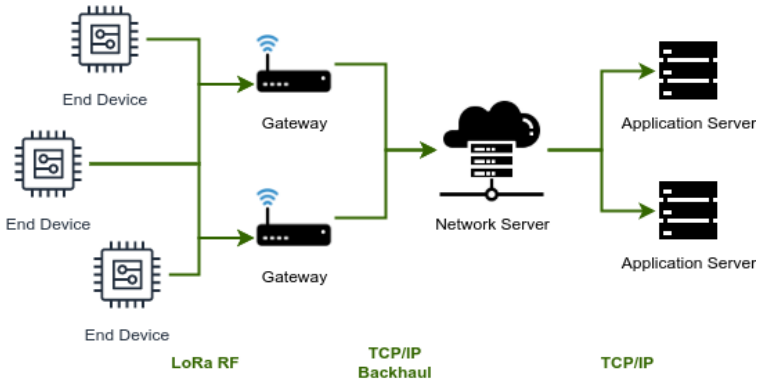


Figura 6.1: Un diagrama de referencia de una arquitectura de red LoRaWAN.

La activación de los End Devices en la red se realiza mediante ABP (Activation by Personalization) u OTAA (Over-the-Air Activation). En ABP, el dispositivo ya ha sido configurado con una dirección fija y claves de sesión (necesarias para establecer comunicación) antes de la instalación. En OTAA, estos se asignan dinámicamente en un proceso llamado Join-Procedure, que involucra al Join Server mencionado anteriormente.

6.2. Estado del arte

Los desafíos de adaptar LoRaWAN a redes con backhaul limitado se han tratado en diferentes trabajos. En [114], se centran en las comunicaciones por satélite y presentan tres problemas principales: la latencia de la red que puede ser mayor que los tiempos de espera de ACK, la duplicación de paquetes que puede cargar el backhaul con múltiples copias del mismo mensaje y la sobrecarga en la red que pueden introducir los protocolos de serialización de la capa aplicación (como JSON). Este último también se analiza en [115, 116], donde proponen una técnica de compresión basada en Protocol Buffers.

En [117] se introduce una arquitectura LoRa-SDN que sienta las bases para procesar paquetes a nivel de Gateway. Software Defined Networks (SDN) es un sistema de software de arquitectura en capas que tiene como objetivo hacer que la administración de la red sea más fácil y escalable. En este modelo, un conmutador SDN, que es un software que se ejecuta en conmutadores de red (switches), se instala en un Gateway LoRa para filtrar paquetes. Un controlador SDN se instala junto con el Network Server para que pueda sincronizar los conmutadores SDN. Incluso si el modelo general puede funcionar con LoRa, su diseño no es compatible con LoRaWAN, ya que asume que puede identificar nodos a nivel de Gateway, algo que el protocolo no ofrece.

Fog computing en LoRaWAN también ha sido objeto de una intensa investigación. Los autores de [118], [119] y [120] proponen, en resumen, dos grupos de arquitecturas que, aunque válidas, no se abordan en este trabajo. En el primero, las claves de seguridad de la aplicación y la red LoRaWAN deben descargarse a los Gateways, para que puedan acceder a los datos del paquete, pero presenta algunos riesgos ya que compromete la seguridad del sistema.

En el segundo, algunos componentes del sistema que suelen estar en la nube se instalan en los Gateways, como el Network Server de LoRaWAN y las bases de datos de la aplicación. Dependiendo de los requisitos, esto puede no parecer adecuado para algunas aplicaciones de IoRT, ya que concentrar la potencia computacional en la nube es generalmente más económico y más fácil de mantener y actualizar. Además, tener una capa gruesa de procesamiento en los Gateways requeriría más recursos computacionales y un mayor consumo de energía.

Otros trabajos, como [121] y [122] tienen como objetivo optimizar el tráfico y el consumo de energía de los dispositivos asignando QoS a los nodos según los requisitos de la aplicación. Se centran principalmente en ajustar la configuración de la parte de radio de la red, que está fuera del alcance de este trabajo.

En [123] los autores proponen un agente que procesa cada paquete en un nodo intermedio, después de su envío y antes de llegar al destino. Este agente puede filtrar, agregar o transformar datos de forma dinámica. Aunque está en el contexto de las comunicaciones en tiempo real y para otros protocolos de comunicación, los principios básicos son similares a los introducidos en este trabajo.

6.3. Propuesta

El Gateway NA-LoRaWAN propuesto en este trabajo se diferencia de los tradicionales porque es capaz de procesar el tráfico en función del End Device transmisor. Nos referimos a esta característica como "Node-Aware", abreviado como NA. Para que un Gateway sea NA, primero debe identificar de forma única qué nodo ha enviado un paquete entrante y luego aplicar un conjunto de reglas para decidir qué se debe hacer con este paquete: reenviarlo o descartarlo. Tener la capacidad de filtrar paquetes por ID de nodo permite no solo aplicar filtros basados en el remitente, sino también clasificar dispositivos en grupos y procesarlos en función de esa clasificación (por ejemplo, crear niveles de prioridad y tratar cada nivel de una manera diferente). Esta es la principal diferencia con un Gateway LoRaWAN tradicional, que, en general, son solo reenviadores de paquetes. Este contraste se ha ilustrado en la Figura 6.2, donde el Gateway en la parte superior reenvía todos los paquetes, mientras que el Gateway en la parte inferior procesa los paquetes y solo reenvía algunos de ellos.

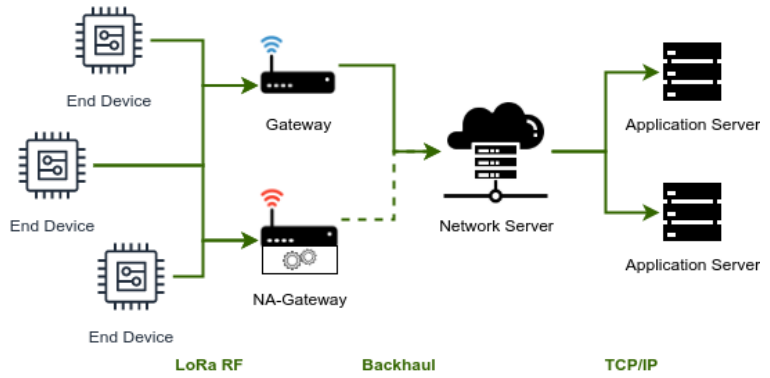


Figura 6.2: Un diagrama de referencia de una arquitectura de red LoRaWAN con un Gateway NA.

Esta sección se divide en dos subsecciones. La primera esboza la arquitectura de la propuesta. La segunda propone una forma de identificar de forma única los nodos en el Gateway.

6.3.1. Arquitectura

Se adopta un enfoque simple inspirado en SDN para rediseñar la interacción entre los Gateways y el Network Server. En la Figura 6.3 se muestra un diagrama de arquitectura de referencia, que se basa en una solución SDN más completa que se puede encontrar en [117].

Los Gateways cuentan con un software de conmutación SDN básico, que periódicamente recibe actualizaciones con la información que necesita para operar, como la tabla de flujo. El software del controlador SDN se instala en un servidor junto con el Network Server y cuenta con permisos para leer sus datos.

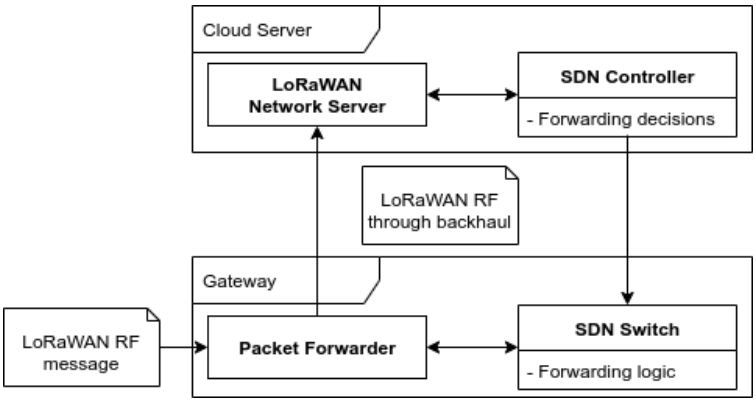


Figura 6.3: Un diagrama de referencia de una arquitectura LoRaWAN-SDN.

6.3.1.1. Identificación de un End Device

No existe en LoRaWAN una forma estándar para que un Gateway asocie un determinado paquete con un End Device único. Este trabajo lo realiza el Network Server, que busca el End Device correspondiente en una base de datos de back-end, en base a la firma criptográfica del mensaje recibido. Dado que el Gateway no conoce la clave criptográfica utilizada (Network Session Key) y no tiene acceso a la base de datos de dispositivos, no puede hacer coincidir el mensaje con un dispositivo. Otra alternativa para el Gateway sería utilizar el DevEUI, que es un identificador único asignado por el fabricante del chip (similar a una dirección MAC). Sin embargo, como solo se envía durante el proceso de unión en OTAA, no será accesible nuevamente en paquetes futuros ni en OTAA ni en ABP y, por lo tanto, el Gateway no conocerá el DevEUI de cada paquete de datos entrante. Finalmente, está DevAddr, que es una dirección codificada en el dispositivo (ABP) o asignada por el Network Server (OTAA). Es efímero y no único, por lo que pueden existir muchos dispositivos con el mismo DevAddr en la misma red, lo que hace que sea inútil para identificar nodos unívocamente. Aún así, si se cambia el proceso de asignación de DevAddr para garantizar que sea único por dispositivo, entonces, en algunos escenarios como los que apunta este trabajo, este atributo sería suficiente ya que está presente en cada paquete que pasa por el Gateway. Esta solución solo sería útil

en aquellas redes donde los nodos tienen poca o nula movilidad (dentro de la cobertura del mismo Gateway).

Un DevAddr en LoRaWAN es una dirección de 32 bits compuesta por un prefijo NetID Type, un NwkID y un NwkAddr. En las redes privadas (definidas en el campo NetID Type), hay espacio para 25 bits de NwkAddr [124] [125] únicos, lo que da como resultado millones de direcciones disponibles. Esto es más que suficiente para unos pocos cientos de nodos como las redes a las que se dirige este trabajo.

Para OTAA, si el Network Server asigna uniformemente el DevAddr, se logra la unicidad de DevAddr. Una vez que se alcanza el límite de DevAddr disponibles, el Network Server podría finalizar las sesiones más antiguas (por ejemplo, un pequeño porcentaje del total de direcciones asignadas) para eliminar de sus registros posibles nodos inactivos mientras libera espacio para aceptar nuevos.

Para ABP, no se requieren cambios ya que el DevAddr se asigna manualmente. Sin embargo, los integradores de dispositivos deben tener especial cuidado para no superponer End Devices con el mismo DevAddr.

Esta solución inutilizaría el roaming, ya que en LoRaWAN este no admite redes privadas. Sin embargo, para los casos de uso de IoRT a los que se dirige este trabajo, esto puede ser una ventaja, ya que facilita el control del tráfico que se envía al backhaul (característica que también ofrecen algunos Gateways LoRaWAN).

6.3.2. Políticas de modelado de tráfico

Identificar el ID de nodo de cada paquete y utilizar la arquitectura propuesta anteriormente, habilita que el Gateway analice y controle el tráfico que recibe y envía. En esta sección, se presentan cuatro políticas para reducir el tráfico enviado al backhaul.

6.3.2.1. Filtrar el tráfico por ID de nodo

Cada conmutador SDN en el Gateway podría mantener una lista blanca local (similar a una tabla de flujo en SDN) de DevAddr para reenviar al Network Server. Esta lista será actualizada por el controlador SDN, descargando un comando a un Gateway cuando, o bien el administrador de red configura un nuevo dispositivo ABP, o cada vez que un nuevo dispositivo concluye un Join Procedure exitoso en OTAA. Si dos Gateways envían un Join Procedure desde el mismo dispositivo, el controlador SDN elegirá el que tenga el RSSI más bajo. En el caso de ABP, depende del administrador de la red decidir qué Gateway retransmitirá cada End Device.

Los conmutadores SDN informan su estado periódicamente (por ejemplo, una vez al día) y el controlador SDN mantiene un registro del estado de todos sus Gateways. Si un Gateway no informa su estado y ya no envía paquetes, el controlador SDN podría migrar o distribuir su lista blanca a los Gateways cercanos.

Al mantener actualizada la lista blanca de paquetes para reenviar, sin repetir el DevAddr entre ellos, los Gateways solo transmitirán una vez cada mensaje al Network Server, evitando la duplicación.

6.3.2.2. Filtrar el tráfico por prioridad de nodo

Si el controlador SDN asocia cada nodo a un nivel de prioridad (por ejemplo, de 1 a 5, siendo 1 el más importante y 5 el menos), podría indicar al Gateway que solo reenvíe paquetes de los niveles de prioridad más altos cuando necesite ahorrar ancho de banda del backhaul. De forma predeterminada, cada nodo tiene la misma prioridad y cada Gateway reenvía a todos los nodos sin importar sus prioridades. Sin embargo, el administrador de la red podría cambiar estas dos propiedades. Una vez que se cambia el nivel de prioridad para reenviar, el Network Server descarga un comando a cada Gateway actualizando su configuración.

6.3.2.3. Filtrar el tráfico por número de paquetes

Otra técnica potencialmente útil para filtrar el tráfico sería limitar el número de paquetes que se reenviarán en un período, independientemente de su tamaño. Este período puede ser de minutos, horas, días, etc. Dos casos de uso para usar esta política podrían ser cuando el proveedor de red de backhaul ofrece una cantidad limitada de paquetes para transmitir a través de su infraestructura, o cuando la aplicación en la nube no requiere leer cada paquete entrante, pero sólo algunos de ellos periódicamente. Un ejemplo de este último caso podría ser si una aplicación está leyendo sólo 1 muestra cada hora de un valor de temperatura, pero los sensores en campo lo transmiten cada 30 minutos, la mitad de los paquetes no son consumidos por la aplicación, por lo que pueden ser filtrados antes de ser reenviados.

Cuando se registran las aplicaciones LoRaWAN, el administrador de la aplicación puede indicar opcionalmente el período y la cantidad de paquetes que los End Devices pueden transmitir (si no se completa esta información, se supone que la aplicación puede recibir datos en cualquier momento). Luego, esta información se agrega para todos los dispositivos y se descarga en los conmutadores SDN. Por lo tanto, si un nodo envía mensajes a una frecuencia superior a la permitida, el Gateway filtrará esos paquetes adicionales localmente.

6.3.2.4. Filtrar el tráfico por presupuesto

Si no se envía información crítica, el controlador SDN podría limitar la cantidad de bytes enviados al backhaul, descargando un presupuesto en bytes a cada Gateway. Este presupuesto puede ser por nodo o por Gateway. Cada vez que el Gateway recibe un paquete, primero verifica que el presupuesto disponible sea mayor que el tamaño del paquete y, si es así, resta el tamaño al presupuesto correspondiente. Cuando el presupuesto disponible es menor que el tamaño del paquete, no lo reenvía y lo rechaza.

6.3.3. Diseño de extensión de LoRaWAN

La implementación de NA-LoRaWAN requiere la extensión de las funcionalidades del Gateway, pero no de los End Devices. Es por esto que los End Devices pueden trabajar indistintamente con Gateways NA o Gateways simples. En el nivel del Network Server, también se deben agregar algunas funcionalidades para implementar el protocolo NA.

A continuación, se introduce un diseño general que describe los cambios y adiciones a LoRaWAN, con el objetivo de permitir la implementación de las 4 políticas de modelado de tráfico presentadas. Como se indicó en las anteriormente, el diseño está inspirado en SDN,

NODE-AWARE VERSION (4 bits)	PACKET TYPE (4 bits)	TIMESTAMP (48 bits)	PAYLOAD (variable length)
0 (current version)	0 (NODE-AWARE ACTIVATION)	Milliseconds since epoch	Optional. Contains specific data that depends on the packet type.
	1 (NODE_AWARE_DEACTIVATION)		
	2 (RESET)		
	4 (REMOVE_DEVADDR_WHITELIST)		
	5 (UPDATE_DEVADDR_WHITELIST_ACK)		

Cuadro 6.1: Estructura de paquetes propuestos

donde hay un software similar al controlador SDN instalado junto con el Network Server, mientras que hay un software similar al conmutador SDN instalado en cada Gateway. Una descripción completa del protocolo de intercambio controlador-conmutador SDN está más allá del propósito de este trabajo, pero se presentan las ideas principales. Toda la comunicación entre el Network Server y el Gateway se realiza entre el controlador SDN y el conmutador SDN. Además de la comunicación, se realizan algunas modificaciones en el Network Server, partiendo de la base de lo descrito en los documentos técnicos de LoRaWAN® Backend [124].

En algunos casos, la descripción no incluye una única definición precisa, sino que se dan algunas alternativas. Estos se consideran como "decisiones de diseño" que deben analizarse y definirse para cada caso de uso objetivo específico.

En la primera subsección, se presenta la activación y desactivación del modo NA en un Gateway. En la siguiente, se detalla el método para identificar los End Devices en los Gateways. Las siguientes 2 subsecciones describen la característica de tolerancia a fallas de los Gateways y cómo actualizarlos en caso de que sea necesario un cambio en la infraestructura de la red. Finalmente, las últimas cuatro subsecciones contienen el diseño de las cuatro políticas de modelado del tráfico.

6.3.3.1. Activación de modo NA

Los End Devices LoRaWAN deben ser compatibles con los Gateways estándar y NA. Esto permite a los diseñadores de redes encontrar el equilibrio adecuado entre la redundancia de paquetes y la optimización del tráfico. En una red podrían haber todos Gateways LoRaWAN estándar, todos Gateways NA o una combinación de ambos. Además, un Gateway debería poder convertirse en NA sobre la marcha, descargando el comando ACTIVATION del Network Server. Para cambiar el modo a Gateway estándar, se debe descargar un comando de DEACTIVATION. Los comandos, descritos en el Cuadro 6.1, se envían desde el controlador SDN al conmutador SDN que se encuentra en el Gateway.

Una vez que se desactiva un Gateway NA, este debe mantener almacenados toda la configuración asociada a este modo, en caso de que se reactive en el futuro. Además, en ambos modos, el Gateway debe procesar y responder a cada comando del controlador SDN. Para limpiar la configuración del Gateway, se puede enviar un comando RESET como se describe en el Cuadro 6.1.

6.3.3.2. Gestión de los DevAddr

No se indica en los documentos técnicos de LoRaWAN cómo el Network Server asigna un DevAddr en OTAA. Para garantizar que no se repitan, se puede utilizar el NetID Type para redes privadas, y un enfoque incremental simple de 1 unidad, comenzando desde 0,

para NwkAddr. Cuando no quede espacio para agregar más direcciones (se han consumido los 25 bits de NwkAddr), el Network Server debe enviar un mensaje ForceRejoinReq a los primeros DevAddr asignados para verificar si todavía están activos o no. Para aquellos que no respondan, su DevAddr se libera y se puede reutilizar. Para redes pequeñas, como las que se enfocan en este trabajo, no debería ser difícil encontrar dispositivos obsoletos entre las millones de direcciones que podría formar el protocolo, por lo que el mensaje ForceRejoinReq podría enviarse en lotes pequeños, como de 10 DevAddr. Para los dispositivos ABP, el administrador de la red debe almacenar sus direcciones manualmente y OTAA debe evitar usarlas para asignaciones/eliminaciones.

En OTAA, cada DevAddr se asigna a un Gateway NA en un proceso llamado "Node-Gateway Assignment", que se describe en el Diagrama de Actividades en la Figura 6.4. Los Gateways NA deben reenviar todos los paquetes Join Request (solicitudes de incorporación) recibidos. Al recibir Join Request duplicados, el Network Server es capaz de evaluar todos los Gateways que transmitieron el mismo paquete, y determinar cuál de ellos se encargará de reenviar el End Device correspondiente. Primero, se aplica el proceso "Deduplication", que consiste en transformar paquetes duplicados en una única transmisión de backend, como lo establece la especificación LoRaWAN. Se configura una ventana de tiempo denominada "Deduplication Window " en el Network Server que indica el tiempo de espera para que los nuevos paquetes entrantes equivalentes sean fusionados en uno solo. De los paquetes duplicados recibidos, los Network Servers deben seleccionar el paquete con el mejor RSSI del Gateway, que es similar a lo que hace el LoRaWAN Network Server de ChirpStack [126]. Después de seleccionar al mejor Gateway y asignar un DevAddr como se describió anteriormente, el Network Server debe almacenar en una base de datos el DevAddr, el DevEUI, la identificación de el Gateway y el último valor RSSI, y luego responder la solicitud sólo al Gateway elegido. Finalmente, si llegan más paquetes iguales fuera de la ventana de deduplicación (debido a los retrasos del backhaul), el Network Server debe verificar si su RSSI es mejor que el almacenado y, en ese caso, actualizar la base de datos con los nuevos datos. Cada vez que llega un nuevo paquete, se debe actualizar el RSSI en la base de datos.

En el caso de ABP, el administrador de red debe asignar manualmente en la base de datos el DevAddr con el Gateway correspondiente.

Una vez que se ha asignado un DevAddr a un Gateway (ya sea automáticamente en OTAA o manualmente en ABP), los Gateways involucradas deben actualizar su lista blanca de DevAddr. Para OTAA, este proceso se describe en la Figura 6.5. Primero, el Network Server debe comunicarse con el controlador SDN para actualizar la lista blanca del Gateway. A continuación, el controlador SDN tiene que descargar al conmutador SDN de el Gateway un paquete especial llamado ADD_DEVADDR_WHITELIST, que tiene la estructura de trama descrita en el Cuadro 6.1. Como carga útil del paquete, debería enviar DevAddr. Una vez que el conmutador SDN recibe el paquete, debe almacenarlo en una memoria no volátil, en una estructura similar a un diccionario que contenga el DevAddr asignado a una marca de tiempo. Para cada actualización realizada en la lista blanca con respecto a DevAddr, primero debe verificar que la marca de tiempo del paquete sea mayor que la almacenada; de lo contrario debería descartarlo. En el caso de que el Network Server migre DevAddr a un nuevo Gateway, el controlador SDN primero envía un paquete REMOVE_DEVADDR_WHITELIST (Cuadro 6.1) con DevAddr al Gateway anterior y, después de recibir un ACK, envía un paquete ADD_DEVADDR_WHITELIST al nuevo Gateway con la información de DevAddr. De esta

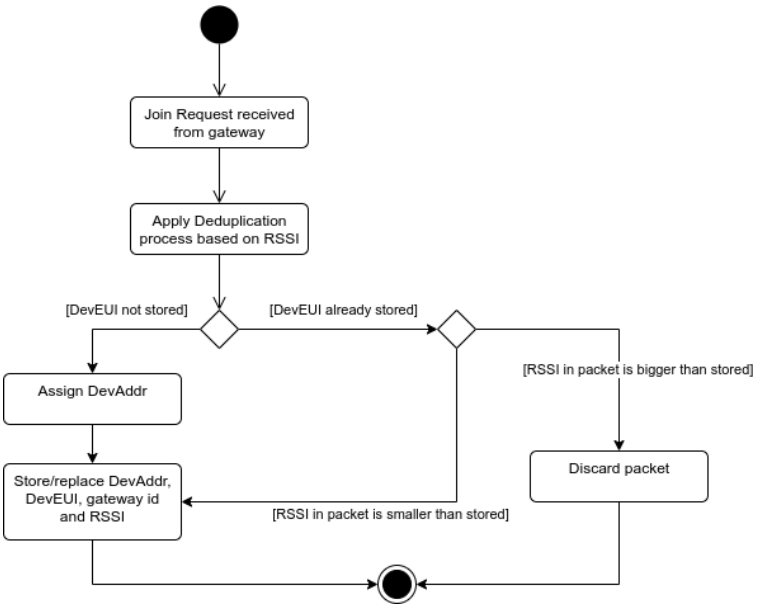


Figura 6.4: Proceso Node-Gateway Assignment (asignación Nodo-Gateway).

manera, el Gateway anterior eliminará primero el DevAddr de la lista blanca y el nuevo Gateway lo agregará. Otra posibilidad será que, dado que podría haber una pérdida de paquetes, este orden se invierta: primero agregue DevAddr en el nuevo Gateway y luego lo elimine del anterior.

Para confirmar la recepción, el conmutador SDN envía un paquete UPDATE_DEVADDR_WHITELIS al controlador SDN como se describe en el Cuadro 6.1, para cada ADD_DEVADDR_WHITELIST y REMOVE_DEVADDR_WHITELIST recibido del Network Server.

6.3.3.3. Tolerancia a fallas

Un aspecto importante en los Gateway LoRaWAN es que si uno de ellos deja de funcionar, los otros Gateways en ejecución siguen transmitiendo todos los mensajes, por lo que la redundancia puede mitigar la pérdida de paquetes causada por un Gateway defectuoso. Los Gateways informan su estado al Network Server mediante el envío de un heartbeat o un mensaje de keep-alice, que es una técnica común ya implementada en stacks populares de LoRaWAN como ChirpStack. En esta propuesta de NA, si un Network Server detecta que una Gateway está inactivo, debe desactivar todas los Gateways NA, transformándolos en reenviadores de paquetes (función básica de los Gateways estándar), de modo que los paquetes asignados al Gateway desconectado tengan la posibilidad de ser reenviados por otros. En esta situación, la recepción de paquetes por el Network Server tiene prioridad sobre la optimización del tráfico del backhaul y por lo tanto la estrategia de tolerancia es la misma que la del LoRaWAN estándar.

Si el Gateway defectuoso se restablece a un estado operativo, el controlador SDN activa de nuevo el modo NA en los Gateways correspondientes.

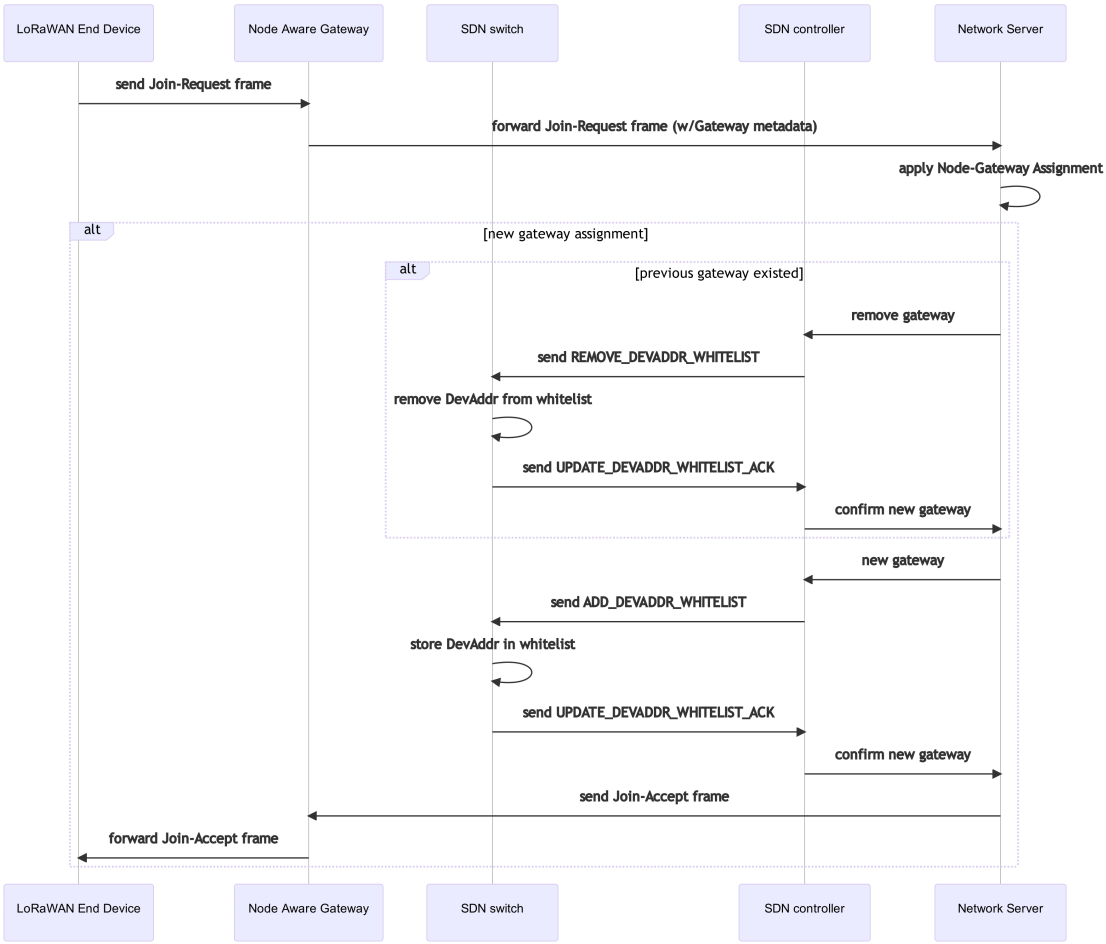


Figura 6.5: Secuencia de actualización de la lista blanca del Gateway.

6.3.3.4. Actualización de la infraestructura de Gateways

Si se modifica la infraestructura local, agregando/eliminando Gateways o cambiando su ubicación, el administrador de la red puede llevar a cabo una actualización de DevAddr manual o automática para redistribuirlos. Por un lado, para las actualizaciones manuales, los paquetes REMOVE_DEVADDR_WHITELIST y ADD_DEVADDR_WHITELIST se pueden enviar a través del controlador SDN, por lo que el DevAddr se migra de un Gateway a otro. Por otro lado, para las actualizaciones automáticas, el mensaje ForceRejoinReq se puede enviar a un grupo, o a todos los dispositivos si es necesario, por lo que se inicia un nuevo Join Procedure y se eligen los mejores Gateways nuevamente. Cuál de las dos opciones es mejor dependerá del análisis que realice el administrador de la red para cada escenario en particular.

6.3.3.5. Filtrar el tráfico por ID de nodo

Cuando un Gateway NA-LoRaWAN recibe un paquete entrante, primero debe verificar si es un paquete de Join Request o un paquete de datos. Para el primero, debe reenviarlo inmediatamente. De ser el segundo, primero debe verificar si el DevAddr se encuentra en su

lista blanca. Si es así, el paquete es reenviado; si no, se descarta. Este proceso se describe en la Figura 6.6.

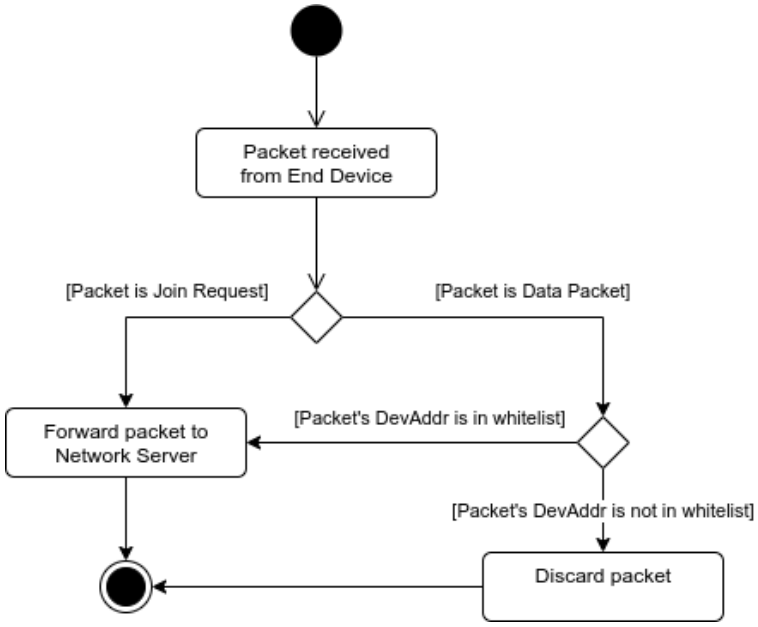


Figura 6.6: Filtrado de paquetes a nivel Gateway.

6.3.3.6. Filtrar el tráfico por prioridad de nodo

Los administradores del sistema deben asignar una prioridad a cada End Device, según los requisitos del caso de uso. Un campo de prioridad, enviado junto con el DevAddr en la carga útil del paquete ADD_DEVADDR_WHITELIST, indica el nivel de prioridad del dispositivo.

Además, también debe enviarse un paquete SET_PRIORITY durante la configuración de la red, que determina el nivel mínimo de prioridad para filtrar en el Gateway. Entonces, por ejemplo, si la carga útil del paquete SET_PRIORITY contiene el nivel 1, todos los paquetes provenientes de nodos con nivel 2 o superior serán descartados.

Cuando un paquete entrante llega a el Gateway, debe verificar, en memoria local, la prioridad del End Device y decidir si reenviarlo o descartarlo. Si el Gateway no contiene información de prioridad para ese End Device, podría por defecto reenviarlo o descartarlo. Esta es una decisión de diseño que puede variar según el caso de uso.

6.3.3.7. Filtrar el tráfico por número de paquetes

En este caso, el Gateway recibe del Network Server el número máximo de paquetes que se pueden reenviar en un período determinado (por hora, diario, etc.). Esto se transmite en la carga útil del paquete SET_MAX_PACKETS.

El Gateway debe filtrar localmente un paquete de un End Device si ya ha enviado todos los paquetes permitidos durante el mismo período. Si no hay información sobre el período, debería reenviar todos los paquetes.

6.3.3.8. Filtrar el tráfico por presupuesto

Aquí se aplica una lógica similar a la descrita en la política de número de paquetes. Cada Gateway recibe un presupuesto que limita la cantidad de bytes que se reenviarán. Es una decisión de diseño si este presupuesto corresponde a el Gateway o a cada End Device. Además, el período asociado con este presupuesto también puede variar. Para algunas aplicaciones, el presupuesto puede ser por horas, para otras, diario, etc.

Una Gateway recibe un paquete SET_BUDGET del Network Server, que contiene en su carga útil el presupuesto máximo que se permite reenviar. Cada vez que una Gateway recibe un paquete de un End Device, debe aplicar la lógica descrita anteriormente en esta política y decidir si reenviarlo o descartarlo.

6.3.4. Simulaciones

Para evaluar el modelo presentado y sus posibles beneficios, se simularon los Gateways NA-LoRaWAN y sus políticas de modelado de tráfico introducidas en este trabajo. La herramienta utilizada se basó en el módulo NS3 LoRaWAN descrito en [127], [128] y [129]. Este módulo se modificó y amplió para agregar una capa adicional de lógica necesaria para ejecutar los filtros de los Gateways. El código se ha publicado en un repositorio público [130].

Es difícil predecir resultados genéricos que puedan aplicarse a todos los casos de uso, ya que hay muchos factores a tener en cuenta y dependerá en última instancia de los requisitos e implementaciones de casos reales, desde la ubicación de el Gateway y el End Device, hasta el análisis de datos cualitativos (por ejemplo, asignar prioridades en la política de modelado del tráfico correspondiente). Para tomar una referencia, las simulaciones se ejecutaron en 100 escenarios que se crearon con configuraciones diferentes pero arbitrarias. Los escenarios se dividen en 5 grupos de 20 instancias según la cantidad de Gateways: 1 Gateway, 2 Gateways, 4 Gateways, 8 Gateways y 16 Gateways. Todos los escenarios tienen un número aleatorio de End Devices que va de 10 a 2000, y también están ubicados aleatoriamente, siguiendo la función *Math.random()* del generador de números pseudoaleatorios (PRNG) proporcionada por el entorno de ejecución de Node.js v14. Como se indicó anteriormente, este trabajo está destinado a pequeñas redes LoRaWAN privadas, por lo que se decidió trabajar con menos de 2000 End Devices y 16 Gateways. A los End Devices se les asignaron niveles de prioridad del 1 al 5 de manera uniforme (para la política de configuración del tráfico correspondiente). Los Gateways se colocaron siguiendo un patrón de cuadrícula, según lo propuesto por [131], que ha demostrado ser un algoritmo eficiente para reducir el número de antenas necesarias. La cobertura de cada Gateway es de alrededor de 9 km y está determinada por el modelo de propagación propuesto en [127], sin tener en cuenta la interferencia de los edificios, ya que los casos de uso objetivos no están destinados a ser implementados en una ciudad. En la Figura 6.7 se puede ver un escenario de ejemplo con 8 Gateways. En este ejemplo, las circunferencias negras representan el área de cobertura estimada de cada Gateway y los puntos azules representan los End Devices dispersos. El rectángulo gris en el fondo delimita el área donde se pueden ubicar los End Devices. De esta imagen se puede concluir que algunos nodos están al alcance de más de un Gateway. Sin embargo, si estos pueden recibir o no mensajes de un nodo, dependerá también del Spreading Factor, que es ajustado automáticamente por cada End Device para alcanzar al menos un Gateway. Por lo tanto, es más adecuado ejecutar una

simulación para contar paquetes duplicados, en lugar de simplemente contar el número de nodos dentro del área de cobertura de un Gateway.

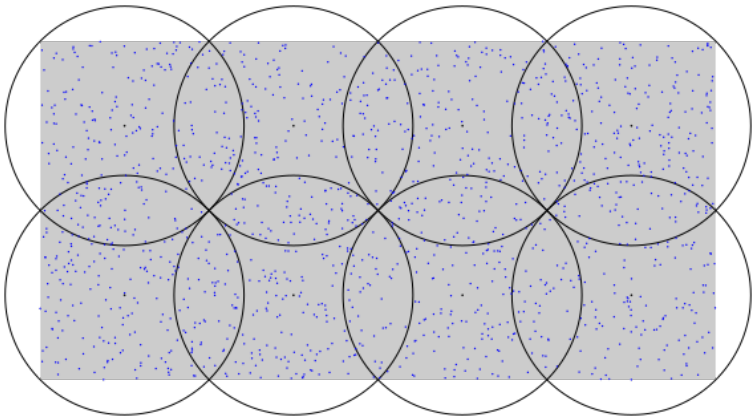


Figura 6.7: Una muestra de los escenarios generados utilizados en las simulaciones.

En la simulación, el Network Server está ubicado en la nube (el controlador) y los Gateways (los conmutadores) contienen cada uno un enlace a la red de backhaul.

Se eligió una duración de 1 hora para cada simulación y, por lo tanto, los paquetes de inicialización tanto de la especificación LoRaWAN (como los Join Request) como los introducidos por este trabajo (como las actualizaciones de la lista blanca del Gateway), no se tomaron en cuenta ya que son solo se envían al principio, y en condiciones de trabajo estables, su impacto se deprecia con el tiempo (por ejemplo, los resultados de la simulación de 1 hora se verán más afectados por los paquetes de inicialización que los resultados de la simulación de 1 mes). Además, para simplificar, se determinó que todos los paquetes sean de tipo uplink (del End Device al Network Server y no al revés) y que se envíen cada 20 minutos en modo no confirmado, con una fase aleatoria proporcionada por el módulo NS3 LoRaWAN. Como observación final, todos los paquetes tienen un tamaño de 19 bytes y, dado que este módulo no cifra los paquetes, son significativamente más pequeños que los reales.

En cada simulación, se rastrearon 3 métricas:

- *Rx Ratio*: total de paquetes transmitidos por Gateways sobre el total de paquetes recibidos por Gateways.
- *RxUnique Ratio*: total de paquetes únicos transmitidos por Gateways sobre el total de paquetes únicos recibidos por Gateways.
- *RxDuplicated Ratio*: total de paquetes duplicados transmitidos por Gateways sobre el total de paquetes duplicados recibidos por Gateways.

El prefijo *Rx* de los nombres de las métricas es desde el punto de vista del Network Server. Como se supone que no hay pérdida de paquetes entre los Gateways y el Network Server, sus valores son exactamente los mismos que si se cambiara el enfoque y se contara la transmisión desde los Gateways. En la descripción de los resultados, se utilizará indistintamente *Tx* desde Gateways o *Rx* desde Network Server.

Los paquetes duplicados y únicos se consideran en todos los Gateways. Entonces, si dos Gateways diferentes reciben el mismo paquete, uno de ellos se cuenta como único y el otro como duplicado. La razón por la que se discriminaron y midieron paquetes únicos y duplicados es para comprender cuántos datos únicos pierde cada modelo como compensación para reducir el tráfico en el backhaul.

6.3.4.1. Variando el número de Gateways

La primera ronda de simulaciones consistió en simular cada una de las 4 políticas de modelado de tráfico y el Gateway estándar (sin ninguna política de modelado de tráfico) sobre el total de los 100 escenarios generados y descritos previamente. Estos resultaron en un total de 500 simulaciones con más de 1500000 paquetes transmitidos.

La configuración realizada fue:

- *Budget*: 19 bytes por nodo cada hora (si se necesitan más bytes para transmitir, los paquetes se descartarán). Esto es igual al tamaño de 1 paquete.
- *Número máximo de paquetes para reenviar*: 1 paquete cada hora (si llegan paquetes adicionales, se descartarán)
- *Nivel de prioridad*: Nivel 1 (los niveles 2 a 5 se filtran)

La figura 6.8 muestra el resumen con los resultados que son los esperados. En las primeras 3 barras, con solo los Gateways estándar (NA-LoRaWAN deshabilitado), se puede ver que todos los paquetes fueron transmitidos, tanto únicos como duplicados. En la política "Filtrar por ID de nodo", sólo se transmitían paquetes únicos, lo que representó el 87,32 % del total de paquetes recibidos por los Gateways, y, para este caso, un ahorro medio del 12,68 % de ancho de banda en el backhaul. Las políticas "Filtrar por presupuesto" y "Filtrar por número de paquetes" rondan el 34,57 % en las 3 métricas, lo que equivale aproximadamente a filtrar 2 de cada 3 paquetes recibidos por los Gateways durante la simulación de 1 hora, correspondiente a la configuración descrita anteriormente. Finalmente, la política "Filtrar por prioridad de nodo" muestra que se transmitieron el 19,88 % de los paquetes, lo que se correlaciona directamente con permitir que solo se reenvíen paquetes del nivel de prioridad 1, filtrando los niveles 2 a 5, producto de una distribución de prioridad uniforme.

En el patrón de cuadrícula utilizado para ubicar los Gateways, existe una correlación estricta entre el número de Gateways y el tamaño del escenario. Por lo tanto, para evaluar si el tamaño del escenario afectó estos resultados, los resultados se dividieron en 3 gráficos en la Figura 6.9, uno para cada métrica, agrupados por el número de Gateways en el eje x. Todas las políticas permanecen con resultados similares a los mostrados anteriormente, excepto la política "Filtrar por ID de nodo" en la Figura 6.9 (a), donde el porcentaje de paquetes transmitidos disminuye a medida que el escenario aumenta: de 100 % cuando sólo hay 1 Gateway, al 80,31 % cuando hay 16.

Sobre la base del contexto de estas simulaciones, se puede concluir que la política "Filtrar por ID de nodo" mejora el ahorro proporcional de ancho de banda a medida que se utilizan más Gateways, sin afectar la pérdida de datos. Las otras políticas son consistentes y escalan con más Gateways.

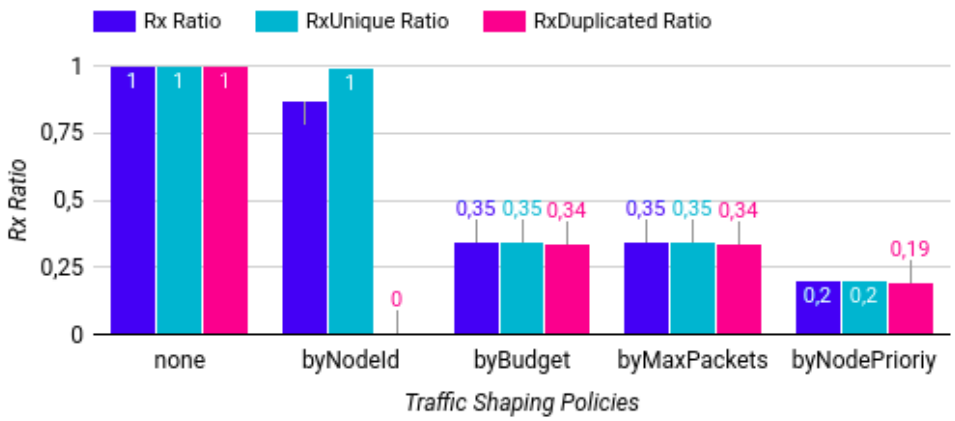


Figura 6.8: Resumen de resultados del primer grupo de simulaciones.

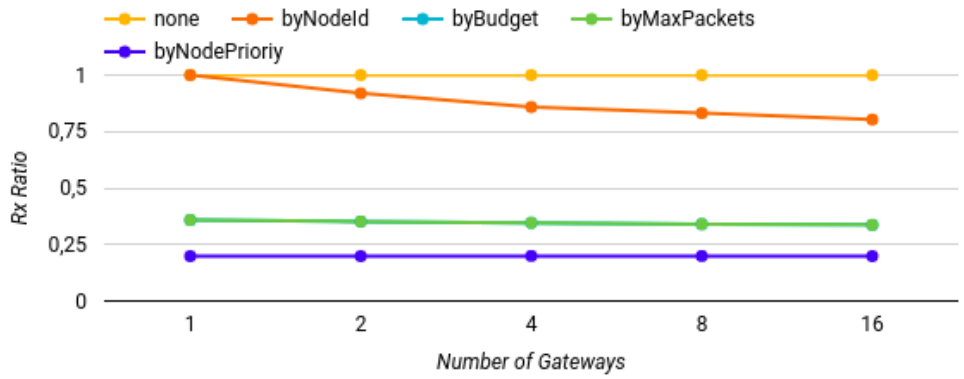
6.3.4.2. Variando las prioridades

Como se explicó anteriormente, los End Devices tenían 5 prioridades (1 a 5) asignadas de manera uniforme. En un nuevo conjunto de pruebas, en cada ejecución se incrementó el nivel de prioridad, comenzando desde 0 (no reenvía ningún paquete) hasta 5 (reenvía todos los paquetes). Para simplificar, estas pruebas se ejecutaron sólo en los 20 escenarios generados con 4 Gateways, y todas las demás configuraciones, aparte de la prioridad, permanecieron iguales. Se realizaron un total de 100 simulaciones con más de 1 millón de paquetes.

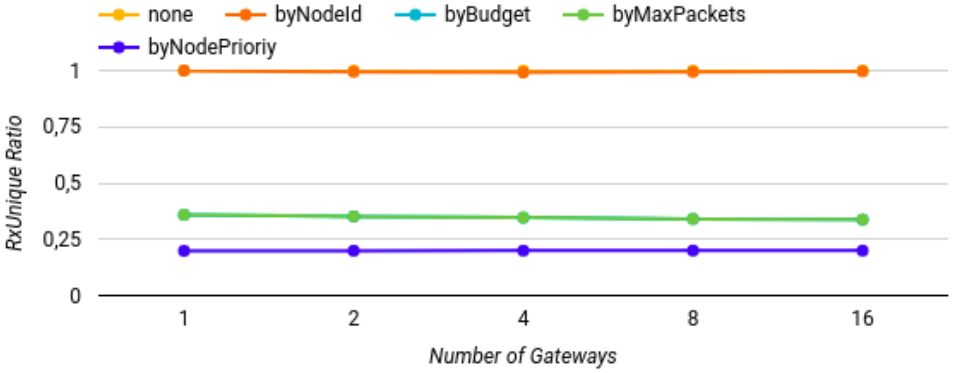
En la Figura 6.10 solo se ha representado *Rx Ratio*, ya que los resultados para *RxUnique* y *RxDuplicated* son similares. Muestra, como era de esperar, que solo la política de prioridad arrojó resultados diferentes en cada prueba. La primera ejecución tuvo 0 % de transmisión de paquetes, la segunda tuvo 19.94 %, la tercera tuvo 39.93 %, la cuarta tuvo 59.83 %, la quinta tuvo 79.91 %, y en la última se transmitió el 100 % de los paquetes. La pendiente de la línea se incrementa en 1/5 aproximadamente, lo que se correlaciona con las 5 prioridades asignadas. A partir de estos resultados, se puede suponer que para la política de prioridades, la compensación de paquetes únicos y duplicados es directamente proporcional a la configuración de las prioridades.

6.3.4.3. Variando el número máximo de paquetes

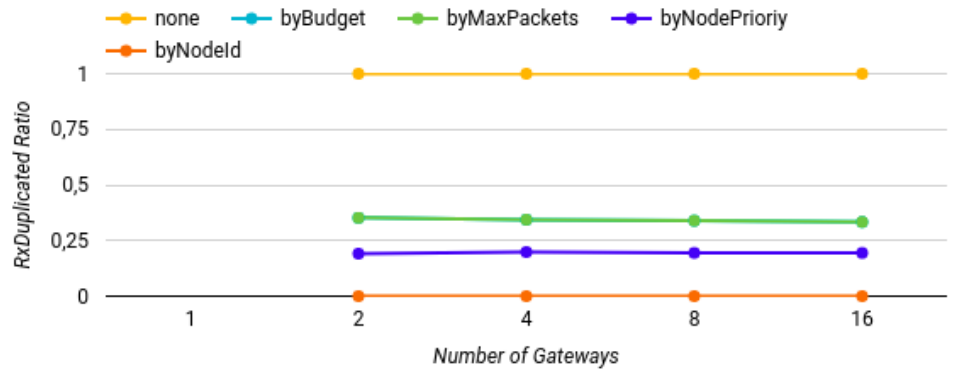
Se ejecutó otro conjunto de pruebas que consistió en cambiar la cantidad máxima de paquetes por End Device a ser reenviados por un Gateway, en un período de 1 hora. En las primeras pruebas se utilizó 1 paquete por hora, en la segunda 2 paquetes y finalmente 3 paquetes. Nuevamente, solo se ejecutaron escenarios con 4 Gateways y la configuración fue similar a las pruebas anteriores. Los resultados de la Figura 6.11 muestran que sólo la política de “Filtrar por número máximo de paquetes” tuvo diferentes salidas en cada ejecución de las pruebas. Como era de esperar, en el primero se transmitieron 1/3 de los paquetes (34,62 %), mientras que en los otros dos se transmitieron 2/3 (68,20 %) y todos ellos respectivamente.



(a)



(b)



(c)

Figura 6.9: Los resultados del primer grupo de simulaciones, discriminados por métricas y número de Gateways. En los 3 gráficos, los valores de las series “byBudget” y “byMaxPackets” están casi al mismo valor, por lo que no se pueden diferenciar fácilmente. En (b) las series “none” y “byNodeId” tienen un valor igual a 1.

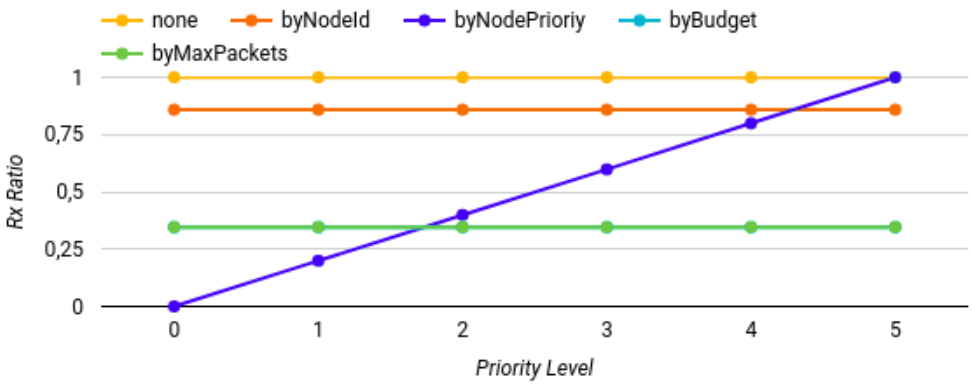


Figura 6.10: Resultados de variar el nivel de prioridad para filtrar en las diferentes políticas de modelado de tráfico.

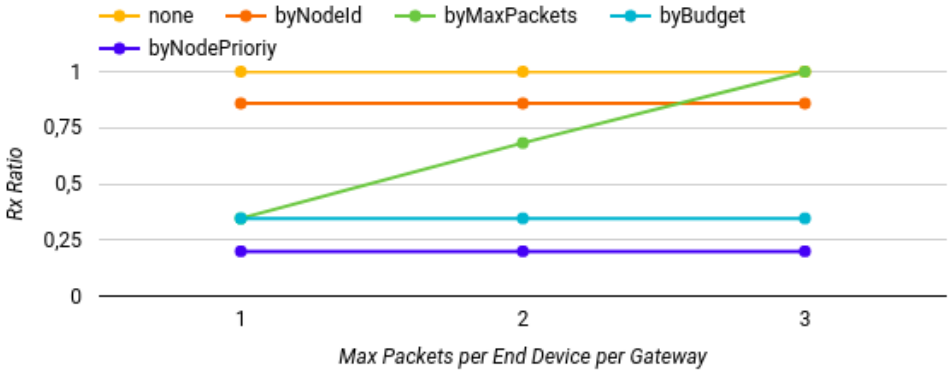


Figura 6.11: Resultados de variar el número de paquetes máximos permitidos en las diferentes políticas de modelado de tráfico.

6.3.4.4. Variando el presupuesto

Utilizando un enfoque muy similar a las pruebas anteriores, esta vez se modificó el presupuesto permitido a transmitir por nodo por Gateway en 1 hora. Esto significa que cada vez que un Gateway recibe un paquete de un End Device, verificará si tiene presupuesto para reenviar ese End Device. Para la primera prueba, se usaron los mismos 19 bytes que antes, luego se aumentó a 38 bytes y finalmente se aumentó a 57 bytes. Como todos los paquetes tienen un tamaño de 19 bytes, en el primer presupuesto solo cabe un paquete, mientras que en el segundo y tercero, 2 y 3 paquetes respectivamente. Aparte del presupuesto, los demás ajustes eran los mismos que en las simulaciones anteriores.

Los resultados de la Figura 6.12 muestran que la única política afectada con el cambio de presupuesto es el "Filtrar por presupuesto". Como era de esperar, en el primer caso se transmitieron casi 1/3 de los paquetes (34,62 %); en el segundo caso casi 2/3 (68,20 %); y finalmente, todos ellos en el último caso.

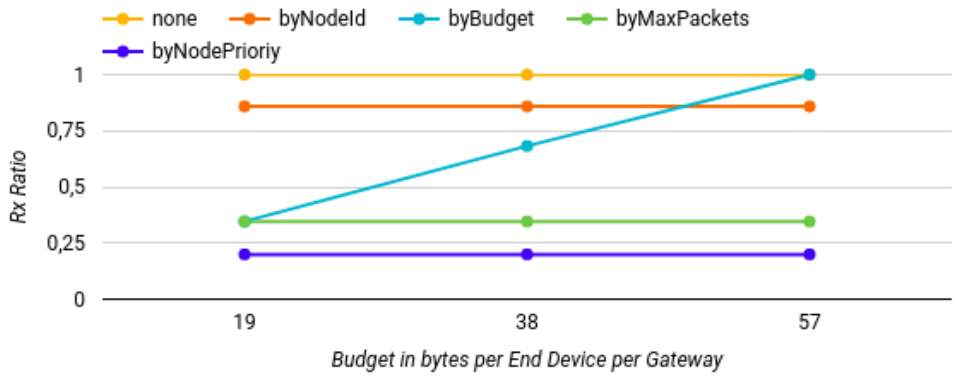


Figura 6.12: Resultados de variar el presupuesto permitido en las diferentes políticas de modelado del tráfico.

6.3.4.5. Discusión

Los resultados obtenidos de las simulaciones indican que hay una reducción sustancial del tráfico del backhaul al utilizar Gateways NA-LoRaWAN. En el caso de la política "Filtrar por ID de nodo ", el 100 % de los paquetes duplicados se descartaron a nivel de Gateway sin pérdida de datos. Para las otras 3 políticas, existe una compensación entre la reducción del tráfico y la pérdida de datos. Si esta compensación es efectiva o no, dependerá de cada escenario y de la configuración de la política.

De acuerdo con las limitaciones de backhaul de IoRT descritas anteriormente, se cree que estas políticas podrían beneficiar a diferentes escenarios. Por ejemplo, en la agricultura, cientos de sensores podrían monitorear diferentes ubicaciones de una o más parcelas en la misma área, todas ellas conectadas a la nube a través de una costosa red de backhaul. Al aplicar estas políticas, podría haber una reducción potencialmente significativa de los costos de la red.

La arquitectura propuesta en este documento se puede aplicar a cualquier política de modelado de tráfico. Cuando es preferible una mayor tolerancia a fallos, la duplicación de paquetes no es necesariamente una mala opción; cuando se debe garantizar el acceso al Network Server a los mensajes de alta prioridad, es preferible la selección de esa política. Los administradores de red deben decidir el enfoque que quieren aplicar en su red para lograr perfiles específicos para el tráfico, discusión que está fuera del alcance de este documento. Sin embargo, es importante destacar el hecho de que el Gateway NA puede manejar diferentes modos de funcionamiento de acuerdo con los requisitos particulares del sistema de red en cada instante. Cabe destacar también que, en cualquier caso, el Network Server y los Gateways tienen que intercambiar mensajes periódicamente, y, realizar la configuración para las implementaciones de las políticas de modelado de tráfico, requiere de algunos intercambios de mensajes adicionales con respecto al funcionamiento normal de LoRaWAN. El sobrecoste se debe justificar en la reducción de tráfico o los beneficios que trae la implementación de las diferentes políticas.

6.4. Contribuciones de código abierto

Para realizar las simulaciones descritas en este capítulo, se realizaron 2 desarrollos que fueron liberados bajo licencia GNU GPLv2[130]. El primero consistió en una modificación realizada sobre un módulo de LoRaWAN para NS3 [132] mediante herencia de clases, lo que permitió crear una nueva capa de software, sin necesidad de sobreescribir el código original. Además, se creó un nuevo ejemplo de uso que consiste en una herramienta de línea de comando que corre una simulación parametrizada que ofrece la posibilidad de elegir la política de modelado de tráfico, el nivel de prioridad, el presupuesto de los gateways, etc. La misma herramienta ofrece un comando de ayuda que explica cómo utilizarla. El segundo desarrollo, es una aplicación desarrollada en Node.js que permite por un lado generar y visualizar nuevos escenarios de simulación (cantidad y ubicación de gateways y dispositivos), y por otro lado ejecutar múltiples simulaciones de NS3 a modo de wrapper, con una sola ejecución. La configuración de este desarrollo fue realizada en Node.js.

6.5. Consideraciones finales

Un nuevo Gateway LoRaWAN con el objetivo de reducir el tráfico enviado por el backhaul fue descrito en este capítulo. Su impacto se vuelve relevante en escenarios donde la red es limitada o costosa, como en IoRT. La idea principal detrás del Gateway, llamada Gateway NA-LoRaWAN, es que puede procesar el tráfico identificando los End Devices. Se introdujeron los conceptos principales de cómo podría funcionar el Gateway y se presentaron diferentes técnicas para filtrar el tráfico. Se describió una propuesta de diseño y una simulación para cada uno de ellos, mostrando que potencialmente puede superar a los Gateways LoRaWAN estándar en cuanto a la cantidad de tráfico que se transmitirá a través de la red de backhaul. Además, cada simulación mostró cómo el tráfico podría escalar y la compensación de los datos del sensor frente al uso de la red, con respecto a diferentes configuraciones. Si es conveniente o no implementar algunas de las políticas de modelado de tráfico en casos de uso reales, dependerá del análisis realizado para cada contexto.

Aún queda trabajo por hacer en futuras investigaciones para seguir validando el Gateway y las técnicas propuestas. A continuación se describen algunas ideas. Primero, analizar los posibles inconvenientes que estos cambios pueden introducir, como tener dos redes funcionando en la misma área o fallas de seguridad introducidas por los nuevos comandos. En segundo lugar, implementar el diseño propuesto y probarlo en campo. En tercer lugar, explorar otras formas de identificar los End Devices en el Gateway para que puedan adaptarse a otros escenarios en los que los nodos se muevan libremente. Finalmente, la combinación de políticas de modelado de tráfico podría mitigar algunos de los problemas potenciales descritos en este documento, al mismo tiempo que ayudaría a reducir el tráfico. Por ejemplo, los nodos con el nivel de prioridad 1 podrían obtener todos sus paquetes reenviados con redundancia, mientras que otras prioridades tendrían los paquetes duplicados filtrados. Esta combinación de políticas, junto con los modos de Gateway (por ejemplo, una combinación de Gateways estándar y NA-LoRaWAN) también debe considerarse para analizarse en trabajos futuros.

Parte IV

Conclusión

Capítulo 7

Conclusiones y Trabajo Futuro

Internet de las Cosas es un nuevo paradigma que, más allá del crecimiento que ha tenido en los últimos años, aún continúa en evolución. El poder interactuar con variables del ambiente a través de Internet, donde la capacidad de procesamiento puede ser elástica y funcionar las 24 horas del día, abre la posibilidad a extender muchas de las barreras que imponen las limitaciones físicas del mundo real. Durante el desarrollo de esta tesis se reailzaron y se evaluaron arquitecturas, diseños, implementaciones y protocolos. Cada uno de los trabajos llevados a cabo, perseguían una problemática clara a la que se propuso una solución.

En la Parte I de este trabajo, se introdujeron conceptos básicos de IoT y algunas ideas de implementación que tienen como objetivo beneficiar a la sociedad en distintos ámbitos como la educación, la salud, el transporte, los servicios públicos y la agrícultura. Una problemática de este último campo, fue diseñada y desarrollada en la Parte II, Capítulo 3. Se propuso una arquitectura para un sistema IoT de monitoreo que determina si un sistema de riego se encuentra en falla, lo que produce un desperdicio de recursos, como el agua y la energía (de la línea eléctrica o combustible). El sistema se planteo utilizando varios componentes Open Source, como las placas de desarrollo SODAQ y el protocolo LoRaWAN. Además, se contempló la posibilidad de que el procesamiento escale rápidamente, utilizando Funciones como Servicio (FaaS). Los resultados del análisis de la implementación, demostraron que durante la experiencia realizada en campo, el sistema cumplió con el objetivo de identificar el correcto estado del sistema de riego.

Otro aspecto del diseño y arquitectura de un sistema IoT fue desarrollado también en la Parte II, pero en el Capítulo 4. El mismo consistió en un sistema de auto-evacuación masiva en ciudades ante desastres naturales. Allí se puso foco en las interacciones entre los distintos componentes, y cómo clasificarlas y validarlas formalmente mediante el uso de IoT-Calculus. Además, se realizó una implementación de un prototipo que permitió extrapolar los resultados a un entorno de una ciudad real. De esta forma, en la Parte II quedaron agrupados dos trabajos que buscan describir un sistema IoT de manera integra, pero uno detallando la implementación tecnológica y una evaluación en un escenario real, y el otro un modelo teórico y su validación formal y empírica mediante prototipos.

La Parte III en cambio, no propuso sistemas, sino más bien, modificaciones a protocolos de comunicación existentes para adaptalos a escenarios específicos. La motivación detrás de estas propuestas, surge a partir de la experiencia de realizar sistemas como los descritos en la anterior Parte. El Capítulo 5 especifica una extensión a MQTT 3.1.1 para que pueda

soportar comunicación en tiempo real suave. Se realizó primero un modelo teórico más general para cualquier protocolo de publicación-suscripción, y luego se detallaron las modificaciones necesarias a realizar en dicho protocolo. Además, se realizó una implementación de las mismas y se publicó un simulador en la web para que pueda ser utilizado y probado. Esto se relaciona directamente con los sistemas de IoT de agricultura como los descritos en el Capítulo 2, ya que pueden requerir medir distintas variables del ambiente y combinarlas de manera sincronizada para mostrar el estado del cultivo o el suelo en un determinado momento. Por otro lado, en el Capítulo 6 se presenta una modificación a LoRaWAN para disminuir el tráfico de la red backhaul. Esto facilitaría la adopción de redes con menos ancho de banda y más caras que las que se encuentran comúnmente en las ciudades. Se realizaron varias simulaciones con un desarrollo propio basado en un módulo LoRaWAN de NS3, que permitieron concluir que el modelo planteado es escalable y reduce el tráfico para los escenarios planteados. Este trabajo es consecuencia de los desafíos futuros que plantea el Capítulo 3, ya que es un primer paso para adaptar LoRaWAN a redes como la satelital que se usa en ese sistema. Por otro lado, también se relaciona con el Capítulo 4, ya que la modificación de LoRaWAN presentada posibilita la utlización de redes alternativas como backhaul, ante la caída de la infrestructura tradicional de comunicaciones frente a un desastre natural.

Para un profesional informático, como el autor de esta tesis, trabajar con IoT presentó un gran desafío con respecto a otros sistemas de software. Por ejemplo, al realizar pruebas empíricas en escenarios similares o reales, IoT requiere de una instalación y mantenimiento físico, que involucra desplazamientos, traslado de equipos, garantizar conectividad, etc. Todas tareas que no suelen encontrarse en desarrollos informáticas más típicos en Argentina, como los que consisten sólo en una aplicación web o móvil.

Luego de analizar las problemáticas desarrolladas en esta tesis, se observó que los escenarios que no cuentan con tecnologías de comunicación tan desarrolladas como las de los centros urbanos, presentan numerosos inconvenientes y deben ser tratadas de manera excepcional. Puede ser mediante la modificación de protocolos existentes, como se llevó a cabo en esta tesis, o la creación de nuevas formas de comunicación. De cualquier manera, es conveniente validarlas formalmente, con simulaciones, y de manera empírica a través de prototipos o escenarios reales, todas cuestiones que fueron abarcadas en estos trabajos.

Siguiendo esta línea planteada, a futuro se buscará avanzar en el Internet de las Cosas Remotas, que contempla aquellos escenarios que cuentan con una infraestructura de comunicación más inestable y/o limitada. La completa adaptación del protocolo LoRaWAN para poder trabajar con Internet satelital es uno de los caminos a tomar. Esto permitiría desplegar fácilmente dispositivos de IoT desarrollados por terceros, utilizando una tecnología de comunicación ya probada y en constante crecimiento. La agricultura, los sistemas de evacuación, y las industrias alejadas de las ciudades, pueden verse beneficiadas de dicho desarrollo.

Appendices

Apéndice A

Estructura de paquetes de la extensión de tiempo real suave para MQTT

A continuación se detallan las estructuras de los paquetes nuevos o modificados introducidos en la extensión propuesta a MQTT versión 3.1.1 para soportar tiempo real suave.

A.1. Paquete de registro

El paquete de registro denominado `REGISTER`, es un nuevo tipo de paquete de control de MQTT introducido por esta extensión. A través del mismo, un publicador informa el tópico o los tópicos que va a utilizar, y sus respectivos parámetros temporales.

A.1.1. Cabecera fija

La Tabla A.1 muestra la estructura de la cabecera fija, similar a la de cualquier otro paquete de control de MQTT. Toma como identificador el número 15, actualmente no utilizado en la versión 3.1.1 de MQTT.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	MQTT Control Packet type (15)				Reserved			
	1	1	1	1	0	0	0	0
byte 2	Remaining Length							

Cuadro A.1: Estructura de la cabecera fija del paquete de registro.

A.1.2. Cabecera variable

La Tabla A.2 muestra la estructura de la cabecera variable. Los primeros 4 bytes informan la marca de tiempo UNIX en segundos del publicador en el instante previo a enviar un mensaje. El byte 5 hace referencia al período en minutos en el cuál el publicador enviará los

mensajes del tópico. Los bytes 6 a 7 contienen el largo del tópico en bytes, que comienza en el byte 8.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 2	Timestamp MSB							
byte 3	Timestamp							
byte 4	Timestamp							
byte 5	Timestamp LSB							
byte 1	Period							
byte 6	Length Topic MSB							
byte 7	Length Topic LSB							
byte 8 ...	Topic Name							

Cuadro A.2: Estructura de la cabecera variable del paquete de registro.

A.1.3. Payload

Este paquete no utiliza payload.

A.2. Paquete de publicación

En este apartado se describe la modificación propuesta al paquete `PUBLISH` de MQTT, donde se agrega un registro temporal al mensaje.

A.2.1. Cabecera fija

Sin cambios con respecto a la especificación original.

A.2.2. Cabecera variable

La Tabla A.3 muestra la propuesta nueva. Del byte 1 al 4 se envía el tiempo UNIX en segundos del publicador en el instante previo a mandar el mensaje. A partir del byte 5 se encuentra el tópico y el identificador del paquete, de la misma forma que aparece en la especificación original.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	Timestamp MSB							
byte 2	Timestamp							
byte 3	Timestamp							
byte 4	Timestamp LSB							
byte 5 ...	Topic and Packet Identifier							

Cuadro A.3: Estructura de la cabecera variable del paquete de publicación.

A.2.3. Payload

Sin cambios con respecto a la especificación original.

A.3. Paquete de suscripción

Las modificaciones propuestas para el paquete `SUBSCRIBE` de MQTT están relacionadas con establecer las condiciones temporales deseadas por el suscriptor.

A.3.1. Cabecera fija

Sin cambios con respecto a la especificación original.

A.3.2. Cabecera variable

La Tabla A.4 ilustra los campos de la cabecera variable del paquete `SUBSCRIBE`. Al inicio del mismo, se encuentra entre los bytes 1 y 4, la marca temporal del publicador en segundos en el instante previo a enviar el mensaje. Luego, en el byte 5, se establece el período en minutos en el que desea recibir los mensajes, y finalmente en el byte 6, se establece la latencia máxima en minutos esperada. La cabecera culmina con el identificador de paquete, tal como se describe en la especificación original.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	Timestamp MSB							
byte 2	Timestamp							
byte 3	Timestamp							
byte 4	Timestamp LSB							
byte 5	Period							
byte 6	Latency							
byte 7	Packet Identifier MSB							
byte 8	Packet Identifier LSB							

Cuadro A.4: Estructura de la cabecera variable del paquete de suscripción.

A.3.3. Payload

Sin cambios con respecto a la especificación original.

Bibliografía

[1] *State of IoT Q4 2020 & Outlook 2021*. IoT Analytics, 2020.

[2] I. Peña-López. *Itu Internet Report 2005: The Internet of Things*. ITU, 2005.

[3] Arun Kumar, Ming Zhao, Kai-Juan Wong, Yong Liang Guan, and Peter Han Joo Chong. A comprehensive study of iot and wsn mac protocols: Research issues, challenges and opportunities. *IEEE Access*, 6:76228–76262, 2018.

[4] C. C. Sobin. A survey on architecture, protocols and challenges in IoT. *Wireless Personal Communications*, 112(3):1383–1429, January 2020.

[5] Gutierrez. *Low-rate wireless personal area networks : enabling wireless sensors with IEEE 802.15.4*. Standards Information Network, IEEE Press, Piscataway, N.J, 2010.

[6] Myra Dideles. Bluetooth. 9(4):11–18, June 2003.

[7] J. Postel. User datagram protocol. STD 6, RFC Editor, August 1980. `http://www.rfc-editor.org/rfc/rfc768.txt`.

[8] Jon Postel. Transmission control protocol. STD 7, RFC Editor, September 1981. `http://www.rfc-editor.org/rfc/rfc793.txt`.

[9] *MQTT Version 3.1.1*, 2014.

[10] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (CoAP). Technical report, June 2014.

[11] Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, 2016.

[12] Dali Ismail, Mahbubur Rahman, and Abusayeed Saifullah. Low-power wide-area networks: opportunities, challenges, and directions. pages 1–6, 01 2018.

[13] Aloÿs Augustin, Jiazi Yi, Thomas Clausen, and William Mark Townsley. A study of lora: Long range amp; low power networks for the internet of things. *Sensors*, 16(9), 2016.

[14] Dan Dragomir, Laura Gheorghe, Sergiu Costea, and Alexandru Radovici. A survey on secure communication protocols for iot systems. In *2016 International Workshop on Secure Internet of Things (SIoT)*, pages 47–62, 2016.

[15] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of lpwan technologies for large-scale iot deployment. *ICT Express*, 5(1):1–7, 2019.

[16] *Proceedings of the Expert Meeting on How to Feed the World in 2050 : 24-26 June 2009, FAO Headquarters, Rome*. FAO, Rome, 2009.

[17] Plan nacional de riego. `https://www.magyp.gob.ar/sitio/areas/riego/plan_riego`.

[18] Plan nacional de agua. `https://www.argentina.gob.ar/sites/default/files/plan_nacional_agua_.pdf`.

[19] *Estudio del potencial de ampliación del riego en Argentina*. FAO, 2015.

[20] Nestor Barrionuevo, German, and Cynthia Waldman. Análisis espacio temporal del riego por pivote central en la provincia de buenos aires en el período 1995-2015. 11 2016.

[21] Martin Tagfliotti, María Bedogni, Ortego Jaime, Silvia Capezio, and Marcelo Huarte. Caracterización fenotípica de genotipos de papa por su comportamiento frente a estrés hídrico a campo. 10 2013.

[22] ENACOM. Mapa de conectividad nacional, Jul 2019.

[23] Pallavi Sethi and Smruti R. Sarangi. Internet of things: Architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017:1–25, 2017.

[24] R. T. Peters and S. R. Evett. USING LOW-COST GPS RECEIVERS FOR DETERMINING FIELD POSITION OF MECHANIZED IRRIGATION SYSTEMS. *Applied Engineering in Agriculture*, 21(5):841–845, 2005.

[25] SODAQ. Sodaq one, Jul 2019.

[26] Partha Pratim Ray. Internet of things for smart agriculture: Technologies, practices and future direction. *Journal of Ambient Intelligence and Smart Environments*, 9(4):395–420, June 2017.

[27] Erwin van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uta, and Alexandru Iosup. Serverless is more: From PaaS to present cloud computing. *IEEE Internet Computing*, 22(5):8–17, September 2018.

[28] United Nations. World population prospects 2019, 2019.

[29] Dennis Mileti. *Disasters by design: A reassessment of natural hazards in the United States*. Joseph Henry Press, 1999.

[30] Gabriel M Eggly, José Mariano Finochietto, Matias Micheletto, Roger Pueyo Centelles, Rodrigo Santos, Sergio F Ochoa, Roc Meseguer, and Javier Orozco. Evacuation supporting system based on iot components. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 31, page 38, 2019.

[31] Matias Micheletto, Vinicius Petrucci, Rodrigo Santos, Javier Orozco, Daniel Mosse, Sergio F. Ochoa, and Roc Meseguer. Flying real-time network to coordinate disaster relief activities in urban areas. *Sensors*, 18(5), 2018.

[32] Ivan Lanese, Luca Bedogni, and Marco Di Felice. Internet of things: A process calculus approach. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, page 1339–1346, New York, NY, USA, 2013. Association for Computing Machinery.

[33] Huibo Bi and Erol Gelenbe. A survey of algorithms and systems for evacuating people in confined spaces. *Electronics*, 8:711, 06 2019.

[34] Najla Al-Nabhan, Nadia Al-Aboody, and A.B.M. Alim Al Islam. A hybrid iot-based approach for emergency evacuation. *Computer Networks*, 155:87 – 97, 2019.

[35] Marin Lujak, Holger Billhardt, Jürgen Dunkel, Alberto Fernández, Ramon Hermoso, and S. Ossowski. A distributed architecture for real-time evacuation guidance in large smart buildings. *Computer Science and Information Systems*, 14:257–282, 01 2017.

[36] Javier Mulero Chaves and Tomaso De Cola. 1 - public warning applications: Requirements and examples. In Daniel Câmara and Navid Nikaein, editors, *Wireless Public Safety Networks 3*, pages 1 – 18. Elsevier, 2017.

[37] Andreas I Miaoudakis, Nikolaos E Petroulakis, Diomedes Kastanis, and Ioannis G Askoxylakis. Communications in emergency and crisis situations. In *Distributed, Ambient, and Pervasive Interactions (DAPI), International Conference on*, pages 555–565. Springer, 2014.

[38] Partha Sarathi Paul, Krishnandu Hazra, Sujay Saha, Subrata Nandi, Sandip Chakraborty, and Sajal Das. Generating crisis maps for large-scale disasters: Issues and challenges. this publication is an outcome of the r&d work undertaken in the itra project of media lab asia entitled. "post-disaster situation analysis and resource management using delay-tolerant peer-to-peer wireless networks (disarm)". In Daniel Câmara and Navid Nikaein, editors, *Wireless Public Safety Networks 3*, pages 67 – 98. Elsevier, 2017.

[39] Cristina Párraga Niebla, Javier Mulero Chaves, and Tomaso De Cola. 8 - design aspects in multi-channel public warning systems. In Daniel Câmara and Navid Nikaein, editors, *Wireless Public Safety Networks 2*, pages 227 – 261. Elsevier, 2016.

[40] Vasani Yash Kishorbhai and Nagekar Nainesh Vasantbhai. Aon: A survey on emergency communication systems during a catastrophic disaster. *Procedia Computer Science*, 115:838 – 845, 2017. 7th International Conference on Advances in Computing & Communications, ICACC-2017, 22-24 August 2017, Cochin, India.

[41] Yuko Murayama and Kayoko Yamamoto. Research on disaster communications. In *IFIP Advances in Information and Communication Technology*, pages 1–11. vol 516. Springer, Cham, 2019.

[42] Matthew L. Spialek and J. Brian Houston. The influence of citizen disaster communication on perceptions of neighborhood belonging and community resilience. *Journal of Applied Communication Research*, 47(1):1–23, 2019.

[43] M. Avvenuti, MG. Cimino, S. Cresci, A. Marchetti, and M. Tesconi. A framework for detecting unfolding emergencies using humans as sensors. *Springerplus*, 5(43), 2016.

[44] Sergio F. Ochoa and Rodrigo Santos. Human-centric wireless sensor networks to improve information availability during urban search and rescue activities. *Information Fusion*, 22:71 – 84, 2015.

[45] M. Romano, T. Onorati, I. Aedo, and P. Díaz. Designing mobile applications for emergency response: Citizens acting as human sensors. *Sensors (Basel, Switzerland)*, 16, 2016.

[46] Paloma Díaz, Teresa Onorati, Marco Romano, and Ignacio Aedo. Designing affordable technologies to integrate citizens in early warning activities. *Proceedings*, 2:1253, 10 2018.

[47] Marco Conti, Andrea Passarella, and Sajal K. Das. The internet of people (iop): A new wave in pervasive mobile computing. *Pervasive and Mobile Computing*, 41:1 – 27, 2017.

[48] Stefan Stieglitz, Deborah Bunker, Milad Mirbabaie, and Christian Ehnis. Sense-making in social media during extreme events. *Journal of Contingencies and Crisis Management*, 26, 08 2017.

[49] J. Brian Houston, Joshua Hawthorne, Mimi Perreault, Eun Park, Marlo Hode, Michael Halliwell, Sarah McGowen, Rachel Davis, Shivani Vaid, Jonathan Mcelderry, and Stanford Griffith. Social media and disasters: A functional framework for social media use in disaster planning, response, and research. *Disasters*, 39, 09 2014.

[50] J. C. Araneda, H. Rudnick, S. Mocarquer, and P. Miquel. Lessons from the 2010 chilean earthquake and its impact on electricity supply. In *2010 International Conference on Power System Technology*, pages 1–7, 2010.

[51] Kathrin Eismann, Oliver Posegga, and Kai Fischbach. Decision making in emergency management: The role of social media. In *ECIS 2018 Proceedings*, pages 2–20, 06 2018.

[52] Theus Hossmann, Franck Legendre, Paolo Carta, Per Gunningberg, and Christian Rohner. Twitter in disaster mode: Opportunistic communication and distribution of sensor data in emergencies. In *Proceedings of the 3rd Extreme Conference on Communication: The Amazon Expedition*, ExtremeCom '11, pages 1–6, New York, NY, USA, 2011. Association for Computing Machinery.

[53] D. G. Reina, M. Askalani, S. L. Toral, F. Barrero, E. Asimakopoulou, and N. Bessis. A survey on multihop ad hoc networks for disaster response scenarios. *International Journal of Distributed Sensor Networks*, 2015:3, 2015.

[54] Christian Reuter. Communication between power blackout and mobile network overload. *International Journal of Information Systems for Crisis Response and Management (IJISCRAM) (accepted)*, 6, 01 2014.

[55] Roger Pueyo Centelles, Roc Meseguer, Felix Freitag, Leandro Navarro, Sergio F. Ochoa, and Rodrigo M. Santos. Loramoto: A communication system to provide safety awareness among civilians after an earthquake. *Future Generation Computer Systems*, 115:150 – 170, 2021.

[56] A. Malizia, T. Onorati, P. Diaz, I. Aedo, and F. Astorga-Paliza. Sema4a: An ontology for emergency notification systems accessibility. *Expert Systems with Applications*, 37(4):3380 – 3391, 2010.

[57] Giammarco Cecchini, Alessandro Bazzi, Barbara M. Masini, and Alberto Zanella. LTEV2vsim: An LTE-v2v simulator for the investigation of resource allocation for cooperative awareness. In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, June 2017.

[58] Khurrum Mustafa Abbasi, Tamim Ahmed Khan, and Irfan Ul Haq. Hierarchical modeling of complex internet of things systems using conceptual modeling approaches. *IEEE Access*, 7:102772–102791, 2019.

[59] Maroun Koussaifi, Sylvie Trouilhet, Jean-Paul Arcangeli, and Jean-Michel Bruel. Ambient intelligence users in the loop: Towards a model-driven approach. In *Federation of International Conferences on Software Technologies: Applications and Foundations*, pages 558–572. Springer, 2018.

[60] Todorka Glushkova and Stanimir Stoyanov. Ambient-oriented modeling of intelligent context-aware systems. 7(1):53–61, 2018.

[61] Robin Milner, Joachim Parrow, and David Walker. *A calculus of mobile processes, part 1.* University of Edinburgh Laboratory for Foundations of Computer Science, 1989.

[62] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. *Mob. Netw. Appl.*, 10(4):519–528, August 2005.

[63] Rodrigo Santos, Daniel Mosse, Taieb Znati, and Louise Comfort. Design and implementation of a witness unit for opportunistic routing in tsunami alert scenarios. *Safety Science*, 90:75 – 83, 2016. Building Community Resilience to Global Hazards: A Sociotechnical Approach.

[64] John W. Eaton, David Bateman, Søren Hauberg, and Rik Wehbring. *GNU Octave version 4.2.1 manual: a high-level interactive language for numerical computations*, 2017.

[65] Shuiguang Deng, Longtao Huang, Hongyue Wu, Wei Tan, Javid Taheri, Albert Y. Zomaya, and Zhaohui Wu. Toward mobile service computing: Opportunities and challenges. *IEEE Cloud Computing*, 3(4):32–41, July 2016.

[66] Marco Stolpe. The internet of things. *ACM SIGKDD Explorations Newsletter*, 18(1):15–34, August 2016.

[67] N Vijayakumar and R Ramya. The real time monitoring of water quality in IoT environment. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE, March 2015.

[68] Yingfeng Zhang, Wenbo Wang, Naiqi Wu, and Cheng Qian. IoT-enabled real-time production performance analysis and exception diagnosis model. *IEEE Transactions on Automation Science and Engineering*, 13(3):1318–1332, July 2016.

[69] Udit Satija, Barathram Ramkumar, and M. Sabarimalai Manikandan. Real-time signal quality-aware ECG telemetry system for IoT-based health care monitoring. *IEEE Internet of Things Journal*, 4(3):815–823, June 2017.

[70] Matias Micheletto, Vinicius Petrucci, Rodrigo Santos, Javier Orozco, Daniel Mosse, Sergio Ochoa, and Roc Meseguer. Flying real-time network to coordinate disaster relief activities in urban areas. *Sensors*, 18(5):1662, May 2018.

[71] Rodrigo M. Santos, Javier Orozco, Sergio F. Ochoa, Roc Meseguer, and Daniel Mosse. Providing real-time message delivery on opportunistic networks. *IEEE Access*, 6:40696–40712, 2018.

[72] M. Finochietto. Real-time mqtt-mosca. `https://github.com/marianofino/realtime-mqtt/tree/e212b31dfdfebefabf76ed010911abcc0ee09969`, 2019.

[73] Chayan Sarkar, Akshay Uttama Nambi S. N., R. Venkatesha Prasad, Abdur Rahim, Ricardo Neisse, and Gianmarco Baldini. DIAT: A scalable distributed architecture for IoT. *IEEE Internet of Things Journal*, 2(3):230–239, June 2015.

[74] Ivan Lanese, Luca Bedogni, and Marco Di Felice. Internet of things: A process calculus approach. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*. ACM Press, 2013.

[75] Adnan Aijaz and A. Hamid Aghvami. Cognitive machine-to-machine communications for internet-of-things: A protocol stack perspective. *IEEE Internet of Things Journal*, 2(2):103–112, April 2015.

[76] Alexios Lekidis, Emmanouela Stachtiari, Panagiotis Katsaros, Marius Bozga, and Christos K. Georgiadis. Using BIP to reinforce correctness of resource-constrained IoT applications. In *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*. IEEE, June 2015.

[77] Ananda Basu, Saddek Bensalem, Marius Bozga, Paraskevas Bourgos, Mayur Maheshwari, and Joseph Sifakis. Component assemblies in the context of manycore. In *Formal Methods for Components and Objects*, pages 314–333. Springer Berlin Heidelberg, 2013.

[78] Benjamin Aziz. A formal model and analysis of an IoT protocol. *Ad Hoc Networks*, 36:49–57, January 2016.

[79] Jorge E. Luzuriaga, Marco Zennaro, Juan Carlos Cano, Carlos Calafate, and Pietro Manzoni. A disruption tolerant architecture based on MQTT for IoT applications. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, January 2017.

[80] In Lee and Kyoochun Lee. The internet of things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440, July 2015.

[81] L.A. Grieco, A. Rizzo, S. Colucci, S. Sicari, G. Piro, D. Di Paola, and G. Boggia. IoT-aided robotics applications: Technological implications, target domains and open issues. *Computer Communications*, 54:32–47, December 2014.

[82] Song Han, Thomas Lin, Deji Chen, and Mark Nixon. WirelessCHARM: An open system low cost wireless marshalling module for industrial environments. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, March 2014.

[83] Salvatore Distefano, Giovanni Merlino, and Antonio Puliafito. A utility paradigm for IoT: The sensing cloud. *Pervasive and Mobile Computing*, 20:127–144, July 2015.

[84] Subha Koley and Prasun Ghosal. An IoT enabled real-time communication and location tracking system for vehicular emergency. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, July 2017.

[85] Hermann Auernhammer. Precision farming — the environmental challenge. *Computers and Electronics in Agriculture*, 30(1-3):31–43, February 2001.

[86] Alex McBratney, Brett Whelan, Tihomir Ancev, and Johan Bouma. Future directions of precision agriculture. *Precision Agriculture*, 6(1):7–23, February 2005.

[87] Petri Luoto, Mehdi Bennis, Pekka Pirinen, Sumudu Samarakoon, Kari Horneman, and Matti Latva-aho. System level performance evaluation of lte-v2x network. In *European Wireless 2016; 22th European Wireless Conference*, pages 1–5, 2016.

[88] Petri Luoto, Mehdi Bennis, Pekka Pirinen, Sumudu Samarakoon, Kari Horneman, and Matti Latva-aho. Vehicle clustering for improving enhanced LTE-v2x network performance. In *2017 European Conference on Networks and Communications (EuCNC)*. IEEE, June 2017.

[89] Aloÿs Augustin, Jiazi Yi, Thomas Clausen, and William Townsley. A study of LoRa: Long range & low power networks for the internet of things. *Sensors*, 16(9):1466, September 2016.

[90] IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications.

[91] Kees Wevers and Meng Lu. V2x communication for its.

[92] Pengfei Wang, Boya Di, Hongliang Zhang, Kaigui Bian, and Lingyang Song. Cellular v2x communications in unlicensed spectrum: Harmonious coexistence with VANET in 5g systems. *IEEE Transactions on Wireless Communications*, 17(8):5212–5224, August 2018.

[93] Alessandro Bazzi, Alberto Zanella, and Barbara Mavi Masini. An OFDMA-based MAC protocol for next-generation VANETs. *IEEE Transactions on Vehicular Technology*, 64(9):4088–4100, September 2015.

[94] Keiichi Yasumoto, Hirozumi Yamaguchi, and Hiroshi Shigeno. Survey of real-time processing technologies of IoT data streams. *Journal of Information Processing*, 24(2):195–202, 2016.

[95] Bjorn Konieczek, Michael Rethfeldt, Frank Golatowski, and Dirk Timmermann. Real-time communication for the internet of things using jCoAP. In *2015 IEEE 18th International Symposium on Real-Time Distributed Computing*. IEEE, April 2015.

[96] Bjoern Konieczek, Michael Rethfeldt, Frank Golatowski, and Dirk Timmermann. A distributed time server for the real-time extension of CoAP. In *2016 IEEE 19th International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, May 2016.

[97] Gabriel Eggly, Mariano Finochietto, Emmanouil Dimogerontakis, Rodrigo Santos, Javier Orozco, and Roc Meseguer. Real-time primitives for CoAP: Extending the use of IoT for time constraint applications for social good. *Proceedings*, 2(19):1257, October 2018.

[98] Sefki Kolozali, Maria Bermudez-Edo, Daniel Puschmann, Frieder Ganz, and Payam Barnaghi. A knowledge-based approach for real-time IoT data stream annotation and processing. In *2014 IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*. IEEE, September 2014.

[99] *Data Distribution Service for Real-time Systems Specification*, 2015.

[100] Tizar Rizano, L. Abeni, and L. Palopoli. Experimental evaluation of the real-time performance of publish-subscribe middlewares. In *REACTION*, 2013.

[101] Pallavi Sethi and Smruti R. Sarangi. Internet of things: Architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017:1–25, 2017.

[102] *Pattern-oriented software architecture: a system of patterns*. Wiley, 1996.

[103] Carlos Rodríguez-Domínguez, Kawtar Benghazi, Manuel Noguera, José Luis Garrido, María Luisa Rodríguez, and Tomás Ruiz-López. A communication model to integrate the request-response and the publish-subscribe paradigms into ubiquitous systems. *Sensors*, 12(6):7648–7668, June 2012.

[104] Basem Al-Madani, Anas Al-Roubaiey, and Zubair A. Baig. Real-time QoS-aware video streaming: A comparative and experimental study. *Advances in Multimedia*, 2014:1–11, 2014.

[105] J.A. Stankovic. Misconceptions about real-time computing: a serious problem for next-generation systems. *Computer*, 21(10):10–19, October 1988.

[106] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, January 1973.

[107] Roger A Light. Mosquitto: server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13):265, May 2017.

[108] M. Collina. Mosca lib/server.js, Nov 2019.

[109] M. Finochietto. Mqtt graphical interface. `http://192.241.222.173:4101/demo/`, 2019.

[110] M. De Sanctis, E. Cianca, G. Araniti, I. Bisio, and R. Prasad. Satellite communications supporting internet of remote things. *IEEE Internet of Things Journal*, 3(1):113–123, 2016.

[111] Zoltán Bánhidi. The impact of broadband networks on growth and development in south america. *Periodica Polytechnica Social and Management Sciences*, 29(1):33–39, September 2020.

[112] Elias Yaacoub and Mohamed-Slim Alouini. Efficient fronthaul and backhaul connectivity for IoT traffic in rural areas. *IEEE Internet of Things Magazine*, 4(1):60–66, March 2021.

[113] Fei Gu, Jianwei Niu, Landu Jiang, Xue Liu, and Mohammed Atiquzzaman. Survey of the low power wide area network technologies. *Journal of Network and Computer Applications*, 149:102459, 2020.

[114] M. R. Palattella and N. Accettura. Enabling internet of everything everywhere: Lpwan with satellite backhaul. In *2018 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–5, 2018.

[115] Ivan I. Lysogor, Leonid S. Voskov, and Sergey G. Efremov. Survey of data exchange formats for heterogeneous LPWAN-satellite IoT networks. In *2018 Moscow Workshop on Electronic and Networking Technologies (MWENT)*. IEEE, March 2018.

[116] Ivan Lysogor, Leonid Voskov, Alexey Rolich, and Sergey Efremov. Study of data transfer in a heterogeneous LoRa-satellite network for the internet of remote things. *Sensors*, 19(15):3384, August 2019.

[117] F. Holik, U. Roedig, and N. Race. Lora-sdn: Providing wireless iot edge network functions via sdn. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1795–1800, 2020.

[118] Jakub Jalowiczor, Jan Rozhon, and Miroslav Voznak. Study of the efficiency of fog computing in an optimized LoRaWAN cloud architecture. *Sensors*, 21(9):3159, May 2021.

[119] Pape Barro, Marco Zennaro, Jules Degila, and Ermanno Pietrosemoli. A smart cities LoRaWAN network based on autonomous base stations (BS) for some countries with limited internet access. *Future Internet*, 11(4):93, April 2019.

[120] Pape Abdoulaye Barro, Marco Zennaro, and Ermanno Pietrosemoli. TLTN – the local things network: on the design of a LoRaWAN gateway with autonomous servers for disconnected communities. In *2019 Wireless Days (WD)*. IEEE, April 2019.

[121] Samir Dawaliby, Abbas Bradai, and Yannis Pousset. Adaptive dynamic network slicing in LoRa networks. *Future Generation Computer Systems*, 98:697–707, September 2019.

[122] Eduardo Lima, Jean Moraes, Helder Oliveira, Eduardo Cerqueira, Sherali Zeadally, and Denis Rosário. Adaptive priority-aware LoRaWAN resource allocation for internet of things applications. *Ad Hoc Networks*, page 102598, July 2021.

[123] Mariano Finochietto, Gabriel M. Eggly, Rodrigo Santos, Javier Orozco, Sergio F. Ochoa, and Roc Meseguer. A role-based software architecture to support mobile service computing in IoT scenarios. *Sensors*, 19(21):4801, November 2019.

[124] Lorawan® backend interfaces technical specification (ts002-1.1.0). Technical report, LoRa Alliance, October 2020. https://lora-alliance.org/wp-content/uploads/2020/11/TS002-1.1.0_LoRaWAN_Backend_Interfaces.pdf.

[125] Lorawan ® l2 1.0.4 specification (ts001-1.0.4). Technical report, LoRa Alliance, October 2020. https://lora-alliance.org/wp-content/uploads/2020/11/LoRaWAN-1.0.4-Specification-Package_0.zip.

[126] Orne Brocaar. Github - chirpstack network server. `https://github.com/brocaar/chirpstack-network-server/blob/1a7ed28f5293d124029093cbb96821e5b9037bc9/internal/uplink/collect.go#L26`, 2021.

[127] Davide Magrin, Marco Centenaro, and Lorenzo Vangelista. Performance evaluation of lora networks in a smart city scenario. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7, 2017.

[128] Martina Capuzzo, Davide Magrin, and Andrea Zanella. Confirmed traffic in lorawan: Pitfalls and countermeasures. In *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 1–7, 2018.

[129] Davide Magrin, Martina Capuzzo, and Andrea Zanella. A thorough study of lorawan performance under different parameter settings. *IEEE Internet of Things Journal*, 7(1):116–127, 2020.

[130] Jose Mariano Finochietto. Github - ns3-lorawan-simulation. `https://github.com/marianofino/ns3-lorawan-simulation/tree/c7f08dc7dfafef899d67747ff14efc225a3ef49e`, 2021.

[131] Lorna Booth, Jehoshua Bruck, Massimo Franceschetti, and Ronald Meester. Covering algorithms, continuum percolation and the geometry of wireless networks. *The Annals of Applied Probability*, 13(2), May 2003.

[132] Stefano Romagnolo Davide Magrin, Martina Capuzzo and Michele Luvisotto. Github - lorawan ns-3 module. `https://github.com/signetlabdei/lorawan/tree/9f373c23076c3e821e6a6104bde8409e1a24a9fe`, 2021.