ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ANÁLISIS COMPARATIVO DE LOS PROTOCOLOS LwM2M Y MQTT-SN PARA ADMINISTRACIÓN DE SISTEMAS IoT

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

ANGEL LEONARDO ALCOCER MALLITASIG

angel.alcocer@epn.edu.ec

DIRECTOR: SORAYA LUCÍA SINCHE MAITA, PhD.

soraya.sinche@epn.edu.ec

DMQ, septiembre 2024

CERTIFICACIONES

Yo, Angel Leonardo Alcocer Mallitasig declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

ANGEL LEONARDO ALCOCER MALLITASIG

Certifico que el presente trabajo de integración curricular fue desarrollado por Angel Leonardo Alcocer Mallitasig, bajo mi supervisión.

SORAYA LUCÍA SINCHE MAITA, PhD.
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Angel Leonardo Alcocer Mallitasig

Soraya Lucía Sinche Maita, PhD.

DEDICATORIA

A mi madre, Rosario, quien nunca cesó de apoyarme incluso en mis momentos más difíciles. Tu amor incondicional y fortaleza fueron la luz que guió mi camino a lo largo de esta travesía académica.

A mis profesores y mentores, Pablo Wilian Hidalgo Lascano y Soraya Lucia Sinche Maita, por su sabia guía, apoyo académico y la enorme dedicación que mostraron para apoyarme en la culminación de mis estudios. Su compromiso y orientación fueron pilares fundamentales que guiaron cada paso de este camino académico, proporcionándome inspiración y motivación constantes

A mis gatitas Panchita y Cachita, cuyo amor y compañía perduran en mi corazón y en mi vida.

Este trabajo está dedicado a ustedes, quienes han sido mi fuerza e inspiración. En cada página escrita queda reflejada mi profunda gratitud hacia ustedes.

AGRADECIMIENTO

A mi madre, Rosario, por tu sacrificio y aliento constante.

A mi mentor, MSc. Pablo Wilian Hidalgo Lascano, y a mi directora de tesis, PhD. Soraya Lucia Sinche Maita, por su constante apoyo y paciencia durante el desarrollo de este trabajo. Su conocimiento y orientación fueron fundamentales para su realización.

A mis gatitas Panchita y Cachita, quienes me acompañaron durante todo este trayecto y lo seguirán haciendo a través de sus recuerdos.

Finalmente, agradezco a todos aquellos que creyeron en mí y me brindaron su apoyo durante este camino.

Cada uno de ustedes ha dejado una huella imborrable en mi trayectoria académica y personal. Este logro no habría sido posible sin ustedes.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de integración curricular abarca el análisis y comparación de los protocolos de administración IoT LwM2M(*Lightweight Machine to Machine*) [1] y MQTT-SN(*Message Queing Telemetry Transport Protocol)*[2], cada uno en su propio escenario. El siguiente trabajo de integración curricular consta de tres capítulos:

El primer capítulo abarca los conceptos necesarios para comprender el funcionamiento de los protocolos de administración IoT y las herramientas a utilizar.

El segundo capítulo consta de la implementación de los escenarios de prueba incluyendo el levantamiento de servidores, codificación de clientes y lectura de sensores. Esta información es enviada por cada cliente a su servidor (*LwM2M*) o Bróker (*MQTT-SN*) cada cierto intervalo de tiempo. Se realizó el estudio de la estructura y funcionamiento de los protocolos para comprender su comportamiento, la cabecera que es insertada por cada protocolo. En ambos escenarios el flujo de tráfico está divido en información y administración para su análisis, estableciendo un protocolo de pruebas que permite analizar el comportamiento de cada protocolo en su escenario de prueba.

Finalmente, en el capítulo tres mediante gráficas se observa la cantidad de tráfico que se intercambia para cada protocolo a partir de gráficas y tablas. Las pruebas abarcan varios ambientes: con envío de información, sin envío de información, durante *Video Streaming* y mientras se realiza la descarga de un archivo. Las conclusiones y recomendaciones reflejan el comportamiento de los protocolos dentro de su escenario en cada ambiente y en que aplicaciones pueden ser empleados. Finalmente se presentan las conclusiones y recomendaciones producto del presente trabajo de integración curricular.

PALABRAS CLAVE: LwM2M, MQTT-SN, Leshan, RSMB, IoT, Administración.

ABSTRACT

The present work is focused on the analysis and comparison of the IoT management protocols LwM2M (Lightweight Machine to Machine) [1] and MQTT-SN (Message Queuing Telemetry Transport Protocol) [2], each in its scenario. This work includes three chapters:

Chapter 1 covers the concepts needed to understand the operation of IoT management protocols and the tools used in the present work.

Chapter 2 involves developing test scenarios, including server surveys, client encoding, and sensor readings. Each client periodically sends this information to its server (LwM2M) or broker (MQTT-SN). The study of the structure and functioning of the protocols was conducted to understand their behavior and the headers they insert. In both scenarios, the traffic flow is divided into information and administration for analysis. A test protocol was established to allow for the behavior analysis of each protocol within its respective scenario.

Finally, Chapter 3 presents the volume of traffic exchanged for each protocol using graphs and tables. Testing is conducted across several environments: with information transmission, without information transmission, during video streaming, and while downloading a file. The findings and recommendations highlight the behavior of protocols within their respective scenarios in each environment and identify suitable applications for their use. Finally, the conclusions and recommendations are presented based on the obtained results.

KEYWORDS: LwM2M, MQTT-SN, Leshan, RSMB, IoT, Management.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Cada día más dispositivos se integran a los sistemas IoT (*Internet of Things*) por lo que la administración se ha convertido en un desafío. En la actualidad existen diferentes protocolos utilizados para administrar sistemas IoT tales como: LwM2M (LightWeight Machine to Machine) [1], MQTT (*Message Queuing Telemetry Transport*) [2] y MQTT-SN (*Message Queuing Telemetry Transport For Sensor Networks*) [2].

Al momento de implementar un sistema de administración de dispositivos IoT es importante seleccionar el protocolo de administración adecuado. Es por esta razón que se hace indispensable comprender el funcionamiento de estos protocolos y su impacto en el tráfico generado dentro de la red. Sin la selección adecuada del protocolo de administración en un Sistema IoT, éste no tendrá un rendimiento eficiente y una adecuada utilización de sus recursos.

Por estas razones, en el presente trabajo de integración curricular se propone realizar un análisis comparativo de los protocolos de administración más utilizados como son LwM2M y MQTT-SN dentro de un sistema IoT. Para este análisis se implementarán escenarios de prueba utilizando sistemas embebidos y diferentes tipos de sensores.

1.1 Objetivo general

Analizar comparativamente los protocolos LwM2M y MQTT-SN para la administración de Sistemas IoT.

1.2 Objetivos específicos

1. Analizar los fundamentos teóricos que permitan comprender el funcionamiento de los protocolos LwM2M y MQTT-SN.
2. Analizar las condiciones de prueba que permitan comparar el rendimiento de los protocolos LwM2M y MQTT-SN.
3. Implementar escenarios de prueba para los protocolos LwM2M y MQTT-SN.
4. Analizar los resultados obtenidos.

1.3 Alcance

El análisis comparativo que se propone se orienta a los protocolos LwM2M y MQTT-SN en escenarios prácticos para la administración de dispositivos IoT. En primer lugar, se estudia

cada uno de los protocolos, en lo que respecta a formatos de trama, intercambio de mensajes y funcionalidad para el proceso de administración

Para el análisis de estos protocolos se implementan dos escenarios prácticos que operan de igual manera. En el escenario 1 se utiliza el protocolo LwM2M, mientras que MQTT-SN se lo implementa en el escenario 2 (ver Figura 1.1 y Figura 1.2).

Para cada escenario, se implementa un servidor basado en el framework Leshan en el caso del protocolo LwM2M y RSMB para la implementación del protocolo MQTT-SN, los cuales son levantados en una misma máquina virtual con Ubuntu.

Los clientes para cada protocolo están levantados en el dispositivo embebido Raspberry Pi 4.

Los clientes están conectados a un Arduino, el cual realiza el proceso de adquisición de datos con diferentes tipos de sensores (BH1750 para luz, DHT22 para temperatura y humedad, y HC-SR04 para proximidad). Estos datos provenientes de los sensores son enviados al Raspberry Pi 4 mediante comunicación serial. Para el cliente LwM2M los datos son tratados a partir de lenguaje Java, mientras que Python es el lenguaje utilizado para el cliente MQTT-SN. El tratamiento para el envío de información al servidor depende del protocolo de administración IoT.

Para cada protocolo el envío de datos es automatizado, de manera de poder analizar los mensajes de administración por cada protocolo y su tiempo de respuesta mediante un analizador de tráfico.

El proceso de comunicación para cada protocolo se describe a continuación:

- **LwM2M:** El servidor LwM2M recibe el mensaje y extrae los datos para presentarlos en la interfaz web proporcionada por LESHAN [3]. Mediante la configuración del cliente LwM2M de Leshan, se recolectan y envían los datos al servidor LwM2M a través de la red Wifi.
- **MQTT-SN**: Para MQTT-SN, la recolección y envío de datos al servidor RSMB se lo realiza en lenguaje Python. Para MQTT-SN, se usa el Broker RSMB, y los datos son procesados de manera que los clientes que se suscriban al tópico determinado observen los datos publicados.

El análisis comparativo de cada protocolo se lo realiza de acuerdo con parámetros tales como retardo y overhead.

Dentro del protocolo de pruebas, el dispositivo Arduino realiza lecturas de los datos de los sensores cada cierto tiempo, el cual es indicado por la Raspberry Pi 4. Se tomarán datos cada 5, 10, 15 y 20 minutos. El análisis se aplica sobre el overhead introducido por el protocolo de administración IoT con respecto a la información enviada, la cual es proporcionada por los sensores DHT22 (Temperatura y Humedad), HC-SR04 (Proximidad) y BH1750(Luz).

Los datos obtenidos están reflejados en gráficas de manera que permiten observar el comportamiento de cada protocolo, así como determinar las condiciones a partir de las cuales se puede obtener un mejor rendimiento en función del tipo de aplicación sobre el cual se va a realizar la administración.

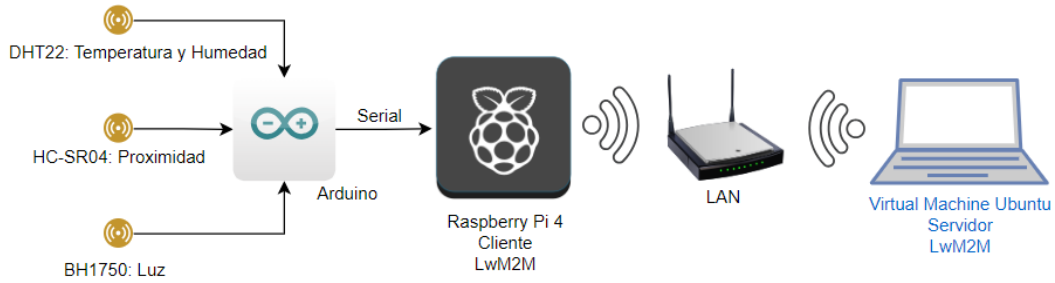

Figura 1.1 Escenario 1 utilizando el protocolo LwM2M

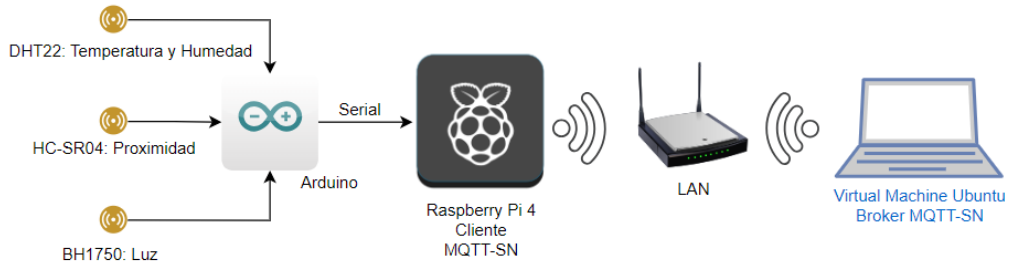

Figura 1.2 Escenario 2 utilizando el protocolo MQTT-SN

1.4 Marco teórico

1.4.1 Conceptos básicos de Sistemas IoT

Un sistema IoT consiste en una red de objetos o cosas, las cuales utilizan tecnologías de comunicaciones, software y sensores, que permitan conectarse con otros dispositivos y sistemas para poder intercambiar datos a través de Internet. [4]

En un sistema IoT puede ser utilizada la tecnología *Machine to Machine* (M2M) la cual permite la comunicación o intercambio de información entre dos dispositivos o máquinas remotas. El entorno M2M está compuesto por las máquinas que intercambian información

entre sí, donde diferentes tipos de sensores recolectan información y la transfieren a un punto central. [4]

Un sistema IoT se encuentra conformado por:

- Cosas: Sensores, actuadores, sistemas embebidos y objetos que se utilizan en la vida cotidiana, del hogar e incluso de la industria.
- Comunicación: Incluye tecnologías de comunicación que permiten que los dispositivos estén interconectados y se tenga un ecosistema IoT.
- Computación y Almacenamiento: donde se realiza el procesamiento de datos, por ejemplo, *Big Data*, *Machine Learning* e Inteligencia Artificial dentro de la Nube.
- Servicios y Aplicaciones: En el mercado IoT, entre las principales una variedad de servicios y aplicaciones tales como: monitorización de tráfico, control de salud, agricultura, industria, *Smart Grids* y ahorro energético, distribución de agua, gestión y monitoreo automatizado y remoto de equipos, mantenimiento predictivo, buen manejo de inventario y control de calidad. [4]

1.4.2 Administración de Sistemas IoT

La administración es una parte esencial en toda red, permitiendo el monitoreo de su estado, controlar su operación, detectar fallas y otras funcionalidades.

Las ventajas de una administración en un sistema IoT abarcan el tener un inventario sobre los recursos del sistema IoT, control sobre fallas, mantenimiento preventivo y correctivo, y seguimiento de recursos.

A medida que a un sistema IoT se incorporan más dispositivos, su administración se vuelve un desafío. Por otra parte, la heterogeneidad de los dispositivos, la cual consiste en la existencia de dispositivos de diferentes fabricantes, es uno de los mayores retos.

Cuando un dispositivo IoT se conecta a una red, este debe ser capaz de asociarse a la misma y poder ser administrado mediante mecanismos estandarizados, los cuales deben ser soportados, sin importar el fabricante.

1.4.3 Protocolos de administración IoT

El uso de protocolos de administración IoT permite gestionar el gran número de dispositivos que forman parte de un sistema IoT. Estos dispositivos disponen de recursos limitados ya sea en capacidad o consumo de energía, por lo que los protocolos de administración a

implementar en sistemas IoT deben garantizar un bajo consumo de energía, memoria y ancho de banda.

En la Tabla 1.1 se encuentran los protocolos principales de administración desarrollados por organizaciones tales como ISO (*International Organization for Standarization*), IETF (*Internet Engineering Task Force*) y OMA(*Open Mobile Alliance*).

Tabla 1.1 Protocolos de administración [5]

Organización	Protocolos de Administración
ISO	CMIP (*Common Management Information Protocol*)
IETF	SNMP (*Simple Network Management Protocol*) NETCONF (*Network Configuration Protocol*) RESTCONF (*Representational State Transfer Configuration Protocol*) CoMI (*CoAP Management Interface*)
OMA	DM (*Device Management Protocol*) LwM2M (*Lightweight Machine to Machine*)
Otros	LNMP (*LowPAN Network Management Protocol*) MQTT (Message Queuing Telemetry Transport) MQTT-SN (Message Queuing Telemetry Transport for Sensor Networks)

1.4.3.1 LwM2M

Introducido por la *Open Mobile Alliance*, está orientado a la administración de dispositivos IoT. Es un protocolo ligero y estructurado, lo que lo hace adecuado cuando se dispone de dispositivos de baja capacidad, tal como los dispositivos embebidos.

Este protocolo usualmente trabaja sobre el puerto 5683 y es apropiado tanto para plano de datos como para administración de dispositivos con capacidad de memoria limitada; además, consume un nivel bajo de energía, por tanto, es apropiado para dispositivos con batería limitada. [1]

LwM2M corre sobre el protocolo CoAP(*Constrained Application Protocol*), siendo este el encargado de administrar las peticiones y respuestas cliente/servidor, por lo tanto, es un pilar importante al momento de estudiar el funcionamiento del protocolo LwM2M.[1]

CoAP se basa en un modelo REST (*Representational State Transfer*), que proporciona aplicaciones RESTful modeladas en la semántica de HTTP (*Hypertext Transfer Protocol*). En este patrón de diseño, el servidor proporciona una URI (Identificador uniforme de recursos) para que los clientes puedan acceder a los recursos del sitio a partir de los métodos GET, PUT, POST y DELETE. [1]

Además, CoAP está diseñado para correr en dispositivos simples y sobre UDP, teniendo un enfoque cliente-servidor.

En la Figura 1.3 se observa el stack de protocolos reducido sobre el cual trabaja LwM2M. CoAP, quien encapsula a LwM2M, puede hacer uso opcional de seguridad mediante el protocolo DTLS (*Datagram Transport Layer Security*) o ser transportado directamente en UDP o sobre SMS en redes celulares. [1]

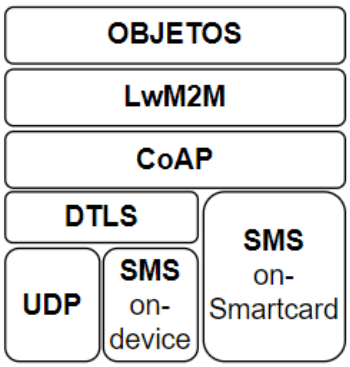

Figura 1.3 Stack de protocolo sobre el cual trabaja LwM2M [1]

En el modelo de recursos definido por el protocolo LwM2M (Figura 1.4), un recurso consiste en cada elemento de información del cliente LwM2M. Los recursos están lógicamente organizados en Objetos, de los cuales se pueden tener múltiples instancias. Los objetos definidos por el protocolo LwM2M tienen asignados cada uno un identificador único. [1]

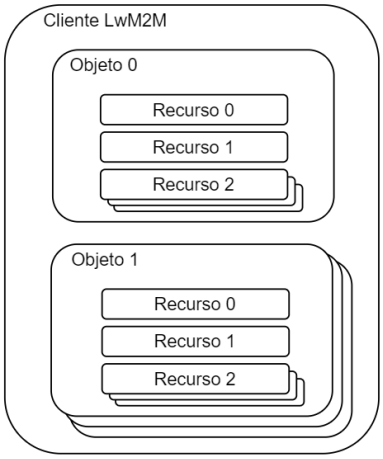

Figura 1.4 Modelo de Recursos LwM2M [1]

En la Figura 1.5 se presenta el paradigma bajo el cual LwM2M opera basado en un esquema cliente-servidor. En este paradigma los dispositivos M2M corresponden a los clientes y el servidor LwM2M a una plataforma, servicio o aplicación M2M.

Para que el dispositivo cliente y el servidor puedan establecer, finalizar y mantener una comunicación se emplean los siguientes procedimientos:

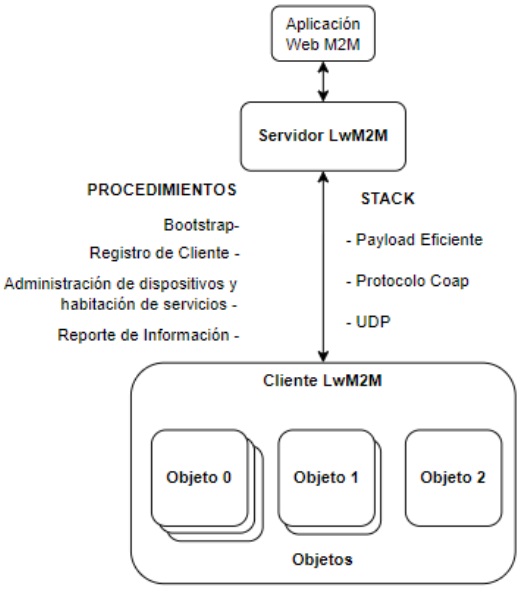

Figura 1.5 Paradigma cliente-servidor LwM2M [1]

1. Bootstrap
2. Registro de Cliente
3. Administración de Dispositivos y Establecimiento de servicio
4. Reporte de Información

1. Bootstrap: Provee información necesaria al cliente LwM2M acerca del o los servidores LwM2M, de manera que este pueda registrarse con uno o más servidores LwM2M.

Existen cuatro modos Bootstrap soportados por esta interfaz: de fábrica, desde una *Smartcard*, iniciado por el Cliente e iniciado por el Servidor. El cliente LwM2M debe soportar al menos un modo Bootstrap. En el presente trabajo se utiliza modo de fábrica.

En el modo de fábrica, el cliente está configurado con la información necesaria antes de ser ejecutado. Ésta puede corresponder a la información Bootstrap del servidor LwM2M-Bootstrap y/o la información Bootstrap del servidor LwM2M. En este modo, el cliente LwM2M puede conectarse a un servidor LwM2M sin la necesidad previa de conectarse a un servidor Bootstrap LwM2M. [1]

2. Registro de cliente: El cliente LwM2M usa esta interfaz para registrarse con el servidor LwM2M, mantener su registro y eliminarlo a través de las operaciones *Register, Update y De-Register* respectivamente como se observa en la Figura 1.6.

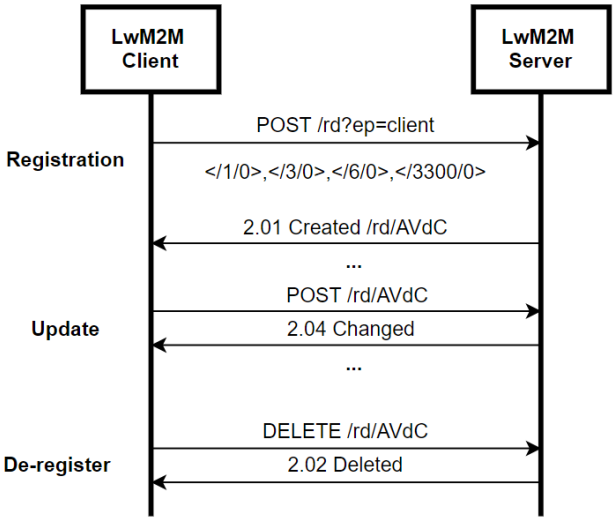

Figura 1.6 Registro de cliente LwM2M [1]

3. **Administración de Dispositivos y Establecimiento del Servicio:** Usada por el servidor LwM2M para acceder a las instancias y a los recursos disponibles del cliente LwM2M registrado (Figura 1.7). Las operaciones disponibles por parte de esta interfaz son *Create, Read, Write, Delete, Execute, Write-Attributes y Discover*.

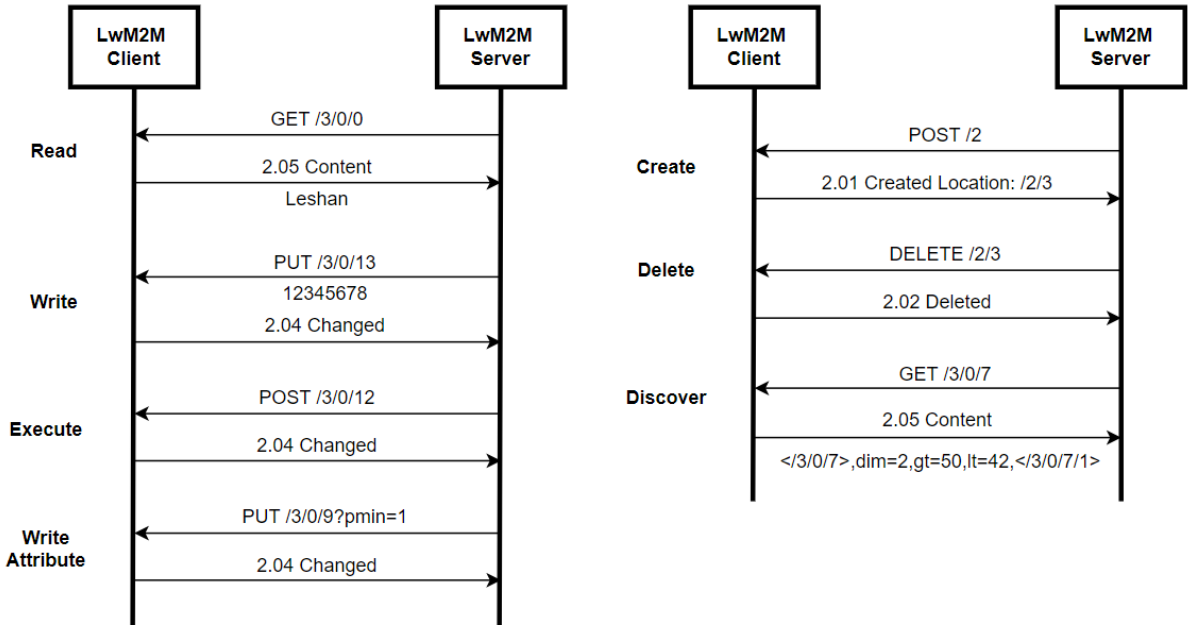

Figura 1.7 Administración de dispositivos y Establecimiento de Servicio [1]

4. **Reporte de Información:** Con la operación Observe, se recibe notificaciones cuando se encuentran disponibles nuevos valores. En la Figura 1.8 se presenta la Interfaz utilizada por el Servidor LwM2M para observar cualquier cambio en un recurso del cliente a partir del envío de una operación Observe.

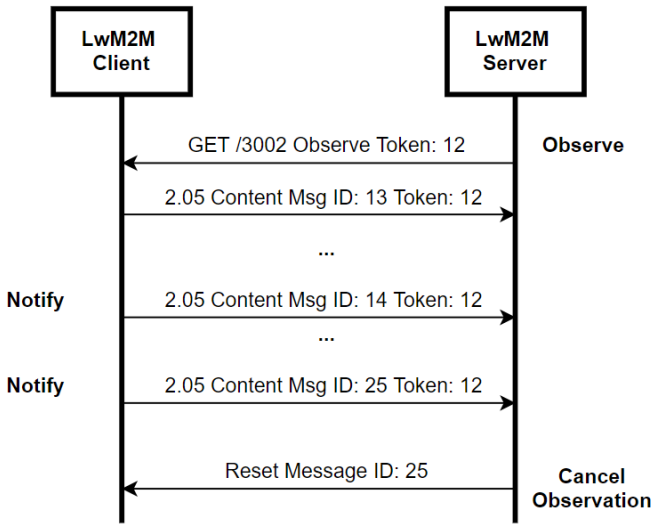

Figura 1.8 Reporte de información [1]

1.4.3.2 MQTT-SN

MQTT-SN es un protocolo ligero orientado a redes inalámbricas de sensores, que trabaja bajo el modelo publicación/suscripción, basado en MQTT (*Message Queuing Telemetry Transport*), con la diferencia de que no requiere una conexión permanente y tiene un *payload* reducido. [2]

MQTT-SN trabaja sobre UDP, como se puede ver en la Figura 1.9, en el puerto 1885. Permite el uso de ID o tópicos de nombre corto en lugar del nombre de tópico para publicación de mensajes. El cliente tiene capacidad para descubrimiento de bróker y este para anunciarse; además, permite el uso de clientes en modo de ahorro de energía solo para enviar información. [2]

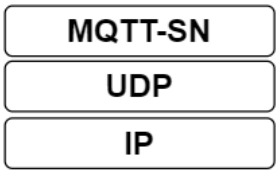

Figura 1.9 Stack de Protocolos MQTT-SN [2]

MQTT-SN maneja varios tipos de tópicos para publicación de mensajes tal como el **ID de Tópico** (el bróker asigna un ID al nombre de tópico), **tópico predefinido** (previamente conocido por el cliente y bróker) y **tópico de nombre corto** (formado por 2 octetos).

En la Figura 1.10 se presenta el paradigma de publicación/subscripción con el que opera MQTT-SN, el cual es un modelo de envío de mensajes asíncrono, donde uno o más clientes se suscriben a uno o varios tópicos, a los cuales otros clientes o sensores puede publicar información, todo esto mediante un Bróker.

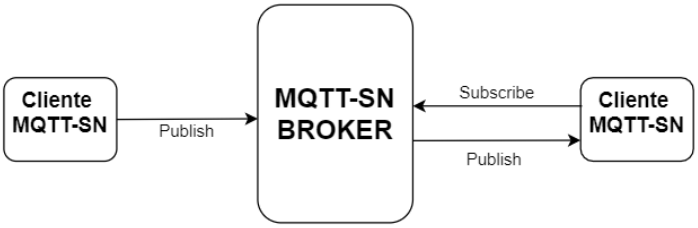

Figura 1.10 Paradigma Publicación-Suscripción MQTT-SN

MQTT-SN maneja niveles de calidad de servicio para publicación de mensajes, los cuales son: **nivel 0** (una sola vez máximo), **nivel 1**(al menos una vez, requiere confirmación), **nivel 2** (más confiable, se garantiza mediante dos flujos entre el remitente y receptor) y **nivel -1** (el cliente solo publica información, no hay una conexión inicial con el Bróker, es decir, se tiene establecido un tópico predefinido en el Bróker) [2]

El dispositivo cliente y el bróker pueden establecer, finalizar y mantener una comunicación mediante los siguientes procedimientos:

1. Anuncio y descubrimiento de Bróker
2. Conexión, Ping y Desconexión
3. Registro, Suscripción, Cancelación de Suscripción y Publicación a Tópicos

1. Anuncio y descubrimiento del bróker: Un Bróker periódicamente puede anunciar su presencia a los dispositivos que se encuentran en la red, permitiendo al cliente encontrar la dirección de red de ese Bróker, esto si el cliente no está previamente configurado con la dirección del Bróker.

2. Conexión, Ping y Desconexión: En la Figura 1.11 se observa cómo el cliente MQTT-SN en este procedimiento crea una conexión entre el cliente y el Broker, supervisa si el Bróker está en línea y cierra la conexión con el Bróker.

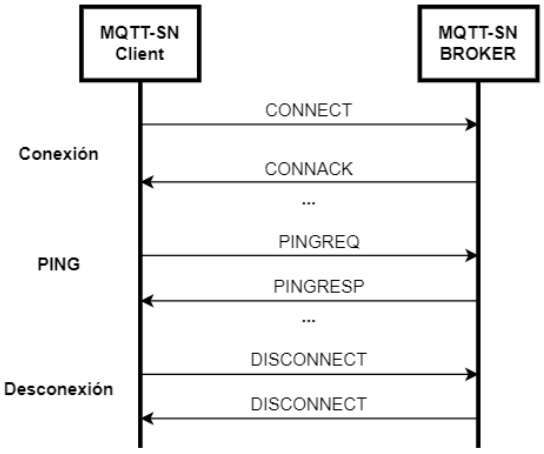

Figura 1.11 Conexión, Ping y Desconexión

3. **Registro, Suscripción, Cancelación de Suscripción y Publicación a Tópicos:** En este procedimiento, como se observa en la Figura 1.12, el cliente registra el tópico en el Broker, el cual asigna y retorna un ID. Una vez establecidos los IDs de tópicos, él o los clientes pueden proceder a la suscripción, publicación o cancelación a tópicos.

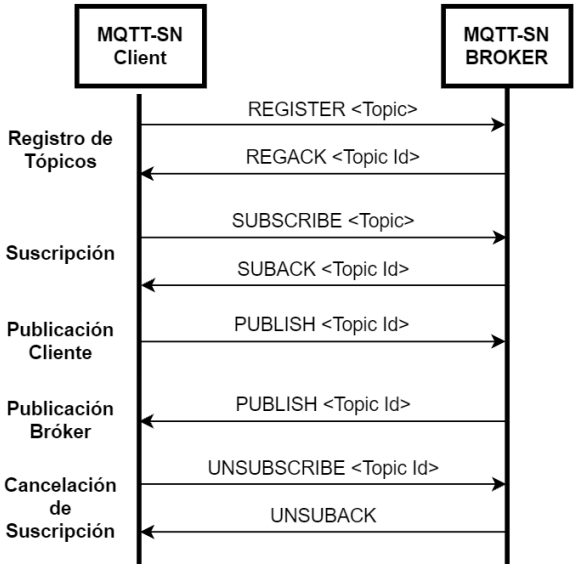

Figura 1.12 Registro, Suscrip, Can-Suscripción y Pub a Tópicos

1.4.4 Plataformas para administración de Sistemas IoT

En el mercado existen varias plataformas para administración de dispositivos IoT, las cuales permiten un mejor manejo de los recursos y su administración. Para el protocolo LwM2M se tiene Eclipse Leshan, mientras que para MQTT-SN está RSMB (*Really Small Message Broker*)

1.4.4.1 Eclipse Leshan

Es un *framework* que maneja el protocolo LwM2M. Como se ve en la Figura 1.13, consta de un servidor, un cliente y un servidor Bootstrap demo para pruebas. Provee librerías Java para desarrollar servidores y clientes propios. El servidor consta de una interfaz web para la administración y control de dispositivos. [3]

Soporta objetos IPSO (Alianza de Protocolo de Internet para Objetos Inteligentes), basados en un modelo de objetos. Consiste en una colección de recursos que posee un identificador único, representando un tipo de sensor físico, actuador u otra fuente de datos. [3]

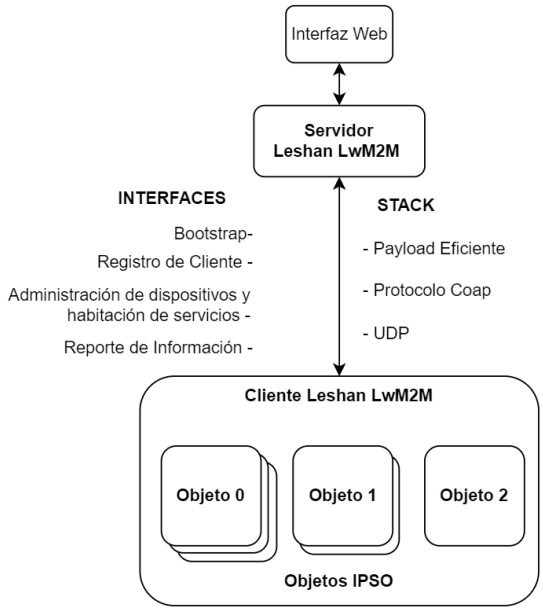

Figura 1.13 Eclipse Leshan, basado en [1]

1.4.4.2 RSMB

Really Small Message Broker funciona para MQTT y MQTT-SN. Se encuentra codificado en lenguaje C, y se puede modificar su funcionamiento a partir de un archivo de configuración.

Permite que un mensaje publicado por un cliente MQTT-SN pueda ser recibido por otro cliente MQTT-SN o por un cliente MQTT y viceversa, comportándose el Bróker como un Gateway entre ambos protocolos (Figura 1.14). [6]

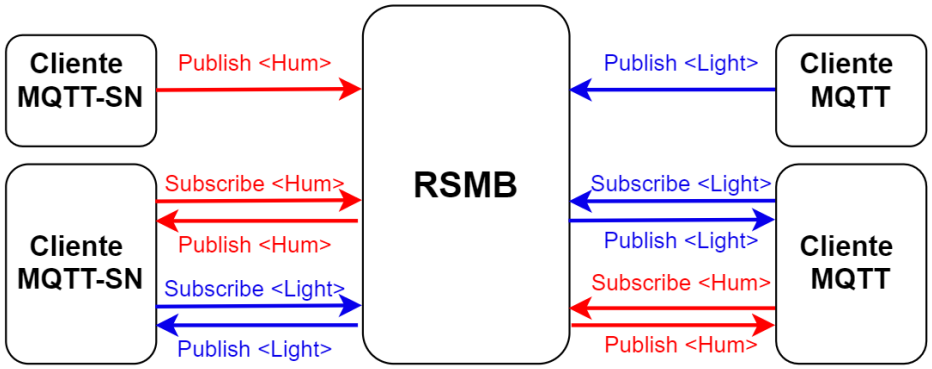

Figura 1.14 Really Small Message Broker [6]

2 METODOLOGÍA

En el presente capítulo se menciona la preparación de escenarios de prueba, codificación en clientes para intercambio de mensajes, instalación de servidor y bróker, análisis de protocolos a nivel de mensaje, establecimiento del protocolo de prueba, todo esto de manera que se pueda obtener la información necesaria para su análisis

2.1 Estructuración de Escenarios de prueba

Para el análisis de los protocolos que son parte del presente trabajo, se implementan dos escenarios de prueba, el primero basado en el protocolo LwM2M y el segundo en MQTT-SN. Los dos escenarios comparten los mismos componentes como se puede observar en la Figura 2.1(LwM2M) y la Figura 2.2(MQTT-SN).

Figura 2.1 Escenario para protocolo LwM2M

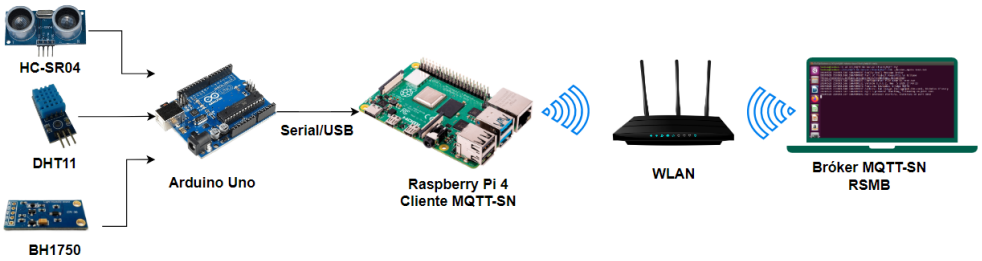

Figura 2.2 Escenario para protocolo MQTT-SN

De acuerdo con un ambiente IoT ambos escenarios están compuestos por:

- Cosas: Sensor de humedad y temperatura (DHT11), de proximidad (HC-SR04) y de luz (BH1750) conectados a un Arduino Uno, para la recolección de datos; los cuales son enviados mediante comunicación serial al dispositivo embebido Raspberry Pi 4 para su procesamiento y manejo por su respectivo protocolo.
- Comunicación: Se establece el envío de mensajes utilizando LwM2M o MQTT-SN según el escenario sobre una red local WiFi.
- Servicios y Aplicaciones: *Frameworks* Leshan y RSMB instalados sobre una máquina virtual Ubuntu.

2.2 Implementación de Servidores y Clientes

En esta sección se explica la configuración y/o programación de los servidores y clientes necesarios para el análisis y comparación de los protocolos objetivo, junto a los comandos para que estos puedan funcionar de forma adecuada.

El servidor LwM2M y el Bróker MQTT-SN se encuentran instalados en una máquina virtual con el Sistema Operativo Ubuntu 16.04. A continuación se explican sus comandos de instalación y uso.

2.2.1 Instalación del Servidor LwM2M Leshan

Para su instalación, LESHAN proporciona un servidor demo LwM2M, el cual puede ser descargado junto al cliente LwM2M y otras librerías con el comando presente en el Código 2.1 en un terminal. [3]

```
git clone https://github.com/eclipse/leshan.git
```

Código 2.1 Descarga de mensaje Leshan

Luego, es necesario realizar la compilación con el Código 2.2. [3]

```
mvn clean install
```

Código 2.2 Instalación de Framework Leshan

Para ejecutar el servidor demo LwM2M se utiliza el comando del Código 2.3 en un terminal, ingresando como parámetro la dirección IP del host donde va a ser ejecutado.

```
sudo java -jar leshan-server-demo-1.0.0-SNAPSHOT-jar-with-dependencies.jar -lh
“Dirección Ip del Host”
```

Código 2.3 Levantamiento de servidor LwM2M

En la Figura 2.3 se observa la interfaz web del servidor demo LwM2M, misma que permite dar seguimiento a los clientes, a partir de la lectura y observación de instancias y sus recursos. Se debe acceder a la dirección IP del host del servidor en el puerto 8080(TCP).

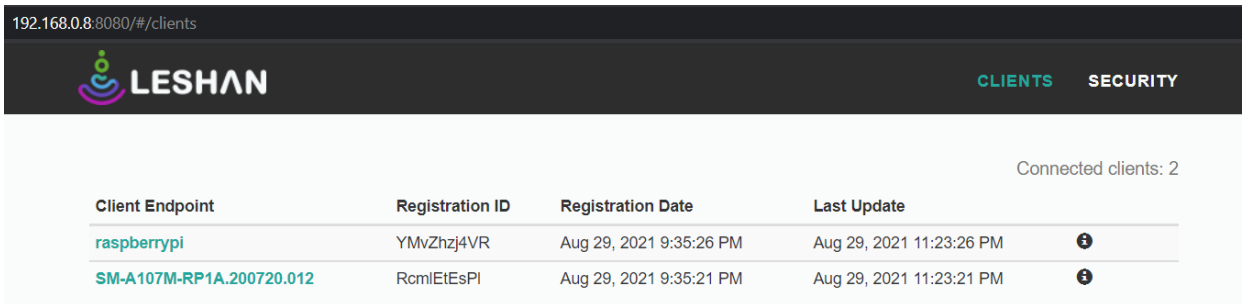

Figura 2.3 Framework Leshan-Servidor Web

2.2.2 Instalación del Bróker MQTT-SN RSMB

Proporcionado por IBM, el bróker RSMB puede ser descargado mediante el comando del Código 2.4. [6]

```
git clone https://github.com/eclipse/mosquitto.rsmb
```

Código 2.4 Descarga de Broker RSMB

Ya descargado, es necesario compilar el Broker ejecutando el comando **make** en el directorio **rsmb/src** para poder ejecutarlo. [6]

El Bróker RSMB y sus funcionalidades pueden ser configuradas a partir de un archivo de configuración. En el Código 2.5 se encuentra en el archivo de configuración y permite que el Broker pueda trabajar con el protocolo MQTT-SN. [6]

```
trace_output protocol            // Se muestra en consola cada mensaje recibido
listener 1885 INADDR_ANY mqtts   // Permite escuchar el puerto 1885 desde
                                 // cualquier dirección Ip para el protocolo
                                 // MQTT-SN
```

Código 2.5 Archivo de configuración RSMB

Para ejecutar el Bróker RSMB se utiliza el comando del Código 2.6 en la terminal adjuntando como parámetro el archivo de configuración. [6]

```
./broker_mqtts test.txt
```

Código 2.6 Ejecución de Broker RSMB junto a archivo de configuración

2.3 Programación para la Lectura de Datos

El código a ser grabado en el dispositivo Arduino Uno abarca la lectura y manejo de información, a partir de una conexión serial de los sensores de luz (BH1750), proximidad (HC-SR04), temperatura y humedad (DHT22). En el Código 2.7 se presenta la lectura de información de cada sensor y su preparación para ser leída por cada cliente del dispositivo Raspberry Pi 4. La información es presentada en el puerto serial mediante separadores, de manera que pueda ser procesada de lado del cliente.

Para que los clientes alojados en el dispositivo Raspberry Pi 4 puedan recopilar esta información y la lectura sea independiente a la ejecución del cliente de cada protocolo, se codifica un *script* en lenguaje Python, el cual mediante conexión serial indica al dispositivo Arduino Uno cuando realizar la lectura. El período para la ejecución del *script* Python está definido dentro del código de cada cliente LwM2M y MQTT-SN que se ejecuta en el dispositivo Raspberry Pi 4.

```
lux = lightMeter.readLightLevel();  // Lectura dato iluminación
h = dht.readHumidity();             // Lectura dato húmedad
t = dht.readTemperature();          // Lectura dato temperaturda

digitalWrite(Trigger, HIGH);        // La medición de distancia se realiza
delayMicroseconds(10);              // mediante ultrasonido
digitalWrite(Trigger, LOW);         //
tr= pulseIn(Echo, HIGH);            // Lectura de dato proximidad
d= tr/58.2;                         // Transformación de información a cm

Serial.print("<");
Serial.print(d);    // Impresión dato de distancia
Serial.print(",");
Serial.print(h);    // Impresión dato de humedad
Serial.print(",");
Serial.print(t);    // Impresión dato de temperatura
Serial.print(",");
Serial.print(lux);  // Impresión dato de luz
Serial.println(">");
```

Código 2.7 Lectura de datos de Sensores desde Arduino Uno

En las líneas del Código 2.8 se establece la conexión, lectura de datos y cierre a partir de lenguaje Python. Es necesario tener en cuenta que se necesita un tiempo mínimo de espera de dos segundos para establecer la conexión antes de realizar la lectura de datos.

```
port_arg= '/dev/ttyACM0'    # Puerto a utilizar para conexión serial
ser = serial.Serial(port_arg, 9600)       # Creación de conexión
time.sleep(2)               # Tiempo de espera para creación de conexión
rawserial = ser.readline()  # Lectura de datos de dispositivo Arduino
ser.close();                # Cierre de conexión
```

Código 2.8 Conexión y obtención de datos desde Raspberry Pi 4

En el Código 2.9 se presenta el manejo de datos para su procesamiento realizado por el cliente. Los datos enviados por el dispositivo Arduino Uno son leído como un arreglo, el cual es procesado para obtener cada dato por separado; estos datos son impresos en consola en formato *float* para poder ser procesados por cada cliente.

```
cookedserial = rawserial.decode().strip()
datasplit = cookedserial.split(',')
distancia = datasplit[0].strip('<')
humedad = datasplit[1].strip(',')
temperatura = datasplit[2].strip(',')
luz = datasplit[3].strip('>')
sensores = {distancia,humedad,temperatura,luz}
print('{0:0.1f} {1:0.1f} {2:0.1f} {3:0.1f}'.format( float(distancia),
float(humedad), float(temperatura), float(luz)))
```

Código 2.9 Conversión de datos para poder ser manejados por el cliente

2.3.1 Código cliente LwM2M

Codificado en lenguaje Java, Leshan proporciona un cliente demo [3], este está modificado mediante la agregación y codificación de las clases correspondientes a los parámetros a medir: Luz, Temperatura, Humedad y Proximidad; además la clase “DHT11”, es la encargada de controlar la lectura de datos de manera periódica como se puede ver en el Código *2.10*.

```
public DHT11Class(long period) {
    this.scheduler = Executors.newSingleThreadScheduledExecutor(new NamedThreadFactory("DHT11"));
    scheduler.scheduleAtFixedRate(new Runnable() {

        @Override
        public void run() {
            try {
                readTemperatureAndHumidity();

            } catch (Exception e) {
                System.out.println("Error reading sensor");
                e.printStackTrace();
            }
        }
    }, 2, period, TimeUnit.SECONDS);
}
```

Código 2.10 Ejemplo de clase para control de lectura periódica de datos

En el Código 2.11, se encuentran las líneas necesarias para llamar al *script* Python con el objetivo de recopilar las mediciones de sensores del dispositivo Arduino UNO.

```
 private synchronized void readTemperatureAndHumidity() throws Exception {

     Runtime rt = Runtime.getRuntime();
     Process p = rt.exec("python lecturaarduino.py");
     BufferedReader bri = new BufferedReader(new InputStreamReader(p.getInputStream()));

     if ((line = bri.readLine()) != null) {
         if (!(line.contains("ERR_CRC") || line.contains("ERR_RNG"))) {
             data = line.split(" ");
             distance = Double.parseDouble(data[0]);
             humidity = Double.parseDouble(data[1]);
             temperature = Double.parseDouble(data[2]);
             light = Double.parseDouble(data[3]);


         } else {
             System.out.println("Error reading sensor value");
         }
     }
     bri.close();
     p.waitFor();

     setChanged();
     notifyObservers();
 }
```

Código 2.11 Llamado a Script Python

Un ejemplo de envío de datos del sensor de proximidad se presenta en el Código 2.12, en el cual a partir de la observación por parte del servidor LwM2M, el cliente enviará el recurso solicitado de acuerdo con el período indicado si existe variación en el valor de este.

```
public void update(Observable o, Object arg) {
    currentDistance = dht11.getDistance();

    Integer changedResource = adjustMinMaxMeasuredValue(currentDistance);
    if (changedResource != null) {
        fireResourcesChange(SENSOR_VALUE, changedResource);

    } else {
        fireResourcesChange(SENSOR_VALUE);
    }
}
```

Código 2.12 Envío de recurso distancia desde Cliente LwM2M

Para la ejecución del cliente se utiliza el comando del Código 2.13 insertando como parámetros la dirección IP del servidor LwM2M y el período de lectura y envío de datos.

```
sudo java -jar rpi_leshan_client_valco.0.jar -u "Dirección Ip Servidor":5683 -
t "Período"
```

Código 2.13 Comando para ejecución de cliente LwM2M

2.3.2 Código cliente MQTT-SN

El cliente MQTT-SN se encuentra codificado en lenguaje Python 3, utilizando librerías para cliente que el bróker RSMB proporciona. [7]

En el Código 2.14 se puede observar la subscripción, donde se obtienen los *Topics ID* para el siguiente proceso de publicación.

```
for top in topics:
  rc, topic_id = client.subscribe(top,0)
  topics_id.append(topic_id)
```

Código 2.14 Obtención de tópicos

En el Código 2.15 se presenta el procedimiento de publicación, leyendo primero las mediciones de los sensores a partir del script Python.

```
def lectura():
    output = subprocess.check_output(['python', 'sensorsleshan.py'])
    sensor = (output.decode().strip()).split()
    topics_pub = topics_id

    for x in topics_pub:

        mensaje = sensor[x-topics_pub[0]]
        print("Pub",x, ": " + mensaje)
        client.publish(x,mensaje,qos=0)
```

Código 2.15 Asignación de lecturas a su respectivo tópico

Para la ejecución del cliente se ejecuta el comando del Código 2.16, donde se ingresa como parámetros: dirección del bróker RSMB, período de lectura y envío de datos y si los mensajes a enviar serán mediciones de sensores y de administración (1) o solo de administración (0).

```
python3 Rasduino.py "Período" "Dirección Ip Broker" "Envío(1) o no(0) de Información de sensores"
```

Código 2.16 Ejecución de cliente MQTT-SN

Las conexiones de dispositivos y sensores para ambos escenarios se presentan en la Figura 2.4, donde se ejecuta el protocolo de pruebas.

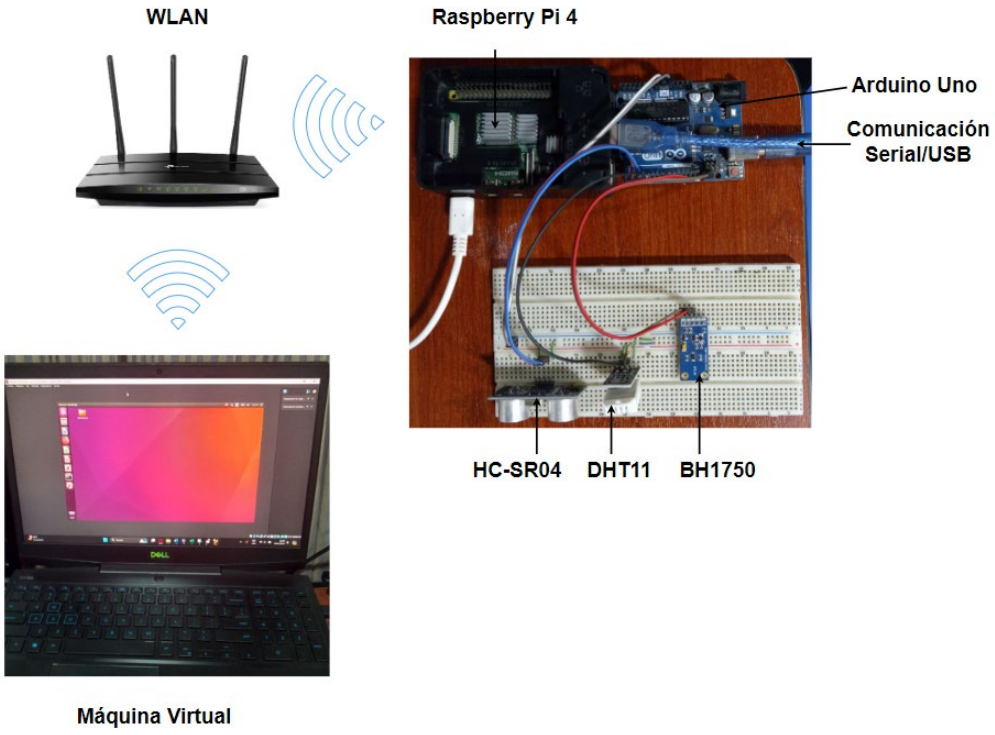

Figura 2.4 Montaje de Escenarios de prueba

2.4 Análisis de los protocolos a Nivel de Mensaje

Para entender el funcionamiento de los protocolos LwM2M y MQTT-SN en un ambiente de administración, es necesario conocer qué tramas se intercambian. Un análisis a nivel de trama permite identificar la forma en que los protocolos establecen, mantienen y terminan la comunicación y cómo intercambian información.

2.4.1 LwM2M

2.4.1.1 Formato de Mensaje LwM2M

El protocolo LwM2M maneja cuatro formatos para la representación de información: Texto plano, Opaco, TLV y JSON. Para la presente comparación se utiliza el formato TLV para la representación de información. La Tabla 2.1 muestra el formato de la Trama LwM2M y sus campos.

Tabla 2.1 Formato de Trama LwM2M, basado en [1]

	Tipo	**Identificador**	**Longitud**	**Valor**
# Bits	**8**	**8-16**	**0-24**	**Variable**

- **Tipo (1 Byte):** indica el tipo de recurso que se va a manejar y su longitud. A continuación, se puede observar los subcampos del campo Tipo(Tabla 2.2) y su función en la trama LwM2M:

Tabla 2.2 Campo Tipo de la Trama LwM2M, basado en [1]

	Tipo del campo Identificador	**Longitud del campo Identificador**	**Tipo de campo Longitud**	**Longitud del campo Valor**
Bit	7-6	5	4-3	2-0

  - Tipo del campo identificador (Bits 7-6): Se tienen cuatro tipos de identificadores, tomando cada uno un valor:

    00: Se tiene una instancia de objeto, en el que el campo valor puede contener uno más recursos.

    01: Cuando se tienen múltiples recursos, se utilizan instancias de recursos para identificar cada uno. Este identificador indica que se tiene una instancia de recurso con información.

    10: Se tiene un recurso múltiple, contiene varias instancias de recursos en el campo valor.

    11: Se tiene un recurso con información.

  - Longitud del campo Identificador (Bit 5): puede ser de dos tamaños, 0 para 8 bits, o 1 para 16 bits de longitud.

  - Tipo de campo Longitud (Bits 4-3): Se tienen cuatro tipos de longitud:

00: No se dispone del campo longitud, seguido del campo Identificador se encuentra el campo valor y su longitud en bytes se establece en los bits 2-0 de este campo.

01: campo Longitud de 8 bits y los bits 2-0 de este campo son ignorados.

10: campo Longitud de 16 bits y los bits 2-0 de este campo son ignorados.

11: campo Longitud de 24 bits y los bits 2-0 de este campo son ignorados

- Longitud del campo Valor(Bits 2-0): En caso de usarse indican la longitud del campo valor.

- **Identificador (1 o 2 Bytes):** Indica la instancia de objeto, el recurso o la instancia del recurso.
- **Longitud (0-3 Bytes):** Indica la longitud del campo Valor.
- **Valor:** Contiene el valor del identificador especificado.

En la Figura 2.5, se observa un ejemplo de bytes a decodificar, correspondientes al protocolo LwM2M.

e2 16 45 6c 78

Figura 2.5 Ejemplo - Bytes de Trama LwM2M

El primer octeto e2 (**1110 0010**), corresponde al campo Tipo y permite verificar si existe o no el campo longitud, como se presenta en la Tabla 2.3.

Tabla 2.3 Campo Tipo de Trama LwM2M

	Tipo del campo Identificador	**Longitud del campo Identificador**	**Tipo de campo Longitud**	**Longitud del campo Valor**
Bit	7-6	5	4-3	2-0
	11	1	00	010

- **Tipo del campo identificador (11):** Se tiene un recurso con información.
- **Longitud del campo Identificador (1):** El campo identificador tiene 16 bits de longitud.
- **Tipo de campo longitud (00):** No se dispone del campo longitud, la longitud del campo valor está definida por los bits 2-0 de este campo.
- **Longitud del campo valor (010):** La longitud del campo valor es de 2 bytes.

En la Tabla 2.4 se observa la información que está siendo transportada:

Tabla 2.4 Trama LwM2M

	Tipo	Identificador	Valor
# Oct	1	2	2
	0xe2	0x1645	0x6c78

Los dos siguientes octetos corresponde al Identificador, con un valor 0x1645 su valor decimal es 5701, el cual corresponde al recurso “Unidad del Sensor”.

Los dos últimos octetos 0x6c78 corresponden al recurso 5701, el cual al estar definido como tipo *string*, tiene un valor equivalente a las unidades de los datos que son luxes (lx).

Entre el cliente y servidor LwM2M se requiere el intercambio de tramas CoAP, ya que en este protocolo se encapsula LwM2M para el envío de información, por lo que es importante el análisis de los mensajes CoAP intercambiados. [1]

2.4.1.2 Formato de mensaje CoAP e Intercambio de Mensajes

El formato de un mensaje CoAP se presenta en la Tabla 2.5. el cual incluye:

Tabla 2.5 Formato de trama CoAP, basado en [8]

Ver	**T**	**TKL**	**Code**	**Message ID**
Token				
Options				
11111111			**Payload**	

- **Ver (2 bits):** Identifica la versión del protocolo CoAP.
- **T (2 bits):** Identifica el tipo del mensaje CoAP. La Tabla 2.6 presenta los tipos de mensajes.

Tabla 2.6 Tipo de mensajes CoAP, basado en [8]

T	**Type**
0	Confirmable
1	Non-Confirmable
2	Acknowledgement
3	Reset

- **TKL (4 bits):** Identifica la longitud del campo variable Token.

Code (8 bits): Indica el método HTTP del mensaje. Está dividido en dos partes: los 3 bits más significativos que indican la clase, y los 5 bits restantes para el detalle. Los tipos de clase se encuentran definidos en la Tabla 2.7, mientras que los métodos en la Tabla 2.8.

Tabla 2.7 Códigos para Tipos de Clase, basado en [8]

Código-Clase	Clase
0	Request
2	Success Response
4	Client Error Response
5	Server Error Response

Tabla 2.8 Códigos para Métodos HTTP, basado en [8]

Code	Int	Método
0.00	0	Empty Message
0.01	1	GET
0.02	2	POST
0.03	3	PUT
0.04	4	DELETE
2.01	65	2.01 Created
2.02	66	2.02 Deleted
2.04	68	2.04 Changed
2.05	69	2.05 Content

- **Message ID (16 bits):** Tiene como fin la detección de mensajes duplicados y emparejar mensajes tipo *Acknowledgement/Reset* con los tipos *Confirmable/Non-confirmable*. [8]
- **Token:** De acuerdo con el valor dado en el campo TKL, el Token tendrá una longitud de 0 a 8 bytes. Este valor es usado para relacionar solicitudes y respuestas. [8]
- **Options:** se pueden incluir varias opciones en el mensaje. Cada opción está definida por su número, longitud y valor, como se presenta en la Tabla 2.9; sin embargo, el número de opción no está definido directamente, para calcularlo se hace uso del valor opción delta.

Tabla 2.9 Formato para Opciones

Option Delta	Length
Option Value	

  - **Option Delta (4 bits):** con este valor se calcula el número de opción, para hacerlo se realiza la suma de los valores delta de las opciones anteriores más el de la opción actual. Para la primera opción, el número de ésta corresponderá al de la opción delta. [8]
  - **Length(4 bits):** Identifica la longitud en bytes del campo valor de la opción.
  - **Value:** Valor correspondiente a la opción.

Los números de opción están definidos en la Tabla 2.10:

Tabla 2.10 Números y Tipos de Opción, basado en [8]

# Opción	Opción
6	Observe
8	Location-Path
11	Uri-Path
12	Content-Format
15	Uri-Query
17	Accept

- *Observe:* Permite iniciar(0) o cancelar(1) la observación del recurso.
- *Location Path:* Indica la ubicación de un recurso como una ruta URI absoluta, al colocarse en una respuesta señala la ubicación de un recurso creado.
- *Uri-Path:* Indica la parte de la ruta absoluta de una URI. Permite especificar el recurso objetivo.
- *Content-Format:* Indica el formato en el que se encuentra el *payload* del mensaje.
- *Uri-Query:* Especifica un argumento indicando el recurso.
- *Accept:* Indica el tipo de formato de contenido que acepta el cliente.

- **Indicador de fin de opciones (8 bits):** O indicador del *payload*, toma el valor 11111111 dando a conocer el fin de las opciones y la continuación del *payload* del mensaje. Si este valor no está presente, indica un *payload* de longitud cero.
- **Payload:** Carga útil del mensaje.

2.4.1.2.1 Intercambio de mensajes en el Procedimiento de Registro

Para poder registrarse en el servidor LwM2M, el cliente lo hace a partir del mensaje CoAP POST como se observa en la Figura 2.6, el servidor confirma el registro con el mensaje 2.05 Created.

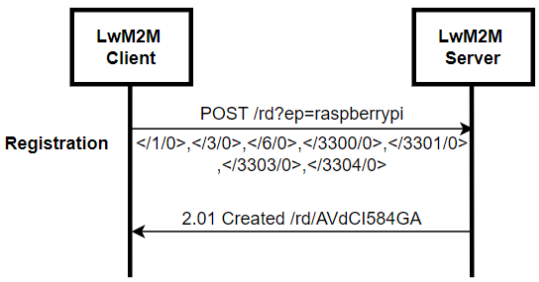

Figura 2.6 Procedimiento de Registro LwM2M, basado en [1]

La Tabla 2.11 presenta el mensaje POST que requiere confirmación, lleva como opciones la URI del Servidor, el formato del *payload*, el enlace de transporte que se va a usar con UDP (U) por defecto, la versión del protocolo LwM2M, el tiempo de vida (lt) y el nombre del nodo o cliente (ep) [8]. Para el *payload* incluye el tipo de recurso, siendo LwM2M el que se va a usar, los objetos y recursos del cliente.

Tabla 2.11 Mensaje CON POST, Solicitud de Registro de Cliente, basado en [8]

Ver	**T**	**TKL**	**Code**	**Message ID**
1	0	8	2	54439
Token: 18f228f24818bd28				
Opt Name: #1: Uri-Path: rd				
Opt Name: #2: Content-Format: application/link-format				
Opt Name: #3: Uri-Query: b=U				
Opt Name: #4: Uri-Query: lwm2m=1.0				
Opt Name: #5: Uri-Query: lt=1800				
Opt Name: #6: Uri-Query: ep=raspberrypi				
11111111	**rt=**"oma.lwm2m", </ 1/0>,</ 3/0>,</ 3300/0>, </ 3301/0>,</ 3303/0>,</ 3304/0>			

Los objetos están definidos por la ruta /{ID del Objeto}/{ID de la instancia del Objeto} y los recursos por la ruta /{ID del Objeto}/{ID de la instancia del Objeto}/{ID del recurso}. [1]

En respuesta del servidor hacia el cliente, el mensaje 2.01 Created incluye el identificador del registro del cliente para operaciones de actualización y eliminación. Este mensaje(Tabla 2.12) indica que el cliente se ha registrado correctamente en el servidor LwM2M.

Tabla 2.12 Mensaje ACK 2.01 Created, Registro Correcto, basado en [8]

Ver	**T**	**TKL**	**Code**	**Message ID**
1	2	8	65	54439
Token: 18f228f24818bd28				
Opt Name: #1: Location-Path: rd				
Opt Name: #2: Location-Path: 1U71Ri0hxc				

2.4.1.2.2 Intercambio de mensajes en el procedimiento de Actualización

En la Figura 2.7, con el fin de refrescar su tiempo de vida de registro, se observa que el cliente lo realiza a partir del envío del mensaje CoAP POST hacia el servidor LwM2M, este mensaje es respondido con el método 2.04 Changed.

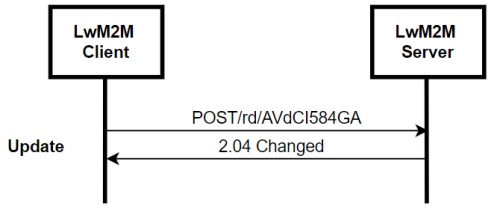

Figura 2.7 Procedimiento de Actualización LwM2M, basado en [1]

El mensaje POST mostrado en la Tabla 2.13 no incluye parámetros más que el identificador de registro, indica que se requiere refrescar el tiempo de vida.

Tabla 2.13 Mensaje CON POST, Solicitud de refresco de tiempo de vida, basado en [8]

Ver	**T**	**TKL**	**Code**	**Message ID**
1	0	8	2	54440
Token: 1cba5853cbafba18				
Opt Name: #1: Location-Path: rd				
Opt Name: #2: Location-Path: 1U71Ri0hxc				

El mensaje 2.04 Changed del servidor (Tabla 2.14) indica que se ha refrescado el tiempo de vida correctamente.

Tabla 2.14 Mensaje ACK 2.04 Changed, Tiempo de Vida reiniciado, basado en [8]

Ver	**T**	**TKL**	**Code**	**Message ID**
1	2	8	68	54440
Token: 1cba5853cbafba18				

2.4.1.2.3 Intercambio de mensajes en el procedimiento de Finalizar registro

El cliente, para eliminar su registro del servidor, utiliza el método CoAP DELETE y es confirmado por el servidor con el método 2.02 Deleted (Figura 2.8).

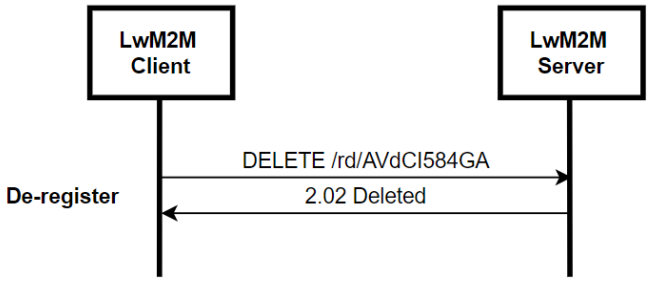

Figura 2.8 Procedimiento de Finalización de Registro LwM2M, basado en [1]

La solicitud de finalización de registro se da mediante el mensaje DELETE como se presenta en la Tabla 2.15. Lleva como parámetros el identificador de registro que recibió el cliente al momento de registrarse.

Tabla 2.15 Mensaje CON DELETE, Solicitud para Finalizar Registro, basado en [8]

Ver	T	TKL	Code	Message ID
1	0	8	4	54759
Token: ef2dead5ae0687a1				
Opt Name: #1: Uri-Path: rd				
Opt Name: #2: Uri-Path: 1U71Ri0hxc				

El servidor retorna el mensaje DELETE que se presenta la Tabla 2.16, confirmando que se eliminó el registro exitosamente.

Tabla 2.16 Mensaje ACK 2.02 DELETED, Eliminación correcta de Registro, basado en [8]

Ver	T	TKL	Code	Message ID
1	2	8	66	54759
Token: ef2dead5ae0687a1				

2.4.1.2.4 Intercambio de mensajes en el procedimiento de Lectura de recursos

Para leer un recurso o instancia, el servidor LwM2M lo solicita a partir del método CoAP GET y la respuesta es enviada por el cliente en el método 2.05 Content, como se observa en la Figura 2.9.

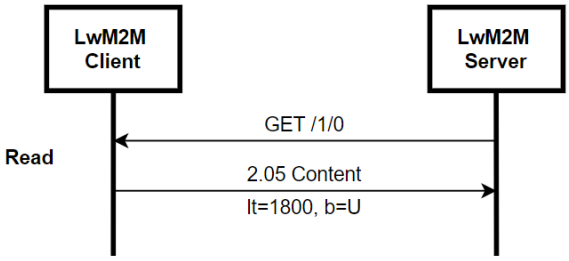

Figura 2.9 Procedimiento de Lectura LwM2M, basado en [1]

En la Tabla 2.17 se presenta el mensaje CoAP GET, que incluye como opciones la ruta del recurso y el formato en que se desea recibir la información, en este caso LwM2M.

Tabla 2.17 Mensaje CON GET, Solicitud de lectura de recurso, basado en [8]

Ver	T	TKL	Code	Message ID
1	0	8	1	40759
Token: c079d875c559c969				
Opt Name: #1: Uri-Path: 1				
Opt Name: #2: Uri-Path: 0				
Opt Name: #3: Accept: application/vnd.oma.lwm2m+tlv				

En la respuesta CoAP 2.05 Content que se observa en la Tabla 2.18 se incluye el *payload* en el formato especificado en la solicitud.

Tabla 2.18 Mensaje ACK 2.05 Content, retorno de información de recurso, basado en [8]

Ver	T	TKL	Code	Message ID
1	2	8	69	40759
Token: c079d875c559c969				
Opt Name: #1: Content-Format: application/vnd.oma.lwm2m+tlv				
11111111	Lightweight M2M TLV			

2.4.1.2.5 Intercambio de mensajes en el procedimiento de Observación

Como se puede ver en la Figura 2.10, la observación de un recurso o instancia de objeto es solicitada por el servidor LwM2M al cliente, con el fin de recibir notificaciones periódicas cada que se den cambios en el recurso. El servidor a partir del método GET indica al cliente qué recurso o instancia quiere observar, y se responde con el método 2.05 Content, el cual incluye la información solicitada.

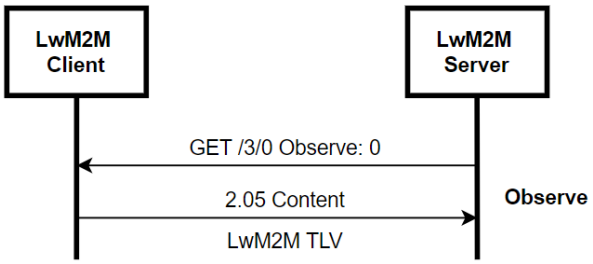

Figura 2.10 Procedimiento de Observación LwM2M, basado en [1]

El mensaje GET presentado en la Tabla 2.19 incluye como opciones la ruta del recurso, el formato en que se desea recibir la información y la opción Observe con valor cero, indicando que el servidor requiere recibir notificaciones periódicas del recurso o instancia de objeto cuando presente cambios.

El *Token* de este mensaje, será el mismo para las notificaciones durante la observación, permitiendo relacionarlas con el mensaje GET que solicitó la observación.

Tabla 2.19 Mensaje CON GET, Solicitud para Observación de Recurso, basado en [8]

Ver	T	TKL	Code	Message ID
1	0	8	1	40760
Token: a70a725eb1e25c92				
Opt Name: #1: Observe: 0				
Opt Name: #2: Uri-Path: 3				
Opt Name: #3: Uri-Path: 0				
Opt Name: #4: Accept: application/vnd.oma.lwm2m+tlv				

La respuesta 2.5 Content que se presenta en la Tabla 2.20, confirma el inicio de la observación incluyéndose el *payload* en el formato especificado en la solicitud. Además,

se tiene la opción Observe con un valor incrementable, con el objetivo de mantener un orden entre los mensajes periódicos que serán enviados en el proceso de Notificación.

Tabla 2.20 Mensaje ACK 2.05 Content, Inicio de Observación de Recurso, basado en [8]

Ver	**T**	**TKL**	**Code**	**Message ID**
1	2	8	69	40760
Token: a70a725eb1e25c92				
Opt Name: #1: Observe: 3				
Opt Name: #2: Content-Format: application/vnd.oma.lwm2m+tlv				
11111111	Lightweight M2M TLV			

2.4.1.2.6 Intercambio de mensajes en el procedimiento de Notificación

De acuerdo con la configuración del cliente, este envía un mensaje tipo 2.05 Content (Tabla 2.21) periódicamente con la información del recurso hacia el servidor que solicitó la observación de acuerdo con un período determinado.

Tabla 2.21 Mensaje NON 2.05 Content, Notificación de Recurso, basado en [8]

Ver	**T**	**TKL**	**Code**	**Message ID**
1	**1**	**8**	**69**	**54448**
Token: a70a725eb1e25c92				
Opt Name: #1: Observe: 8				
Opt Name: #2: Content-Format: application/vnd.oma.lwm2m+tlv				
11111111	**Lightweight M2M TLV**			

2.4.1.2.7 Intercambio de mensajes en el procedimiento de Cancelar Observación

Cuando ya no se requiere la observación de un recurso o la instancia de objeto, el servidor envía un mensaje de tipo Reset, que se presenta en la Figura 2.11, el cual no lleva parámetros más que el Message ID correspondiente al del mensaje Notify del recurso o instancia de objeto que ya no se quiere escuchar. El cliente al recibir este mensaje anulará la observación del recurso u objeto sin necesidad de enviar una confirmación.

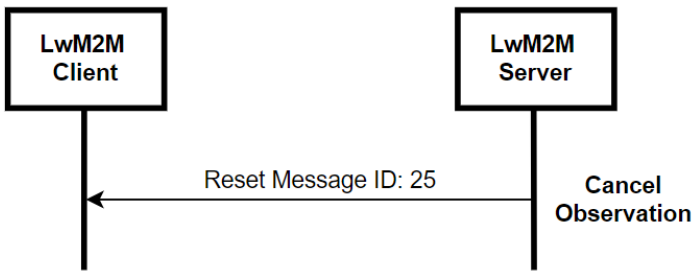

Figura 2.11 Procedimiento de Cancelación de Observación LwM2M, basado en [1]

El mensaje Reset es enviado por el servidor para cancelar la observación(Tabla 2.22).

Tabla 2.22 Mensaje Reset, Anulación de Observación de recurso, basado en [8]

Ver	T	TKL	Code	Message ID
1	3	0	0	54754

2.4.2 MQTT-SN

Entre cliente y bróker MQTT-SN las tramas, que poseen una estructura básica, permiten establecer, mantener y terminar la conexión; además de la suscripción y publicación de tópicos.

La trama MQTT-SN está compuesta por dos partes como se presenta en la Tabla 2.23: cabecera variable y una parte opcional de tamaño variable que estará presente o no de acuerdo con el tipo de mensaje.[2]

Tabla 2.23 Formato de Mensaje MQTT-SN

	Cabecera	Parte Variable
Byte	2 o 4	N

La cabecera incluye los campos presentes en la Tabla 2.24**:**

Tabla 2.24 Campo Cabecera de Mensaje MQTT-SN

	Longitud	Tipo de Mensaje
Byte	1 o 3	1

- **Longitud (1 o 3 Bytes):** Indica el número total de octetos contenidos en el mensaje, incluyéndose los octetos de este campo.

- **Tipo de Mensaje (1 Byte):** Especifica el tipo del mensaje, puede tomar uno de los valores especificados en la Tabla 2.25. [2]

Tabla 2.25 Tipos de Mensaje MQTT-SN, basado en [2]

Tipo	Valor	Tipo	Valor
0x04	CONNECT	0x05	CONNACK
0x0C	PUBLISH	0x18	DISCONNECT
0x12	SUBSCRIBE	0x13	SUBACK
0x14	UNSUBSCRIBE	0x15	UNSUBACK
0x16	PINGREQ	0x17	PINGRESP

La parte variable depende del tipo de mensaje, se encuentra conformada por los siguientes campos.

- **ClientId:** Campo variable de 1 a 23 caracteres utilizados para identificar el cliente.
- **Data:** Corresponde al *payload* del mensaje PUBLISH. Es de longitud variable y contiene los datos que serán publicados.
- **Duration (2 Bytes):** Especifica la duración en segundos de un período de tiempo.
- **Flags (1 Byte):** Está conformado por las siguientes banderas según la Tabla 2.26:

Tabla 2.26 Campo Banderas de Mensaje MQTT-SN, basado en [2]

	DUP	QoS	Retain	Will	CleanSesion	TopicIdType
Bit	7	6-5	4	3	2	1-0

  - *DUP:* Indica si el mensaje es enviado por primera vez o no, es utilizado en mensajes PUBLISH. [2]
  - *QoS:* nivel de QoS en mensajes PUBLISH enviados por un cliente; “0x00” para QoS 0, “0x01” para QoS 1, “0x10” para QoS 2 y “0x11” para QoS -1.
  - *Retain:* Si un mensaje es publicado con retención, un cliente nuevo al suscribirse al tópico recibirá la información retenida en un mensaje PUBLISH. [2]
  - *Will:* Se utiliza solo en el mensaje CONNECT, si el cliente lo coloca en 1, este solicitará el tópico y el mensaje *Will* o de voluntad. Si un cliente se desconecta sin avisar al Bróker, este enviará el mensaje *Will* a los suscritos al Tópico *Will*. [2]
  - *CleanSession:* Establece si se requiere (“1”), o no (“0”) una sesión limpia, es decir, en una sesión limpia el bróker eliminará todas las suscripciones del cliente. [2]
  - *TopicIdType:* contiene el campo *Topic Id* o *Topic Name*, pudiendo ser Id de Tópico(“0x00”), Tópico Predefinido(“0x01”) o Tópico de nombre corto(“0x10”).
- **MsgId (2 Bytes):** Permite emparejar mensajes con su respectiva respuesta.
- **ProtocolID (1 Byte):** Indica el nombre del protocolo y su versión en el mensaje CONNECT.

- **ReturnCode (1 Byte):** Corresponde a los códigos de respuesta, se indica su valor y su respectivo mensaje en la Tabla 2.27.

Tabla 2.27 Tipos de mensaje de respuesta, basado en [2]

Código	Mensaje	
0x00	Aceptado	
0x01	Congestión	Rechazado
0x02	Topic ID Inválido	
0x03	No soportado	

- **TopicId (2 Byte):** Contiene el valor del *Topic Id*.
- **TopicName:** De longitud variable, contiene un *string* que indica el nombre del tópico.

2.4.2.1 Intercambio de mensajes en el procedimiento de Conexión

Para que el cliente MQTT-SN pueda establecer una conexión con el Broker, se emplea la trama CONNECT que es confirmada con la trama CONNACK, como se muestra en la Figura 2.12.

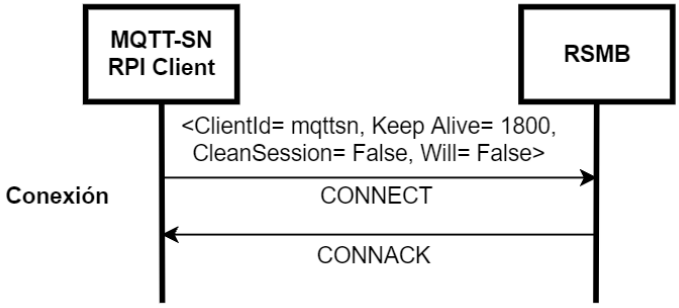

Figura 2.12 Procedimiento de Conexión MQTT-SN

A partir del mensaje MQTT-SN CONNECT enviado por el cliente se establece la conexión con el Broker, como se muestra en la Tabla 2.28, su trama indica el nombre y la versión del protocolo, siendo la versión v 1.0 para MQTT-SN. El campo *Duration* especifica el tiempo de vida o período en el cual se enviará un PINGREQ al servidor. Además, lleva el *ClientId* que corresponde al nombre del cliente. [2]

Tabla 2.28 Mensaje CON CONNECT, Solicitud de Conexión, basado en [2]

	Length	MsgType	Flags	ProtocolId	Duration	ClientId
Byte	0	1	2	3	4-5	(6:n)
Valor (Hex)	0x0d	0x04	0x00	0x01	0x0708	0x6d717474736e (Hex) mqttsn (string)

En el campo Flags que se indica en la Tabla 2.29 se utilizan solamente los bits Will y *CleanSession*.

Tabla 2.29 Campo Flags para Mensaje Connect

	DUP	QoS	Retain	Will	CleanSesion	TopicIdType
Bit	0x0	0x00	0x0	0x0	0x0	0x00

El mensaje MQTT-SN CONNACK es enviado desde el Broker al cliente como confirmación de registro (Tabla 2.30).

Tabla 2.30 Mensaje ACK CONNACK, Registro Correcto, basado en [2]

	Length	MsgType	ReturnCode
Byte	0	1	2
Valor (Hex)	0x03	0x05	0x00

2.4.2.2 Intercambio de mensajes en el procedimiento de Desconexión

Cuando un cliente requiera cerrar la conexión envía un mensaje DISCONNECT al Bróker como se observa en la Figura 2.13.

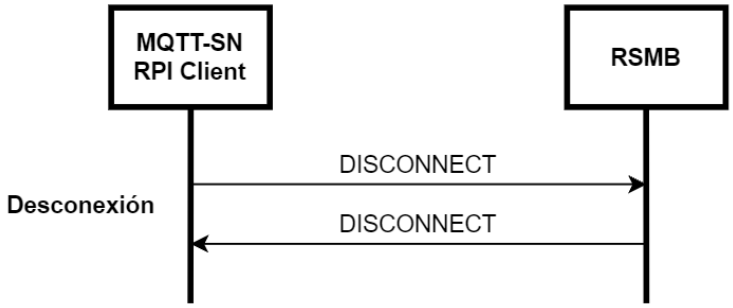

Figura 2.13 Procedimiento de Desconexión MQTT-SN

En la trama Disconnect que se muestra en la Tabla 2.31, si el cliente requiere entrar al estado dormido, llevará el campo opcional *Duration*. [2] El bróker responde este mensaje a partir de otro DISCONNECT.

Tabla 2.31 Mensaje DISCONNECT, Solicitud y Respuesta, basado en [2]

	Length	MsgType	Duration (Opcional)
Byte	0	1	2-3
Valor (Hex)	0x02	0x18	

2.4.2.3 *Intercambio de mensajes en el procedimiento de Ping*

Para poder mantener la conexión el cliente envía un mensaje PINGREQ al bróker para verificar que este activa la conexión, recibiendo un PINGRESP como respuesta como se presenta en la Figura 2.14.

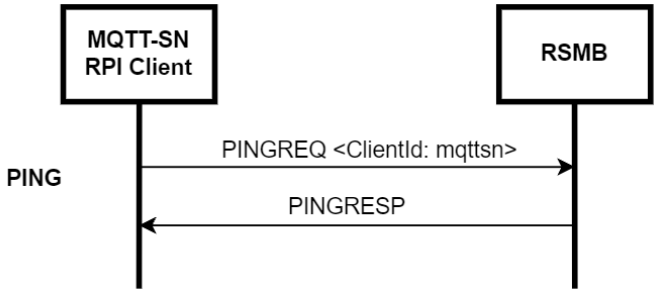

Figura 2.14 Procedimiento de Actualización MQTT-SN

Mediante el mensaje PINGREQ el cliente verifica si el bróker está activo. Incluye el ID del cliente como campo opcional (Tabla 2.32). [2]

Tabla 2.32 Mensaje CON PINGREQ, verificar si el Bróker esta activo, basado en [2]

	Length	MsgType	ClientId
Byte	0	1	2:n
Valor (Hex)	0x08	0x16	0x6d717474736e

Como respuesta al mensaje PINGREQ, el Bróker envía el mensaje PINGRESP que se muestra en la Tabla 2.33.

Tabla 2.33 Mensaje ACK PINGRESP, Confirmación de Broker activo, basado en [2]

	Length	MsgType
Byte	0	1
Valor (Hex)	0x02	0x17

2.4.2.4 *Intercambio de mensajes en el procedimiento de Publicación*

Con el objetivo de publicar información ya sea desde el cliente o el bróker(Figura 2.15), se emplea el mensaje PUBLISH.

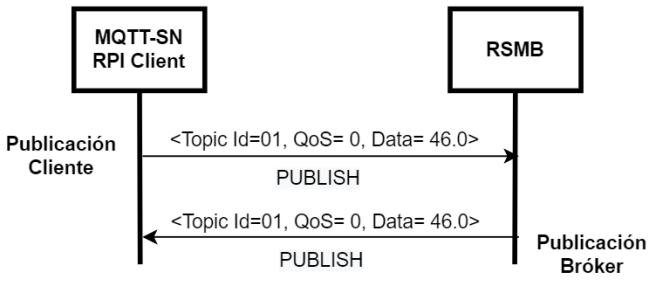

Figura 2.15 Procedimiento de Publicación MQTT-SN

El mensaje PUBLISH es utilizado por clientes y Bróker para publicar información para un tópico como se presenta en la Tabla 2.34.

Tabla 2.34 Mensaje PUBLISH, publicación a tópicos, basado en [2]

	Length	MsgType	Flags	TopicId	MsgId	Data
Byte	0	1	2	3-4	5-6	(7:n)
Valor (Hex)	0x08	0x0c	0x00	0x0007	0x0000	0x33 (Hex) 3(Char)

En el campo Flags que se presenta en la Tabla 2.35, el bit DUP indica si el mensaje es enviado por primera vez o no, QoS el nivel para el mensaje, utilizado en casos de nivel QoS 1 y 2, en QoS 0 toma un valor igual a 0x00, *Retain* define si el mensaje será o no retenido y *TopicIDType* indica el tipo de tópico contenido en el campo *TopicId*. [2]

Tabla 2.35 Campo Flags de Mensaje Publish

	DUP	QoS	Retain	Will	CleanSesion	TopicIdType
Bit	0x0	0x00	0x0	0x0	0x0	0x00

2.4.2.5 Intercambio de mensajes en el procedimiento de Suscripción

Para que un cliente pueda escuchar la información publicada en un tópico específico, es necesario que este se suscriba al tópico mediante el uso del mensaje SUBSCRIBE, recibiendo como confirmación el mensaje SUBACK (Figura 2.16).

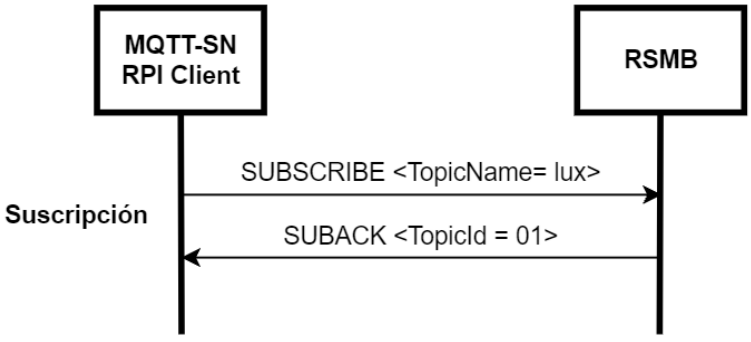

Figura 2.16 Procedimiento de Suscripción MQTT-SN

El mensaje SUBSCRIBE es usado por el cliente para suscribirse a un tópico, el formato del mensaje se presenta en la Tabla 2.36. La información con la que se suscribirá ya sea el nombre del tópico, Id predefinido del tópico o un nombre corto dependerá de los bits *TopicIdType* del campo *Flags*. *MsgId* permite asociar el mensaje SUBSCRIBE con su respectivo SUBACK. [2]

Tabla 2.36 Mensaje CON SUSCRIBE, solicitud para suscripción, basado en [2]

	Length	MsgType	Flags	MsgId	TopicName
Byte	0	1	2	3-4	5:n
Valor (Hex)	0x08	0x12	0x00	0x0001	0x616359 (Hex) acY (string)

En el campo *Flags* del mensaje Suscribe (Tabla 2.37) se utilizan los bits DUP, QoS y *TopicIdType*. [2]

Tabla 2.37 Campo Flags de Mensaje Suscribe

	DUP	QoS	Retain	Will	CleanSesion	TopicIdType
Bit	0x0	0x00	0x0	0x0	0x0	0x00

Como confirmación a una suscripción exitosa, el Bróker MQTT-SN responde con el mensaje SUBACK (Tabla 2.38), adjuntando el *TopicId* que podrá ser usado por el cliente y el servidor para publicar mensajes, este campo no es relevante en el caso de suscripciones a tópicos de nombre corto. *MsgId* permite asociar el mensaje SUBACK con su respectivo SUBSCRIBE. [2]

Tabla 2.38 Mensaje ACK SUBACK, Suscripción exitosa, basado en [2]

	Length	MsgType	Flags	TopicId	MsgId	ReturnCode
Byte	0	1	2	3-4	5-6	7
Valor (Hex)	0x08	0x13	0x00	0x0004	0x0001	0x00

En el campo *Flags* (Tabla 2.39) únicamente se utiliza los bits QoS para indicar su nivel de calidad de servicio.

Tabla 2.39 Campo Flags de Mensaje Suback

	DUP	QoS	Retain	Will	CleanSesion	TopicIdType
Bit	0x0	0x00	0x0	0x0	0x0	0x00

2.4.2.6 Intercambio de mensajes en el procedimiento de Cancelar suscripción

Cuando el cliente ya no requiere escuchar los mensajes publicados a un tópico, cancela la suscripción a ese tópico mediante el mensaje UNSUBSCRIBE, como se muestra en la Figura 2.17, confirmándolo el servidor con el mensaje UNSUBACK.

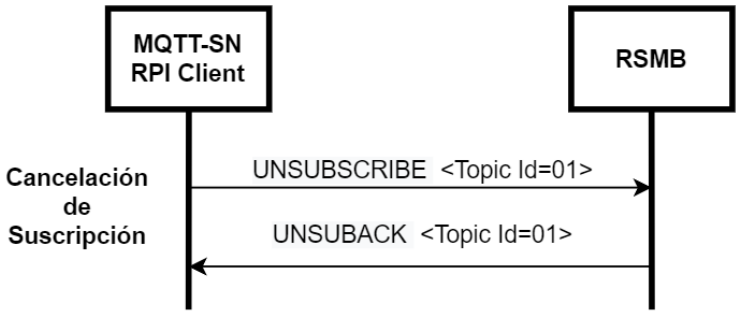

Figura 2.17 Procedimiento de De-Suscripción MQTT-SN

A continuación, en la Tabla 2.40 se muestra el formato del mensaje UNSUBSCRIBE para solicitar la anulación de la suscripción al tópico.

Tabla 2.40 Mensaje CON Unsubscribe, solicitud para anular suscripción, basado en [2]

	Length	MsgType	Flags	MsgId	TopicId
Byte	0	1	2	3-4	5:n
Valor (Hex)	0x08	0x14	0x00	0x0001	0x616359 (Hex) acY (string)

En el campo *Flags* únicamente se utiliza los bits *TopicIdType* indicando el tipo del tópico del cual se va a cancelar la suscripción (Tabla 2.41). [2]

Tabla 2.41 Campo Flags de Mensaje Unsubscribe

	DUP	QoS	Retain	Will	CleanSesion	TopicIdType
Bit	0x0	0x00	0x0	0x0	0x0	0x00

Como respuesta a una cancelación de suscripción exitosa, el bróker lo confirma con el mensaje UNSUBACK(Tabla 2.42). El campo *MsgId* identifica este mensaje con su respectivo UNSUBSCRIBE.

Tabla 2.42 Mensaje ACK UNSUBACK, Suscripción cancelada, basado en [2]

	Length	MsgType	MsgId
Byte	0	1	2-3
Valor (Hex)	0x04	0x15	0x0001

2.5 Protocolo de Pruebas

Para comparar los protocolos LwM2M y MQTT-SN en un ambiente de administración, se establece el protocolo de pruebas al cual estarán sometidos los escenarios. El tráfico se divide en dos tipos: Administración e Información.

Se ha determinado como tráfico de administración al encargado de establecer, mantener, finalizar la conexión, mensajes para observación/suscripción y cancelación, mensajes para mantener la conexión y datos acerca del dispositivo cliente.

En la Tabla 2.43, se establece el tráfico de administración correspondiente al cliente LwM2M.

Tabla 2.43 Tráfico de administración – LwM2M

Objeto	# Objeto	# Recurso	Recurso	Tipo
LwM2M Server	/1	/1/0/0	Short Server ID	Integer
		/1/0/1	Tiempo de vida	Integer
		/1/0/6	Almacenamiento de notificaciones si el dispositivo está offline	Boolean
		/1/0/7	Modo de enlace usado	String
Device	/3	/3/0/0	Fabricante	String
		/3/0/1	Número de Modelo	String
		/3/0/2	Número Serial	String
		/3/0/3	Versión de Firmware	String
		/3/0/9	Nivel de Batería	Integer
		/3/0/10	Memoria Disponible	Integer
		/3/0/11	Código de Error	Integer
		/3/0/13	Hora Actual	Time
		/3/0/14	Desface UTC	String
		/3/0/15	Zona Horaria	String
		/3/0/16	Modo de enlace soportado	String
		/3/0/17	Tipo de Dispositivo	String
		/3/0/18	Versión de Hardware	String
		/3/0/19	Versión de Software	String
		/3/0/20	Estado de Batería	Integer
		/3/0/21	Memoria Total	Integer
Generic Sensor (Proximidad)	/3300	/3300/0/5601	Valor mínimo medido	Float
		/3300/0/5602	Valor máximo medido	Float
		/3300/0/5603	Valor mínimo de rango	Float
		/3300/0/5604	Valor máximo de rango	Float
		/3300/0/5701	Unidad del Sensor	String
		/3301/0/5751	Tipo de Sensor	String
Illuminance	/3301	/3301/0/5601	Valor mínimo medido	Float
		/3301/0/5602	Valor máximo medido	Float
		/3301/0/5603	Valor mínimo de rango	Float
		/3301/0/5604	Valor máximo de rango	Float
		/3301/0/5701	Unidad del Sensor	String
Temperature	/3303	/3303/0/5601	Valor mínimo medido	Float
		/3303/0/5602	Valor máximo medido	Float
		/3303/0/5603	Valor mínimo de rango	Float
		/3303/0/5604	Valor máximo de rango	Float
		/3303/0/5701	Unidad del Sensor	String
Humidity	/3304	/3304/0/5601	Valor mínimo medido	Float
		/3304/0/5602	Valor máximo medido	Float
		/3304/0/5603	Valor mínimo de rango	Float
		/3304/0/5604	Valor máximo de rango	Float
		/3304/0/5701	Unidad del Sensor	String

En la Tabla 2.44, se establece la información de administración del dispositivo Raspberry Pi 4 donde se ejecuta el cliente MQTT-SN.

Tabla 2.44 Tráfico de administración – MQTT-SN

Tópico	Descripción
Sistema	Sistema Operativo
Hostname	Usuario del dispositivo
RpiVersion	Versión Raspberry Pi
Procesador	Arquitectura del procesador
Fecha	Fecha y Hora Actual
Ram	Memoria RAM
Memdisk	Capacidad de almacenamiento
Memfree	Almacenamiento disponible
UMem	Unidad utilizada para memorias
UDist	Unidad para medida de proximidad
UHum	Unidad para medida de humedad
UTemp	Unidad para medida de temperatura
ULuz	Unidad para medida de iluminación

El valor retornado por cada sensor forma parte del tráfico de información, ésta puede verse en la Tabla 2.45 para el caso de LwM2M y en la Tabla 2.46 para MQTT-SN.

Tabla 2.45 Tráfico de Información – LwM2M

Objeto	# Objeto	# Recurso	Recurso	Tipo
Generic Sensor (Proximidad)	/3300	/3300/0/5700	Valor del Sensor	Float
Illuminance	/3301	/3301/0/5700	Valor del Sensor	Float
Temperature	/3303	/3303/0/5700	Valor del Sensor	Float
Humidity	/3304	/3304/0/5700	Valor del Sensor	Float

Tabla 2.46 Tráfico de Información – MQTT-SN

Tópico	Descripción
Dist	Valor Sensor de Proximidad
Hum	Valor Sensor de Humedad
Temp	Valor Sensor de Temperatura
Luz	Valor Sensor de Iluminación

Para el análisis se utiliza la herramienta para captura de tráfico Wireshark, de código libre y gratuito basado en librerías *PCAP*. Para la captura de tráfico se puede aplicar filtros a los mensajes capturados, teniendo una mejor vista del tráfico objetivo. [9]

Una vez establecido el tráfico a capturar se establece el protocolo de pruebas al cual estará sometido:

1. En un ambiente con envío de Información se gráfica y analiza el número de tramas de información, administración, mensajes perdidos y eficiencia de acuerdo con capturas de tráfico, en los cuales el envío de información se da en los períodos 1, 5, 10, 15, 20 y 30 min durante 24 horas para cada protocolo.

2. En un ambiente sin envío de información de sensores se gráfica y analiza el número de tramas de administración y mensajes perdidos de acuerdo con capturas de tráfico, en los cuales el envío de tráfico solo de administración se da en los períodos 1, 5, 10 y 15 min durante 12 horas para cada protocolo.

3. Análisis de retardo de respuesta de mensajes que requieren confirmación para cada protocolo.

   El tiempo de respuesta en que el servidor responde al cliente se calcula a partir de la diferencia de tiempos entre el mensaje CON y su respectivo ACK, tiempos proporcionados por la herramienta Wireshark. El resultado obtenido permite identificar qué protocolo presenta menor retardo de respuesta.

   Para LwM2M, los mensajes que se consideran para el cálculo del retardo se observan en la Tabla 2.47, mientras que para MQTT-SN se los presenta en la Tabla 2.48.

Tabla 2.47 Mensajes que requieren confirmación – LwM2M

	CON	**ACK**
Registro de Cliente	POST	2.01 Created
Observación	GET	2.05 Content
Keep Alive	POST	2.04 Changed
Cancel-Observación	2.05 Content	Empty Message
Desconexión	DELETE	2.02 Deleted

Tabla 2.48 Mensajes que requieren confirmación – MQTT-SN

	CON	**ACK**
Registro de Cliente	CONNECT	CONNACK
Suscripción	SUBSCRIBE	SUBACK
Keep Alive	PINGREQ	PINGRESP
Unsubscribe	UNSUBSCRIBE	UNSUBACK
Desconexión	DISCONNECT	DISCONNECT

4. Comportamiento en un ambiente de Video Streaming para cada protocolo, se consumirá el servicio de una misma película para cada escenario, con el fin de

observar la influencia de estos protocolos de administración IoT sobre el servicio descrito.

5. Comportamiento en un ambiente de Descarga de Archivo de tamaño 900 MB para cada protocolo de administración IoT, con esto se observa el comportamiento del servicio de descarga de archivo de acuerdo con el escenario donde se ejecuta.

En estos dos últimos protocolos de prueba para cada escenario se captura, analiza y grafica el tráfico de administración e información, junto a mensajes perdidos, retardo y eficiencia.

2.6 Filtros de tráfico Wireshark

De manera que se pueda analizar el tráfico para cada protocolo, se aplican filtros que permitan separar el tráfico objetivo de todo el tráfico de red.

En el Código 2.17 se encuentra el filtro Wireshark utilizado para capturar las mediciones proporcionadas por los sensores para el cliente LwM2M del dispositivo Raspberry Pi 4.

```
((((coap.opt.uri_path_recon=="/3301/0/5700")||(coap.opt.uri_path_recon==
"/3300/0/5700"))||(coap.opt.uri_path_recon=="/3304/0/5700"))||
(coap.opt.uri_path_recon=="/3303/0/5700"))&&(coap.code==69)
```

Código 2.17 Filtro de tráfico – Información cliente LwM2M

La información de administración para LwM2M es capturada a partir del filtro mostrado en el Código 2.18.

```
coap && !(((((coap.opt.uri_path_recon=="/3301/0/5700")||
(coap.opt.uri_path_recon=="/3300/0/5700"))||(coap.opt.uri_path_recon=="/3304/
0/5700"))||(coap.opt.uri_path_recon=="/3303/0/5700"))&&(coap.code==69))
```

Código 2.18 Filtro de tráfico – Administación cliente LwM2M

En el Código 2.19 se encuentra el filtro Wireshark utilizado para capturar las mediciones proporcionadas por los sensores para el cliente MQTT-SN del dispositivo Raspberry Pi 4.

```
(udp.port eq 1885)&&(data.data[1]==0C)&&(data.data[4]==01 ||data.data[4]==02
||data.data[4]==03||data.data[4]==04)
```

Código 2.19 Filtro de tráfico – Información cliente MQTT-SN

La información de administración para MQTT-SN es capturada a partir del filtro mostrado en el Código 2.20.

```
(udp.port eq 1885)&&!((data.data[1]==0C)&&(data.data[4]==01 ||data.data[4]==02
||data.data[4]==03||data.data[4]==04))
```

Código 2.20 Filtro de tráfico – Administración cliente MQTT-SN

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Una vez realizadas las pruebas y capturas de tráfico para cada escenario, en el presente capítulo se presentan las gráficas, tablas y conclusiones respectivas.

3.1 Cabecera Insertada

De acuerdo con el análisis de estructura de la trama para cada protocolo, CoAP junto a LwM2M son mucho más estructurados que MQTT-SN, permitiendo a CoAP adjuntar opciones que indican la URI para acceder a un objeto, a uno o varios recursos, el formato usado para el *payload* es LwM2M, el cual maneja varios formatos de mensaje como *int, float o string*.

MQTT-SN maneja una estructura simplificada, en la cual no se puede especificar el tipo de dato a usar, manejándose en general el formato de datos tipo *String*.

Si se toma en cuenta que LwM2M trabaja junto a CoAP, MQTT-SN agrega una cabecera de menor tamaño y de longitud fija en comparación a la agregada por CoAP y LwM2M que es de longitud variable de acuerdo con el mensaje y al objeto o número de recursos observados.

En la Tabla 3.1 se puede observar la cabecera introducida por cada protocolo al momento de la observación/publicación a un recurso/tópico.

Tabla 3.1. Cabecera Introducida por cada protocolo

Cabecera Insertada	Tamaño (Bytes)
CoAP - LwM2M	22
MQTT-SN	7

Al ser más estructurados los protocolos CoAP y LwM2M, su cabecera permite una mejor administración al abarcar más información y su identificación en un solo mensaje. Mientras que MQTT-SN al usar una cabecera más liviana incrementa su eficiencia frente a LwM2M al momento de llevar información.

Como ventaja frente a MQTT-SN, LwM2M al permitir observar varios recursos en un solo mensaje, permite reducir el número de tramas transmitidas a una sola, como fue el caso de los detalles del dispositivo (Tabla 3.2).

Tabla 3.2. Cabecera Introducida – Características Dispositivo

Cabecera Insertada	Tamaño (Bytes)
CoAP – LwM2M	54
MQTT-SN	7

En la Tabla 3.3 se puede observar el caso donde solo se toma en cuenta la cabecera mínima agregada solo por LwM2M, resultando más eficiente que MQTT-SN.

Tabla 3.3. Cabecera mínima introducida

Cabecera Mínima Insertada	Tamaño (Bytes)
LwM2M	2
MQTT-SN	7

A continuación, se presenta un ejemplo de análisis de los mensajes a nivel de Byte para el protocolo LwM2M (Figura 3.1 a Figura 3.5) y MQTT-SN (Figura 3.6):

1. **LwM2M y CoAP:**

68 45 2c 20 27 46 c9 e9 6d c5 85 e1 61 04 62 2d 16 ff e8 15 e1 08 00

Figura 3.1. Bytes CoAP

El primer Byte (Figura 3.2) indica la versión y tipo de mensaje CoAP y la longitud del campo Token.

68 = 0110 1000

Figura 3.2. Bits de Versión, Tipo y Longitud de Token para mensaje CoAP

- Versión del Protocolo CoAP(01): Versión 1.
- Tipo de Mensaje CoAP(10): Acknowledgement(2).
- Longitud campo Token(1000): 8 Bytes.

Método HTTP(45): 010 Tipo(2), 00101 Detalle(5) lo que corresponde al mensaje CoAP 2.05 Content.

Message ID(2c 20 = 0010 1100 0010 0000): El ID de mensaje es 11296

Token(27 46 c9 e9 6d c5 85 e1): mediante este valor se asocia el recurso a su objeto, en este caso a *Generic Sensor*(Proximidad).

Opciones(61 04 62 2d 16 ff):

La primera opción(Figura 3.3) según su análisis es de tipo Observe:

61 04 = 0000 0001 0000 0100

Figura 3.3. Opción CoAP 1

- Opción Delta 1 (0110): Al ser la primera observación del recurso, el valor de la opción es 6 correspondiente a Observe.
- Longitud del campo valor de la opción(0001): 1 Byte
- Valor de la Opción Observe(0000 0100): el número de secuencia Observe es 4

La segunda opción(Figura 3.4) según su análisis es de tipo *Content Format*:

62 2d 16 = 0110 0010 0010 1101 0001 0110

Figura 3.4. Opción CoAP 2

- Opción Delta 2 (0110): Se suma el valor Delta de la opción actual(6) a la suma de los anteriores valores Delta (6), es decir 6+6= 12(Content Format) por lo que esta opción indica el formato del contenido o *payload* del mensaje CoAP.
- Longitud del campo valor de la opción(0010): 2 Bytes
- Valor de la Opción *Content Format* (0010 1101 0001 0110): application/vnd.oma.lwm2m+tlv(11542), el formato del payload del mensaje CoAP corresponde a LwM2M TLV.

Fin de opciones(1111 1111)

Trama LwM2M:

e8 15 e1 08 00

Figura 3.5. Bytes LwM2M

- Campo Tipo de la trama LwM2M2 (e8): 1110 1000
    - Tipo del campo identificador(11): Recurso con información
    - Longitud del campo identificador(1): 2 Bytes
    - Tipo de campo Longitud(01): El campo Longitud de la trama LwM2M es de 1 Byte y se ignoran los 3 siguientes bits.
    - Longitud del campo Valor(000): ignorados
- Campo Identificador de la trama LwM2M(15 e1): Recurso 0001 0101 1101 0001 = 5601 o valor mínimo medido.
- Campo Longitud del campo Valor de la trama LwM2M(08): 8 Bytes .
- Valor(00): La medición de la lectura de datos es 0.

2. MQTT-SN:

Los bytes mostrados a continuación pertenecen a un mensaje MQTT-SN de publicación, el análisis de estos bytes permite entender mejor la estructura manejada por MQTT-SN.

08 0c 00 00 0a 00 00 34

Figura 3.6. Bytes MQTT-SN

Longitud de trama(08): El número total de Octetos de la trama es 0000 1000 = 8.

Tipo de mensaje(0c): Se tiene un mensaje tipo PUBLISH.

Banderas(00):

- DUP(0): No se toma en cuenta ya que se usa un nivel QoS igual a 0.
- QoS(00): Nivel de calidad de servicio 0.
- Retain(0): El mensaje no será retenido para ser enviado a un cliente nuevo que se conecte.
- TopicIDType(00): Se está utilizando ID de Tópico.

TopicID(00 0a): El ID de Tópico asignado es 0000 1010 = 10.

MsgID(00 00): No es usado ya que el mensaje PUBLISH al usarse QoS 0 no está asociado con ninguna respuesta

Data(34): Transformándolo a código ASCII se tiene un valor igual a 4.

3.2 Captura de Tráfico

A continuación, se puede observar un ejemplo de análisis de la captura de tráfico en el escenario LwM2M, con envío de Información de sensores cada minuto. En la Tabla 3.4 se encuentra el número de mensajes totales, perdidos y el porcentaje de mensajes de información respecto al total de mensajes enviados.

Tabla 3.4. Mensajes de información y administración totales y mensajes perdidos

	Información/Sensores	Administración	Total
Mensajes Totales	5757	1751	7508
Mensajes Perdidos	15	5	20
% Mensajes Perdidos	0,26%	0,28%	0,27%

De la Tabla 3.4 se puede obtener el porcentaje de Información enviada respecto al total que es igual a 76,67%.

En la Figura 3.7 se observa que en el tráfico de información durante la hora 19 tiene una caída de tráfico, esto corresponde a mensajes perdidos durante este período, esto debido a fallas de conexión o alta cantidad de tráfico.

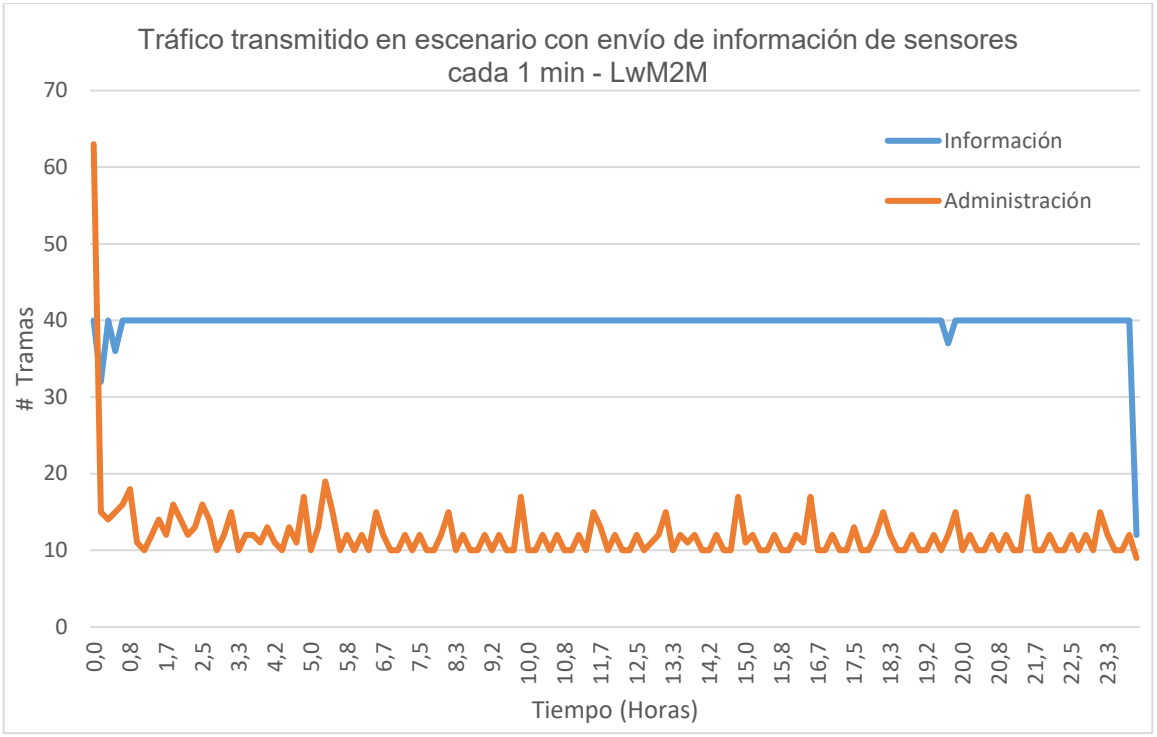

Figura 3.7. Escenario con envío de información de sensores cada minuto – LwM2M

Para analizar el número de mensajes perdidos se verificó en la captura de tráfico que cada mensaje de confirmación tenga respuesta y que exista una correcta observación de acuerdo con el período de tiempo. En la Figura 3.8 se puede observar como un mensaje CON no obtuvo su respectivo ACK en tiempos posteriores, además no hubo un mensaje que tenga se mismo ID, correspondiendo a un mensaje perdido.

No.	Time	Source	Destination	Protocc	Length	Info
85	17.422290	192.168.0.11	192.168.0.12	CoAP	70	CON, MID:39959,
90	18.455664	192.168.0.11	192.168.0.12	CoAP	70	CON, MID:39960,
91	18.468553	192.168.0.12	192.168.0.11	CoAP	65	ACK, MID:39960,
99	20.111156	192.168.0.11	192.168.0.12	CoAP	70	CON, MID:39961,
100	20.123362	192.168.0.12	192.168.0.11	CoAP	73	ACK, MID:39961,
104	23.429762	192.168.0.11	192.168.0.12	CoAP	70	CON, MID:39962,
105	23.444485	192.168.0.12	192.168.0.11	CoAP	72	ACK, MID:39962,
109	24.837251	192.168.0.11	192.168.0.12	CoAP	70	CON, MID:39963,
111	24.897993	192.168.0.12	192.168.0.11	CoAP	72	ACK, MID:39963,

Figura 3.8. Ejemplo de pérdida de paquete CoAP en archivo de captura

En la Tabla 3.5 se tiene un ejemplo del retardo para cada mensaje que requiere confirmación durante el envío de información de sensores cada minuto, el promedio de

estos para cada período permite obtener una referencia para entender el comportamiento del protocolo LwM2M respecto a retardo.

Tabla 3.5. Tiempo de respuesta para mensajes que requieren confirmación

		Retardo (ms)
Registro de Cliente	**POST/2.01 Created**	6,66
Observación	**GET/2.05 Content**	35,32
Keep Alive	**POST/2.04 Changed**	2,13
Cancel-Observación	**2.05 Content/Empty Message**	0,32
Desconexión	**DELETE/2.02 Deleted**	0,82

El mismo análisis es aplicado al escenario manejado por MQTT-SN, de esta manera, con la información recolectada para cada escenario, se realiza el análisis del comportamiento de cada protocolo de administración.

3.2.1 Con Envío de Información

Se debe considerar que el cliente LwM2M proporcionado por LESHAN está codificado de manera que, durante la observación de un recurso, el cliente envía el dato solo en el caso que exista un cambio en el recurso.

La pérdida de mensajes se encuentra sometida al estado de la conexión a Internet de la red local.

En la Figura 3.9 y Tabla 3.6 se puede observar el número de mensajes capturados en el equipo donde se ejecutó cada servidor. En un ambiente con envío de información se puede verificar que el bróker MQTT-SN manejó aproximadamente el doble de mensajes de mediciones de sensores, lo cual se debe a que el bróker debe publicar información a los tópicos a los que está suscrito el cliente, este número depende del número de clientes conectados.

Los mensajes de información de administración manejados por el servidor LwM2M son menores, ya que la observación de cierta información de administración como rangos, valores máximos y mínimos envían notificaciones al servidor solo cuando existe cambios. Además, gracias a la estructura que proporciona el protocolo LwM2M, este permite que en un mensaje de notificación se realice la observación de varios recursos como la correspondientes a la información del dispositivo, reduciendo el número de mensajes transmitidos por el cliente.

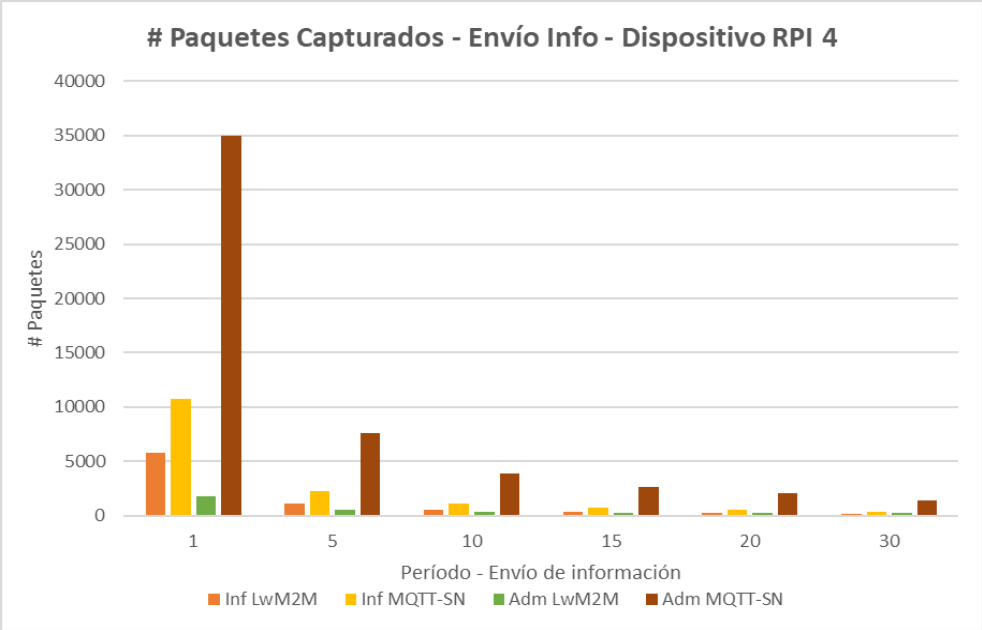

Figura 3.9. Mensajes capturados durante envío de información

Tabla 3.6. Mensajes capturados durante envío de información por cada período

Período [min]	Datos LwM2M	Datos MQTT-SN	Adm LwM2M	Adm MQTT-SN
1	5757	10710	1751	34996
5	1160	2272	503	7572
10	584	1150	331	3914
15	391	770	300	2688
20	296	584	261	2064
30	200	390	236	1436

La Tabla 3.7 muestra la eficiencia para cada protocolo, esta se calcula promediando el número de mensajes que abarca información de sensores respecto al número de mensajes total. Los valores para el protocolo MQTT-SN son bajos, ya que como se puede observar en la Figura 3.9, el tráfico referente a administración también es enviado durante cada período de tiempo, siendo este muy alto frente a la información de sensores.

Tabla 3.7. Eficiencia Información/Tráfico Total

	Eficiencia	
Período [min]	**Inf LwM2M**	**Inf MQTT-SN**
1	%76.68	%23.43
5	%69.75	%23.08
10	%63.83	%22.71
15	%56.58	%22.27
20	%53.14	%22.05
30	%45.87	%21.36

Se puede observar una mayor pérdida de mensajes en la captura de tráfico durante el envío de mensajes, en períodos cortos tales como 1 y 5 minutos, comparados con intervalos de tiempo de muestra mayores, tales como 30 minutos, donde las pérdidas fueron bajas e incluso nulas. En el escenario MQTT-SN, el Bróker RSMB al someterse a un alto nivel de tráfico, presenta en ocasiones errores de no publicación de los mensajes recibidos al cliente suscrito; sin embargo, esto no demuestra un mal desempeño del protocolo MQTT-SN sino de la herramienta Bróker RSMB, la cual ya no se encuentra en desarrollo. Por lo tanto, ambos protocolos muestran estabilidad al momento de manejar tráfico de información de sensores junto tráfico de administración.

En la Tabla 3.8 y Figura 3.10 se muestra los mensajes perdidos en cada período de tiempo de captura para ambos protocolos de administración.

La captura durante el período de 1 minuto en el escenario con MQTT-SN muestra una mayor pérdida de mensajes, ya que el Bróker no está optimizado para manejar una alta cantidad de tráfico. Además, al manejar un nivel de calidad de servicio igual a 0, el Bróker no está obligado a confirmar la publicación del mensaje.

El escenario donde se ejecutó el protocolo de administración LwM2M presentó una menor cantidad de pérdidas cuando existe alta cantidad de tráfico. Por tanto, el período de envío de mensajes es un factor a considerar al momento de la ejecución de cada cliente en su respectivo escenario.

Tabla 3.8**.** Mensajes Perdidos por cada período de tiempo con envío de información

	Información						**Administración**					
	LwM2M			**MQTT-SN**			**LwM2M**			**MQTT-SN**		
Periodo [min]	**Total**	**Perdidos**	**%**	**Total**	**Perdidos**	**%**	**Total**	**Perdidos**	**%**	**Total**	**Perdidos**	**%**
1	5757	15	0,26%	10710	18	0,17%	1751	5	0,28%	34996	40	0,11%
5	1160	0	0,00%	2272	8	0,35%	503	0	0,00%	7572	8	0,11%
10	584	0	0,00%	1150	2	0,17%	331	0	0,00%	3914	0	0,00%
15	391	1	0,26%	770	6	0,77%	300	1	0,33%	2688	4	0,15%
20	296	0	0,00%	584	0	0,00%	261	0	0,00%	2064	4	0,19%
30	200	0	0,00%	390	2	0,51%	236	0	0,00%	1436	8	0,55%

A partir de los mensajes *Keep Alive* se garantizó que el canal de comunicación estuviese funcional, de manera que los clientes puedan enviar mensajes a su servidor/bróker correctamente a pesar de que existan pérdidas de mensajes,

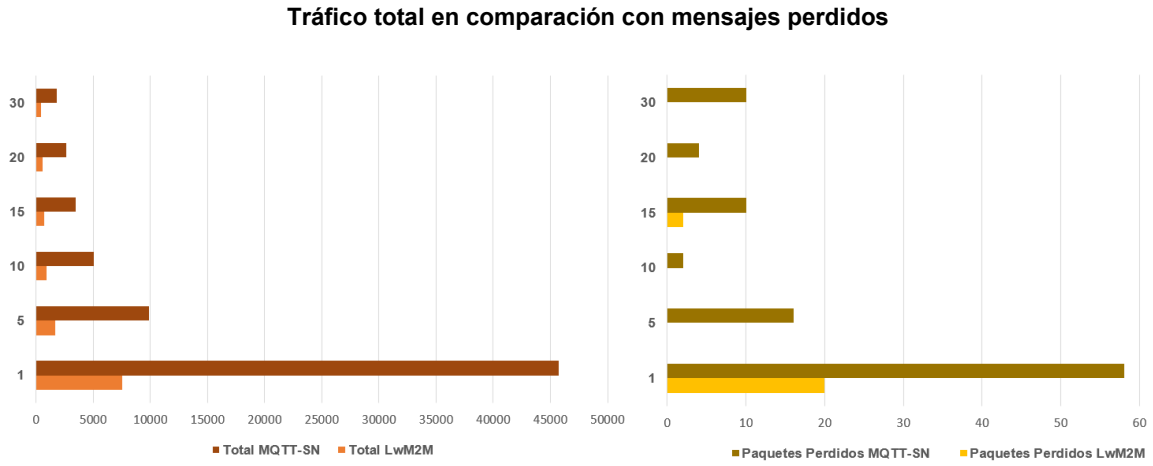

Figura 3.10. Tráfico Total y mensajes perdidos con envío de información

En la Tabla 3.9 se puede ver los mensajes perdidos para cada periodo de tiempo de cada protocolo de administración, las posibles y mayores causas de la pérdida de mensajes en ambos escenarios se debieron al canal de comunicación, ya que se utiliza una red WiFi, donde la transmisión de datos se dio en períodos de tiempo cortos, creándose un flujo de tráfico alto, ya que aparte de los escenarios planteados, en la red local siempre existirá tráfico ajeno al analizado.

Tabla 3.9. Número de mensajes perdidos y porcentaje respecto al tráfico total

	LwM2M		%	MQTT-SN		%
1	7508	20	0,27%	45706	58	0,13%
5	1663	0	0,00%	9844	16	0,16%
10	915	0	0,00%	5064	2	0,04%
15	691	2	0,29%	3458	10	0,29%
20	557	0	0,00%	2648	4	0,15%
30	436	0	0,00%	1826	10	0,54%

3.2.2 Sin Envío de Información

La información de administración presentada en la Tabla 2.43 y Tabla 2.44 abarca la necesaria para que cada protocolo pueda iniciar la conexión, mantenerla y finalizarla, además de la necesaria para la suscripción(MQTT-SN) u observación(LwM2M) y las características del dispositivo.

De acuerdo con las capturas de tráfico, LwM2M emplea un número mucho menor de tramas de administración, el manejo de recursos, instancias y objetos permite enviar mensajes más estructurados y con más información, comparado al cliente MQTT-SN que se programó para enviar una característica por tópico.

En la Tabla 3.10, y la Figura 3.11, se observa que, a menores períodos de tiempo, el tráfico manejado por LwM2M no disminuye considerablemente en comparación a MQTT-SN, siendo un protocolo adecuado para administración, ya que envía el tráfico necesario para la gestión del cliente cuando es necesario.

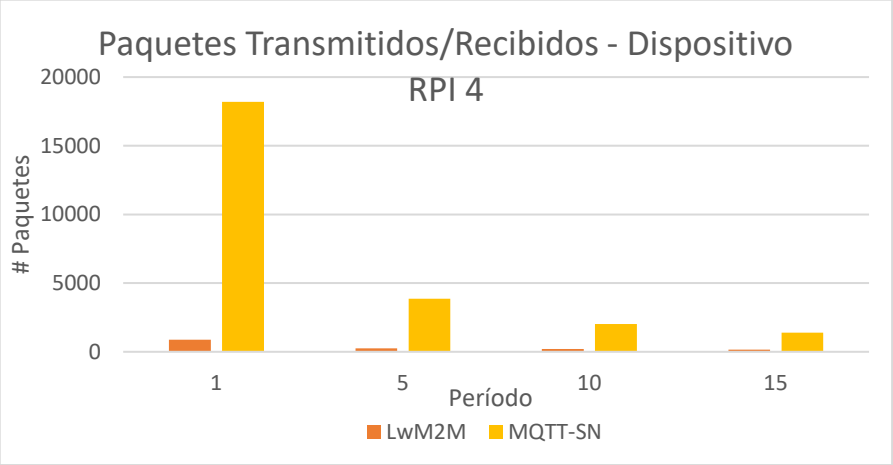

Figura 3.11. Mensajes capturados en escenario sin envío de información

Tabla 3.10. Mensajes capturados por cada período sin envío de información

Período [min]	LwM2M	MQTT-SN
1	883	18190
5	253	3866
10	212	2020
15	159	1396

En la Tabla 3.11 se puede observar el número de mensajes de administración empleados para conexión, iniciar o cancelar la observación/suscripción, *Keep Alive* y cierre de conexión. Este número de mensajes es constante a pesar del período de tiempo. Mediante esta información es posible obtener el tráfico que abarca características del dispositivo y especificaciones de los sensores.

Tabla 3.11. Tramas de administración para mensajes que requieren confirmación

	# Tramas Administración	
Mensajes Intercambiados	**LwM2M**	**MQTT-SN**
Establecimiento – Conexión	2	2
Observación – Suscripción	46	26
Mantenimiento – Conexión	52	50
Can-Obser/De-Sub	46	26
Cierre – Conexión	2	2
Total	148	106

El número de tramas empleadas para iniciar y finalizar la conexión es igual en ambos protocolos, difiriendo en el mantenimiento de la conexión o tramas *Keep Alive*, esto debido a que en LwM2M, el cliente, a pesar de ser configurado para enviar estas tramas cada media hora, el período de envío es menor, equivalente a 1600 segundos o 26.67 minutos.

La eficiencia presentada en la Tabla 3.12, se calcula tomando en cuenta el tráfico que abarca las características del dispositivo cliente y de los sensores, esto frente al tráfico de administración total que se observa en la Tabla 3.10.

Tabla 3.12. Eficiencia de tráfico de características de dispositivo frente a tráfico total

			Eficiencia	
Período [min]	**LwM2M**	**MQTT-SN**	**LwM2M**	**MQTT-SN**
1	**735**	**18084**	83,24%	99,42%
5	**105**	**3760**	41,50%	97,26%
10	**64**	**1914**	30,19%	94,75%
15	**11**	**1290**	6,92%	92,41%

Se puede observar que MQTT-SN transmite un mayor número de tramas de información del dispositivo y sensores respecto a LwM2M, esto no significa que LwM2M transmita menos información, ya que ésta la transmite en un solo mensaje o cuando presenta cambios.

De acuerdo a la Figura 3.12 y la Tabla 3.13, en el escenario con el cliente LwM2M, no se presentaron pérdidas durante su ejecución, mientras que para MQTT-SN, las pérdidas fueron bajas, mostrando ambos protocolos estabilidad frente a alto tráfico en un ambiente de administración.

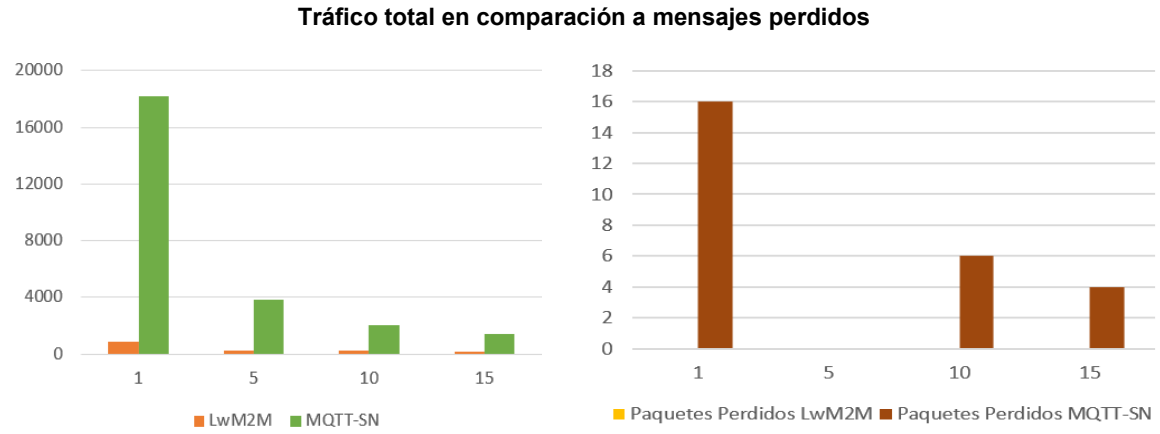

Figura 3.12. Mensajes perdidos frente a tráfico total sin envío de información

Tabla 3.13. Mensajes perdidos por cada período de tiempo sin envío de información

	LwM2M			MQTT-SN		
	Total	**Perdidos**	**%**	**Total**	**Perdidos**	**%**
1	883	0	0,00%	18190	16	0,09%
5	253	0	0,00%	3866	0	0,00%
10	212	0	0,00%	2020	6	0,30%
15	159	0	0,00%	1396	4	0,29%

3.2.3 Análisis de Retardo

A partir del promedio de los tiempos de retardo presentes para cada protocolo de administración en su respectivo escenario, en la Tabla 3.14 se puede observar el tiempo en que el servidor tarda en responder a mensajes que requieren confirmación.

Tabla 3.14. Retardo de respuesta a mensajes que requieren confirmación

	Retardo (ms)			
RPI 4	**Con envío**		**Sin envío**	
	LwM2M	**MQTT-SN**	**LwM2M**	**MQTT-SN**
Registro	40.77	0.19	33.15	0.14
Obs/Sub	18.66	0.24	20.12	0.22
Keep Alive	4.86	13.15	9.53	7.73
Cancel - Obs/Unsub	4.00	0.16	3.16	0.14
Desconexión	5.68	0.32	2.10	0.23

MQTT-SN presenta un menor retardo entre mensajes que requieren confirmación frente a CoAP/LwM2M, concluyendo que el bróker MQTT-SN responde con mayor rapidez. Esto debido al tamaño de trama empleado por LwM2M que es mucho mayor, requiriendo un mayor tiempo de procesamiento.

Además, MQTT-SN proporciona un establecimiento de conexión más rápida en comparación a LwM2M, esto debido a que LwM2M al momento de crear la conexión también indica los objetos o recursos a utilizarse, debiendo procesar esta información previo a la creación del cliente.

Es necesario tomar en cuenta que la respuesta a mensajes que requieren confirmación es esencial para cumplir con el protocolo de pruebas en cada escenario, esto debido a que en el caso de no haber una confirmación a una solicitud de observación o suscripción, no habrá notificación o publicación para el recurso o tópico, lo que implica pérdida de información y la cancelación de la captura de tráfico en Wireshark.

3.2.4 Video Streaming

En la captura de tráfico durante el envío de *video streaming* (2 horas y 4 minutos) en cada cliente, en su respectivo escenario, hubo pausas durante la reproducción de video; sin embargo, estos tiempos de carga fueron bajos sin afectar la visualización del *stream*.

El cliente raspberry pi para cada escenario tuvo un desempeño satisfactorio durante la ejecución del *video streaming,* ya que este último no fue afectado por la alta afluencia de tráfico, esto se debe a que ambos protocolos, además de manejar formatos de mensaje corto, no utilizan mucho ancho de banda.

Los factores que influyeron durante el proceso realizado en este protocolo de prueba fueron la velocidad de transmisión y la estabilidad manejada por el canal de comunicación. Debido a que se tiene una red estable, el *video streaming* se reprodujo con fluidez, sin ser afectado por la inserción de tráfico por parte de los protocolos de administración IoT. En la Tabla 3.15 se puede ver que el tráfico manejado por el cliente MQTT-SN es más elevado frente a LwM2M; sin embargo, esto no afectó a la reproducción del video.

Tabla 3.15. Porcentaje de número de mensajes de información vs total de mensajes - *Video Streaming*

	Número de Mensajes	
	LwM2M	**MQTT-SN**
Información	504	888
Administración	212	3102
Total	716	3990
Porcentaje de información enviada	70,39%	22,26%

En el escenario manejado por LwM2M, al momento de iniciar la conexión, que considera el envío del mensaje con la información de registro y los recursos a observar, se pudo ver un tiempo de carga alto durante el inicio del *video streaming*, esto se da ya que se está enviando un mensaje de gran tamaño, a medida que continúa el *streaming* hubo tiempos de carga menores.

El comportamiento de ambos protocolos durante este escenario de *Video Streaming* refleja estabilidad frente a alto flujo de tráfico, esto se puede observar en la Figura 3.13 y su respectiva Tabla 3.16 donde el número de mensajes perdidos no representa un porcentaje considerable dentro del tráfico total.

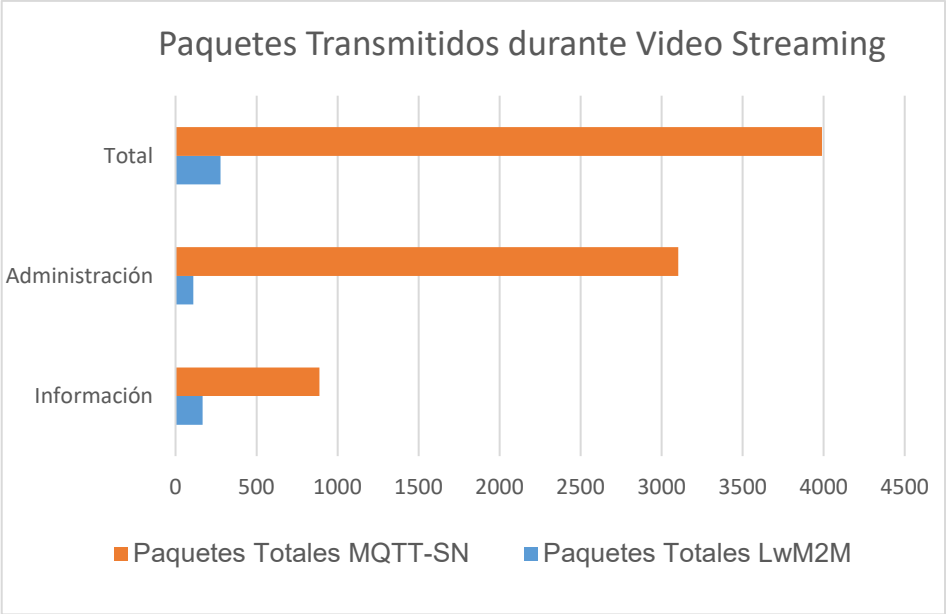

Figura 3.13. Mensajes transmitidos durante *Video Streaming*

Tabla 3.16. Mensajes perdidos y porcentaje respecto al total durante *Video Streaming*

	Información		**Administración**		**Total**	
	LwM2M	**MQTT-SN**	**LwM2M**	**MQTT-SN**	**LwM2M**	**MQTT-SN**
Mensajes Totales	504	888	212	3102	716	3990
Mensajes Perdidos	4	50	0	28	4	78
% Mensajes Perdidos	0,79%	5,33%	0,00%	0,89%	0,56%	1,92%

El tiempo de respuesta durante la ejecución de *video streaming* (Tabla 3.17) no presentó valores altos comparados a los análisis en otros escenarios, deduciendo que ambos protocolos tienen comportamiento estable durante la transmisión de video.

El *video streaming* al ser una aplicación de tiempo real presentó un tiempo de respuesta estable, ya que el contenido pudo ser observado sin ralentizaciones notables.

Tabla 3.17. Tiempo de retardo para mensajes confirmables durante *video streaming*

RPI 4	**Retardo (ms)**	
	LwM2M	**MQTT-SN**
Registro	105,23	0,296
Obs/Sub	17,73	0,281
Keep Alive	3,66	0,175
Cancel - Obs/Unsub	2,09	0,166
Desconexión	6,05	0,395

3.2.5 Descarga de Archivo

Ambos escenarios durante la descarga de un archivo de 990MB, presentaron un comportamiento estable, el tiempo de descarga no fue elevado. Se obtuvo un tiempo de descarga de 41 minutos para el escenario manejado por LwM2M y 39 minutos para MQTT-SN.

En la Tabla 3.18 se puede observar que en el escenario LwM2M se manejó menos cantidad de tráfico que en MQTT-SN; sin embargo, la descarga del archivo desde el cliente no fue más rápida dependiendo del estado de red WiFi. En un ambiente de administración ambos protocolos presentaron estabilidad frente a descarga de archivos.

Tabla 3.18. Porcentaje de número de mensajes de información vs total de mensajes - Descarga de Archivo

	LwM2M	**MQTT-SN**
Mensajes de Información	168	282
Mensajes de Administración	109	1030
Total	277	1312
Porcentaje de información enviada	60,65%	21,49%

En la Figura 3.14 se puede observar los mensajes transmitidos durante la descarga del archivo y en la Tabla 3.19 a los mensajes perdidos. El número bajo de mensajes perdidos para MQTT-SN se debió a fallas del Bróker durante la descarga del archivo, debido a que hubo un alto flujo de tráfico; mientras que LwM2M no presentó pérdidas, demostrando una vez más la estabilidad de ambos clientes frente a la presencia de un alto tráfico.

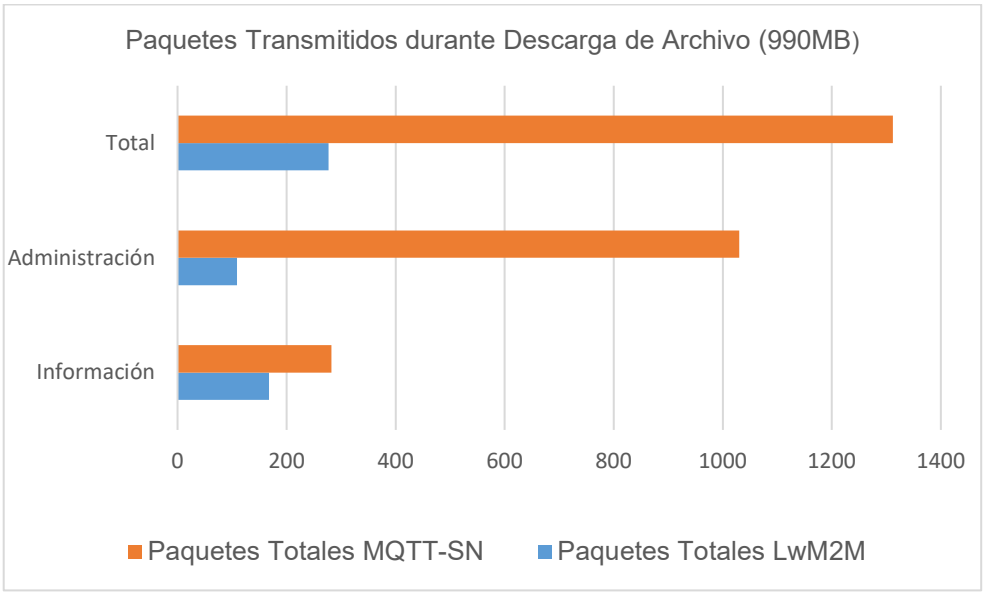

Figura 3.14. Mensajes Transmitidos durante descarga de archivo

Tabla 3.19. Mensajes perdidos respecto al total de tráfico durante descarga de archivo

	Información		Administración		Total	
Mensajes	**LwM2M**	**MQTT-SN**	**LwM2M**	**MQTT-SN**	**LwM2M**	**MQTT-SN**
Totales	168	282	109	1030	277	1312
Perdidos	0	6	0	6	0	12
% Perdidos	0,00%	2,08%	0,00%	0,58%	0,00%	0,91%

En la Tabla 3.20 se puede observar el tiempo que le tomó a cada mensaje que requiere confirmación recibir su respectiva respuesta. En un ambiente de descarga de un archivo, estos tiempos de respuesta no fueron elevados. Por lo tanto, en ambos escenarios los protocolos LwM2M y MQTT-SN mostraron estabilidad.

Tabla 3.20. Tiempo de retardo para mensajes confirmables durante descarga de archivo

	Retardo (ms)	
	LwM2M	**MQTT-SN**
Registro	40,93	0,108
Obs/Sub	166,84	0,107
Keep Alive	15,85	0,075
Cancel - Obs/Unsub	0,55	0,086
Desconexión	1,27	0,113

3.3 Conclusiones

El análisis del escenario con LwM2M en un ambiente de administración permitió determinar que este protocolo proporciona objetos y/o recursos predefinidos, junto a la capacidad de agregar nuevos objetos y recursos, garantiza escalabilidad, flexibilidad y estandarización, al permitir cargar un gran número de sensores y que estos puedan estar definidos dentro del sistema o agregarlos sin problema gracias al manejo de Objetos IPSO.

MQTT-SN, al estar orientado a redes de sensores, permite asignar un tópico a la medición que proporciona el sensor sin importar el tipo, por lo que puede abarcar un gran número de sensores en el sistema IoT, de esta manera, provee escalabilidad y flexibilidad.

Respecto a la estandarización, MQTT-SN, a pesar de tener aplicaciones, no es muy utilizado en ambientes comerciales, por lo que no existe una norma que indique el manejo de la información, esto parece una desventaja; sin embargo, permite acoplar cualquier tipo de sensor y facilita el manejo de datos a partir de la programación del cliente.

La cabecera insertada por el protocolo LwM2M varía de acuerdo con el tipo de información manejada; además, es necesario tomar en cuenta a CoAP junto a LwM2M al momento de analizar la cabecera insertada ya que proporciona información del recurso u objeto leído. Sin embargo, si se toma en cuenta la cabecera mínima de 2 bytes insertada por LwM2M,

ésta es menor que los 7 bytes insertados por MQTT-SN, proporcionando una mayor eficiencia en el manejo de información.

La conexión de red es un factor importante en el desempeño del protocolo en su respectivo escenario, ya que durante el procedimiento de actualización o *Keep Alive*, se permite refrescar el tiempo de vida, en el caso de LwM2M, si el mensaje de refresco se pierde, el servidor dejará de escuchar los recursos debiendo reiniciarse el registro del cliente, esto conlleva a la cancelación automática de la observación. Al no haber recursos observados, la captura de tráfico en *wireshark* tendrá que ser descartada ya que no habrá tráfico de información de sensores. En el presente trabajo, se presentaron pocos casos, siendo las fallas de conexión de red la principal razón de este error.

En MQTT-SN, si el cliente no recibe el mensaje PINGRESP luego de enviar el mensaje PINGREQ al Bróker, este buscará reconectarse al Bróker o conectarse a otro; por lo tanto, en los escenarios planteados, al haber como pérdida máxima un mensaje PINGRESP, la conexión al Bróker junto a las suscripciones no se perderá.

El Broker RSMB utilizado para MQTT-SN, al no encontrarse en desarrollo, no permite varias características tal como el uso de tópicos predefinidos o clientes dormidos. Además, presentó errores al momento de no publicar los datos enviados por el cliente.

Es necesario tomar en cuenta que, en la ejecución del cliente MQTT-SN, al momento de enviar información de sensores el cliente hacia el Broker habrá un retardo mínimo de 2 segundos, debido al establecimiento de la conexión entre el Raspberry Pi y el Arduino Uno antes de proceder con la lectura y envío de datos. Por lo tanto, es necesario tomar en cuenta que a pesar de tener un protocolo IoT de administración que sea rápido, la forma de recolección de datos de sensores también introducirá un retardo en el ambiente IoT.

De acuerdo con el funcionamiento del cliente Leshan LwM2M, cuando se captura tráfico, por cada 100 mensajes NON se enviará un mensaje CON, es decir, si de un objeto en observación se han enviado 100 mensajes NON, se enviará un mensaje CON con el valor del recurso, por lo que este mensaje requerirá confirmación. De esta manera se asegura una conexión persistente entre cliente y servidor LwM2M.

A pesar de existir en el mercado *gateways* transparentes que permiten la conversión entre los protocolos MQTT y MQTT-SN, el Bróker RSMB permitió verificar el funcionamiento propio del protocolo sin necesidad de intermediarios. Sin embargo, al no encontrarse en desarrollo el Bróker RSMB, no existe en el mercado un bróker propio para MQTT-SN.

A diferencia de MQTT-SN, para LwM2M si existen servidores en el mercado que se encuentran estandarizados y en desarrollo tal como LESHAN, de manera que existe más flexibilidad al momento de escoger soluciones para trabajar con LwM2M.

La arquitectura publicación/suscripción de MQTT-SN permite a los clientes ser parte de la administración ya que estos pueden suscribirse a los tópicos referentes a un dispositivo y observar su estado. En LwM2M esta administración está centralizada y llevada a cabo por el servidor, quien puede ver el estado de todos los dispositivos conectados.

Debido a que el Bróker debe recibir el tráfico publicado y volverlo a publicar al cliente suscrito, la cantidad de tráfico en este caso se duplica, lo cual se pudo observar en las gráficas correspondientes al escenario manejado por MQTT-SN. Por lo tanto, a medida que se agreguen más clientes al escenario el tráfico incrementará.

El comportamiento del protocolo MQTT-SN en el cual se publica a los tópicos cada cierto período, permite afirmar que sus aplicaciones están centradas a seguimiento estricto del funcionamiento de dispositivos y su administración como la Telemedicina.

LwM2M al ser más estructurado, permite una administración más centralizada, siendo una aplicación objetiva la administración dentro de las *Smart Grids* e incluso la Industria 4.0.

Entre la descarga de archivo y *Video Streaming*, los escenarios mostraron estabilidad durante la ejecución de ambos servicios, siendo la cantidad de tráfico manejada por cada protocolo la diferencia más notable entre cada escenario. A partir de estas pruebas se concluye que ambos protocolos son lo suficientemente estables y no infieren en el desempeño de servicios como el *streaming* de video y la descarga de un archivo.

De acuerdo con los escenarios analizados se deduce que mientras mayor sea el tráfico y tiempo que esté ejecutándose el bróker RSMB, mayor será la probabilidad de fallos de este, debiéndose tomar varias capturas hasta tener la menor cantidad de pérdidas.

En la herramienta Wireshark, no existe un filtro que permita capturar y observar el tráfico detallado para MQTT-SN, siendo necesario realizar una interpretación de los mensajes capturados y clasificarlo por protocolo de transporte, puerto y/o parte del código del mensaje, por lo que para futuros trabajos se puede desarrollar un filtro de este protocolo para Wireshark.

El presente trabajo puede extenderse al uso en aplicaciones móviles Android; además del análisis de consumo de batería por aplicación, extendiendo el estudio de ambos protocolos y elección del mejor para un determinado escenario de acuerdo con protocolos establecidos.

El correcto desempeño de cada protocolo de administración IoT en su respectivo escenario permitió concluir que los dispositivos embebidos tal como Raspberry Pi 4 son ideales para implementar aplicaciones IoT, donde los clientes hacen uso bajo de recursos y de red.

3.4 Recomendaciones

Durante las capturas de tráfico realizadas en un período de 24 horas, existieron fallas de conexión, de sensores, del dispositivo Raspberry Pi 4, tomándose en cuenta las capturas con mejores resultados y comportamiento similar.

El análisis en el presente trabajo usa un canal de comunicación basado en WiFi, pudiendo expandirse a otras tecnologías de red; además, puede ser orientado a aplicaciones más específicas como administración de ambientes dentro de Smart Grids, Telemedicina, Sistemas de Riego entre otros.

Respecto al hardware donde se implementaron los clientes, es necesario tomar en cuenta fallas propias del dispositivo embebido. ya que, en el ambiente real, el dispositivo Raspberry Pi 4 presentó fallas en su conexión a la red Wifi, esto debido a fallas propias de la placa al momento de conectar un monitor por el puerto HDMI y/o usar resoluciones altas, perdiéndose o reduciéndose la capacidad de conexión a la red.

Al momento de realizar capturas es necesario tomar en cuenta la configuración de los equipos que estarán trabajando por largos períodos de tiempo, debido a que, según su configuración de ahorro de energía, al ingresar a este estado de ahorro se puede llegar a cancelar la captura de tráfico o la ejecución del servidor.

Al momento de analizar el número de mensajes perdidos, se debe tomar en cuenta los tiempos de desfase, debido a que al irse acumulando puede haber períodos en los que pareciera que no hubo tráfico, esto no significa que se perdieron mensajes, sino que, debido al desfase de tiempo, no hubo mensajes transmitidos en ese tiempo.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] Open Mobile Alliance. (2018). *Lightweight Machine to Machine Technical Specification*. Disponible en: https://www.openmobilealliance.org/release/LightweightM2M/V1_0_2-20180209-A/OMA-TS-LightweightM2M-V1_0_2-20180209-A.pdf. Último Acceso: 17 de julio de 2024.

[2] Stanford-Clark, A., & Truong, H. L. (2013). *MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2*. Disponible en: https://groups.oasis-open.org/higherlogic/ws/public/download/66091/MQTT-SN_spec_v1.2.pdf/latest. Último Acceso: 17 de julio de 2024.

[3] Eclipse. (2020). *Leshan*. Disponible en: https://github.com/eclipse-leshan/leshan. Último Acceso: 17 de julio de 2024.

[4] Salazar, J., & Silvestre, Y. S. (2020). *INTERNET DE LAS COSAS*. Disponible en: https://techpedia.eu/topic/obj/866. Último Acceso: 17 de julio de 2024.

[5] Sinche, S., Raposo, D., Armando, N., Rodrigues, A., Boavida, F., Pereira, V., & Silva, J. S. (2020). A Survey of IoT Management Protocols and Frameworks. *IEEE Communications Surveys and Tutorials*, *22*(2), 1168–1190. Disponible en: https://doi.org/10.1109/COMST.2019.2943087. Último Acceso: 17 de julio de 2024.

[6] IBM Corporation. (2008). *Getting started with the Really Small Message Broker Really Small Message Broker and other IBM technologies*. Disponible en: https://github.com/eclipse/mosquitto.rsmb/blob/master/rsmb/doc/gettingstarted.htm. Último Acceso: 17 de julio de 2024.

[7] Cope S. (2024, January 26). *Using The Python MQTT-SN Client*. Disponible en: http://www.steves-internet-guide.com/python-mqttsn-client/. Último Acceso: 17 de julio de 2024.

[8] IETF. (2014). *The Constrained Application Protocol*. Disponible en: https://datatracker.ietf.org/doc/html/rfc7252. Último Acceso: 17 de julio de 2024.

[9] Sharpe, R., Warnicke, E., & Lamping, U. (1991). *Wireshark User's Guide*. Disponible en: https://www.wireshark.org/docs/wsug_html/. Último Acceso: 17 de julio de 2024.

5 ANEXOS

ANEXO I. Servidor Leshan LwM2M

ANEXO II. Código del Cliente Leshan LwM2M

ANEXO III. Bróker RSMB

ANEXO IV. Código del Cliente MQTT-SN

ANEXO V. Script Python para recolección de datos a partir de conexión serial del Arduino

ANEXO VI. Script Arduino para lectura de datos de sensores

ANEXO VII. Archivos PCAP de las capturas de tráfico para cada protocolo de administración.

ANEXO VIII. Archivos Excel sobre análisis de Tráfico para cada protocolo de administración.

ANEXO I

Servidor Leshan LwM2M

El ejecutable para el servidor Leshan LwM2M se encuentra cargado en el repositorio One Drive: Código Servidor Leshan LwM2M

ANEXO II

Código del Cliente Leshan LwM2M

El archivo ejecutable .jar y el código Java del Cliente Leshan LwM2M se encuentran cargados en el repositorio One Drive: Código Cliente Leshan LwM2M

ANEXO III

Bróker RSMB

El ejecutable para el Bróker MQTT-SN se encuentra cargado en el repositorio One Drive:

Bróker RSMB

ANEXO IV

Código del Cliente MQTT-SN

El código del Cliente Python 3 MQTT-SN se encuentra cargado en el repositorio One Drive:
Código Cliente MQTT-SN

ANEXO V

Script Python para recolección de datos a partir de conexión serial del Arduino

El Script Python para recolección de datos del Arduino se encuentra cargado en el repositorio One Drive: Script Python para Recolección de Datos de Sensores

ANEXO VI

Script Arduino para lectura de datos de sensores

El Script Arduino para lectura de datos de sensores se encuentra cargado en el repositorio One Drive: Script Arduino para lectura de datos de sensores

ANEXO VII

Archivos PCAP de las capturas de tráfico para cada protocolo de administración

Los archivos .pcap de las capturas de tráfico se encuentran cargados en el repositorio One Drive: Archivos PCAP

ANEXO VIII

Archivos Excel sobre análisis de Tráfico para cada protocolo de administración

Los archivos Excel que abarcan el análisis de tráfico se encuentran cargados en el repositorio One Drive: Archivos Excel sobre análisis de Tráfico