IoT/MQTT Network Simulation Wireless Protocol Performance Analysis

Course: CompE560: Computer and Data Networks

Project Group 2: IOT/MQTT

Members: Akshita Golakiya, Ruturaj Uttarwar, Omar Aviles, Haripriya Kotapati, Alden Cam, Matthew Chang, Ulises Urbina, Brandon Nguyen, Daniel Self, Rogelio Barajas

Project Overview

- Goal: Build an IoT/MQTT simulator.
- Focus: Wireless Protocol Performance Analysis.
- Features: MQTT broker with clients, GUI to show devices and metrics, and 3 main experiments.

INDUSTRY CHALLENGES

- Most IoT projects don’t succeed. about 3 out of 4 fail, usually early in testing.
- A big reason is **bad connectivity**. more than a third of companies report issues, and only a tiny fraction (around 2%) get the reliability they need.
- Poor connectivity has real consequences:
    a. Bad data → bad decisions.
    b. Reputation damage when systems don’t work.
    c. Higher costs from downtime and inefficiency.
- The main technical problems: packet loss, delays, and instability.

TARGET PROBLEMS WE SOLVE

Build a working MQTT simulation with reconnects, QoS 0/1, and session management.

Compare Wi-Fi 802.11n and BLE 5.x in terms of speed, range, and energy use.

Run the three required experiments:

- Duty cycle vs. latency and battery life (E1)
- Wi-Fi vs BLE performance comparison (E2)
- Broker failover and recovery (E3)

Provide a simple GUI that shows devices on a map, metrics in real-time, and graphs for analysis.

PROJECT CHALLENGES

Realistic Assessment of Our Challenges

Technical Challenges: • Integration Complexity

- Getting 10 people’s code to work together will be tough

• Realistic Modeling Balance

- We need to balance accuracy with time. we can’t build real radios, so we’ll use simplified models.

• Performance Scaling

- Scaling to 100+ devices and still showing results in real time will test our simulator’s efficiency.

GUARANTEED DELIVERABLES

- A working MQTT simulation with QoS 0 and 1, reconnect logic, and session management.
- A side-by-side comparison of Wi-Fi 802.11n and BLE 5.x, including range, speed, and energy use.
- All three required experiments:
    - E1: Duty cycle vs. latency and battery life
    - E2: Wi-Fi vs BLE comparison
    - E3: Broker failover and recovery
- A functional GUI that shows device placement, live metrics, and experiment results.

FEATURES WE INCLUDE

Core Implementation

- **Simplified PHY Models**: Wi-Fi and BLE profiles with speed, range, and energy from published standards. (We’re not doing complex radio wave math.)
- **MQTT Features**: QoS 0 and 1, reconnect logic, session persistence, and publish/subscribe messaging.
- **Mobility & Topology**: About 70% of devices will stay fixed, while 30% move around on a grid. We’ll also model coverage changes and basic broker failover.
- **Experiments**: We’ll study energy vs performance trade-offs and run scenarios that reflect real IoT deployments.

FEATURES WE EXCLUDE

Features We’re Not Doing

- We won’t simulate **real radio physics**. too complex for this project. Instead, we’ll use simplified models with speed, range, and energy stats.
- We won’t add **advanced Wi-Fi features** like beacons or associations. They don’t add much learning value here.
- We won’t build **multiple MQTT brokers**. just one broker with failover support.
- We won’t spend time on a **super-polished GUI**. it’ll be functional and clear, not fancy.
- We won’t implement **Zigbee**. it’s too complex with mesh networking and duty cycles. We’ll stick to Wi-Fi vs BLE since that gives us plenty to analyze.

SIMULATOR ARCHITECTURE

Simulator Architecture – How It Works

- **Energy Analysis**: Tracks battery use in real time, models Wi-Fi vs BLE power, and estimates battery life under different duty cycles.
- **Network Topology**: Simulates device placement and mobility, calculates range and signal strength, and shows connectivity on a map.
- **MQTT Management**: Handles publish/subscribe messaging, tests different QoS levels, and monitors broker queues and persistence.
- **Failure Recovery**: Simulates broker failures, measures how long reconnections take, and checks how many messages get lost or recovered.

ROUTING ALGORITHMS

Routing Algorithms – Our Strategy

- **Hierarchical Design:** IoT devices → Gateways → Broker → Cloud.
- **Distance-Based Routing:** Devices connect to the nearest gateway, with backup options if one fails.
- **MQTT Topic Routing:** Broker organizes messages by topic, ensures QoS, and handles offline devices with retained messages.
- **Why We Chose This:**
    - Matches real-world IoT setups.
    - Scales well to 100+ devices.
    - Easy to test because failures are clear and measurable.

PROJECT TIMELINE

Weeks 1–3 (Foundation): Team roles, Wi-Fi + BLE models, MQTT basics, and core simulator framework.

Weeks 4–6 (Core Implementation): Add MQTT QoS, integrate wireless protocols, basic GUI, and energy tracking.

Weeks 7–9 (Integration & Testing): Put everything together, debug, add mobility and experiments, and refine GUI.

Weeks 10–12 (Experiments & Delivery): Run all experiments, analyze results, and finish presentation + documentation.

Risk Management: Weekly integration checkpoints, fallback options if something breaks, and focus on must-have features first.

REFERENCES

[1] M. M. Rahman et al., "Failure Analysis of IoT-based Smart Agriculture System,"

IEEE STI 2023.

[2] A. Abdulhamid et al., "Quantitative failure analysis for IoT systems,"

Int. J. Syst. Assur. Eng. Manag., 2025.

[3] K.-X. Shi et al., "Fault diagnosis method for WSN nodes," Sci. Rep., 2024.

[4] S. Bi et al., "Failure Analysis in Critical Communication Infrastructures,"

arXiv, 2024.

[5] IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput, IEEE Std 802.11n-2009, Oct. 2009.