ANALISIS PENDETEKSIAN SERANGAN DENIAL OF SERVICE (DOS) MENGGUNAKAN LOGIKA FUZZY METODE MAMDANI PADA JARINGAN INTERNET OF THINGS (IOT)

DENIAL OF SERVICE (DOS) DETECTION ANALYSIS USING FUZZY LOGIC MAMDANI METHOD ON INTERNET OF THINGS (IOT) NETWORK

Farid Muhammad[1], Ida Wahidah[2], Arif Indra Irawan[3]
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]**faridm@student.telkomuniversity.ac.id**, [2]**wahidah@telkomuniversity.ac.id**,
[3]**arifirawan@telkomuniversity.ac.id**

Abstrak

Munculnya *Internet of Things* (IoT) memudahkan masyarakat dalam melakukan banyak hal. IoT terjadi karena adanya hubungan antara jaringan internet dengan suatu perangkat. *Message Queueing Telemetry Transport* (MQTT), merupakan salah satu protokol yang digunakan pada jaringan IoT. MQTT bersifat *lightweight* dan dapat menangani banyak *client*, sehingga ckup banyak digunakan dan dipilih. Keterbukaan dalam sistem MQTT membuat protokol mudah untuk terkena serangan, salah satunya adalah *Denial of Service* (DoS). Serangan DoS dapat menyebabkan sebuah *server* atau jaringan tidak dapat diakses oleh *user* sehingga tidak dapat digunakan dengan baik. Oleh karena itu, diusulkan penelitian Tugas Akhir untuk menganalisis serangan DoS pada jaringan IoT dengan membuat sebuah rancangan *Intrusion Detection System* (IDS). Algoritma yang akan digunakan dalam pembuatan rancangan pendeteksian serangan DoS adalah Logika *fuzzy* metode mamdani. Penelitian ini dilakukan untuk menganalisis efisiensi algoritma dalam mendeteksi serangan dan *Quality of service* (QoS) dari jaringan. Pengerjaan dibuat pada MATLAB dan akan dilakukan perbandingan QoS dengan Cooja *Simulator*. Hasil pengujian didapatkan bahwa pendeteksian dengan *synthetic network* belum bisa dibilang akurat dalam pendeteksiannya, dikarenakan belum bisa menampilkan hasil secara numerik. Skenario beban trafik dan tanpa beban trafik mempengaruhi hasil QoS MATLAB dan Cooja ketika dilakukan perbandingan. Rata-rata keseluruhan *recall* pada pendeteksian di Cooja adalah 98.62% pada pengujian 20 *node.*

Kata kunci : Deteksi, DoS, MQTT, Logika *Fuzzy*.

Abstract

The appearing of Internet of Things (IoT) really helps the society for doing things. IoT can be happen because there are a connection between a network and a device. Message Queueing Telemetry Transport (MQTT) is one of the protocol that used in IoT. MQTT is frequently selected for IoT protocol because it is lightweight and can handle many clients. The openness of MQTT system make the protocol become vulnerable to be attack. One of the examples is Denial of Services (DoS) attack. DoS attack make a server or a network cannot be access by the user, and it is not working well. Based on that, This Final Assignment will be doing an analysis of DoS attack detection on IoT network by creating an Intrusion Detection System (IDS). The algorithm that use to create a detection system design is a Mamdani Fuzzy Logic. The purposes of this is to analyze the algorithm efficiency and the Quality of Service (QoS) from the network. The work will be create on MATLAB and compare it with another QoS from Cooja Simulator. The results shos that detection with a synthetic network not accurate yet, because it cannot show the accuracy in numeric. Attack and no attack scenario affect the MATLAB and Cooja QoS results when doing comparison. Recall overall on Cooja detection is 98.62% on 20 node testing.

Keywords: *Detection*, DoS, *MQTT*, *Fuzzy Logic*.

1. Pendahuluan

Internet of Things (IoT) merupakan sebuah teknologi pada jaringan yang memungkinkan suatu perangkat saling terhubung dengan *internet* dan dapat melakukan *data sharing* [1]. Dalam infrastruktur IoT, terdiri dari jaringan yang sudah ada dan internet beserta pengembangan jaringannya, sehingga IoT menawarkan objek, sensor, kemampuan koneksi yang dapat menyediakan layanan, dan aplikasi kooperatif yang independen. Message Queueing Telemetry Transport (MQTT) merupakan salah satu protokol yang digunakan pada jaringan IoT. MQTT dikembangkan oleh International Business Machines (IBM) pada tahun 1999 dan mendapatkan standarisasi pada tahun 2013 [2]. MQTT mempunyai tiga komponen penting, yaitu *Publisher*, *Subscriber*, dan *Broker*. Banyak aplikasi yang memanfaatkan protokol MQTT seperti *health care monitoring*, dikarenakan MQTT merupakan protokol yang *lightweight*, dan mampu menangani ribuan *client* jarak jauh hanya dengan satu *server*. Protokol MQTT memiliki kelemahan, yaitu sifat keterbukaan protokol yang dapat menyebabkan

serangan-serangan dapat terjadi , seperti serangan *Denial of Services* (DoS) [3]. Penelitian sebelumnya telah melakukan pembuatan sebuah *Intrusion Detection System* (IDS) pada protokol MQTT untuk mendeteksi serangan DoS. Penelitian [3] bernama *Secure*-MQTT dan membandingkannya dengan MQTT-S untuk melakukan pendeteksian serangan DoS.

Penelitian ini bertujuan untuk menganalisis pendeteksian serangan DoS dalam suatu jaringan IoT dengan menggunakan algoritma *Fuzzy Logic* metode mamdani. Penelitian ini merupakan salah satu penelitian untuk mengetahui efisiensi dan efektifitas algoritma yang digunakan pada penelitian dan mengetahui hasil *Quality of Service* (QoS) dari jaringan yang diteliti. Pengerjaan penelitian ini dilakukan pada MATLAB dengan membuat sebuah *synthetic network* yang berusaha dibentuk menyerupai jaringan yang menggunakan protokol MQTT sederhana. Kemudian, hasil yang didapatkan pada MATLAB akan dibandingkan dengan hasil simulasi yang didapatkan pada Cooja *Simulator* pada bagian cara melakukan pendeteksian dan juga QoS dari sistem.

2. Dasar Teori dan Metodologi

2.1 *Intrusion Detection System* (IDS)

IDS merupakan sebuah alat yang bertujuan untuk melakukan *monitoring* sebuah data trafik pada jaringan. IDS melakukan identifikasi dan melakukan proteksi dari intrusi yang dapat mengancam keamanan, privasi, dan kesatuan pada jaringan [4]. Sistem akan memberikan sebuah peringatan untuk memberitahu apakah aktivitas tersebut tergolong *malicious* atau tidak. Dalam IDS yang tipikal, IDS tersusun atas sensor, mesin analisis, dan sistem pelaporan. Sensor menyebar pada daerah *network* dan *host* yang berbeda-beda. Tugas sensor adalah mengirimkan koleksi data kepada mesin analisis, yang bertanggung jawab untuk menginvestigasi koleksi data, dan mendeteksi intrusi yang sedang berlangsung.

Terdapat dua tipe IDS yang biasa digunakan, ada *Host-based Intrusion Detection System* (HIDS) dan *Network-based Intrusion Detection System* (NIDS). HIDS pada umumnya ada pada sebuah alat di komputer dan melakukan *monitoring* aktivitas jahat yang terjadi dalam sistem tersebut, dan NIDS melakukan *monitoring* aktivitas jahat pada segmen jaringan [5].

2.2 *Message Queueing Telemetry Transport (MQTT)*

MQTT merupakan sebuah protokol yang didesain secara spesifik untuk komunikasi "*Machine to machine*". Protokol MQTT berjalan pada *Transmission Control Protocol*/*Internet Protocol* (TCP/IP) dan mempunyai paket data yang rendah sehingga konsumsi pada *power supply* pun cukup kecil. Protokol ini merupakan protokol *data-agnostic* yang dapat mengirimkan data dalam berbagai macam bentuk seperti *binary data*, *text*, *Extensible Markup Language* (XML), *JavaScript Object Notation* (JSON) dan protokol ini menggunakan model *publish*/*subscribe* daripada model *client-server* [6]. MQTT mengikuti prosedur *TCP-based connection establishment*. Sebuah alat akan mengirimkan sebuah *request message*, yaitu *CONNECT*, untuk melakukan koneksi dengan *broker*. Ketika request telah diterima, *broker* akan mengirimkan *acknowledgment*, *CONNACK*, kepada alat yang mengirimkan pesan *request* tersebut. Kemudian, alat IoT akan mengirimkan atau *publish* pesan pada *topic* tertentu kepada *broker*, dan alat penerima *subscribe* pesan dari *broker*.

Dalam MQTT, terdapat QoS *level*, yang merupakan sebuah persetujuan antara pengirim pesan dan penerima pesan yang mendefinisikan sebuah jaminan dalam pengiriman pesan yang spesifik. Terdapat tiga QoS level, pertama adalah *at most once* (0) yang menjamin pengiriman *best-effort*, *at least once* (1) yang menjamin pengiriman terkirim setidaknya sekali kepada penerima, dan *exactly once* (2), *level* tertinggi dan menjamin setiap pesan diterima hanya sekali oleh penerima yang dituju [7].

2.3 Serangan *Denial of Service* (DoS)

Serangan DoS merupakan salah satu jenis serangan terhadap komputer atau jaringan pada internet yang bertujuan untuk mengacaukan atau merusak sistem di dalamnya [8]. Tipe serangan ini biasa mengarah kepada titik akhir suatu sistem, *communication nodes*, dan kanal komunikasi. Serangan ini menghabiskan *resource* yang dimiliki oleh jaringan atau komputer sampai *resource* tersebut habis dan tidak bisa berfungsi dengan benar, dan mencegah pengguna lain untuk memperoleh akses layanan dari komputer atau jaringan yang diserang secara tidak langsung. Serangan DoS yang biasa terjadi salah satunya adalah penyerang melakukan *traffic flooding* dengan paket yang banyak, lalu dikirimkan kepada *server* yang ingin diserang.

2.4 *Fuzzy Logic*

Fuzzy Logic dikenalkan pertama kali oleh Prof. Lotfi A. Zadeh pada tahun 1965. Himpunan *fuzzy* merupakan dasar dari teori logika *fuzzy*. Pada teori himpunan fuzzy, peranan anggota derajat keanggotaan sebagai penentu keberadaan elemen dalam suatu himpunan sangat penting. Derajat keanggotaan menjadi ciri utama dari penalarana dengan *fuzzy logic* tersebut [9]. *Fuzzy logic* digunakan untuk menerjemahkan suatu besaran yang diekspresikan menggunakan bahasa (linguistik) dan *fuzzy logic* menunjukkan sejauh mana suatu nilai itu benar dan sejauh mana nilai itu salah. *Fuzzy Logic* memiliki 3 tahap, yaitu *fuzzification*, *inference*, dan *defuzzification*.

2.4.1 *Membership Function* (MF)

MF merupakan suatu kurva untuk menunjukkan pemetaan titik-titik suatu *input* data ke dalam nilai keanggotaannya yang memiliki interval dari 0 sampai dengan 1.

2.4.2 Mamdani *Fuzzy Inference System*

Fuzzy Inference System merupakan sistem yang dapat mengevaluasi semua *rule* secara simultan untuk menghasilkan kesimpulan dan urutan *rule* secara sembarang. Oleh karena itu, semua aturan atau *rule* harus didefinisikan terlebih dahulu sebelum membangun sebuah *Fuzzy Inference System* yang akan digunakan untuk menginterprestasikan sebuah *rule* tersebut. Salah satu *Fuzzy Inference System* adalah Mamdani.

Metode Mamdani juga dikenal dengan nama metode MAX-MIN. Menggunakan MAX pada komposisi antar fungsi implikasi dan MIN pada fungsi implikasi. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Dalam metode Mamdani, konsekuensi dalam *base of rules* dihitung dengan ide spesialis. T-norm^ (minimum) biasa diadopsi untuk logika konektif "*and*", diekspresikan dengan persamaan [10]:

$$\mu_A(x) and\ \mu_B(x) = MIN\{\mu_A(x), \mu_B(x)\} \tag{2.1}$$

Di mana $\mu_A(x)$ dan $\mu_B(x)$ merupakan MF yang mendefinisikan *fuzzy set* A dan B. Untuk logika konektif "*or*" s-norm, V (maksimum) persamaannya adalah:

$$\mu_A(x)\ OR\ \mu_B(x) = MAX\{\mu_A(x), \mu_B(x)\} \tag{2.2}$$

Beberapa tahap yang dilakukan untuk mendapatkan *output* dari metode Mamdani ini adalah:

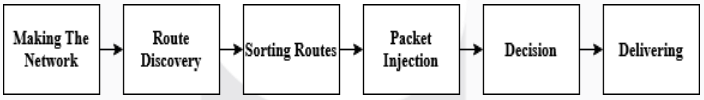
1. Saat melakukan evaluasi aturan dalam mesin inferensi, metode Mamdani menggunakan fungsi MIN dan komposisi antar-*rule* menggunakan fungsi MAX untuk menghasilkan *fuzzy* baru.
2. Proses defuzifikasi pada metode ini menggunakan metode centroid dengan rumus berikut:

$$Z = \frac{\int \mu(z).z\ dz}{\int \mu(z) dz} \tag{2.3}$$

Dengan Z merupakan titik pusat daerah *fuzzy*, dan $\mu(z)$ merupakan kumpulan dari *output* MF.

2.5 Perancangan dan Konfigurasi Sistem Pada MATLAB

Dalam penelitian ini dibuat sistem untuk melakukan pendeteksian serangan pada MATLAB. Berikut merupakan diagram sistem yang dirancang:



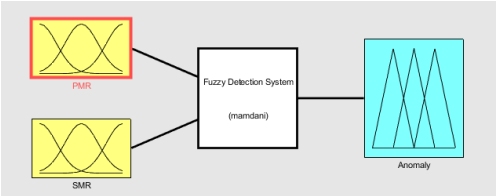
Gambar 1 Diagram Sistem *Synthetic Network* di MATLAB

Synthetic network merupakan pembentukan sebuah jaringan yang bersifat artifisial dan terdiri atas beberapa *node* dan garis yang merepresentasikan sebuah hubungan antara satu *node* dengan *node* lainnya. Terdapat enam bagian dalam pembuatan jaringan ini. Tahap pertama dimulai dengan *Making the Network*, pembentukan jaringan, dimulai dari jumlah *node* (*source* dan *destination node*), posisi masing-masing *node*, *radio propagation*, dan hubungan antara *node* satu dengan lainnya. *Route Discovery*, program melakukan pencarian rute yang sesuai dan tepat untuk jaringan yang akan dibuat. *Sorting Rout*es, jaringan akan dilakukan sortir. *Packet Injection*, tahapan ini dilakukan pengiriman paket pada jaringan. Terdapat pengiriman paket *CONNECT*, *PUBLISH*, dan *SUBSCRIBE*, serta kapasitas pesan yang dapat disesuaikan dengan standar paket pada MQTT. *Decision*, menggunakan algoritma *fuzzy logic* metode mamdani untuk melakukan pemilihan pada pengirman paket jaringan yang ada. Terakhir, *Delivering* dilakukan pengiriman paket yang telah diinjeksi kepada setiap *node* yang ada pada jaringan. Terdapat kondisi *IF-ELSE statement* untuk mengetahui *path* pada jaringan yang normal, abnormal, dan *attack*.

2.5.1 MF Pada MATLAB

MF yang digunakan pada MATLAB terdiri atas dua *input* variabel, *Publish Message Rate* (PMR) dan *Subscribe Message Rate* (SMR). Untuk *output* hanya terdiri atas satu, yaitu *anomaly*. Variabel PMR memiliki MF yang terdiri atas *Low*, *Medium*, dan *High* dengan rentang 0 sampai dengan 10. Untuk variabel SMR pun

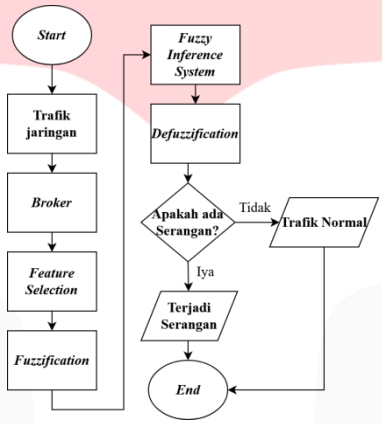
MF memiliki rentang yang sama dengan PMR. Variabel *output* Anomaly memiliki MF yang terdiri atas *Discard*, Normal, Abnormal, dan *Attack*, dan memiliki rentang dari 0 sampai dengan 10. Bentuk MF yang digunakan pada penelitian ini adalah Gaussian.



Gambar 2 MF pada penelitian di MATLAB

2.6 Perancangan dan Konfigurasi Sistem Pada Cooja

Penelitian selanjutnya dilakukan pada *Cooja Simulator*. Simulasi yang dilakukan pada pengujian ini bertujuan untuk melakukan perbandingan hasil deteksi yang dilakukan serta QoS yang didapatkan pada pengujian MATLAB. Terdapat diagram alir untuk penelitian di simulasi ini dan dapat dilihat pada Gambar 4.



Gambar 3 Diagram alir sistem deteksi pada Cooja

Pada jaringan yang menggunakan protokol MQTT akan mengirimkan sebuah *network traffic* pada sebuah jaringan yang telah dibuat. Paket-paket yang telah masuk pada *broker* diambil oleh *Feature Selection* dalam bentuk *node* yang akan mengambil informasi berupa pake *PUBLISH* dan *SUBSCRIBE* yang masuk pada *broker*. Informasi yang telah masuk, akan diambil oleh *fuzzy logic* yang dibentuk dalam sebuah *node* baru untuk membantu dalam pendeteksian serangan.

Paket yang telah masuk ke dalam *broker*, akan diambil oleh *feature selection node. Feature selection* akan mengambil informasi tentang paket *PUBLISH* dan *SUBSCRIBE* yang merupakan variabel *input* dalam penelitian di simulasi Cooja. Paket *PUBLISH* dan *SUBSCRIBE* ini akan menjadi dua buah variabel baru, yaitu *Publish Message Rate* (PMR) dan *Subscribe Message Rate* (SMR). Persamaannya adalah sebagai berikut

$$PMR = \frac{Publish\ masuk}{Paket\ masuk} \tag{3.1}$$

$$SMR = \frac{Subscribe\ masuk}{Paket\ masuk} \tag{3.2}$$

2.6.1 *Feature Selection* pada *node*

Feature Selection pada penelitian ini bertugas untuk mengambil informasi tentang paket *PUBLISH* dan *SUBSCRIBE* yang masuk ke dalam *broker* MQTT. Pengambilan informasi ini adalah dengan mengunakan *broker status* yang tersedia di dalam Mosquitto [11]. Terdapat tiga *broker status* yang akan digunakan pada penelitian ini, diantaranya adalah sebagai berikut:

1. *$SYS/broker/clients/total* : Berfungsi untuk memberikan informasi tentang jumlah klien yang aktif dan tidak aktif (paket *CONNECT* dan *CONNACK*)

2. *$SYS/broker/messages/received* : Berfungsi untuk memberikan informasi tentang keseluruhan jumlah semua jenis paket yang masuk ke dalam *broker*
3. *$SYS/broker/publish/messages/received* : Berfungsi untuk memberikan informasi tentang jumlah paket *PUBLISH* yang masuk ke dalam *broker*.

2.7 MF pada Cooja

Terdapat dua MF *input* dan satu MF *output* yang digunakana penelitian pengujian di cooja. Dua *input* ini adalah PMR dan SMR. PMR dan SMR memiliki MF yang terdiri atas *LOW*, *MEDIUM*, dan *HIGH* dengan rentang 0 sampai 0.5. Satu *output* ini merupakan *ANOMALY*, terdiri atas NORMAL, ABNORMAL, dan *ATTACK* dengan rentang dari 0 sampai dengan 1. Bentuk MF yang digunakan pada penelitian ini adalah bentuk segitiga.

2.8 *Fuzzy Rules* pada pengujian MATLAB dan Cooja

Dalam algoritma *Fuzzy Logic*, terdapat *rules* yang digunakan untuk membuat sistem dapat mendeteksi apakah itu termasuk serangan atau tidak. *Rules* yang terdapat pada *Fuzzy Logic* Mamdani penelitian ini ada Sembilan rules. Contoh dari isi *rules* yang ada pada penelitian ini adalah sebagai berikut:

1. IF PMR = *Low and* SMR = *Low* THEN *Anomaly* = Normal
2. IF PMR = *Low* and SMR = *Medium* THEN *Anomaly* = Abnormal
3. IF PMR = *Low* and SMR = *High* THEN *Anomaly* = *Attack*

2.9 Parameter Pengujian MATLAB dan Cooja

Terdapat tiga parameter di mana MATLAB menggunakan dua parameter (*Delivery Time*, dan *Throughput*) , dan Cooja memiliki tiga parameter *(Recall*, *Delivery Time*, *Throughput*), di mana hanya digunakan dua parameter untuk perbandingan hasil QoS, yaitu *Delivery time* dan *Throughput*. Penjelasan parameter lebih lanjut adalah sebagai berikut:

1. *Delivery Time*, merupakan pengukuran lamanya waktu yang dibutuhkan untuk paket yang dikirim kepada *destination node*. Satuan yang digunakan adalah milisekon (ms).
2. *Throughput*, yaitu kecepatan *transfer data* secara efektif, yang diukur dalam satuan bps (*bit per second*)/kbps (*kilobit per second*)/mbps *(megabit per second*). *Throughput* juga merupakan total keseluruhan paket yang sukses diterima dibagi dengan waktu pengiriman paket tersebut [12]. Persamaannya adalah:

$$Throughput = \frac{Paket\ data\ diterima}{waktu} \tag{3.3}$$

3. *Recall*, merupakan parameter yang menentukan seberapa presisi sistem *fuzzy logic* dalam mendeteksi sebuah serangan yang terjadi. Persamaannya adalah:

$$Recall = \frac{N_{TP}}{N_{TP}+N_{FN}} \tag{3.4}$$

Di mana N_{TP} merupakan jumlah *True Positive* dan N_{FN} merupakan jumlah *False Negative*.

3. Pembahasan

3.1. Skenario Pengujian

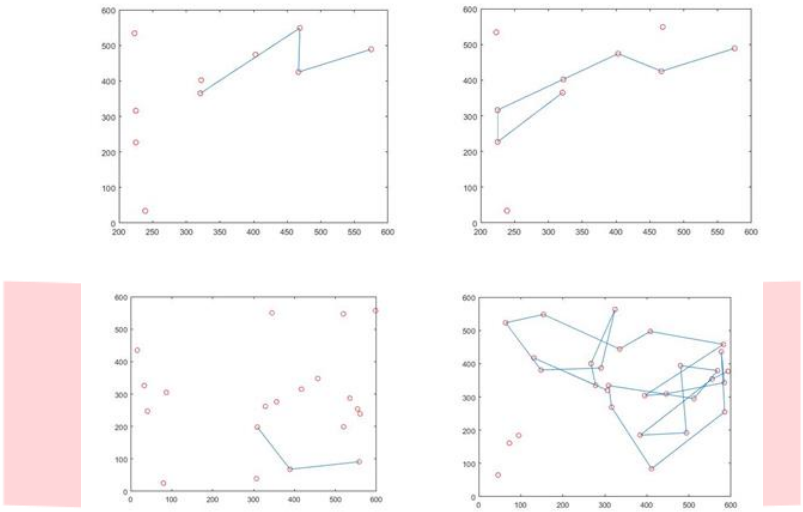
Skenario pengujian penelitian Tugas Akhir ini dilakukan sepenuhnya dalam perangkat lunak MATLAB dan Cooja. Pengujian dilakukan dengan skenario pengaruh jumlah *node* terhadap hasil yang didapatkan, kemudian selain itu pun, akan dilihat pengaruh dari hasil ketika terdapat beban trafik dan pada saat tidak terdapat beban trafik pada sistem yang sedang diuji. Jumlah *node* yang akan diuji pada MATLAB berjumlah 10, 20, dan 30 *node* dan untuk pengujian pada Cooja berjumlah 10, 20, dan 30 *node* (*feature selection node* tidak dianggap sebagai *node* klien). Pada pengujian MATLAB pun, selain hasil *Delivery Time* dan *throughput*, akan menunjukkan juga keluaran *synthetic network* yang bertindak sebagai pendeteksian serangan DoS pada MATLAB.

Pada pengujian Cooja, dari masing-masing jumlah *node* yang diuji terdapat 20% *node* yang bertindak sebagai penyerang, dan pengujian dilakukan selama 15 menit. Hasil QoS pada MATLAB dan Cooja kemudian akan dilakukan perbandingan.

3.3 Hasil *Synthethic Network* pada MATLAB

Terdapat hasil berupa *synthetic network* sederhana. Pembentukan *synthetic network* ini sebagai bentuk untuk melakukan deteksi apakah terjadi serangan atau trafik normal. *Synthetic Network* ditunjukkan berupa

path-path pendeteksian yang ditentukan dari jumlah *hop* yang telah ditentukan pada *threshold* (1 ≥ *decision* <3 (normal), 3 ≥ *decision* <5 (abnormal), *decision* ≥5 (*attack*)). Beberapa *path* adalah sebagai berikut:

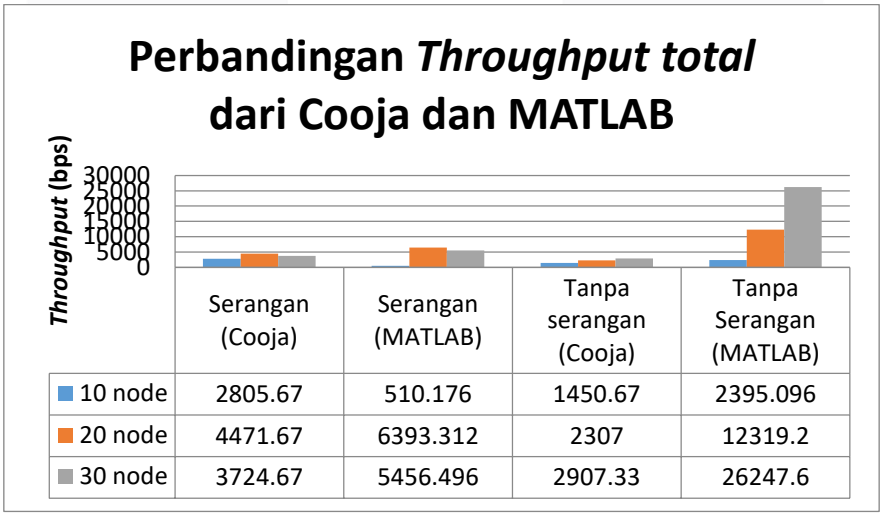


Gambar 1 Hasil *Synthetic Network*

Gambar 1 menunjukkan beberapa *synthetic network* yang diambil dari pengujian berjumlah 10, 20, dan 30 *node* dengan dua gambar di atas adalah 10 *node* pada skenario tanpa beban trafik dan dengan beban trafik, dua gambar di bawah adalah 20 *node* dengan beban trafik dan 30 *node* tanpa beban trafik. Dapat dilihat bahwa pada 10 *node* yang terdapat beban trafik dan tanpa beban trafik memiliki perbedaan. Bentuk *path* pada 10 *node* tanpa beban trafik menunjukkan keadaan normal, karena jumlah *hop* sebanyak tiga, dan pada pengujian dengan beban trafik, *synthetic network* termasuk kategori *attack/flood*, karena jumlah *hop* telah lebih dari lima. Pada *synthetic network* di jumlah 20 *node*, terdapat kesalahan deteksi, menjadi terdeteksi sebagai normal, dan pada jumlah 30 *node*, tanpa serangan terdeteksi sebagai serangan karena jumlah *hop* yang lebih dari lima. Secara keseluruhan, ini belum dapat dibilang akurat, karena akurasi belum bisa didapatkan secara numerik dan tidak dapat mengetahui *node* yang bertindak sebagai penyerang.

3.3 Perbandingan Hasil *Throughput* pada MATLAB dan Cooja

Hasil yang didapatkan adalah sebagai berikut:



	Serangan (Cooja)	Serangan (MATLAB)	Tanpa serangan (Cooja)	Tanpa Serangan (MATLAB)
10 node	2805.67	510.176	1450.67	2395.096
20 node	4471.67	6393.312	2307	12319.2
30 node	3724.67	5456.496	2907.33	26247.6

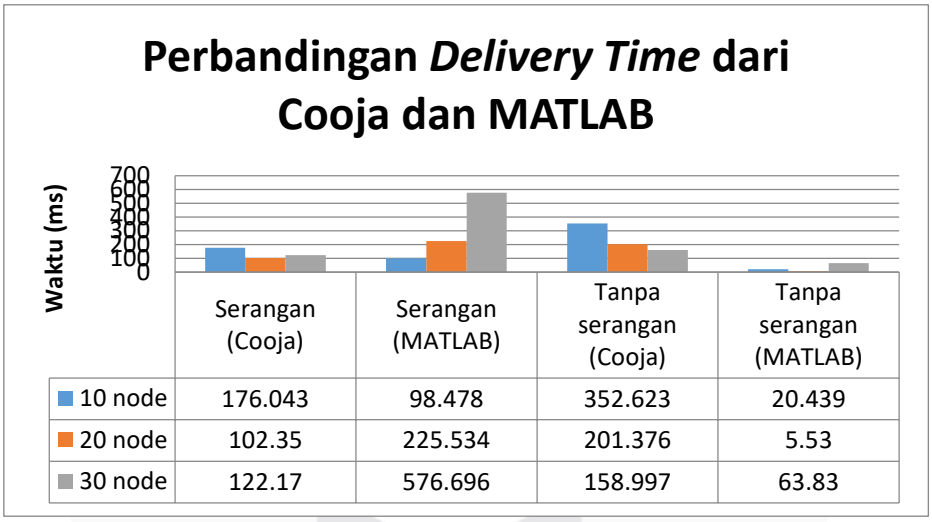
Gambar 2 Hasil Perbandingan *Throughput* MATLAB dan Cooja.

Secara keseluruhan, terlihat bahwa hasil *throughput* yang didapatkan pada pengujian Cooja mendapatkan hasil naik dan turun pada skenario serangan, dan pada skenario tanpa serangan, hasil yang didapatkan relatif mengalami kenaikan ketika jumlah *node* bertambah. Pada pengujian MATLAB, pengujian pada skenario serangan dan tanpa serangan relatif mengalami kenaikan setiap jumlah *node* bertambah. Hasil pada pengujian MATLAB pun merupakan hasil terbesar yang didapatkan pada dua skenario berbeda dibandingkan dengan

hasil yang didapatkan pada pengujian di Cooja. Pada pengujian Cooja, hasil *throughput* pada serangan memiliki hasil yang lebih tinggi dibandingkan pada skenario tanpa serangan, dengan hasil , sedangkan hasil pada MATLAB merupakan sebaliknya, di mana hasil pada skenario tanpa serangan, memiliki hasil yang lebih tinggi dibandingkan dengan hasil pada skenario serangan.

3.4 Perbandingan Hasil *Delivery Time* pada MATLAB dan Cooja

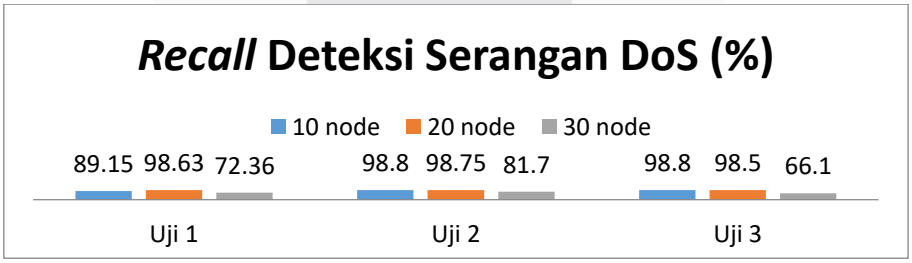
Hasil yang didapatkan pada pengujian *delivery time* pada pengujian dua *simulator* ada pada gambar 3. Secara keseluruhan, pengujian pada skenario serangan, waktu yang didapatkan pada pengujian di Cooja mengalami penurunan ketika bertambah menjadi 20 *node* dan kemudian mengalami kenaikan lagi pada jumlah 30 *node.* Pada pengujian di MATLAB, waktu yang didapatkan cenderung mengalami kenaikan setiap jumlah *node* bertambah dari 10 *node*, kemudian 20 *node*, dan 30 *node*. Waktu paling baik pada skenario serangan didapatkan pada MATLAB pada pengujian 10 *node* dengan waktu 98.478 ms, dan waktu terlama pada pengujian 30 *node* di MATLAB dengan 576.696 ms. Pada skenario tanpa serangan, pengujian di Cooja mendapatkan hasil yang cenderung menurun ketika bertambah jumlah *node*, sedangkan pada MATLAB, didapatkan hasil dengan perbedaan yang cukup signifikan dibandingkan dengan Cooja. Pada skenario tanpa serangan, hasil terbaik didapatkan pada pengujian MATLAB di 20 *node* dengan hasil 5.53 ms, dan waktu terlama didapatkan pada pengujian Cooja di 10 *node* dengan hasil 352.623 ms. Untuk pengujian dengan Cooja, sebaliknya hasil yang didapatkan pada skenario tersebut berbanding terbalik dengan hasil pada MATLAB dengan skenario serangan. Hasil pada pengujian 10 *node* menghasilkan waktu terlama dibandingkan dengan 30 *node* yang memiliki hasil tercepat. Hal tersebut dapat terjadi salah satunya adalah pada saat Cooja *simulator* sedang melakukan *running program* yang dikerjakannya. Pada simulasi, terdapat *speed time* yang bisa diatur, di mana 100% *speed time* sama dengan *real-time*. Pada saat pengujian sedang berlangsung, terkadang *speed time* pada saat pengujian bisa berubah-ubah persentase nya, sehingga kecepatan pun berubah menjadi lambat.



Gambar 3 Hasil *Delivery Time* pada MATLAB dan Cooja

3.5 Hasil *Recall* pada Pengujian di Cooja

Hasil *recall* yang didapatkan pada pengujian Cooja adalah sebagai berikut:



Gambar 4 Hasil *Recall* Deteksi pada Cooja

Pengujian pada masing-masing jumlah *node* di sini dilakukan sebanyak tiga kali pengujian dan direpresentasikan dalam bentuk persen (%). Secara keseluruhan, dari tiga pengujian yang dilakukan, memiliki perbedaan hasil yang didapatkan. Pada pengujian pertama, didapatkan *recall* tertinggi pada pengujian dengan

jumlah 20 *node* dengan hasil 98.63%, *recall* tertinggi pada pengujian kedua didapatkan pada pengujian dengan jumlah 10 *node* dengan hasil 98.8%, dan *recall* tertinggi pada pengujian ketiga didapatkan pada pengujian dengan jumlah 10 *node* juga, dengan hasil 98.8%. Dapat dilihat juga bahwa semakin banyak jumlah *node* yang diuji, maka persentase *recall* pun akan terjadi penurunan, seperti pada pengujian dengan jumlah 30 *node* yang merupakan hasil *recall* terendah yang didapatkan pada ketiga pengujian yang dilakukan. Hasil *recall* pada pengujian ini termasuk tinggi. Hasil didapatkan tinggi dapat terjadi karena pada saat sedang melakukan pendeteksian serangan DoS, *fuzzy logic* dapat mendeteksi sebuah serangan pada skenario ketika memang sedang terjadi serangan pada jaringan tersebut. Rata-rata keseluruhan *recall* pada 10 *node* adalah 95.58%, 98.62% pada 20 *node* dan 73.39% pada 30 *node*.

4. Kesimpulan

Dari pengujian dan analisis penelitian ini, dapat disimpulkan bahwa pendeteksian dengan *synthetic network* masih belum dapat dibilang akurat dalam pendeteksiannya, dikarenakan belum dapat menampilkan akurasi secara numerik, dan belum dapat diketahui mana *node* yang bertindak sebagai penyerang. Performa sistem pengujian di MATLAB dan Cooja ketika terdapat beban trafik dan tanpa beban trafik sangat mempengaruhi *output* yang dihasilkan oleh sistem. Pada perbandingan QoS yang didapatkan, pada skenario serangan, *throughput* paling besar didapatkan pada MATLAB pada jumlah 30 *node* dengan hasil 5456.496 bps, dan pada tanpa serangan, hasil terbesar didapatkan pada MATLAB pada jumlah 30 *node* dengan hasil 26247.6 bps. Pada bagian *delivery time*, pada skenario serangan, waktu tercepat didapatkan pada pengujian MATLAB dengan jumlah 10 *node* dengan hasil 98.478 ms, dan pada skenario tanpa serangan, hasil tercepat didapatkan pada pengujian MATLAB dengan jumlah 20 *node* dengan hasil 5.53 ms. Pengujian *recall* pada Cooja mendapatkan hasil rata-rata tertinggi pada pengujian jumlah 20 *node* dengan 98.62%.

Daftar Pustaka:

[1] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, no. February, pp. 25–37, 2017.

[2] A. Al-fuqaha, S. Member, M. Guizani, M. Mohammadi, and S. Member, "Internet of Things : A Survey on Enabling," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] A. P. Haripriya and K. Kulothungan, "Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things," *Eurasip J. Wirel. Commun. Netw.*, vol. 2019, no. 1, 2019.

[4] M. F. Elrawy and A. I. Awad, "Intrusion detection systems for IoT-based smart environments : a survey," pp. 1–20, 2018.

[5] R. T. Gaddam and M. Nandhini, "An analysis of various snort based techniques to detect and prevent intrusions in networks: Proposal with code refactoring snort tool in Kali Linux environment," *Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2017*, no. Icicct, pp. 10–15, 2017.

[6] R. A. Atmoko, R. Riantini, and M. K. Hasin, "IoT real time data acquisition using MQTT protocol," *J. Phys. Conf. Ser.*, vol. 853, no. 1, 2017.

[7] The HiveMQ Team, "Quality of Service 0,1 & 2 - MQTT Essentials: Part 6," 2015. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. [Accessed: 22-Oct-2019].

[8] V. Zlomislić, K. Fertalj, and V. Sruk, "Denial of service attacks, defences and research challenges," *Cluster Comput.*, vol. 20, no. 1, pp. 661–671, 2017.

[9] S. Abidah, "Analisis komparasi metode tsukamoto dan sugeno dalam prediksi jumlah siswa baru," *J. Teknol. Inf. dan Komun.*, vol. 8, no. 2, pp. 57–63, 2013.

[10] E. Pourjavad and A. Shahin, "The Application of Mamdani Fuzzy Inference System in Evaluating Green Supply Chain Management Performance," *Int. J. Fuzzy Syst.*, vol. 20, no. 3, pp. 901–912, 2018.

[11] R. Light, "Mosquitto man page | Eclipse Mosquitto." [Online]. Available: https://mosquitto.org/man/mosquitto-8.html. [Accessed: 26-Oct-2020].

[12] ETSI, "EG 203 165 - V1.1.1 - Speech and multimedia Transmission Quality (STQ); Throughput Measurement Guidelines," vol. 1, pp. 1–30, 2012.