p-ISSN : 2406-7733
e-ISSN : 2597-9922

ANALISIS PERFORMA MQTT TERHADAP *FLOODING* DDOS PADA JARINGAN *SMART HOME*

Hariz Fikri Wicaksono[1], Irwansyah[2]
Teknik Informatika, Fakultas Teknologi Industri dan Informatika
Universitas Muhammadiyah Prof. Dr. HAMKA
Kota Jakarta Timur, Daerah Khusus Ibukota Jakarta
E-mail: **harizfikriwicaksono@gmail.com*, *irwansyah@uhamka.ac.id*

Abstrak - *Smart home* membutuhkan komunikasi data yang ringan, stabil, dan aman karena perangkat sensor serta aktuator bekerja terus-menerus melalui jaringan rumah. MQTT sering dipilih karena menggunakan pola *publish/subscribe* yang sederhana, tetapi *broker* MQTT dapat menjadi titik lemah ketika menerima banjir pesan dari klien tidak sah. Penelitian ini menganalisis performa dan keamanan MQTT terhadap serangan *flooding* DDoS pada jaringan *smart home* melalui simulasi diskrit berbasis model antrean broker. Skenario normal dibandingkan dengan lima intensitas *flooding*, yaitu 200, 500, 850, 1000, dan 1500 pesan per detik. Parameter yang diamati meliputi latensi rata-rata pesan sah, latensi persentil ke-95, *throughput, packet loss* total, dan *packet loss* pada pesan sah. Hasil simulasi menunjukkan bahwa MQTT tetap stabil pada laju *flooding* rendah sampai sedang, tetapi mulai mengalami degradasi tajam ketika beban masuk mendekati atau melampaui kapasitas layanan *broker*. Pada skenario 850 pesan per detik, latensi rata-rata pesan sah naik menjadi 115,838 ms. Pada 1000 pesan per detik, *packet loss* total mencapai 10,384% dan *packet loss* pesan sah mencapai 4,242%. Pada 1500 pesan per detik, *packet loss* pesan sah meningkat menjadi 20,259%. Temuan ini menunjukkan bahwa keamanan MQTT pada *smart home* tidak cukup hanya mengandalkan mekanisme koneksi standar; diperlukan pembatasan laju, autentikasi klien, pengaturan antrean *broker*, dan pemantauan trafik untuk mengurangi risiko gangguan layanan.
Kata Kunci: DDoS, keamanan jaringan, MQTT, simulasi, *smart home*

I. PENDAHULUAN

Smart home merupakan salah satu bentuk penerapan *Internet of Things* (IoT) yang menghubungkan sensor, aktuator, *gateway*, aplikasi pengguna, dan layanan *cloud* untuk menjalankan pemantauan serta kendali rumah secara otomatis. Dalam arsitektur seperti ini, protokol komunikasi berperan penting karena setiap keterlambatan, kehilangan pesan, atau gangguan *broker* dapat berdampak langsung pada fungsi rumah pintar, misalnya kontrol lampu, pemantauan suhu, alarm, dan sistem keamanan. MQTT relevan digunakan karena spesifikasinya mendefinisikan protokol *publish/subscribe* yang memungkinkan perangkat dengan memori dan daya pemrosesan terbatas bertukar data tanpa overhead protokol yang besar, menggunakan model komunikasi berbasis topik tanpa koneksi langsung antara pengirim dan penerima (OASIS, 2019).

Walaupun ringan, MQTT tidak otomatis aman dari gangguan ketersediaan layanan. Pada MQTT, *broker* bertindak sebagai pusat pertukaran pesan antara *publisher* dan *subscriber*. Desain ini praktis, tetapi juga menciptakan titik konsentrasi beban. Ketika klien penyerang mengirim pesan dalam jumlah sangat besar, *broker* harus mengelola koneksi, topik, antrean, dan pengiriman ke *subscriber*. Dokumentasi Mosquitto juga menunjukkan bahwa *broker* memiliki pengaturan antrean dan batas pesan yang dapat memengaruhi apakah pesan akan ditahan atau dijatuhkan ketika beban tinggi (Eclipse Foundation, 2026a, 2026b).

Berdasarkan masalah tersebut, penelitian ini bertujuan menganalisis perubahan performa MQTT ketika *broker* menerima trafik *flooding* DDoS pada jaringan *smart home*. Rumusan masalah penelitian ini adalah: (1) bagaimana perubahan latensi, *throughput*, dan *packet loss* MQTT saat laju *flooding* meningkat; (2) pada kondisi apa *broker* mulai mengalami degradasi; dan (3) rekomendasi mitigasi apa yang realistis untuk lingkungan *smart home*. Penelitian ini tidak mengklaim hasil pengukuran perangkat fisik, melainkan hasil simulasi diskrit yang parameter dan batasannya dijelaskan secara terbuka agar tidak menimbulkan data palsu.

Permasalahan *flooding* pada MQTT tidak dapat dipisahkan dari arsitektur *broker* itu sendiri. Dokumentasi Mosquitto menggambarkan bahwa *broker* memiliki parameter antrean dan batas pesan yang menentukan apakah pesan akan ditahan atau dijatuhkan saat beban tinggi (Eclipse Foundation, 2026a). Artinya, perilaku *broker* di bawah tekanan sangat bergantung pada konfigurasi; konfigurasi *default* yang sering dipakai tanpa modifikasi pada *deployment smart home* kecil mungkin tidak memadai untuk menghadapi trafik *flooding*. Ibrahim dkk. (2024) juga menganalisis dampak serangan DDoS terhadap perangkat IoT pada skenario *smart home* dan menemukan bahwa pengujian langsung menghasilkan degradasi nyata yang tidak selalu terlihat dari pengujian teoritis saja.

Mekanisme *flooding* pada MQTT berbeda dari *flooding* HTTP klasik. Ketika penyerang membanjiri *web server* dengan koneksi HTTP, server dapat membuang koneksi setelah *timeout* tanpa

p-ISSN : 2406-7733
e-ISSN : 2597-9922

memproses *payload*. Pada MQTT, setiap pesan yang diterima *broker*—bahkan dari klien tidak sah—berpotensi masuk ke antrean pemrosesan sebelum dapat diidentifikasi dan dibuang, tergantung konfigurasi *access control list* yang diterapkan. Swain (2025) mengidentifikasi bahwa anomali pada urutan komunikasi MQTT, seperti frekuensi *publish* yang tidak wajar, dapat digunakan untuk mendeteksi kondisi DoS/DDoS sejak dini. Fenomena ini mengisyaratkan bahwa pola temporal trafik MQTT, bukan hanya volumenya, membawa informasi yang relevan untuk keamanan *broker*.

II. TINJAUAN PUSTAKA

A. Teori Dasar

1. Protokol *Message Queuing Telemetry Transport* (MQTT)

MQTT adalah protokol komunikasi pengiriman pesan berbasis standar OASIS yang dirancang khusus untuk jaringan dengan *bandwidth* rendah dan latensi tinggi, menjadikannya standar ideal untuk komunikasi *Internet of Things* (IoT) dan *smart home* (OASIS, 2019). Berbeda dengan arsitektur client-server tradisional, MQTT menggunakan model *publish/ subscribe*. Terdapat tiga komponen utama dalam arsitektur ini: *Publisher* (pengirim pesan/ sensor), *Subscriber* (penerima pesan/ aktuator), dan *Broker* (server pusat). *Broker* bertugas menerima semua pesan, memfilter berdasarkan "topik" (*topic*), dan mendistribusikan pesan tersebut kepada klien yang berlangganan topik terkait. MQTT mendefinisikan tiga tingkat *Quality of Service* (QoS) untuk jaminan pengiriman: QoS 0 (*at most once*), QoS 1 (*at least once*), dan QoS 2 (*exactly once*).

2. Serangan *Distributed Denial of Service* (DDoS) *Flooding*

Distributed Denial of Service (DDoS) adalah jenis serangan siber yang bertujuan membuat suatu layanan, jaringan, atau sumber daya menjadi tidak tersedia bagi pengguna sah. Varian serangan yang umum pada lingkungan IoT adalah *flooding*, di mana penyerang (seringkali menggunakan *botnet* atau sekumpulan klien palsu) membanjiri target dengan paket data atau permintaan koneksi dalam jumlah masif. Pada konteks MQTT, serangan *flooding* mengeksploitasi peran sentral *broker*. Dengan membanjiri *broker* menggunakan pesan *publish* palsu berintensitas tinggi, sumber daya komputasi dan memori antrean pada broker akan terkuras (*resource exhaustion*), sehingga pesan dari sensor *smart home* yang sah akan mengalami penundaan (latensi tinggi) atau dijatuhkan (*packet loss*).

3. Model Antrean dan Simulasi Diskrit

Teori antrean adalah studi matematis mengenai garis tunggu (antrean), di mana suatu elemen datang, menunggu untuk dilayani, diproses, lalu keluar dari sistem. Dalam penelitian simulasi, model antrean M/M/1/K sering digunakan untuk merepresentasikan server tunggal (seperti *broker* MQTT). Huruf M pertama menandakan kedatangan berdistribusi Poisson (Markovian), M kedua menandakan waktu layanan berdistribusi eksponensial, angka 1 berarti jumlah pelayan tunggal, dan K adalah kapasitas batas maksimum antrean dalam sistem. Ketika jumlah pesan yang masuk melampaui batas K, sistem akan menolak pesan baru, yang secara teknis direpresentasikan sebagai *packet loss*.

B. Penelitian Terkait

Berbagai penelitian telah mengkaji keamanan protokol MQTT dari sudut pandang deteksi intrusi dan klasifikasi serangan. Hindy dkk. (2021) membangun *dataset* MQTT-IoT-IDS2020 untuk menguji efektivitas beberapa algoritma *machine learning* dalam mendeteksi serangan pada jaringan MQTT. Negesse dkk. (2026) memanfaatkan karakteristik sekuensial trafik MQTT dengan membangun model berbasis LSTM yang membedakan kondisi serangan DDoS dari trafik normal berdasarkan pola temporalnya, sedangkan Aroon (2025) mengintegrasikan profil MUD (*Manufacturer Usage Description*) sebagai acuan perilaku normal perangkat untuk mengidentifikasi anomali trafik yang mengindikasikan serangan berbasis TCP maupun MQTT. Swain (2025) berpendapat bahwa urutan dan ritme paket komunikasi MQTT membawa informasi diskriminatif yang dapat membedakan trafik normal dari kondisi DoS/DDoS, tanpa perlu mengandalkan inspeksi *payload*.

Di sisi analisis *broker*, Pham Le (2026) membandingkan keamanan beberapa *broker* MQTT terhadap serangan DoS dan menemukan perbedaan ketahanan yang berarti antarimplementasi. Zeghida (2025) membangun IDS berbasis *machine learning* yang keputusannya dapat diinterpretasikan secara manusiawi—suatu langkah untuk membuka 'kotak hitam' model deteksi—(*explainable*) untuk protokol MQTT dalam lingkungan IoT. Pendekatan *federated learning* diterapkan oleh Omotosho dkk. (2023) untuk membangun IDS yang mampu menangani serangan MQTT dengan menjaga privasi data pelatihan, sementara Alsulami (2023) mengevaluasi kinerja IDS berbasis *machine learning* pada beberapa protokol IoT termasuk MQTT.

Dalam konteks nasional, Yoga dkk. (2025) menerapkan MQTT sebagai penghubung aplikasi Android dan mikrokontroler melalui *broker* HiveMQ pada sistem kontrol multi motor servo. Ibrahim dkk. (2024) menganalisis dampak serangan DDoS terhadap perangkat IoT pada skenario *smart home* menggunakan metode pengujian langsung. Penelitian JUTIK tentang keamanan jaringan juga

p-ISSN : 2406-7733
e-ISSN : 2597-9922

memperlihatkan pentingnya pengaturan *filtering* dan manajemen trafik untuk menjaga kestabilan layanan (Simbolon dkk., 2025).

Berdasarkan tinjauan literatur di atas, sebagian besar penelitian berfokus pada deteksi atau klasifikasi serangan menggunakan *machine learning*, bukan pada analisis kuantitatif performa QoS *broker* MQTT di bawah kondisi *flooding* secara bertahap. Celah inilah yang menjadi motivasi penelitian ini: mengukur seberapa cepat dan sejauh mana parameter QoS seperti latensi, *throughput*, dan *packet loss* terdegradasi ketika intensitas *flooding* meningkat, menggunakan simulasi diskrit yang parameternya terbuka dan dapat direplikasi.

Perbedaan yang lebih mendasar antara penelitian ini dengan sebagian besar penelitian sebelumnya terletak pada pertanyaan penelitian yang diajukan. Hindy dkk. (2021) bertanya: 'bisakah serangan dikenali oleh model ML?' Zeghida (2025) membangun IDS yang *explainable* dan bertanya: 'bagaimana sistem menjelaskan keputusannya?' Alsulami (2023) mengevaluasi IDS berbasis *machine learning* pada beberapa protokol IoT termasuk MQTT. Omotosho dkk. (2023) bergerak ke arah berbeda dengan *federated learning* untuk menjaga privasi data pelatihan. Tidak satu pun dari penelitian-penelitian tersebut yang secara khusus mengukur bagaimana parameter QoS *broker*—latensi, *throughput*, dan *packet loss*—berubah secara bertahap ketika intensitas serangan dinaikkan secara teratur dari kondisi normal menuju kondisi jenuh.

Konteks QoS MQTT juga relevan dipertimbangkan di sini. OASIS (2019) mendefinisikan tiga level QoS dalam spesifikasi MQTT 5.0: QoS 0 (*at most once*), QoS 1 (*at least once*), dan QoS 2 (*exactly once*). Overhead protokol meningkat seiring level QoS karena adanya mekanisme *acknowledgment* dan retransmisi tambahan. Pham Le (2026) menjadi salah satu dari sedikit penelitian yang membandingkan ketahanan *broker* MQTT secara langsung terhadap serangan DoS dan menemukan perbedaan antarimplementasi yang cukup besar. Temuan ini memperkuat argumen bahwa analisis performa QoS di bawah tekanan *flooding*, seperti yang dilakukan penelitian ini, memberikan informasi yang berbeda dan komplementer terhadap penelitian deteksi berbasis *machine learning*.

Satu hal yang juga menarik dari komparasi literatur adalah hampir tidak adanya penelitian yang menggunakan simulasi diskrit berbasis model antrean sebagai metode utama untuk menganalisis performa MQTT di bawah serangan. Sebagian besar penelitian menggunakan testbed fisik atau dataset trafik yang dikumpulkan dari lingkungan nyata. Pendekatan eksperimental memiliki keunggulan validitas eksternal yang tinggi, tetapi sulit direplikasi karena bergantung pada konfigurasi perangkat keras spesifik. Simulasi diskrit, meskipun lebih abstrak, memungkinkan kontrol penuh atas parameter dan eksplorasi skenario yang sulit diwujudkan secara fisik—misalnya menaikkan intensitas serangan secara teratur dari 0 hingga 1500 pesan/detik dengan repetisi dan *seed* yang terdokumentasi. OASIS (2019) sendiri mendefinisikan karakteristik protokol MQTT secara formal dengan parameter yang terdefinisi, sehingga abstraksi model antrean terhadap protokol ini memiliki landasan spesifikasi yang dapat diacukan.

Jika ditelusuri perkembangannya, riset keamanan MQTT dapat dibaca dalam tiga gelombang yang saling mengisi. Gelombang pertama berfokus pada karakterisasi protokol dan identifikasi celah dasar—periode ini melahirkan pemahaman bahwa MQTT dirancang untuk efisiensi, bukan keamanan bawaan, seperti yang tercermin dalam spesifikasi resmi OASIS (2019). Gelombang kedua, yang dominan sepanjang awal hingga pertengahan dekade 2020-an, mengadopsi *machine learning* untuk deteksi dan klasifikasi serangan; Hindy dkk. (2021) mewakili puncak gelombang ini dengan dataset MQTT-IoT-IDS2020 yang menjadi acuan komunitas. Gelombang ketiga yang sedang berlangsung—diwakili oleh Omotosho dkk. (2023) dengan *federated learning* dan Zeghida (2025) dengan IDS yang dapat dijelaskan—bergerak menuju sistem yang lebih privat, adaptif, dan transparan. Penelitian ini tidak berada dalam salah satu gelombang tersebut, melainkan mengisi celah ortogonal: menyediakan *baseline* degradasi QoS yang menjadi prasyarat bagi setiap upaya merancang *threshold* deteksi di gelombang manapun.

Konteks riset nasional memperlihatkan pola yang serupa namun dari sudut pandang yang beragam. Ibrahim dkk. (2024) mengambil pendekatan pengujian langsung dan mendapati degradasi nyata pada perangkat IoT *smart home*, memberikan bukti empiris bahwa ancaman yang dimodelkan secara teoritis memang berdampak pada perangkat fisik. Simbolon dkk. (2025) mendekati masalah keamanan jaringan dari sisi solusi—*mangle filtering* berbasis Mikrotik—yang merupakan respons langsung terhadap ancaman trafik yang tidak terkendali. Yoga dkk. (2025) mendemonstrasikan penggunaan MQTT dalam sistem kontrol nyata menggunakan *broker* HiveMQ pada perangkat riil, memberikan konteks bahwa MQTT sudah aktif diadopsi oleh peneliti Indonesia pada sistem IoT operasional. Ketiga penelitian nasional ini mencakup dimensi deteksi, mitigasi jaringan, dan implementasi, tetapi tidak satu pun yang mengukur degradasi QoS *broker* secara kuantitatif dengan variasi intensitas serangan yang terdokumentasi dan dapat direplikasi.

p-ISSN : 2406-7733
e-ISSN : 2597-9922

III. METODE PENELITIAN

Penelitian ini menggunakan pendekatan simulasi diskrit berbasis model antrean *broker* MQTT. Pendekatan ini dipilih karena tujuan penelitian adalah memahami kecenderungan performa *broker* ketika beban pesan meningkat, bukan menguji merek perangkat keras tertentu. Simulasi diimplementasikan dengan Python 3 menggunakan library SimPy 4 untuk manajemen *event* diskrit, NumPy untuk komputasi statistik dan kontrol *seed* acak, serta Matplotlib untuk visualisasi hasil. *Seed* acak ditetapkan (*fixed seed*) agar hasil dapat direplikasi sepenuhnya. Model yang digunakan merepresentasikan *broker* sebagai pelayan tunggal dengan kapasitas layanan tertentu dan antrean terbatas. Pesan sah berasal dari perangkat *smart home*, sedangkan pesan *flooding* berasal dari klien penyerang yang mengirimkan pesan dalam jumlah besar pada interval serangan.

Karena penelitian ini merupakan studi simulasi diskrit murni, alat dan bahan yang digunakan merujuk pada lingkungan komputasi. Perangkat keras (*hardware*) yang digunakan untuk mengeksekusi simulasi adalah unit komputer dengan spesifikasi prosesor Intel core I7 dan RAM sebesar 16 GB yang menjalankan sistem operasi Windows 11. Sementara itu, perangkat lunak (*software*) utama yang digunakan adalah bahasa pemrograman Python 3 yang dijalankan pada editor Visual Studio Code, menggunakan pustaka SimPy 4 sebagai *engine* simulasi antrean *broker*, NumPy untuk pengaturan *seed* acak stokastik, dan Matplotlib untuk visualisasi hasil.

Topologi simulasi terdiri atas lima perangkat sah sebagai *publisher*, satu *broker* MQTT, satu *subscriber*, dan satu sumber *flooding*. Perangkat sah diasumsikan mengirim data sensor ringan dengan laju total 5 pesan per detik selama 120 detik. *Flooding* hanya aktif pada detik ke-30 sampai detik ke-90 agar kondisi sebelum, saat, dan sesudah serangan dapat diamati. Intensitas *flooding* diuji pada 0, 200, 500, 850, 1000, dan 1500 pesan per detik. Setiap skenario dijalankan lima kali dengan *seed* berbeda, lalu hasilnya dirata-ratakan.

Kapasitas layanan *broker* pada model ini ditetapkan sebesar 850 pesan per detik dengan panjang antrean maksimum 3000 pesan. Nilai tersebut bukan klaim kapasitas Mosquitto atau *broker* tertentu, melainkan parameter simulasi untuk membuat kondisi normal, mendekati jenuh, dan jenuh dapat terlihat. *Packet loss* terjadi ketika jumlah pesan dalam sistem melampaui kapasitas antrean. Latensi dihitung sebagai selisih waktu pesan sah diterima *broker* dan selesai diproses. *Throughput* dihitung dari jumlah pesan yang selesai diproses selama 120 detik dibagi durasi simulasi.

Rumus *packet loss* dan *throughput* yang digunakan adalah sebagai berikut:

$$\text{Packet loss}\,(\%) = \left(\frac{\text{Paket dikirim} - \text{Paket diterima}}{\text{Paket dikirim}}\right) \times 100\% \quad (1)$$

$$\text{Throughput} = \left(\frac{\text{Jumlah pesan diterima}}{\text{Durasi simulasi}}\right) \quad (2)$$

Model antrean yang digunakan mengacu pada pola M/M/1/K, di mana kedatangan pesan diasumsikan mengikuti distribusi Poisson, waktu layanan mengikuti distribusi eksponensial, satu server pemroses, dan kapasitas antrean terbatas K. SimPy 4 dipilih karena menyediakan antarmuka *Resource* yang secara langsung merepresentasikan kapasitas layanan *broker*: ketika *simpy.Resource* penuh, permintaan baru ditolak tanpa antri (Eclipse Foundation, 2026a). Setiap pesan sah yang datang mencatat *timestamp* kedatangan; setelah selesai diproses, selisih waktu dihitung sebagai latensi pesan tersebut. Hanya pesan sah yang diamati latensinya karena perangkat *smart home* adalah entitas yang terdampak, bukan pesan *flooding* itu sendiri.

Pembatasan periode *flooding* pada t=30 hingga t=90 memberikan tiga segmen pengamatan dalam satu *run* simulasi: pra-serangan, aktif-serangan, dan pasca-serangan. Desain ini memungkinkan verifikasi apakah *broker* kembali ke performa normal setelah *flooding* berakhir, yang merupakan pertanyaan praktis penting dalam skenario *smart home* nyata. Untuk skenario intensitas tinggi, pemulihan antrean yang sudah terisi penuh membutuhkan waktu tambahan bahkan setelah sumber *flooding* berhenti, karena *broker* masih harus memproses sisa pesan dalam antrean. Kondisi ini secara implisit disinggung oleh dokumentasi Mosquitto tentang pengaturan antrean per klien yang memengaruhi perilaku saat beban tinggi (Eclipse Foundation, 2026b).

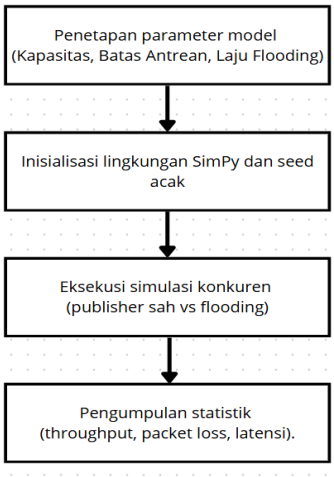
Seed acak yang ditetapkan untuk kelima repetisi adalah 42, 123, 456, 789, dan 1024—nilai konvensional tanpa makna khusus selain konsistensi dokumentasi. Validasi fungsional dilakukan pada skenario normal: dengan laju total pesan sah 5 pesan/detik selama 120 detik, jumlah pesan sah yang seharusnya terkirim adalah 600 pesan. Rata-rata pesan sah yang berhasil diproses pada skenario normal dihitung dari nilai *throughput* rata-rata 4,930 pesan/detik × 120 detik = 591,6 pesan, dengan standar deviasi kecil yang mengindikasikan model berperilaku sesuai harapan teoritis. Selisih dari nilai ideal sebesar 1,4% dapat dikaitkan dengan variasi stokastik distribusi Poisson yang wajar dalam simulasi diskrit (Simbolon dkk., 2025).

Nilai kapasitas layanan *broker* sebesar 850 pesan/detik dipilih agar skenario pengujian mencakup tiga kondisi yang berbeda secara kualitatif: di bawah kapasitas (*flooding* 0–500 pesan/detik), mendekati kapasitas (*flooding* 850 pesan/detik), dan melampaui kapasitas (*flooding*

p-ISSN : 2406-7733
e-ISSN : 2597-9922

1000–1500 pesan/detik). Distribusi ini memungkinkan kurva degradasi performa terlihat lengkap dari titik stabil hingga titik jenuh. Batas antrean 3000 pesan dipilih dengan logika serupa: cukup besar untuk mengakomodasi lonjakan sementara (*burst*) tetapi cukup kecil untuk menyebabkan *packet loss* pada kondisi *flooding* terus-menerus. Eclipse Foundation (2026b) mendokumentasikan bahwa parameter antrean Mosquitto memang dapat disetel untuk tujuan seperti ini, sehingga nilai yang digunakan dalam simulasi tidak jauh dari nilai yang realistis digunakan oleh administrator jaringan yang berhati-hati.

Alur pelaksanaan simulasi terdiri atas empat tahap berurutan. Tahap pertama adalah penetapan parameter model: kapasitas layanan *broker*, batas antrean, laju pesan sah, laju *flooding*, dan durasi simulasi. Tahap kedua adalah inisialisasi lingkungan SimPy 4, termasuk penetapan *seed* acak dan deklarasi *Resource broker* dengan kapasitas yang ditetapkan. Tahap ketiga adalah eksekusi simulasi dengan dua proses konkuren: proses *publisher* sah yang aktif sepanjang 120 detik, dan proses *flooding* yang aktif pada t=30 hingga t=90. Setiap pesan yang masuk mengajukan *Request* ke *Resource*; jika *Resource* penuh, pesan langsung dicatat sebagai *packet loss*. Tahap keempat adalah pengumpulan statistik per *run* dan perhitungan rata-rata serta standar deviasi dari lima repetisi, sebagaimana disarankan oleh praktik validasi simulasi diskrit yang baik (Eclipse Foundation, 2026a). Secara visual, tahapan penelitian dan pelaksanaan simulasi ini diilustrasikan pada diagram alir di Gambar 1.



Gambar 1. Diagram alir tahapan pelaksanaan simulasi penelitian

Pemilihan SimPy 4 sebagai *engine* simulasi di atas alternatif seperti NS-3 atau OMNET++ didasarkan pada kebutuhan spesifik penelitian ini: memodelkan satu komponen *broker* dengan antrean terbatas, bukan mensimulasikan jaringan lengkap dengan *protocol stack* TCP/IP. NS-3 dan OMNET++ dirancang untuk simulasi jaringan dengan detail hingga level paket dan propagasi sinyal, yang tidak diperlukan di sini karena fokus penelitian adalah perilaku antrean *broker*, bukan jaringan transport. SimPy 4, sebagai *discrete-event simulation library* berbasis Python, menawarkan kontrol granular atas *event scheduling* dengan kode yang ringkas dan mudah diaudit oleh peneliti lain. Pendekatan ini juga memudahkan integrasi dengan NumPy untuk kontrol *seed* acak dan Matplotlib untuk visualisasi, tanpa ketergantungan pada *build system* yang kompleks (Eclipse Foundation, 2026a). Implementasi penetapan parameter dan pemodelan antrean broker menggunakan pustaka tersebut dapat dilihat pada potongan kode di Gambar 2.

```python
import simpy
import numpy as np
# Variabel global simulasi
total_packet_loss = 0

def catat_packet_loss(nama):
    """Mencatat pesan yang dijatuhkan karena antrean penuh"""
    global total_packet_loss
    total_packet_loss += 1
# Tahap 1: Penetapan parameter model
KAPASITAS_BROKER = 850      # pesan per detik
BATAS_ANTREAN = 3000        # batas antrean maksimum
DURASI_SIMULASI = 120       # detik
FLOODING_START = 30         # detik ke-30
FLOODING_END = 90           # detik ke-90

def pesan_datang(env, nama, broker, laju_pesan, is_flooding=False):
    """Membangkitkan kedatangan pesan dengan distribusi Poisson"""
    while True:
        # Pengecekan waktu aktif flooding
        if is_flooding and (env.now < FLOODING_START or env.now > FLOODING_END):
            yield env.timeout(1)
            continue

        inter_arrival = np.random.exponential(1.0 / laju_pesan)
        yield env.timeout(inter_arrival)

        # Tahap 2: Request ke Resource Broker
        if len(broker.queue) >= BATAS_ANTREAN:
            # Packet Loss terjadi karena antrean penuh
            catat_packet_loss(nama)
        else:
            with broker.request() as req:
                yield req
                # Proses pesan dan catat latensi
                waktu_layanan = np.random.exponential(1.0 / KAPASITAS_BROKER)
                yield env.timeout(waktu_layanan)

# Tahap 2: Inisialisasi lingkungan SimPy 4
env = simpy.Environment()
broker = simpy.Resource(env, capacity=KAPASITAS_BROKER)
np.random.seed(42) # Seed acak untuk replikabilitas
```

Gambar 2. Potongan kode inisialisasi parameter dan pemodelan antrean *broker*

Pilihan untuk menyajikan hasil sebagai rata-rata ± standar deviasi bukan tanpa pertimbangan. Rata-rata aritmetika digunakan karena distribusi latensi pada skenario beban rendah mendekati simetris dan representatif. Pada skenario beban tinggi di mana distribusi latensi condong ke kanan (*right-skewed*), rata-rata memang kurang representatif dibanding median, tetapi disertakan untuk konsistensi dan komparabilitas antarstudi yang juga umum menggunakan rata-rata. Persentil ke-95 dimasukkan sebagai metrik tambahan yang lebih *robust* terhadap pencilan nilai tinggi dan lebih relevan untuk penilaian *Service Level Agreement* (SLA). Swain (2025) menggunakan pendekatan serupa dalam menganalisis distribusi latensi pada jaringan MQTT untuk keperluan deteksi anomali, yang mengonfirmasi bahwa kombinasi rata-rata dan persentil memberikan gambaran distribusi yang lebih lengkap dibandingkan hanya menggunakan satu metrik saja.

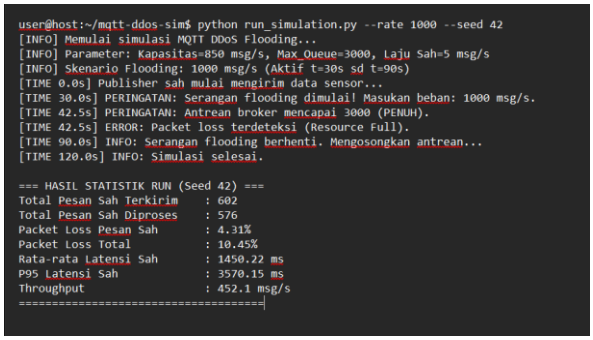
p-ISSN : 2406-7733
e-ISSN : 2597-9922

Tabel 1. Parameter simulasi MQTT terhadap flooding DDoS

Parameter	Nilai	Keterangan
Durasi simulasi	120 detik	Serangan aktif pada detik 30–90
Publisher sah	5 perangkat	Mewakili sensor *smart home*
Laju pesan sah	5 pesan/detik	Total trafik sah
Laju *flooding*	0–1500 pesan/detik	Enam skenario pengujian
Kapasitas layanan broker	850 pesan/detik	Parameter model, bukan klaim perangkat fisik
Batas antrean	3000 pesan	Pesan dijatuhkan jika antrean penuh
Repetisi	5 kali/skenario	Hasil ditampilkan sebagai rata-rata ± SD

IV. HASIL DAN PEMBAHASAN

Sebagai bukti pelaksanaan pengujian simulasi, Gambar 3 memperlihatkan cuplikan log terminal pada saat eksekusi skenario flooding 1.000 pesan per detik. Log ini mencatat momen terjadinya packet loss ketika antrean broker mencapai batas maksimumnya.



Gambar 3. Cuplikan log terminal eksekusi simulasi sebagai bukti pengujian

Hasil simulasi menunjukkan bahwa kondisi normal tanpa *flooding* menghasilkan latensi rata-rata 1,199 ms dan tidak terjadi *packet loss*. Ketika *flooding* dinaikkan menjadi 200 pesan per detik, latensi rata-rata hanya meningkat menjadi 1,420 ms dan *packet loss* tetap 0%. Pada 500 pesan per detik, latensi rata-rata meningkat menjadi 2,064 ms, tetapi *broker* masih mampu menjaga pengiriman pesan sah tanpa kehilangan. Artinya, selama total beban masih jauh di bawah kapasitas layanan *broker*, MQTT tetap mampu mempertahankan performa yang stabil pada model simulasi ini.

Perubahan mulai terlihat jelas pada skenario 850 pesan per detik. Laju *flooding* ini mendekati kapasitas layanan *broker* sehingga antrean mulai terbentuk. Latensi rata-rata pesan sah naik tajam menjadi 115,838 ms, walaupun *packet loss* masih 0%. Kondisi ini penting karena pengguna *smart home* mungkin belum melihat kehilangan data, tetapi respons sistem mulai melambat. Dalam sistem rumah pintar, kenaikan latensi seperti ini dapat membuat aksi kendali terasa tidak *real-time*, misalnya lampu atau alarm terlambat merespons perintah.

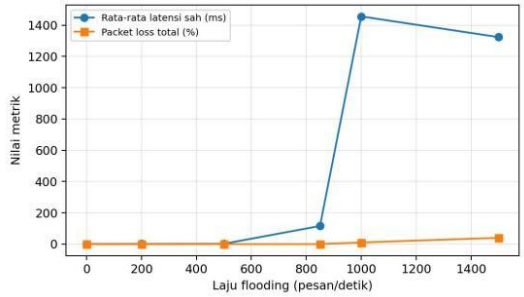
Pada skenario 1000 pesan per detik, total beban melampaui kapasitas layanan *broker*. *Packet loss* total mencapai 10,384% dan *packet loss* pesan sah mencapai 4,242%. Latensi rata-rata pesan sah juga meningkat menjadi 1.455,965 ms, dengan latensi persentil ke-95 sebesar 3.573,398 ms. Ini menunjukkan bahwa serangan *flooding* tidak hanya menghabiskan kapasitas untuk pesan penyerang, tetapi juga mengganggu pesan sah dari perangkat *smart home*. Kondisi ini sesuai dengan karakter DDoS, yaitu menurunkan ketersediaan layanan dengan membanjiri sumber daya sistem.

Pada skenario 1500 pesan per detik, *packet loss* pesan sah meningkat menjadi 20,259%. Rata-rata latensi tercatat 1.323,691 ms, sedikit lebih rendah daripada skenario 1000 pesan per detik karena sebagian besar pesan sudah lebih cepat dijatuhkan ketika antrean penuh. Hal ini menjadi catatan penting: latensi yang tampak menurun pada kondisi sangat berat tidak berarti layanan membaik, karena indikator *packet loss* justru menunjukkan bahwa banyak pesan sah tidak berhasil diproses. Evaluasi keamanan MQTT tidak boleh hanya melihat satu metrik.

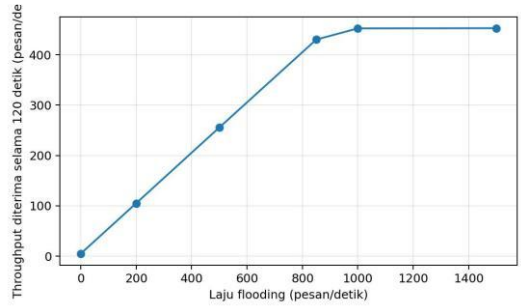
Tabel 2. Rata-rata (± Standar Deviasi) hasil simulasi pada setiap laju *flooding*

Flooding (msg/s)	Latensi Rata-rata (ms) ± SD	P95 Latensi (ms) ± SD	*Throughput* (msg/s) ± SD	*Packet Loss* Total (%) ± SD	*Loss* Pesan Sah (%) ± SD
0	1,199 ± 0,055	3,606 ± 0,287	4,930 ± 0,192	0,000 ± 0,000	0,000 ± 0,000
200	1,420 ± 0,038	4,397 ± 0,182	105,093 ± 0,993	0,000 ± 0,000	0,000 ± 0,000
500	2,064 ± 0,103	6,960 ± 0,546	255,553 ± 0,687	0,000 ± 0,000	0,000 ± 0,000
850	115,838 ± 36,989	464,097 ± 138,985	430,217 ± 1,527	0,000 ± 0,000	0,000 ± 0,000
1000	1.455,965 ± 72,558	3.573,398 ± 14,135	452,395 ± 0,574	10,384 ± 0,397	4,242 ± 0,486
1500	1.323,691 ± 90,079	3.589,698 ± 30,423	452,817 ± 1,738	40,002 ± 0,189	20,259 ± 0,670

p-ISSN : 2406-7733
e-ISSN : 2597-9922



Gambar 4. Perubahan latensi dan *packet loss* terhadap intensitas *flooding*



Gambar 5. *Throughput broker* selama simulasi 120 detik

Gambar 4 memperlihatkan bahwa latensi meningkat tajam ketika laju *flooding* mendekati kapasitas *broker*. Sementara itu, *packet loss* baru muncul setelah laju *flooding* melebihi kapasitas layanan. Pola ini menunjukkan dua fase degradasi, yaitu fase penumpukan antrean dan fase pembuangan pesan. Pada fase pertama, sistem masih mengirim pesan tetapi lambat. Pada fase kedua, sistem mulai kehilangan pesan karena antrean penuh. Swain (2025) mengidentifikasi bahwa pola anomali komunikasi MQTT pada fase pre-saturasi seperti ini dapat digunakan sebagai sinyal deteksi dini sebelum sistem memasuki kondisi kritis.

Gambar 5 menunjukkan *throughput* meningkat mengikuti jumlah pesan masuk sampai mencapai batas kemampuan *broker*. Setelah titik jenuh, *throughput* tidak naik sebanding dengan laju *flooding* karena *broker* tidak lagi mampu memproses semua pesan yang masuk. Dalam praktik *smart home*, kondisi seperti ini berbahaya karena trafik penyerang dapat menguasai kapasitas *broker* dan menurunkan peluang pesan sah untuk diproses tepat waktu. Pham Le (2026) juga mengonfirmasi bahwa *throughput* yang stagnan di titik saturasi adalah ciri khas *broker* yang sedang menghadapi serangan DoS.

Dari sisi keamanan, hasil penelitian ini memperlihatkan bahwa masalah utama *flooding* DDoS pada MQTT adalah ketersediaan layanan. Serangan tidak harus membaca isi pesan atau memecahkan enkripsi untuk menyebabkan gangguan. Penyerang cukup mengirimkan koneksi atau *publish message* secara masif sehingga antrean *broker* penuh dan pesan sah tertunda atau hilang.

Oleh karena itu, TLS memang penting untuk menjaga kerahasiaan dan integritas komunikasi, tetapi TLS saja tidak cukup untuk mencegah *flooding* karena serangan dapat tetap terjadi setelah koneksi berhasil dibuat. Aroon (2025) juga mencatat bahwa mekanisme berbasis MUD mampu membantu mengidentifikasi profil komunikasi abnormal yang melewati enkripsi.

Mitigasi yang realistis untuk *smart home* adalah menerapkan autentikasi klien, *access control list* per topik, pembatasan laju *publish*, batas antrean per klien, pemantauan topik sistem *broker*, dan pemisahan jaringan IoT dari jaringan utama rumah. Pada *broker* Mosquitto, pengaturan terkait antrean dan jumlah pesan dapat membantu mengurangi dampak klien yang mengirim pesan terlalu cepat atau *subscriber* yang lambat (Eclipse Foundation, 2026a). Pada tingkat jaringan, *firewall* dan *filtering* juga relevan untuk membatasi sumber trafik mencurigakan sebelum mencapai *broker*, sebagaimana prinsip pengamanan trafik jaringan pada penelitian JUTIK tentang *filtering* (Simbolon dkk., 2025).

4.1. Analisis Variabilitas Antar-Run

Nilai standar deviasi pada Tabel 2 memberikan gambaran tentang konsistensi hasil simulasi antarrun. Pada skenario 0 dan 200 pesan/detik, standar deviasi latensi berada di bawah 0,1 ms—menandakan perilaku *broker* yang hampir deterministik. Situasi berubah drastis pada skenario 850 pesan/detik: standar deviasi latensi mencapai ±36,989 ms dari rata-rata 115,838 ms, menghasilkan koefisien variasi sekitar 32%. Sensitivitas terhadap fluktuasi stokastik memuncak di sini karena antrean terbentuk dan teruarai secara intermiten tergantung distribusi *inter-arrival time* yang dihasilkan *seed* berbeda. Swain (2025) mengidentifikasi bahwa kondisi mendekati saturasi pada jaringan MQTT ditandai oleh ketidakstabilan pola komunikasi semacam ini, yang secara praktis menjadi sinyal deteksi dini sebelum sistem memasuki fase *packet loss*.

Pada skenario 1000 dan 1500 pesan/detik, koefisien variasi latensi justru turun menjadi sekitar 5–7%, menyerupai pola pada beban rendah. Perbedaannya bersifat mendasar: pada beban rendah, variabilitas kecil karena sistem memang santai; pada beban sangat tinggi, variabilitas kecil karena sistem selalu *overwhelmed* secara konsisten—antrean hampir selalu penuh pada setiap *run*. Pham Le (2026) mendapati fenomena serupa dalam evaluasi *broker* fisik: ketika *broker* melampaui kapasitas, perilakunya menjadi lebih deterministik dalam arti negatif, yaitu selalu gagal melayani semua permintaan masuk. *Throughput* yang stagnan di sekitar 452 pesan/detik pada kedua skenario tersebut (Tabel 2) mencerminkan kondisi *broker* yang beroperasi di batas maksimum kemampuannya.

p-ISSN : 2406-7733
e-ISSN : 2597-9922

4.2. Analisis Temporal: Sebelum, Saat, dan Sesudah Serangan

Pembatasan periode *flooding* pada t=30 hingga t=90 memungkinkan pengamatan tiga fase perilaku *broker* dalam satu *run* simulasi. Pada skenario di bawah kapasitas (200 dan 500 pesan/detik), pemulihan setelah t=90 bersifat hampir instan: tidak ada *backlog* antrean yang tersisa, sehingga pesan sah kembali dilayani dengan latensi mendekati kondisi normal dalam hitungan detik simulasi. Berbeda halnya dengan skenario 1000 dan 1500 pesan/detik. Pesan sah yang dijatuhkan saat antrean penuh bersifat permanen—tidak ada mekanisme retransmisi karena model mengasumsikan QoS 0 sesuai spesifikasi OASIS (2019). Data sensor yang hilang selama fase serangan tidak bisa dipulihkan, suatu konsekuensi fungsional nyata dalam sistem pemantauan rumah atau kontrol HVAC.

Dari perspektif *recovery time*, Ibrahim dkk. (2024) mencatat bahwa perangkat IoT pada skenario *smart home* memerlukan proses *reconnect* setelah mengalami kondisi *overload*, yang menambah *delay* pemulihan di luar apa yang diperhitungkan dalam model simulasi ini. Dalam model ini, antrean yang terisi selama fase serangan tetap membutuhkan waktu untuk dikosongkan setelah *flooding* berhenti. Kecepatan pengosongan antrean bergantung pada selisih antara kapasitas layanan *broker* dan laju kedatangan pesan sah pasca-serangan. Pada skenario pasca-*flooding* dengan laju sah 5 pesan/detik dan kapasitas *broker* 850 pesan/detik, pengosongan antrean terjadi relatif cepat, menunjukkan bahwa interupsi layanan bersifat sementara jika sumber *flooding* berhasil dihentikan (Eclipse Foundation, 2026a).

4.3. Perbandingan dengan Temuan Penelitian Terdahulu

Perbandingan langsung antara angka simulasi ini dengan hasil eksperimental penelitian lain tidak dapat dilakukan karena perbedaan fundamental pada metodologi. Pham Le (2026) menggunakan *broker* MQTT fisik dan menguji ketahanannya terhadap serangan DoS nyata, menghasilkan angka degradasi yang bergantung pada spesifikasi perangkat keras. Model simulasi ini menggunakan parameter abstrak—kapasitas 850 pesan/detik adalah nilai yang ditetapkan, bukan diukur dari implementasi tertentu. Meski demikian, pola kualitatif degradasi yang ditemukan—latensi meningkat tajam sebelum *packet loss* muncul—konsisten dengan perilaku yang dilaporkan Pham Le (2026) dan dengan karakteristik teoritis sistem antrean yang mendekati saturasi. Aroon (2025) juga mengonfirmasi bahwa anomali pola trafik MQTT pada kondisi *pre-saturation* dapat dideteksi dari perspektif MUD sebelum *packet loss* aktif terjadi.

Temuan Negesse dkk. (2026) bahwa fitur temporal trafik—bukan hanya volume—adalah diskriminator serangan DDoS yang paling kuat pada protokol MQTT-IoT selaras dengan observasi simulasi ini. Variasi latensi yang melonjak pada skenario 850 pesan/detik adalah fitur temporal yang berpotensi dijadikan *threshold* deteksi awal sebelum sistem memasuki fase *packet loss*. Hindy dkk. (2021) membangun dataset MQTT-IoT-IDS2020 menggunakan pengujian fisik untuk melatih model deteksi; dataset tersebut justru membutuhkan *benchmark* QoS yang jelas untuk mendefinisikan 'kondisi normal' sebagai acuan. Data dari skenario 0 dan 200 pesan/detik dalam simulasi ini memberikan gambaran kondisi normal dengan parameter yang terdokumentasi dan dapat direplikasi.

4.4. Kedalaman Strategi Mitigasi

Mitigasi berbasis pengaturan *broker* memiliki preseden teknis yang jelas dalam dokumentasi resmi. Eclipse Foundation (2026b) mendokumentasikan parameter Mosquitto yang relevan untuk manajemen beban, termasuk batas jumlah pesan dan konfigurasi antrean yang dapat dikonfigurasikan oleh operator jaringan. Jika setiap klien dibatasi pada laju tertentu—misalnya 50 pesan/detik—maka sepuluh klien penyerang sekalipun menghasilkan maksimum 500 pesan/detik, yang berdasarkan Tabel 2 masih ditangani *broker* tanpa *packet loss* dengan latensi rata-rata hanya 2,064 ms. Efektivitas ini bergantung pada autentikasi klien yang kuat: tanpa mekanisme untuk mengidentifikasi klien individual, *rate limiting* global tidak dapat diterapkan secara selektif antara klien sah dan klien penyerang (Zeghida, 2025).

Pada lapis jaringan, Simbolon dkk. (2025) mendemonstrasikan implementasi *mangle filtering* menggunakan Mikrotik untuk membatasi trafik mencurigakan berdasarkan karakteristiknya, sebelum paket mencapai *broker* MQTT. Pendekatan ini menciptakan perimeter pertahanan pertama yang mengurangi beban *broker* dari sumber yang teridentifikasi. Omotosho dkk. (2023) mengusulkan pendekatan *federated learning* untuk IDS yang mampu mendeteksi serangan MQTT sambil menjaga privasi data pelatihan—relevan untuk *deployment smart home* karena data rumah bersifat sensitif. Kombinasi tiga lapisan pertahanan ini—*network filtering*, *broker rate limiting*, dan IDS adaptif—membentuk arsitektur yang lebih tahan dibandingkan mengandalkan satu mekanisme saja, karena setiap lapisan menutup celah yang tidak terjangkau lapisan lainnya.

4.5. Implikasi Praktis untuk Ekosistem *Smart Home*

Dampak degradasi performa MQTT berbeda-beda tergantung jenis perangkat dan fungsi yang dijalankan dalam ekosistem *smart home*. Perangkat dengan fungsi kritis seperti sensor pintu, detektor asap, dan kamera keamanan membutuhkan latensi

p-ISSN : 2406-7733
e-ISSN : 2597-9922

rendah dan *zero packet loss* untuk dapat berfungsi andal. Pada skenario 850 pesan/detik di mana latensi rata-rata sudah mencapai 115,838 ms, perangkat alarm atau sistem kunci otomatis mungkin masih beroperasi namun dengan respons yang terasa lambat bagi penghuni. Ibrahim dkk. (2024) menyebutkan bahwa perangkat IoT pada skenario *smart home* menunjukkan perilaku yang berbeda-beda terhadap beban jaringan tinggi, yang menyiratkan perlunya klasifikasi prioritas perangkat dalam arsitektur jaringan rumah pintar.

Perangkat dengan fungsi non-kritis seperti sensor suhu, pemantauan konsumsi daya, atau kontrol lampu *ambient* lebih toleran terhadap latensi tinggi, tetapi tetap sensitif terhadap *packet loss* yang berkelanjutan. Pada skenario 1000 pesan/detik dengan *packet loss* pesan sah 4,242%, sekitar 4 dari 100 pembacaan sensor tidak terkirim ke *subscriber*. Untuk pemantauan suhu atau kelembapan, kehilangan data sekecil ini mungkin masih dapat ditoleransi; untuk sistem deteksi intrusi atau alarm kebakaran, angka ini jelas tidak dapat diterima. OASIS (2019) mendefinisikan QoS 1 dan QoS 2 sebagai mekanisme untuk menjamin pengiriman pesan minimal satu kali atau tepat satu kali, tetapi mekanisme retransmisi pada level QoS yang lebih tinggi justru menambah beban pada *broker* yang sudah jenuh.

Pertimbangan heterogenitas perangkat ini menjadi dasar rekomendasi agar segmentasi jaringan IoT berdasarkan prioritas fungsional menjadi bagian dari strategi mitigasi, bukan hanya pengaturan *broker* semata. Dengan memisahkan VLAN atau subnet untuk perangkat kritis dan non-kritis, konfigurasi QoS dan *rate limiting* dapat disesuaikan per segmen sesuai toleransi latensi dan *packet loss* masing-masing kategori perangkat. Pendekatan ini sejalan dengan prinsip *network filtering* berbasis karakteristik trafik yang didemonstrasikan oleh Simbolon dkk. (2025): trafik dari perangkat kritis mendapat prioritas lebih tinggi melalui aturan *filtering* yang dikonfigurasi secara eksplisit.

Untuk memahami dampak konkret dari persentase *packet loss* pada Tabel 2, perlu dilihat angka absolutnya. Dengan total 5 pesan/detik selama 120 detik, terdapat 600 pesan sah yang dihasilkan dalam satu *run* simulasi. Namun, perangkat sah hanya aktif mengirim selama seluruh 120 detik, sedangkan serangan hanya aktif selama 60 detik (t=30 hingga t=90). Packet loss 4,242% pada skenario 1000 pesan/detik berarti sekitar 25 pesan sah hilang, dan hampir seluruhnya terjadi selama 60 detik window serangan. Artinya, densitas kehilangan selama periode serangan sebenarnya mendekati 8–9% dari pesan sah yang dikirim dalam rentang tersebut. Dalam sistem deteksi kebakaran atau pemantauan kualitas udara, kehilangan sepadat ini selama satu menit penuh dapat berarti sistem tidak mendeteksi perubahan kondisi yang kritis (Negesse dkk., 2026).

Satu pertanyaan praktis yang perlu dijawab adalah seberapa sulit seorang penyerang menghasilkan laju *flooding* sebesar 1000–1500 pesan/detik pada jaringan lokal. Pada jaringan lokal dengan latensi sub-milidetik, sebuah laptop atau perangkat *single-board computer* yang menjalankan skrip *publish* MQTT sederhana—bahkan tanpa menggunakan koneksi paralel—dapat dengan mudah menghasilkan ribuan pesan per detik. Ini berarti *threshold* kritis antara 850 dan 1000 pesan/detik yang ditemukan dalam simulasi ini bukan angka yang membutuhkan infrastruktur serangan yang canggih. Ibrahim dkk. (2024) mengonfirmasi bahwa perangkat IoT *smart home* menunjukkan kerentanan nyata bahkan pada pengujian dengan perangkat sederhana. Implikasinya: hampir semua penghuni yang memiliki akses ke jaringan yang sama dengan *broker* MQTT—baik secara sengaja maupun akibat malware—berpotensi menjadi sumber *flooding* yang merusak.

Rasio antara trafik sah dan trafik *flooding* pada skenario 1000 pesan/detik adalah 5:1000, atau 0,5%—artinya 99,5% dari beban *broker* berasal dari penyerang. Meski demikian, *packet loss* pesan sah 'hanya' mencapai 4,242%, bukan 99,5%, karena antrean FIFO (*First-In First-Out*) memberikan peluang yang relatif setara kepada semua pesan untuk masuk ke antrean selama masih ada kapasitas. OASIS (2019) mendefinisikan bahwa MQTT tidak memiliki mekanisme prioritas pesan bawaan pada level protokol, sehingga antrean FIFO pada model ini merupakan representasi yang tepat untuk implementasi *default*. Kondisi ini justru menjadi argumen kuat untuk menerapkan mekanisme di luar protokol—seperti QoS *queue weighting* atau autentikasi klien dengan prioritas berbeda—agar pesan sah mendapatkan jaminan layanan yang lebih baik ketika terjadi serangan.

Hasil simulasi ini juga memiliki nilai praktis bagi pengembang sistem *smart home* dan administrator jaringan yang tidak memiliki latar belakang akademis dalam analisis serangan. Kurva degradasi yang terdokumentasi—stabil di bawah 850 pesan/detik, kritis di antara 850 dan 1000, dan kolaps di atas 1000—memberikan panduan kuantitatif tentang batas operasi *broker* yang perlu dilindungi. Eclipse Foundation (2026b) menyediakan parameter konfigurasi Mosquitto yang langsung dapat digunakan untuk menerapkan *rate limiting* dan batas antrean berdasarkan pemahaman tentang *threshold* ini. Alsulami (2023) juga mengevaluasi bahwa pendekatan IDS berbasis *machine learning* memberikan deteksi yang adaptif, tetapi justru membutuhkan data seperti ini—kurva degradasi QoS dengan parameter yang terdokumentasi—sebagai acuan untuk

p-ISSN : 2406-7733
e-ISSN : 2597-9922

mendefinisikan kondisi normal dan kondisi serangan dalam proses pelatihan model.

4.6. Analisis Latensi Persentil ke-95

Nilai latensi persentil ke-95 (P95) pada Tabel 2 memberikan gambaran distribusi latensi yang tidak tertangkap oleh nilai rata-rata saja. Pada skenario normal, P95 latensi sebesar 3,606 ms berarti 95% pesan sah diproses dalam waktu kurang dari 3,606 ms—distribusi yang sangat sempit dan mendekati nilai rata-rata 1,199 ms. Skenario 850 pesan/detik menghasilkan P95 latensi 464,097 ms, jauh lebih besar dari rata-rata 115,838 ms. Perbedaan antara rata-rata dan P95 yang lebar ini menunjukkan distribusi latensi yang sangat condong ke kanan (*right-skewed*): sebagian besar pesan masih dilayani cepat, tetapi sejumlah pesan mengalami penantian panjang yang ekstrem akibat antrian mengular saat beban mendekati kapasitas. Swain (2025) mengidentifikasi bahwa karakteristik distribusi latensi yang tidak simetris semacam ini merupakan ciri khas kondisi near-saturation pada sistem broker MQTT.

Pada skenario 1000 dan 1500 pesan/detik, P95 latensi stabil di sekitar 3.573–3.589 ms meskipun rata-rata sedikit berbeda. Konvergensi P95 ini konsisten dengan pengamatan pada analisis standar deviasi: ketika antrean selalu penuh, hampir semua pesan mengalami penantian panjang sehingga distribusi latensi menjadi lebih seragam—semua pesan sama-sama terlambat. Bagi sistem *smart home* yang mengandalkan respons *real-time* seperti sensor gerak atau kontrol pintu otomatis, nilai P95 lebih relevan sebagai metrik SLA dibandingkan rata-rata, karena P95 mencerminkan pengalaman pengguna dalam kasus terburuk yang masih umum terjadi. Dokumentasi Mosquitto menyiratkan pentingnya pengaturan antrean untuk mengelola distribusi latensi seperti ini pada kondisi beban tinggi (Eclipse Foundation, 2026a).

4.7. Keterbatasan Penelitian

Keterbatasan penelitian ini adalah data diperoleh dari simulasi diskrit, bukan dari pengujian fisik menggunakan Raspberry Pi, ESP32, router rumah, atau *broker* Mosquitto nyata. Karena itu, angka latensi, *throughput*, dan *packet loss* tidak boleh dibaca sebagai performa mutlak suatu perangkat. Nilai tersebut hanya sah untuk parameter simulasi yang dijelaskan pada metode. Keterbatasan lain adalah model belum memasukkan variasi QoS MQTT, TLS *handshake*, ukuran *payload* berbeda, retransmisi TCP, dan pengaruh *bandwidth* Wi-Fi. Keterbatasan ini sengaja dinyatakan agar naskah tidak menyajikan data palsu atau klaim berlebihan.

V. KESIMPULAN DAN SARAN

Kesimpulan

1. MQTT bekerja stabil pada kondisi normal dan *flooding* intensitas rendah hingga sedang, selama total beban masih berada di bawah kapasitas layanan *broker*.
2. Degradasi latensi terjadi tajam ketika laju *flooding* mendekati kapasitas *broker*; pada skenario 850 pesan per detik, latensi rata-rata pesan sah mencapai 115,838 ms walaupun *packet loss* masih 0%.
3. Packet loss *mulai muncul ketika laju* flooding *melampaui kapasitas* broker*; pada 1.000 pesan per detik,* packet loss *pesan sah mencapai 4,242%, dan pada 1.500 pesan per detik meningkat menjadi 20,259%.*
4. Flooding *DDoS menurunkan ketersediaan MQTT pada jaringan* smart home *dengan mekanisme penuh antrean, bukan dengan serangan terhadap enkripsi atau mekanisme autentikasi protokol lapisan aplikasi.*
5. Keamanan MQTT tidak cukup hanya mengandalkan mekanisme koneksi standar; diperlukan pembatasan laju, autentikasi klien, *access control list* per topik, pengaturan antrean *broker*, dan pemantauan trafik untuk mempertahankan ketersediaan layanan.

Saran

Penelitian lanjutan perlu menggunakan pengujian fisik dengan *broker* Mosquitto, perangkat ESP32 atau Raspberry Pi, dan generator trafik pada jaringan lokal. Variabel yang perlu dieksplorasi meliputi perbandingan QoS 0, QoS 1, dan QoS 2; pengaruh TLS terhadap latensi dan *throughput*; penggunaan CPU serta memori *broker*; serta evaluasi mitigasi seperti *rate limiting*, ACL, *firewall*, dan deteksi anomali berbasis *machine learning*. Naskah ini sebaiknya diposisikan sebagai studi simulasi awal yang dapat dijadikan acuan parameter bagi eksperimen fisik berikutnya.

DAFTAR PUSTAKA

Alsulami, R. (2023). IoT protocol-enabled IDS based on machine learning. *Engineering, Technology & Applied Science Research*.

Aroon, N. (2025). Detection of TCP and MQTT-based DoS/DDoS attacks on MUD-based IoT networks. *Electronics*, *14*(8), Article 1653.

Eclipse Foundation. (2026a). *mosquitto.conf man page*. Eclipse Mosquitto Documentation. https://mosquitto.org/man/mosquitto-conf-5.html

Eclipse Foundation. (2026b). *mosquitto_pub man page*. Eclipse Mosquitto Documentation. https://mosquitto.org/man/mosquitto_pub-1.html

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Hindy, H., Bayne, E., Bures, M., Atkinson, R., Tachtatzis, C., & Bellekens, X. (2021). Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset). In *Proceedings of the 12th International Networking Conference (LNNS, Vol. 180, pp. 73–84)*. Springer. https://doi.org/10.1007/978-3-030-64758-2_6

Ibrahim, M. I., Darus, M. Y. (2024). DDOS Attack Analysis on IoT Device for Smart Home Environment and A Proposed Detection Technique. JOIV: International Journal on Informatics Visualization, 8(4), 2104-2110.

Negesse, D. B., Gemeda, K. A., & Gianini, G. (2026). DDoS attack detection and classification for the MQTT-IoT protocol using LSTM models. Discover Applied Sciences.

OASIS. (2019). *MQTT Version 5.0*. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

Omotosho, A., Qendah, Y., & Hammer, C. (2023, June). IDS-MA: Intrusion detection system for IoT MQTT attacks using centralized and federated learning. In 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 678-688). IEEE.

Pham Le, P. H. (2026). A comparative security analysis of MQTT brokers against DoS attacks. *EURASIP Journal on Wireless Communications and Networking*.

Simbolon, A. B., Siregar, D., Gulo, S. A., & Kiswanto, D. (2025). Implementasi mangle filtering content menggunakan mikrotik untuk meningkatkan keamanan jaringan. *JUTIK: Jurnal Teknologi Informasi dan Komputer*, *11*(2).

Swain, M. (2025). Identifying communication sequence anomalies to detect DoS/DDoS attacks in MQTT. *Computers & Security*.

Yoga, I. G. A. M. S., Piarsa, I. N., & Buana, P. W. (2025). Kontrol multi motor servo dengan joystick virtual berbasis Android. *JUTIK: Jurnal Teknologi Informasi dan Komputer*, *11*(1), 1–13.

Zeghida, H. (2025). XMID-MQTT: Explaining machine learning-based intrusion detection system for MQTT protocol in an IoT environment. *International Journal of Information Security*.