TMAC: โพรโทคอลชั้นแมคเพื่อการสื่อสารระยะไกลสำหรับโพรโทคอลเอ็มคิวทีทีบนเครือข่ายลอร่า

TMAC: A telemetry MAC protocol for MQTT over LoRa networks

ธงชัย เจือจันทร์[1], ทวีวัฒน์ มูลจัด[1] และ เอกธนัช เหลืองศิริวรรณ[1*]
Thongchai Chuachan[1], Taweewat Moonjat[1] and Aktanat Luengsiriwan[1*]

Received: 29 August 2022 ; **Revised**: 13 January 2023 ; **Accepted**: 16 February 2023

บทคัดย่อ

เอ็มคิวทีทีเป็นโพรโทคอลที่ออกแบบและพัฒนาสำหรับการเชื่อมต่อที่มีแบนด์วิดท์ต่ำ และถูกนำไปประยุกต์ใช้อย่างแพร่หลายสำหรับอินเทอร์เน็ตในสรรพสิ่ง ซึ่งส่วนใหญ่เอ็มคิวทีทียังคงถูกประยุกต์ใช้เพียงบนเครือข่ายรูปแบบทีซีพี/ไอพีเดิม แต่ปัจจุบันมีความต้องการใช้เอ็มคิวทีทีสำหรับเครือข่ายระยะไกลมากขึ้น โดยงานวิจัยก่อนหน้ายังไม่มีการพัฒนาโพรโทคอลเอ็มคิวทีทีให้ผสานใช้ร่วมกับเครือข่ายระยะไกลได้อย่างมีประสิทธิภาพ งานวิจัยนี้จึงออกแบบและพัฒนาโพรโทคอล TMAC เพื่อขยายการเชื่อมต่อโพรโทคอลเอ็มคิวทีทีถูกใช้สำหรับเครือข่ายระยะไกล และทดลองโพรโทคอลจากงานวิจัยนี้บนระบบเทสเบด ซึ่งผลการทดลองพบว่า โพรโทคอล TMAC สามารถเชื่อมต่อได้ในระยะไม่ต่ำกว่า 5 กิโลเมตรและมีโอเวอร์เฮดของเน็ตเวิร์คต่ำ

คำสำคัญ: เอ็มคิวทีที เครือข่ายลอร่า ไอโอที

Abstract

MQ Telemetry Transport (MQTT) has been designed and developed for an ultra-low bandwidth communication and has been applied to used in Internet of Things (IoT). Recently, MQTT has been used in traditional TCP/IP networks. Yet, the ultra-low bandwidth and long-range communication of MQTT have been considerably increased. Previous proposed techniques still struggle to make MQTT compatible with Long-Range (LoRa) networks. In this paper, we have design and implement the Telemetry Media Access Control (TMAC). Our TMAC helps increase ranges of MQTT's data communications. We have experimented our TMAC using a network testbed. Experimental results have illustrated that our TMAC can transfer data over 5 kilometers with low network overhead.

Keywords: MQTT, LoRa Networks, IoT

[1] สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสุรินทร์ จังหวัดสุรินทร์
[1] Computer science, Faculty of Science and Technology Surin Rajabhat University, Surin Province
* Corresponding author Email: aktanat@srru.ac.th

บทนำ

MQ Telemetry Transport (MQTT) (HiveMQ, 2020) เป็นหนึ่งในโพรโทคอลที่ถูกใช้อย่างแพร่หลายสำหรับการเชื่อมเข้าสู่ระบบเครือข่ายอินเทอร์เน็ตของอุปกรณ์อินเทอร์เน็ตในสรรพสิ่ง (Internet of Things: IoT) โดย MQTT ถูกนำมาประยุกต์ใช้เพื่อควบคุมสั่งการและรับข้อมูลจากอุปกรณ์เซ็นเซอร์ (Sensors) ผ่านเครือข่าย Ethernet และ WiFi เป็นต้น ซึ่งทำให้ MQTT มีความสำคัญอย่างมากสำหรับ IoT ในปัจจุบัน

จากความต้องการประยุกต์ใช้ IoT สำหรับเครือข่ายระยะไกล หน่วยงาน Internet Engineering Task Force (IETF) (IETF, 2021) จึงออกแบบมาตรฐาน Low-Power Wide Area Network (LPWAN) (Heile *et al.*, 2018) เพื่อเพิ่มศักยภาพของ IoT ให้สามารถเชื่อมต่อผ่านเครือข่ายไร้สายที่มีระยะครอบคลุมพื้นที่การเชื่อมต่อกว้างขึ้น โดย Long-Range (LoRa) (Ferré & Giremus, 2018; Vangelista, 2017) เป็นหนึ่งเทคโนโลยีการกล้าสัญญาณ (Modulation) ของเครือข่ายไร้สายที่ถูกนำมาใช้ แต่ LoRa ขาดโพรโทคอลควบคุมการเชื่อมต่อสื่อสารระหว่างอุปกรณ์ และไม่สามารถนำ MQTT ผสานเป็นส่วนหนึ่งกับ LoRa ได้

ก่อนหน้านี้มีการพัฒนาโพรโทคอล Long Range Wide Area Networks (LoRaWAN) (Petrariu *et al.*, 2019) ทำงานเทียบได้กับชั้น Media Access Control (MAC) (Ansari & Zhang, 2013) ที่คอยกำกับการรับและส่งเฟรมข้อมูลของ LoRa ซึ่ง LoRaWAN มีความสามารถ เช่น เชื่อมต่อแบบสองทาง (Full Duplex) มีกลไกควบคุมการส่งเฟรมข้อมูล และสามารถผสาน LoRaWAN ให้ถูกใช้งานกับ MQTT ผ่านทางแม่ข่าย (Server) ของ LoRaWAN ได้ แต่ LoRaWAN ไม่สามารถผสานเข้ากับ MQTT ได้โดยตรงที่เกตเวย์ดัง Figure 3 (ในส่วนของ App Server) ไม่สามารถสื่อสารได้แบบปลายทางสู่ปลายทาง (End-to-End) มีความหน่วงของการเชื่อมต่อสูง (High Latency) และส่งผลกระทบต่อประสิทธิภาพการเชื่อมต่อสื่อสารดัง (Kietzmann *et al.*, 2022) ได้กล่าวไว้

ด้านงานวิจัยที่มีแนวคิดที่จะนำ MQTT มาผสานเข้ากับ LoRa (Bhawiyuga *et al.*, 2019 ; Huang *et al.*, 2019 ; Spinsante *et al.*, 2017 ; Sun *et al.*, 2020) นั้น ส่วนใหญ่ใช้วิธีสร้างเกตเวย์และรับข้อมูลจากอุปกรณ์ IoT ในชั้น LoRa โดยไม่มีการพัฒนาชั้น MAC และถึงแม้จะมีโอเวอร์เฮดด้านการประมวลผล (Computational Overhead) ไม่สูง แต่จะเกิดโอเวอร์เฮดของเครือข่าย (Network Overhead) สูงหากอุปกรณ์เกิดการเชื่อมต่อล้มเหลว เกิดข้อมูลสูญหายโดยที่ไม่มีกลไกตรวจสอบ ทำให้การเชื่อมต่อโดยใช้ LoRa นั้นไม่มีความน่าเชื่อถือ (Unreliable) และไม่มีวิธีจัดการเมื่อต้องส่งข้อมูลที่มีขนาดใหญ่กว่าแบนด์วิดท์ของ LoRa และไม่สามารถเชื่อมต่อแบบ Full Duplex ได้

ส่วนการประกาศมาตรฐาน SCHC over LoRaWAN (Audebert *et al.*, 2021) ถูกออกแบบให้มีกลไก Fragmentation and Reassembly (FaR) อยู่บน LoRaWAN โดยใช้กลไกของ Static Context Header Compression and Frag-mentation (SCHC) (Minaburo *et al.*, 2020) ซึ่งช่วยให้การพัฒนามีความสะดวกมากขึ้นและสามารถนำ MQTT ผนวกกับ LoRa ได้โดยให้ MQTT อยู่บนโพรโทคอล SCHC แต่การเพิ่มชั้นเครือข่าย MQTT ให้อยู่บน SCHC และ LoRaWAN นั้นจะทำให้มีโอเวอร์เฮดของเครือข่ายสูง และไม่เหมาะกับการนำไปใช้กับเครือข่ายที่มีทั้งแบนด์วิดท์และทรัพยากรประมวลผลต่ำ

งานวิจัยนี้จึงออกแบบและพัฒนาโพรโทคอล Telemetry Media Access Control (TMAC) สำหรับการสื่อสารโดยใช้ MQTT บนเครือข่าย LoRa ที่มีคุณสมบัติ 1) ปรับปรุงโพรโทคอล MQTT ให้ผสานใช้งานร่วมกับ LoRa ได้ 2) เพิ่มปริมาณการส่งข้อมูลโดยพัฒนากลไก FaR 3) มีกลไกป้องกันการสูญหายของข้อมูลในชั้น MAC และ 4) เข้ากันได้กับ MQTT ที่ถูกใช้อยู่ในปัจจุบัน โดย TMAC ถูกพัฒนาโดยใช้ MicroPython ติดตั้งและทดสอบโพรโทคอลบนฮาร์ดแวร์รุ่น ESP32 TTGO-v1.0 แล้วทดสอบบนเครือข่าย Test-bed เพื่อศึกษาโอเวอร์เฮดของเครือข่าย และดีเลย์ (Delay) และกลไกการกู้คืน (Recovery) แพ็กเกตที่เกิดสูญหาย ซึ่งผลการทดลองพบว่าโพรโทคอลจากงานวิจัยนี้เกิดโอเวอร์เฮดของเครือข่ายต่ำ และเกิดการสูญหายและมีการกู้คืนสัดส่วนน้อยที่การส่งสัญญาณระยะ 5 กิโลเมตร

ทบทวนวรรณกรรมที่เกี่ยวข้อง

1. โพรโทคอล MQTT

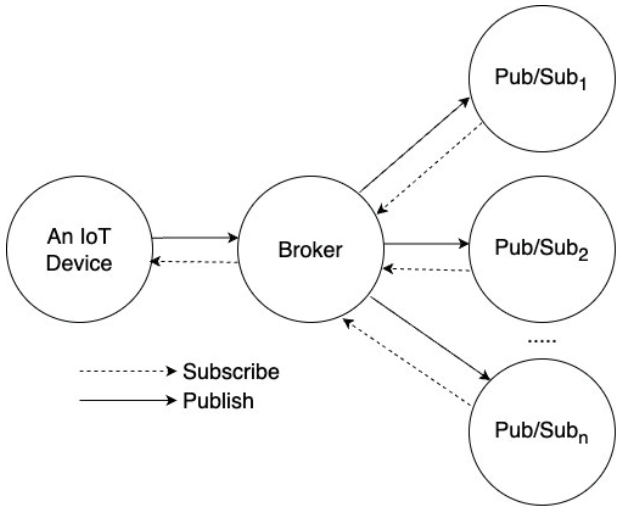

Figure 1 An MQTT system architecture

MQ Telemetry Transport (MQTT) เป็นหนึ่งในโพรโทคอลที่ถูกใช้สำหรับการรับและส่งคำสั่งควบคุมระหว่างอุปกรณ์ IoT และผู้ใช้ โดยอาศัยตัวกลาง (Broker) ดังแสดงใน Figure 1 ทั้งอุปกรณ์ IoT และผู้รับข้อมูล (Pub/Sub ใน Figure 1) จะต้องเชื่อมไปยัง Broker ก่อนและถ้าหาก *Pub/Sub* จะสื่อสารไปยังอุปกรณ์ IoT (*An IoT Device*) จะต้องส่งคำสั่งไปที่ Broker โดยใช้ชื่อเรียก เช่น */room1/light* เป็นต้น ซึ่งเรียกว่าการ Publish ข้อมูล จากนั้นอุปกรณ์ใดๆ ก็ตามที่เชื่อมไปยัง Broker และมีการรอรับข้อมูล (Subscribe) ชื่อเดียวกัน Broker จะส่งข้อมูลดังกล่าวให้กับทุกอุปกรณ์ที่ Subscribe ไว้

โพรโทคอล MQTT ในปัจจุบันทำงานอยู่บนชั้นทรานสปอร์ต (Transport layer) ส่งข้อมูลโดยใช้ TCP (Socolofsky & Kale, 1991) และส่งข้อมูลได้แบบเรียลไทม์ (Realtime) ซึ่งเกิดจากปัจจัยการออกแบบที่คำนึงถึงประสิทธิภาพการใช้พลังงานแบตเตอร์รี ใช้แบนด์วิดท์ต่ำ และกำลังเป็นเครื่องมือสำคัญสำหรับการเชื่อมต่อระหว่างอุปกรณ์ IoT ที่ต้องการครอบคลุมพื้นที่ส่งสัญญาณในระยะไกลได้

2. โพรโทคอล LoRaWAN

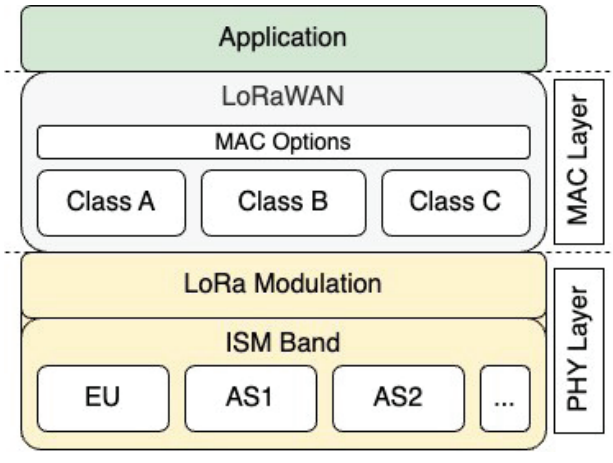

Figure 2 LoRaWAN protocol stack

Long Range Wide Area Networks (LoRaWAN) (Petrariu *et al.*, 2019) เป็นโพรโทคอลในชั้น MAC ที่สนับสนุนให้เกิดการเชื่อมต่อแบบสองทาง (Bi-directional Communication) สนับสนุนการเชื่อมต่อสื่อสารขณะที่อุปกรณ์เกิดการเคลื่อนที่ (Mobility) และเชื่อมต่อได้ในระยะไกล ซึ่งใน Figure 2 เป็นชั้นการทำงานของ LoRaWAN โดย LoRaWAN ทำงานอยู่บนชั้นกายภาพ (PHY layer) LoRa และ LoRaWAN จะมีชุดรูปแบบการทำงานทั้งหมด 3 คลาส ประกอบด้วย Class A, Class B และ Class C ใน Class A อุปกรณ์ IoT จะอยู่ในสถานะเดินเครื่องเปล่า (Idle) และจะเริ่มทำงานตามเวลาที่กำหนดและกลับสู่สถานะ Idle เพื่อทำงานในรอบถัดไป ส่วน Class B ปรับปรุงจาก Class A โดยมีการกำหนดช่วงเวลาทำงานเป็นช่วงเวลาที่กำหนดเพื่อรอรับข้อมูล ทำให้ Class B มีแพ็กเกต Beacon เพื่อใช้ประสานเวลา (Synchronize) เป็นระยะเพื่อให้ทุกอุปกรณ์ IoT มีเวลาที่ตรงกันสำหรับใช้บริหารการส่งข้อความจากหลายอุปกรณ์ และ Class C จะทำงานตลอดเวลา ทำให้ Class C เตรียมรับและส่งข้อมูลตลอดเวลา โดยไม่คำถึงถึงการใช้พลังงาน ซึ่งเหมาะกับการใช้กับระบบที่ต้องรับและส่งข้อมูลตลอดเวลา เช่น Smart Grid เป็นต้น

สำหรับสถาปัตยกรรม LoRaWAN ดัง Figure 3 นั้นเป็นภาพรวมทั้งหมด ซึ่งประกอบด้วย 4 ส่วน คือ 1) อุปกรณ์ IoT ที่เชื่อมต่อเข้าสู่เครือข่าย LoRaWAN (LoRa Devices) 2) ส่วนเกตเวย์ (LoRa Gateway) ซึ่งจะคอยรับและส่งแพ็กเกตในชั้น LoRaWAN ระหว่างอุปกรณ์ IoT และเกตเวย์ จากนั้นเกตเวย์จะแทนที่ชั้น LoRaWAN ด้วย TCP/IP และส่งไปยัง A Network Server 3) ส่วน A Network Server จะคอยรับข้อมูลจากหลายเกตเวย์ เพื่อตรวจสอบการซ้ำของแพ็กเกตจากอุปกรณ์ IoT และส่งต่อให้ส่วนที่ 4) ซึ่งเป็นส่วนของ An Application Server โดยข้อมูลในชั้น App จะนำไปใช้ประโยชน์ต่อโดยนักพัฒนา เช่น เชื่อมต่อโดยใช้โพรโทคอล MQTT, Rest API, CoAP และอื่นๆ ซึ่งจะเห็นว่าสถาปัตยกรรมทั้งหมดของ LoRaWAN นั้นมีความซับซ้อน ทำให้อาจมีดีเลย์สูง

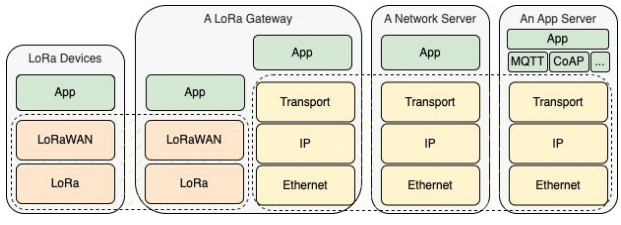

Figure 3 LoRaWAN communication architecture

3. งานวิจัยก่อนหน้า

งานวิจัยที่พยายามผสาน LoRa ร่วมกับ MQTT ก่อนหน้า (Bhawiyuga *et al.*, 2019; Spinsante *et al.*, 2017; Sun *et al.*, 2020) มีลักษณะคล้ายกัน คือ อาศัยการสร้างเกตเวย์เพื่อนำข้อมูลที่ได้รับจากชั้น LoRa เข้าไปไว้ในส่วนข้อความของ MQTT แล้วแทรกด้วย TCP/IP ก่อนที่จะส่งข้อมูลต่อไปยัง Broker และ Broker ก็จะส่งต่อไปยังผู้รับข้อมูล (Subscribers) ถึงแม้ LoRa จะถูกผสานจากชั้น LoRa ไปยัง MQTT ได้ แต่จะเกิดปัญหาเมื่อจำนวนอุปกรณ์ที่เชื่อมต่อมีจำนวนมาก ซึ่งอาจทำให้เกิดแพ็กเกตสูญหายโดยไม่มีกลไกตรวจสอบ และสื่อสารแบบสองทางไม่ได้ ซึ่งเป็นหนึ่งในปัญหาที่ถูกแก้ไขได้จากงานวิจัยนี้

ด้าน Changqing และคณะ (Sun *et al.*, 2020) ได้พัฒนาการผสาน LoRa และ MQTT เข้าด้วยกันโดยการใช้ Raspberry PI เป็นเกตเวย์ LoRa แล้วนำข้อมูลที่ได้จากชั้น

LoRa ส่งต่อไปยังแม่ข่ายกลาง 2 ช่องทางคือ การใช้ MQTT และ Rest API เมื่อข้อมูลถูกเก็บที่แม่ข่ายบนระบบคลาวด์แล้วก็จะนำไปใช้ประโยชน์ด้านอื่นๆ แต่วิธีการนี้เป็นเพียงการเพิ่มช่องทางการส่งข้อมูลจากเกตเวย์สู่ระบบคลาวด์ ซึ่งในสภาพแวดล้อมใช้งานจริงการนำข้อมูลเข้าสู่ระบบคลาวด์โดยมีสองช่องทางอาจไม่มีความจำเป็น และไม่ช่วยให้การสื่อสารโดยใช้ LoRa มีประสิทธิภาพมากขึ้น

Amin และคณะฯ (Tayebi *et al.*, 2022) วิเคราะห์และออกแบบการเชื่อมเครือข่ายของอุปกรณ์ IoT ที่ใช้ LoRaWAN และไม่ใช้ LoRaWAN จากนั้นทำการผสานทั้งสองส่วนเข้าด้วยกันเพื่อใช้ระบบหลังบ้าน (Backend) ของ LoRaWAN ได้ เช่น การเชื่อมต่อจาก CoAP มายัง Backend ของ LoRaWAN ได้เป็นต้น และถึงแม้ Amin และคณะฯ จะช่วยให้อุปกรณ์ที่เชื่อมต่อด้วย LoRaWAN และไม่ใช้ LoRaWAN ใช้งานร่วมกันก่อนที่จะผสานเข้าสู่ MQTT ได้ แต่ก็เป็นการเพิ่มความซับซ้อนของการเชื่อมต่อเครือข่ายและยากต่อการจัดการ และมีโอกาสที่จะเกิดดีเลย์และโอเวอร์เฮดในเครือข่ายสูง

4. การผสาน LoRa กับ MQTT โดยใช้ LoRAWAN และ SCHC

ความพยายามที่จะนำ TCP/IP ให้สามารถสื่อสารได้บน LoRa ของมาตรฐาน RFC 9011 (Audebert *et al.*, 2021) เป็นวิธีอาศัย LoRaWAN เป็นกลไกส่งเฟรมสำหรับ SCHC ซึ่งคอยห่อหุ้ม (Encapsulate) แพ็กเกตจากการสื่อสารด้วย TCP/IP ไปยังอุปกรณ์ IoT บน LoRaWAN ดังนั้น RFC 9011 จึงเป็นจุดเชื่อมต่อที่ทำให้ TCP/IP ถูกขยายการให้ใช้งานบนเครือข่าย LoRa ได้ แต่วิธีนี้เป็นการเพิ่มจำนวนชั้นของเครือข่ายจำนวนมหาศาลโดยไม่จำเป็น ซึ่งกลไกการเชื่อมต่อด้วย TCP/IP หากนำมาใช้บน LoRaWAN จะเกิดโอเวอร์เฮดในระบบเครือข่ายสูงจนส่งผลให้การขยายตัวของการใช้ IoT ช้าลงเนื่องจากปัญหาด้านประสิทธิภาพได้

การออกแบบโพรโทคอล TMAC

1. สถาปัตยกรรมของ TMAC

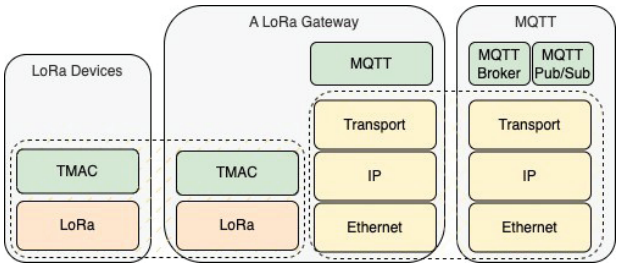

Figure 4 TMAC Protocol Stack and Architecture

งานวิจัยนี้ออกแบบโพรโทคอล Telemetry Media Access Control (TMAC) เพื่อขยายการใช้โพรโทคอล MQTT บนเครือข่าย LoRa โดยมีสถาปัตยกรรมดัง Figure 4 ซึ่งมี 3 ส่วน คือ

- Lora Devices คือ อุปกรณ์ IoT ที่สนับสนุนการเชื่อมต่อด้วย LoRa และในอุปกรณ์จะมีชั้นเครือข่าย 2 ชั้น คือ LoRa และ TMAC
- A LoRa Gateway เป็นส่วนที่คอยแปลโพรโทคอล TMAC ที่ได้รับจากจุดเชื่อมต่อ LoRa (LoRa Interface) มาสร้างเป็นแพ็กเกต MQTT ที่อยู่บนเครือข่าย TCP/IP และส่งต่อไปยัง MATT Broker
- MQTT คือส่วนของ MQTT Broker และกลุ่มอุปกรณ์ Pub/Sub บนระบบคลาวด์

2. โพรโทคอล TMAC

2.1 ขั้นตอน Bootstrapping ของ TMAC

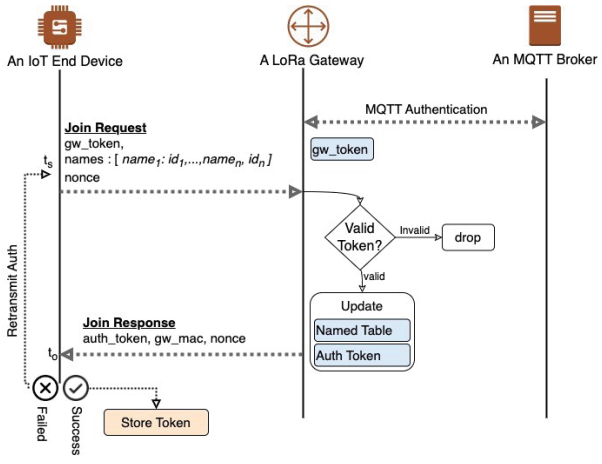

Figure 5 An TMAC bootstrapping procedure

ขั้นตอนการเชื่อมต่อของ TMAC และ MQTT เป็นดัง Figure 5 โดยจะต้องเริ่มติดตั้ง LoRa Gateway โดยยืนยันตัวตนและเชื่อมต่อไปยัง Broker จนสำเร็จ และที่เกตเวย์จะมีค่าคอนฟิกสำหรับการให้สิทธิ์การเข้าถึงเครือข่ายของอุปกรณ์ IoT คือ gw_token และเมื่อมีการปลุกเครื่อง (Bootstrap) อุปกรณ์ IoT จะเริ่มทำงานดัง Figure 5 โดยอุปกรณ์ IoT จำเป็นต้องมีการคอนฟิกค่าเบื้องต้นไว้ดังนี้

- *gw_token* คือ รหัสที่ใช้สำหรับยืนยันขอสิทธิ์การเชื่อมต่อเครือข่ายผ่านเกตเวย์
- *names* คือ รายการชื่อที่ใช้สำหรับการ Publish และ Subscribe จากอุปกณ์ IoT ไปยัง Broker โดยข้อมูลส่วนนี้จะแบ่งเป็น 2 ส่วน คือ ชื่อ (name) และหมายเลขชื่อ (id)

• *nonce* เป็นหมายเลขจากการสุ่มเพื่อป้องกันการซ้ำซ้อนของข้อมูลและใช้เป็นเครื่องมือระบุแพ็กเกต

เมื่อเริ่มปลุกเครื่องอุปกรณ์ IoT จะส่งข้อมูลข้างต้นไปแบบกระจาย (Broadcast) เมื่อเกตเวย์ได้รับและพบว่าเป็นแพ็กเกตประเภท Join Request (กล่าวไว้ในหัวข้อ 3.3 การแบ่งและรวมเฟรมของ TMAC) เกตเวย์จะตรวจสอบ *gw_token* เพื่อยืนยันและให้สิทธิ์เชื่อมต่อ โดยถ้าหาก *gw_token* ถูกต้อง เกตเวย์จะเก็บข้อมูล names ลงตาราง MQTT Names และ Auth Tokens ดัง Table 1 และ Table 2 ตามลำดับ

ในตาราง MQTT Names ดัง Table 1 ประกอบด้วยหมายเลข *MAC*, *name_id* และ *name* โดย *MAC* เป็นข้อมูลที่ดึงมาจากเฟรมของ TMAC (กล่าวไว้ในหัวข้อ 3.3 การแบ่งและรวมเฟรมของ TMAC) ส่วน *name_id* คือ หมายเลขเอกลักษณ์ของ *name* มี ซึ่ง *name* นี้คือชื่อสำหรับเชื่อมต่อกับ Broker และถูกใช้สำหรับการ Publish และ Subscribe ข้อมูลตามกลไกของ MQTT

Table 1 MQTT Names

MAC	name_id	name
MAC_1	1	/p
MAC_2	2	/n
...	...	...
MAC_n	n	/x

ส่วนตาราง Auth Tokens ใน Table 2 จะอัพเดตเมื่อเกตเวย์ยืนยันเลข *gw_token* สำเร็จ โดยเกตเวย์จะสุ่มเลข *auth_token* ขนาด 16 บิต จับคู่กับเลข *MAC* ซึ่งมีขนาด 48 บิต โดย *auth_token* นี้จะใช้สำหรับการยืนยันตัวตนคู่กับการใช้เลข *MAC* สำหรับการ Publish ข้อมูล

หลังจากที่ตาราง MQTT Names และ Auth Tokens ถูกอัพเดตแล้ว เกตเวย์จะส่ง *auth_token*, *gw_mac* และเลข *nonce* ของเกตเวย์กลับไปยังอุปกรณ์ IoT เป็นแพ็กเกตประเภท Join Response (กล่าวไว้ในหัวข้อ 3.3 การแบ่งและรวมเฟรมของ TMAC) โดย *auth_token* คือ หมายเลขที่ใช้สำหรับยืนยันอุปกรณ์ IoT ที่ผ่านการอนุญาตโดยเกตเวย์แล้ว สามารถใช้สำหรับส่งคำสั่ง Publish ข้อมูลได้ ส่วน *gw_mac* เป็นหมายเลข *MAC* ของเกตเวย์ ถูกใช้สำหรับการส่ง Publish ไปยังเกตเวย์ได้แบบตรง (Unicast) ส่วน *nonce* เป็นเลขเดียวกันกับ *nonce* ที่ได้รับจากแพ็กเกต Join Request เพื่อยืนยันว่าเป็นแพ็กเกต Join Response ที่ตอบสนองจากการส่ง Join Request ที่ถูกต้อง

Table 2 Auth Tokens

MAC	auth_token
MAC_1	Tk_1
MAC_2	Tk_2
...	...
MAC_n	Tk_n

ส่วนกลไกป้องกันการสูญหายของแพ็กเกตในขั้นตอน Bootstrapping ของ TMAC นั้น หลังจากเริ่มส่ง Join Request ระบบจะเก็บเวลาเริ่มทำงาน t_s จากนั้นรอแพ็กเกตตอบกลับ ซึ่งถ้าหากแพ็กเกตถูกส่งสำเร็จอุปกรณ์ IoT จะได้รับแพ็กเกต Join Response แต่ถ้าหากไม่ได้รับแพ็กเกต Join Response จนถึง t_o ซึ่งหมายถึงการหมดเวลา (Timeout) อุปกรณ์ IoT จะหน่วงเวลารอ (Backoff) ด้วยการสุ่มระหว่าง 1 ถึง 10 วินาที และเมื่อหมดเวลา Backoff แล้ว TMAC จะเริ่มส่งแพ็กเกต Join Request และเข้าสู่ขั้นตอน Bootstrapping ใหม่

2.2 การ Publish ข้อมูลด้วย TMAC

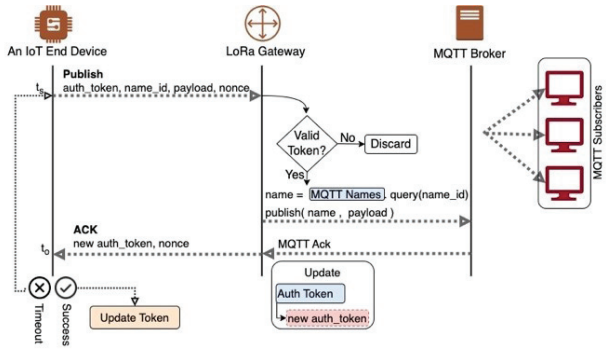

Figure 6 An IoT device publishes MQTT payloads

หนึ่งในกลไกของ MQTT คือการ Publish ข้อมูลไปยัง Broker ซึ่งการ Publish โดยใช้ TMAC บนเครือข่าย LoRa นั้นเป็นไปตาม Figure 6 ซึ่งเริ่มต้นส่งข้อมูลจากอุปกรณ์ IoT ที่ประกอบด้วย *auth_token*, *name_id*, *payload* และ *nonce* เมื่อแพ็กเกตส่งถึงเกตเวย์ เกตเวย์จะตรวจสอบ *auth_token* ในตาราง Auth Tokens (ตาม Table 2) โดยใช้เลข MAC และ *auth_token* หากไม่ตรงกันเกตเวย์จะละทิ้งแพ็กเกตดังกล่าวทิ้ง แต่ถ้าหากถูกต้องเกตเวย์จะค้นหา *name* จากตาราง MQTT Names (ใน Table 1) แล้วทำการ Publish ข้อมูล *payload* ไปยัง Broker

หลังจากเกตเวย์ส่งแพ็กเกตไปยัง Broker และได้รับแพ็กเกต PUBACK จาก Broker แล้ว เกตเวย์จะสร้าง *auth_token* ใหม่ และอัพเดตข้อมูลลงในตาราง Auth Tokens และส่ง *auth_token* กลับไปยังอุปกรณ์ IoT โดยใช้แพ็กเกตประเภท

Acknowledgement (ACK) ซึ่งประกอบด้วย *auth_token* และ *nonce* และหลังจากแพ็กเกต ACK ถูกส่งถึงอุปกรณ์ IoT แล้ว อุปกรณ์ IoT จะอัพเดตค่า *auth_token* ไว้ใช้สำหรับการ Publish ข้อมูลในครั้งถัดไป

สำหรับแพ็กเกตที่เริ่มส่ง ณ t_s จนถึง t_o แล้วแต่ยังไม่ได้รับแพ็กเกต ACK จะเกิด Timeout ขึ้น อุปกรณ์ IoT จะเข้าสู่กระบวนการส่งใหม่ (Retransmission) โดยกลับไปที่ขั้นตอน t_s

2.3 การรับข้อมูลด้วยวิธี Subscribe ผ่าน TMAC

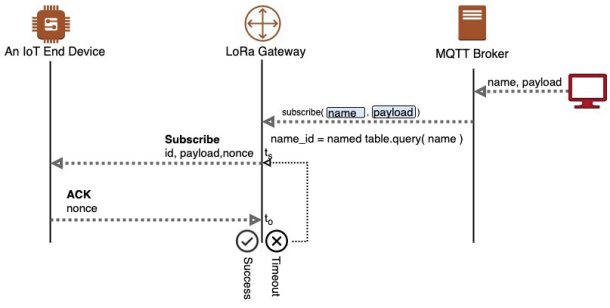

Figure 7 A subscription of IoT devices over TMAC

การ Subscribe ข้อมูลจาก Broker ผ่าน LoRa ของโพรโทคอล TMAC นั้น มีกลไกดัง Figure 7 ซึ่งเริ่มจากเกตเวย์ได้รับแพ็กเก็ต Subscribe จาก Broker จากนั้น เกตเวย์จะค้นหา *name_id* และ *MAC* จากการนำส่วน Topic ของแพ็กเกต Subscribe ค้นหาในตาราง MQTT Names แล้วประกอบเป็นแพ็กเกต TMAC ซึ่งมี *name_id* และ *payload* เป็นส่วนประกอบส่งไปยังอุปกรณ์ IoT พร้อมกับเริ่มนับเวลา t_s และหลังจากที่แพ็กเกตถูกส่งถึงอุปกรณ์ IoT แล้ว อุปกรณ์ IoT จะส่งแพ็กเกตประเภท ACK กลับมายังเกตเวย์ เพื่อยืนยันการได้รับข้อมูล แต่ถ้าหากไม่ได้รับ ACK จนถึง t_o เกตเวย์จะเริ่ม Backoff และส่งแพ็กเกตใหม่

3. การส่งเฟรมแพ็กเกต TMAC

3.1 เฟรมแพ็กเกต TMAC

	48-bit	4-bit	4-bit	8-bit	8-bit	variable	16-bit
	MAC	f_cnt	f_idx	p_type	p_len	payload	crc
An Example	748f3cc24b0d	3	0	1	255	*{ sensros: { ,...*	a2
	748f3cc24b0d	3	1	1	255	*temp: 28.4,.....*	8c
	748f3cc24b0d	3	2	1	240	*.....}}*	fa

Figure 8 TMAC frame

การส่งเฟรม TMAC ตาม Figure 8 มีทั้งหมด 5 ประเภท โดยมีข้อมูลแบ่งออกเป็น 2 ส่วน คือ ส่วนหัวแพ็กเกต (Header) ของ TMAC ประกอบด้วยหมายเลข MAC, f_cnt, f_idx, p_type, p_len, และ crc มีขนาดรวมกันเท่ากับ 88 บิต (11 ไบต์) และส่วน payload มีขนาดที่ยืดหยุ่นตามค่าในฟิลด์ p_len โดยมีรายละเอียดดังนี้

- MAC คือ หมายเลข MAC ของอุปกรณ์ IoT
- f_cnt คือ จำนวนเฟรมที่ใช้ทั้งหมดสำหรับการส่ง เฟรมแพ็กเกตมีขนาดใหญ่กว่า 255 ไบต์ ซึ่งเป็นขนาดเฟรมที่ใหญ่ที่สุดที่สามารถส่งได้ต่อครั้งด้วยไลบรารี sx127x ผ่านเครือข่าย LoRa
- f_idx คือ ดัชนี (index) ของเฟรมแพ็กเกต โดยจะมีค่าไม่เกิน f_cnt
- p_type คือประเภทของเฟรมแพ็กเกต TMAC ซึ่งกล่าวไว้ในหัวข้อ 3.3.2
- p_len คือ ขนาดของ payload
- payload คือข้อมูลที่ถูกส่งโดยใช้ TMAC
- crc คือ ค่าที่ใช้ตรวจสอบความถูกต้องของเฟรม

3.2 ประเภทของเฟรมแพ็กเกต TMAC

การส่งเฟรมแพ็กเกตของ TMAC ตาม Figure 8 ในส่วนของ p_type จะมีหมายเลขขนาด 8 บิต สำหรับระบุประเภทของเฟรมต่อไปนี้

- Join Request คือ เฟรมสำหรับการส่งขอรับการอนุญาตเข้าเชื่อมต่อเครือข่าย (ได้รับ *auth_token*) ซึ่ง Join Request หมายเลข MAC จะระบุเลข MAC ของผู้ส่งจากอุปกรณ์ IoT และเป็นแพ็กเกตประเภท Broadcast ถ้าเกตเวย์ได้รับ p_type เป็น Join Request ก็จะนำแพ็กเกต Join Request มาประมวลผล ส่วนอุปกรณ์ IoT อื่นๆ จะไม่สนใจแพ็กเกต Join Request ดังกล่าว
- Join Response จะส่งแพ็กเกตแบบ Unicast โดยเลข MAC ในเฟรมแพ็กเกต จะหมายถึงเลข MAC ของผู้รับ เช่น ถ้าเกตเวย์ส่งแพ็กเกต Join Response และระบุเลข MAC เป็น 748f3cc24b0d แล้วอุปกรณ์ IoT ที่มีเลข MAC ดังกล่าวเท่านั้นที่จะนำเฟรมแพ็กเกตไปประมวลผล อุปกรณ์ IoT อื่นๆ ก็จะไม่สนใจเฟรมแพ็กเกตดังกล่าว
- Publish เป็นแพ็กเกตที่ถูกส่งจากอุปกรณ์ IoT และการส่งจะระบุ MAC ของเกตเวย์ เพื่อให้เกตเวย์ส่งต่อแพ็กเกตดังกล่าวไปยัง Broker
- Subscribe คือ เป็นประเภทแพ็กเก็ตที่ต้องส่งจากเกตเวย์ไปยังอุปกรณ์ IoT ผ่านเครือข่าย LoRa ทำให้

แพ็กเก็ตประเภทนี้จะระบุเลข MAC เป็นของอุปกรณ์ IoT ที่ต้องรับแพ็กเก็ต Subscribe

- ACK เป็นแพ็กเกตแจ้งผลการส่งแพ็กเกตสำเร็จ ดังนั้นจึงระบุเลข MAC ของผู้รับในแพ็กเกต ACK

3.3 กลไกการ Fragmentation

ในการแพ็กเกตโดยใช้ TMAC บนเครือข่าย LoRa แพ็กเกตจะถูกแบ่งออกเป็นเฟรมให้มีขนาดไม่เกินกว่าที่ LoRa จะสามารถรับและส่งข้อมูลได้ เช่น การส่งแพ็กเกตบนเครือข่าย LoRa โดยใช้ไลบรารี sx127x จะทำได้ไม่เกิน 255 ไบต์ เป็นต้น ซึ่งหน่วยนี้เป็นหน่วยที่ระบุแพ็กเกตขนาดใหญ่ที่สุดที่จะส่งได้ Maximum Transmission Unit (MTU) ดังนั้น กลไกการ Fragmentation จะช่วยให้เฟรมแพ็กเกตมีขนาดไม่เกิน MTU ดังนี้

- หาจำนวนของเฟรมแพ็กเกตได้จาก c = p / (255-h) เมื่อ c คือ จำนวนเฟรมแพ็กเกต ที่จะถูกสร้าง p คือ ความยาวของ payload และ h คือ ความยาวของหัวแพ็กเกต TMAC

- หลังจากได้ c แล้วก็จะแบ่ง payload ออกตามจำนวน c ซึ่งหาได้จาก s = p/c เมื่อ s คือขนาดของข้อมูลในแต่ละเฟรมแพ็กเกต p คือ ขนาดของ payload และ c คือค่าที่ได้จากการหาจำนวนของเฟรมแพ็กเกต

- ข้อมูลที่ถูกแบ่งจาก payload แล้วจะนำมาเติม Header ของ TMAC และเริ่มส่งทีละเฟรม โดยทิ้งระยะห่างระหว่างเฟรมเป็นเวลา 100 มิลลิวินาที (ms) ซึ่งเป็นค่าที่เหมาะสม (Optimal) และน้อยที่สุดที่ทำให้การส่งเฟรมสำเร็จ

3.4 กลไกการ Reassembly

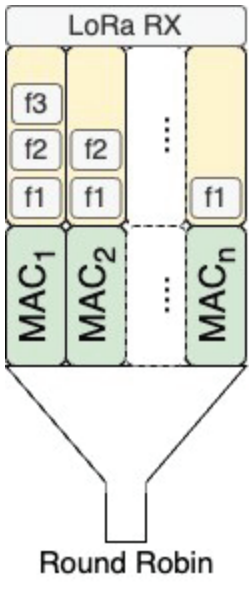

Figure 9 Reassembly bucket

หลังจากที่แพ็กเกตถูกส่งแล้ว ผู้รับจะตรวจสอบหมายเลข MAC ว่าเป็นแพ็กเกตของตัวเองหรือไม่ ถ้าหากหมายเลข MAC ตรงกับหมายเลข MAC ของเครื่อง ผู้รับจะสร้างตะกร้าดัง Figure 9 และรับแต่ละเฟรมแพ็กเกตไว้ที่ตะกร้าตามเลข MAC โดยหลังจากที่ได้รับ f_idx ครบตาม f_cnt แล้ว ตะกร้าของเลข MAC นั้นจะมีสถานะเป็นพร้อมนำออก และ TMAC จะนำเฟรมแพ็กเกตทั้งหมดมารวมเป็นแพ็กเกตและนำไปประมวลผลทีละแพ็กเกตตามลำดับด้วยวิธี Round Robin

สำหรับหมายเลข MAC ที่ถูกนำไปสร้างตะกร้าแต่จำนวน f_idx รวมแล้วไม่เท่ากับ f_cnt เกินกว่าเวลาที่กำหนด (Timeout) MAC ตะกร้าดังกล่าวจะถูกล้างออกจากหน่วยความจำ

4. การพัฒนาโพรโทคอลต้นแบบ

การพัฒนาโพรโทคอล TMAC พัฒนาโดยใช้ภาษา MicroPython และใช้ไลบรารี sx127x ในการกล้าสัญญาณ LoRa ระหว่างอุปกรณ์ IoT และเกตเวย์ ส่วนการสื่อสารระหว่างเกตเวย์และ Broker นั้นเกตเวย์พัฒนาโดยใช้ไลบรารี mqtt.simple สำหรับ esp32 และ Broker ใช้วิธีติดตั้งซอฟต์แวร์ Mosquito MQTT Broker บนระบบปฏิบัติการลินุกส์ Ubuntu 20.04 LTS

การทดลองและประเมินผล

1. แผนผังการทดลอง

การออกแบบการทดลองของงานวิจัยเพื่อประเมินประสิทธิภาพและประสิทธิผลของโพรโทคอล TMAC โดยมีแผนผังดัง Figure 10 ซึ่งประกอบด้วยคอมพิวเตอร์ Labtop, MQTT Broker, LoRa Gateway และ กลุ่มอุปกรณ์ IoT โดยคอมพิวเตอร์ Laptop และ LoRa Gateway จะเชื่อมไปยัง MQTT Broker ผ่านเครือข่ายไร้สาย (Wireless) ช่วงความถี่ 2.4 GHz และเกตเวย์จะเชื่อมกับกลุ่มอุปกรณ์ $IoT_{1\text{-}5}$ ผ่านเครือข่าย LoRa ห่างกันระยะ 1, 2, 3, 4, และ 5 กิโลเมตร ตามลำดับ โดยข้อมูลที่ส่งมี 3 ส่วน คือ

- อุปกรณ์ $IoT_{1\text{-}5}$ เข้าร่วมเครือข่าย (Join Networks) กับ LoRa Gateway เพื่อทดสอบการส่งเฟรมแพ็กเกตประเภท Join Request และ Join Response

- การส่งข้อมูล Publish จากอุปกรณ์ $IoT_{1\text{-}5}$ โดยงานวิจัยนี้ส่งข้อมูลขนาด 250, 500 และ 750 ไบต์ตามลำดับ เพื่อทดสอบการส่งเฟรมแพ็กเกตที่มีการแบ่งเป็นหลายเฟรมแพ็กเกตและเฟรมแพ็กเกตเดียว

- การ Subscribe ข้อมูลโดยอุปกรณ์ $IoT_{1\text{-}5}$ จาก Laptop ผ่าน MQTT Broker เพื่อทดสอบการส่งแพ็กเก็ตประเภท Subscribe ผ่านเครือข่าย LoRa

Figure 10 The test-bed scenario

หลังจากกำหนดการส่งข้อมูลแล้วการทดลองนี้ได้กำหนดค่าพารามิเตอร์อื่นๆ เช่น Timeout ไว้ที่ 5 วินาที (sec) พื้นที่การทดลองเป็นอ่างเก็บน้ำเพื่อควบคุม Line-of-Sight ของอุปกรณ์ IoT_{1-5} ให้เป็นแบบไม่มีสิ่งกีดขวาง และชั้น PHY ของ LoRa ดัง Table 3

Table 3 LoRa's parameter Settings

MAC	ID
Frequency	923.2Mhz
TX power	14
Signal bandwidth	125kHz
Spread factor	7
Coding rate	4
Preamble length	8
Implicit header	False

ฮาร์ดแวร์ที่ใช้สำหรับการทดลองโพรโทโทคอล TMAC ของวิจัยนี้ในส่วนของ MQTT Broker และ Laptop ใช้คอมพิวเตอร์โน้ตบุ๊คหน่วยประมวลผล Intel Core i5 หน่วยความจำสำรอง 8 GB ติดตั้งระบบปฏิบัติการลินุกส์ Ubuntu 20.04 LTS ส่วนอุปกรณ์ IoT_{1-5} และ LoRa Gateway ใช้ ESP32 LILYGO T-Beam V1.1 หน่วยประมวลผลรุ่น CH9102 หน่วยความจำ SPRAM 8MB และ FLASH 4MB ติดตั้ง MicroPython และโพรโทคอล TMAC จากงานวิจัยนี้

2. ตัววัด

ตัววัด (Metrics) ด้านประสิทธิภาพและประสิทธิผลของงานวิจัยกำหนดขึ้นเพื่อวัดประสิทธิภาพการเชื่อมต่อสื่อสารในระยะไกล ปริมาณการส่งข้อมูล (Throughput) ดีเลย์ (Delay) และปริมาณการส่งแพ็กเก็ตใหม่เมื่อเกิดแพ็กเกตสูญหาย (Retransmission) ดังต่อไปนี้

- Throughput โดยส่วนใหญ่ใช้ประเมินปริมาณการส่งข้อมูลสำเร็จ การใช้ Throughput ในการประเมินผลทำให้สามารถประเมินประสิทธิผลของการส่งแพ็กเก็ตของ TMAC ที่ผสานกับ MQTT ได้ และรายงานในหน่วยกิโลบิตต่อวินาที (kbps)

- Delay เพื่อวัดความล่าช้าในการส่งแพ็กเกตจากต้นทางไปยังปลายทาง เช่น จาก Labtop ไปยัง IoT_1, IoT_2 และ IoT_3 ตาม Figure 8 โดย Delay ใช้เป็นเครื่องมือประเมินประสิทธิภาพการจัดการแพ็กเกตโดยใช้ TMAC ได้ และรายงานในหน่วย ms

- Retransmission Ratio ใช้ประเมินอัตราการกู้คืน (Recovery) แพ็กเกตที่สูญหายของโพรโทคอล TMAC ในหน่วยร้อยละ

การทดลองตามตัววัดในงานวิจัยนี้ทำซ้ำอย่างน้อย 40 ครั้งและกำหนดช่วงความเชื่อมั่นไว้ที่ร้อยละ 95

3. ผลการทดลอง

3.1 ดีเลย์จากการส่งแพ็กเกต Join Request และ Join Response

ใน Figure 11 เป็นผลการทดลองส่ง Join Request จากอุปกรณ์ IoT จนกว่าจะได้รับ Join Response จากเกตเวย์จากระยะทาง 1-5 กิโลเมตร จะเห็นว่าการเกิดดี-เลย์อยู่ในระดับใกล้เคียงกัน แต่ถ้าดูในระยะ 4 กิโลเมตรมี Error bar สูงนั้นเกิดจากอุปกรณ์ IoT ไม่ได้รับ Join Response จากเกตเวย์จำนวน 2 ครั้งแต่เนื่องจากได้มีการกำหนดค่า Timeout ไว้ที่ 5 sec (หรือ 5,000 ms) ทำให้เกิด Error bar ในช่วงที่กว้าง แต่โดยเฉลี่ยแล้วระยะทางไม่ส่งผลต่อดีเลย์ของแพ็กเกต Join Request และ Join Response โดย TMAC สามารถประสานระหว่าง LoRa และ MQTT ได้ในเวลาที่ต่ำกว่า 2,500 ms ในระยะทาง 5 กิโลเมตร ซึ่งจะเห็นว่าระยะทางไม่ส่งผลให้เกิดเน็ตเวิร์คโอเวอร์เฮดสูงขึ้น

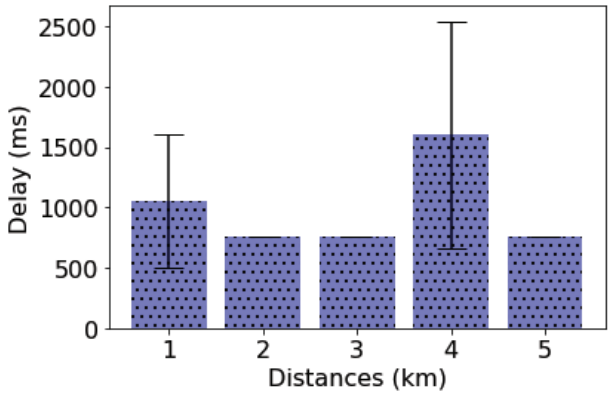

Figure 11 TMAC network join delay

3.2 ดีเลย์จากการการ Publish ข้อมูล

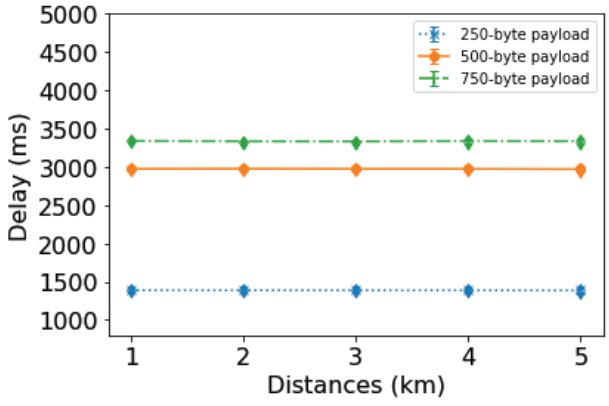

Figure 12 The TMAC delay of 250, 500 and 750 bytes data transmission

การ Publish ข้อมูลจากอุปกรณ์ IoT เพื่อผสานเข้ากับ MQTT นั้น TMAC สามารถช่วยให้ Publish ข้อมูลผ่านเครือข่าย LoRa ได้ปริมาณมากขึ้น โดยการทดลองได้ทดสอบส่ง Payload ขนาดตั้งแต่ 250, 500 และ 750 ไบต์ และ Publish สำเร็จร้อยละ 100 และเกิดดีเลย์ขึ้นดัง Figure 12 ซึ่งแกน x คือระยะทางในหน่วยกิโลเมตร และแกน y คือค่าดีเลย์ที่เกิดขึ้น โดยจะเห็นว่าการส่งแพ็กเกตที่ไม่เกิด Fragmentation ในทุกระยะทางจะมีดีเลย์ต่ำสุดที่ 1,386.1±0.9 ms และสูงสุดที่ 1,388.1±0.8 ms และถ้าหากมีการ Fragmentation มีดีเลย์ของการส่งแพ็กเก็ตขนาด 500 ไบต์ตั้งแต่ 2,969.6±1.2 ms และสูงสุดที่ 2,975.8±1.2 ms ส่วนแพ็กเก็ตที่มี Payload ขนาด 750 ไบต์จะมีดีเลย์อยู่ตั้งแต่ 3,333.0±0.9 ms และสูงสุดที่ 3,338.0±1.3 ms ทำให้ TMAC เป็นโพรโทคอลที่ช่วยเพิ่มปริมาณการส่งข้อมูลผ่านเครือข่าย LoRa ช่วยป้องกันแพ็กเกตสูญหาย และมีดีเลย์อยู่ในระดับต่ำกว่า 3,500 ms ในระยะ 5 กิโลเมตร

3.3 ผลการใช้แบนด์วิดท์เมื่อ Publish และ Subscribe ข้อมูล

เมื่อมีการ Publish และ Subscribe ข้อมูลผ่านเครือข่าย LoRa โดยใช้ Payload ขนาด 250, 500 และ 750 ไบต์ จากระยะทางตั้งแต่ 1-5 กิโลเมตร ดัง Figure 13 โดยแกน x คือระยะทางที่ทำการทดสอบ และแกน y คือ แบนด์วิทด์ที่ใช้ระหว่างการสื่อสารบนเครือข่าย LoRa ในหน่วย kbps และคาดว่าการส่งข้อมูลในระยะที่ไกลขึ้นจะเกิดโอเวเฮดสูงขึ้นและจะทำให้แบนด์วิดท์ลดลง แต่ผลการทดลองพบว่าแบนด์วิดท์ที่ใช้รับและส่งข้อมูลผ่านเครือข่าย LoRa จะอยู่ระหว่าง 3.52±0.1 kbps ถึง 3.56±0.1 kbps และเป็นระดับแบนด์วิดท์ที่สามารถใช้การได้สำหรับกับการรับและส่งคำสั่งควบคุมอุปกรณ์ IoT จากระยะไกลได้

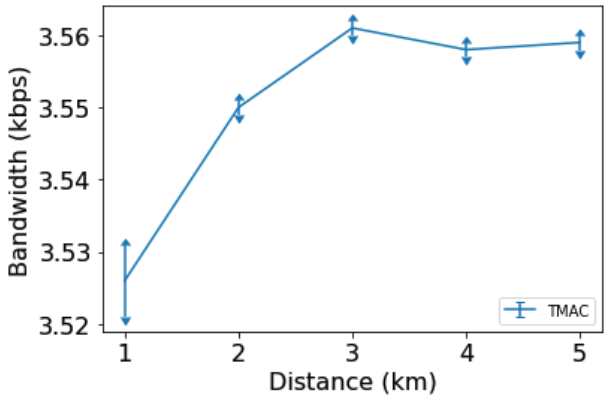

Figure 13 The bandwidth of TMAC publish and ssubscribe transmission

3.4 การเกิด Retransmission

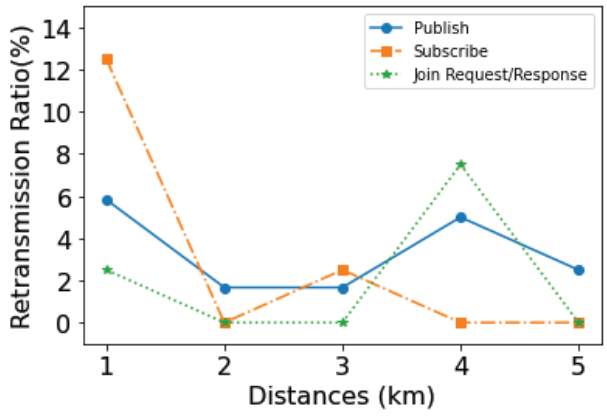

Figure 14 TMAC Retransmission Ratio

สัดส่วนการเกิด Retransmission ดัง Figure 14 คือจำนวนร้อยละของการส่งแพ็กเกตใหม่ของผลการทดลองทั้งหมด เมื่อเกิดการเชื่อมต่อล้มเหลวผ่านเครือข่าย LoRa ที่ระยะทาง 1-5 กิโลเมตรตามแนวแกน x และสัดส่วนร้อยละตามแนวแกน y และหากดูข้อมูลในแกน y จะเห็นว่าสัดส่วนการเกิด Retransmission ต่ำกว่าร้อยละ 10 โดยมีเพียงการส่ง Subscribe จากเกตเวย์ในระยะ 1 กิโลเมตรเท่านั้นที่มีสัดส่วน Retransmission สูงร้อยละ 12 โดยการทดลองนี้จะเห็นว่า TMAC ช่วยให้การส่งข้อมูลได้ร้อยละ 100 โดยมีสัดส่วนการ Retransmission ไม่เกินร้อยละ 12 ที่ระยะทาง 1-5 กิโลเมตร

สรุป

โพรโทคอล MQTT เป็นโพรโทคอลที่มีความสำคัญสำหรับการเชื่อมต่อของอุปกรณ์ IoT แต่ด้วยความต้องการใช้งานของ MQTT บนเครือข่าย LoRa มีสูงขึ้น และยังขาดโปรโทคอลในชั้น MAC ที่คอยประสานและผนวก MQTT เข้ากับ LoRa ได้อย่างมีประสิทธิภาพ โดยก่อนหน้านี้มีความพยายามพัฒนาวิธีการที่หลากหลาย แต่ก็ยังมีจุดที่ต้องปรับปรุงแก้ไข เช่น โอเวอร์เฮดของเครือข่าย ดีเลย์ ขนาดของ

แพ็กเกตมีจำนวนจำกัด ความซับซ้อนการผสาน MQTT เข้ากับ LoRa การเพิ่มชั้นเครือข่ายบน LoRa เพื่อให้สนับสนุน MQTT และการใช้ LoRa โดยไม่อาศัยชั้น MAC ซึ่งทำให้การเชื่อมต่อไม่มีความน่าเชื่อถือ งานวิจัยนี้จึงออกแบบโพรโทคอล TMAC สำหรับผสาน MQTT ให้เข้ากับ LoRa แล้วทดสอบในระบบเครือข่าย Test-bed ในระยะ 1-5 กิโลเมตรโดยใช้ขนาด Payload ที่หลากหลาย และพบว่าโพรโทคอล TMAC สามารถช่วยให้การส่งแพ็กเกตได้ร้อยละ 100 มีดีเลย์ไม่เกิน 3,500 ms ในระยะ 1-5 กิโลเมตร และมีโอ-เวอร์เฮดเครือข่ายต่ำ ซึ่ง Retransmission สูงสุดเพียงร้อยละ 12 ได้

เอกสารอ้างอิง

Ansari, N., & Zhang, J. (2013). *Media Access Control and Resource Allocation: For Next Generation Passive Optical Networks*. Springer Publishing Company, Incorporated.

Audebert, V., Catalano, J., Coracin, M., Gourrierec, M. L., Sornin, N., & Yegin, A. (2021). *Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN* (RFC No. 9011; Standards Track). IETF.

Bhawiyuga, A., Amron, K., Primanandha, R., Kartikasari, D. P., Arijudin, H., & Prabandari, D. A. (2019). LoRa-MQTT Gateway Device for Supporting Sensor-to-Cloud Data Transmission in Smart Aquaculture IoT Application. *2019 International Conference on Sustainable Information Engineering and Technology (SIET)*, 187–190. https://doi.org/10.1109/SIET48054.2019.8986124

Ferré, G., & Giremus, A. (2018). LoRa Physical Layer Principle and Performance Analysis. *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 65–68. https://doi.org/10.1109/ICECS.2018.8617880

Heile, B., Minaburo, A., PAradells, J., Perkins, C., Ponsard, B., Ratilainen, A., SUM, C.-S., Toutain, L., Yegin, A., & Zuniga, J. C. (2018). *Low-Power Wide Area Network (LPWAN) Overview* (RFC No. 8376). IETF.

HiveMQ. (2020). *MQTT & MQTT 5 Essentials* (Vol. 1). HiveMQ GmbH.

Huang, A., Huang, M., Shao, Z., Zhang, X., Wu, D., & Cao, C. (2019). A Practical Marine Wireless Sensor Network Monitoring System Based on LoRa and MQTT. *2019 IEEE 2nd International Conference on Electronics Technology (ICET)*, 330–334. https://doi.org/10.1109/ELTECH.2019.8839464

IETF. (2021). *Internet Engineering Task Force*. https://www.ietf.org/

Kietzmann, P., Alamos, J., Kutscher, D., Schmidt, T. C., & Wählisch, M. (2022). Delay-Tolerant ICN and Its Application to LoRa. *Proceedings of the 9th ACM Conference on Information-Centric Networking*, 125–136. https://doi.org/10.1145/3517212.3558081

Minaburo, A., Toutain, L., Gomez, C., Barthel, D., & Zuniga, J. C. (2020). *SCHC: Generic Framework for Static Context Header Compression and Fragmentation* (RFC No. 8724). IETF. https://datatracker.ietf.org/doc/html/rfc8724

Petrariu, A. I., Lavric, A., & Coca, E. (2019). LoRaWAN Gateway: Design, Implementation and Testing in Real Environment. *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 49–53. https://doi.org/10.1109/SIITME47687.2019.8990791

Socolofsky, T. J., & Kale, C. J. (1991). *A TCP/IP Tutorial* (RFC) [1180]. IETF.

Spinsante, S., Ciattaglia, G., Del Campo, A., Perla, D., Pigini, D., Cancellieri, G., & Gambi, E. (2017). A LoRa enabled building automation architecture based on MQTT. *2017 AEIT International Annual Conference*, 1–5. https://doi.org/10.23919/AEIT.2017.8240560

Sun, C., Zheng, F., Zhou, G., & Guo, K. (2020). Design and Implementation of Cloud-based Single-channel LoRa IIoT Gateway Using Raspberry Pi. *2020 39th Chinese Control Conference (CCC)*, 5259–5263. https://doi.org/10.23919/CCC50068.2020.9189480

Tayebi, A., Veltri, L., Zanichelli, F., & Caselli, S. (2022). Interworking between LoRaWAN and non-LoRa IoT Systems. *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops)*, 415–420. https://doi.org/10.1109/PerComWorkshops53856.2022.9767494

Vangelista, L. (2017). Frequency Shift Chirp Modulation: The LoRa Modulation. *IEEE Signal Processing Letters*, *24*(12), 1818–1821. https://doi.org/10.1109/LSP.2017.2762960