Rancang Bangun *Evolved Network Management System Application* (EVOMAC) Pada Ekosistem Infrastruktur Jaringan Universitas Udayana

Adhitya Bayu Rachman Pratama[1], Gede Sukadarmika[2], Nyoman Putra Sastra[3]

[Submission: 06-01-2022, Accepted: 01-02-2022]

***Abstract*—Network Management System is a device with piles of capabilities to help the network engineer to utilize independent network component especially in a wide framework. Network Management System have the capabilities to identify, configure, monitor, update, and troubleshoots the wired also wireless network devices as well. Udayana University already using two different network management system such as Cacti and OpenNMS. In the 2018 research done by Yohanes, N.H., et all. shows that OpenNMS is a better network management system than Cacti, since it has various functions than Cacti. Hence the OpenNMS were chosen as an integrated component in their application for a network management system purposes. However, the 2018 research was still focusing in accommodate a single API method, which is GETMethod only. This makes their application is only owning one function, to deliver information about outages in the corresponding network. Thus, EVOMAC present as a solution not only to simplify the workflow of the network engineers, but is also able to modify the network via node adding and deleting. EVOMAC is an acronym to Evolved Network Management System Application which constructed from OpenNMS, PostgreSQL, Telegram, and NodeJS. Unlike its predecessor in 2018 research which only utilizes one API method, EVOMAC is utilizing four basic API methods such as POSTMethod, PUTMethod, GETMethod, and DELETEMethod. All of those API methods are the foundation of the command functions /start, /nodes, /addnode, /deletenode, /alarm, /category, and /requisition. Each of the command function has their own response time duration since it is related to the speed of the connected internet and the amount of people who has access to it. Also, EVOMAC has a much more variative outage categories than the previous application research done in 2018. The three categories are MINOR, MAJOR, and CRITICAL. While the previous application research only has two categories MINOR and MAJOR.**

***Keywords*— EVOMAC, Simplify, Solution, Respond, API**

***Intisari*—Sistem Manajemen Jaringan merupakan sebuah perangkat dengan kapabilitas membantu *network engineer* untuk mengelola komponen jaringan independent pada *framework* jaringan yang lebih besar. Universitas Udayana telah menggunakan beberapa *platform* untuk keperluan *network management* seperti Cacti dan OpenNMS. Namun, pada penelitian tahun 2018 yang telah dilakukan oleh Yohanes, N.H., dkk. menunjukkan bahwa OpenNMS lebih baik dari Cacti. Sehingga hal tersebut yang membuat OpenNMS menjadi komponen pengintegrasi dalam aplikasi network management system pada penelitian tahun 2018 tersebut. Penelitian pada aplikasi itu masih berfokus pada pemanfaatan satu metode API saja yaitu GETMethod, sehingga aplikasi tersebut hanya memiliki fungsi untuk menginformasikan gangguan atau kejadian dalam jaringan. Maka dari itu, dalam penelitian ini EVOMAC digagaskan untuk memberikan sebuah solusi berupa penyederhanaan *workflow* bagi para administrator jaringan, hingga memberikan kemudahan modifikasi jaringan via penambahan serta penghapusan node. EVOMAC merupakan kependekan dari *Evolved Network Management System Application* yang terkonstruksi dari OpenNMS, PostgreSQL, Telegram, dan NodeJS. Tak seperti aplikasi pendahulu pada penelitian tahun 2018 yang hanya memberdayakan satu metode API, EVOMAC memberdayakan empat metode API yaitu, GETMethod, POSTMethod, PUTMethod, dan DELETEMethod. Seluruh metode API tersebut merupakan dasar dari fungsi perintah pada EVOMAC seperti, /start, /nodes, /addnode, /deletenode, /alarm, /category, dan /requisition. Pada penelitian ini ditemukan adanya perbedaan durasi respons tiap *command function*, hal tersebut dipengaruhi oleh kecepatan koneksi jaringan internet, serta jumlah pengakses koneksi jaringan internet tersebut. Selain itu, kategori gangguan dari EVOMAC lebih variatif dari aplikasi pendahulu pada tahun 2018 yang hanya menggunakan dua kategori gangguan MAJOR, MINOR. Sementara EVOMAC sudah menggunakan tiga kategori MAJOR, MINOR, dan CRITICAL. Penambahan kategori galat tersebut bertujuan untuk melakukan tindak preventif sedini mungkin saat gangguan dalam jaringan terjadi.**

***Kata Kunci*— EVOMAC, Simplifikasi, Solusi, Respons, API**

I. PENDAHULUAN

Sistem manajemen jaringan adalah sebuah aplikasi ataupun seperangkat aplikasi dengan kapabilitas membantu *network engineer* dalam mengelola komponen jaringan independen pada *framework* jaringan yang lebih besar [1]. *Network Management System* (NMS) mampu mengidentifikasi, mengonfigurasi, memonitor, memperbaharui, dan melakukan *troubleshooting* pada perangkat jaringan baik *wired* maupun *wireless* dalam sebuah *network* [2]. Kemudian, sebuah sistem manajemen kontrol berperan untuk menampilkan data performansi yang telah terkumpul dari tiap komponen jaringan,

[1]*Mahasiswa, Universitas Udayana, Jl. Pelabuhan Benoa, Benoa Residence, Denpasar Selatan, Denpasar 80852 Indonesia (tlp: 0857-3888-7297; e-mail: pratamarachman07@gmail.com)*
[2, 3] *Dosen, Program Studi Teknik Elektro Fakultas Teknik Universitas Udayana, Jln. Jalan Kampus Bukit Jimbaran 80361 INDONESIA (telp: 0361-703315; fax: 0361-4321; e-mail: sukadarmika@unud.ac.id, putra.sastra@unud.ac.id)*

sehingga para *network engineer* mampu melakukan perubahan jika diperlukan.

Network Management System (NMS) mampu mengidentifikasi, mengonfigurasi, memonitor, memperbaharui, dan melakukan *troubleshooting* pada perangkat jaringan baik *wired* maupun *wireless* dalam sebuah *enterprise network*. Kemudian, sebuah sistem manajemen kontrol berperan untuk menampilkan data performansi yang telah terkumpul dari tiap komponen jaringan, sehingga para *network engineer* mampu melakukan perubahan jika diperlukan [1].

Universitas Udayana telah menggunakan beberapa *platform* untuk keperluan *network management*, seperti Cacti™ dan OpenNMS™. Cacti merupakan sebuah *network monitoring tools* yang bekerja berdasarkan fasilitas *Simple Network Management Protocol* (SNMP) dan *Round Robin Database Tools* (RRD Tools) sehingga statistik performa jaringan dalam kurun waktu tertentu dapat dimonitor dengan baik. Ada keterbatasan penggunaan Cacti pada infrastruktur jaringan Universitas Udayana [3], [4]. Contohnya pada diagnosa kasus saat seorang *network administrator* tidak terdapat pada *working area* dan membutuhkan reliabilitas (kehandalan) tinggi terhadap ketersediaan informasi tentang "kesehatan" jaringan termonitor. Cacti belum dilengkapi sistem *Network Map,* sehingga menyulitkan *network administrator* dalam melakukan pemantauan jaringan tertentu. Selain itu pula Cacti belum memiliki fitur *alert,* sehingga penanganan pada jaringan yang mendapatkan suatu *outage* (masalah) tidak dapat dilakukan secara cepat dan akurat [3].

OpenNMS dengan integrasi bot Telegram™ sudah cukup andal dalam menjadi *network monitoring platform* untuk memberikan informasi pada *administrator* terkait kesehatan jaringan. Namun penelitian tersebut masih terbatas pada pemberian notifikasi *outage* jaringan dan hanya memanfaatkan satu metode API yaitu GETMethod [3], [4]. Sehubungan dengan era *internet of things*, jumlah penggunaan *network device* dewasa ini mengalami lonjakan secara eksponensial. Hal tersebut tentu saja memengaruhi kemudahan proses *updating* instalasi dari segi *physical* maupun *logical* [5]. Maka dari itu penelitian ini menawarkan sebuah solusi dengan menambahkan *feature* "*remote node adding*" untuk mempermudah pekerjaan *network administrator*. Dengan adanya *feature* tersebut *administrator* dapat menambahkan *node* kapanpun dan dimanapun jika diperlukan. Dilengkapi dengan adanya *feature* pemantauan jaringan, aplikasi ini tentu dapat dikategorikan sebagai hasil evolusi dari aplikasi pendahulu. Hingga pada akhirnya, muncul sebuah gagasan untuk menciptakan *Evolved Network Management System Application* (EVOMAC) yang memanfaatkan empat metode API, antara lain GETMethod, POSTMethod, PUTMethod, dan DELETEMethod.

II. Manajemen Jaringan

Manajemen jaringan adalah berbagai macam proses, peralatan, dan aplikasi yang digunakan untuk mengadministrasi, mengoperasi, dan mengawasi sebuah infrastruktur jaringan. Manajemen performa dan analisis kesalahan juga termasuk dalam manajemen jaringan [6], [7]. Pengertian lebih mudahnya, manajemen jaringan adalah sebuah proses yang menjaga jaringan tetap sehat, sehingga mampu mendukung bisnis yang sehat pula [8].

Berkiblat dari *Open System Interconnection* (OSI), manajemen jaringan yang kredibel dan memiliki akuntabilitas tinggi harus memenuhi standard *Fault, Configuration, Accounting, Performance, and Security* (FCAPS). FCAPS merupakan model dan *framework* yang dicetuskan oleh *International Standard Organization* (ISO) *for Telecommunications Network Management* [9].

A. Fault Management

Manjemen kesalahan memfasilitasi administrator untuk memonitor infrastruktur jaringan tertentu agar dapat menentukan tindak lanjut jika terjadi galat atau kegagalan (down) [10].

B. Configuration Management

Manajemen konfigurasi memiliki fungsi untuk memonitor informasi pada konfigurasi jaringan, sehingga memudahkan pengelolaan *software* dan *hardware* [11].

C. Accounting Management

Fungsi dari manajemen akunting adalah untuk kalkulasi utilisasi jaringan dari seorang individu atau kelompok tertentu agar informasi penggunaan jaringan dapat dilacak hingga dapat dilakukan proses transaksi sebagai tujuan akunting. Manajemen akunting sering disebut sebagai *billing management*, hal ini karena pengaplikasian statistik pada jaringan telekomunikasi yang memungkinkan administrator untuk mendata kuota penggunaan *hardware* maupun *software* oleh suatu *user* [12].

D. Performance Management

Manajemen performansi atau dapat disebut juga sebagai manajemen kinerja fokusnya untuk memastikan bahwa performa dari suatu jaringan telekomunikasi bekerja dalam taraf yang *aceptable* [13]. Hal tersebut bertujuan agar administrator dapat melakukan preparasi untuk pengembangan jaringan di masa depan, dan untuk memelihara efektifitas serta efisiensi jaringan di masa kini. Penanganan performa jaringan meliputi *throughput, network response time, packet loss density, link utilization, utilization percentage, error rate,* dan sebagainya. Secara general, informasi tersebut dikumpulkan melalui manajemen *Simple Network Management Protocol* (SNMP) yang dipantau dan dikonfigurasi secara aktif agar dapat menotifikasi administrator untuk selalu memantau apakah performa jaringan berada di batas atas kinerja, atau berada di bawah standar kinerja yang telah ditetapkan [14]. Pengawasan jaringan secara aktif merupakan langkah fundamental untuk mewujudkan *credible network management*, karena dengan diterapkannya hal tersebut administrator mampu mengidentifikasi "percikan api" dalam suatu jaringan sebelum terjadi "kebakaran", lebih tepatnya mendeteksi adanya potensi masalah dalam suatu jaringan agar dapat dicegah supaya masalah yang lebih besar tidak terjadi [14], [15]. Dengan melakukan pengawasan jaringan secara kredibel, pemeliharaan kesehatan pada jaringan tersebut dapat terjaga.

E. Security Management

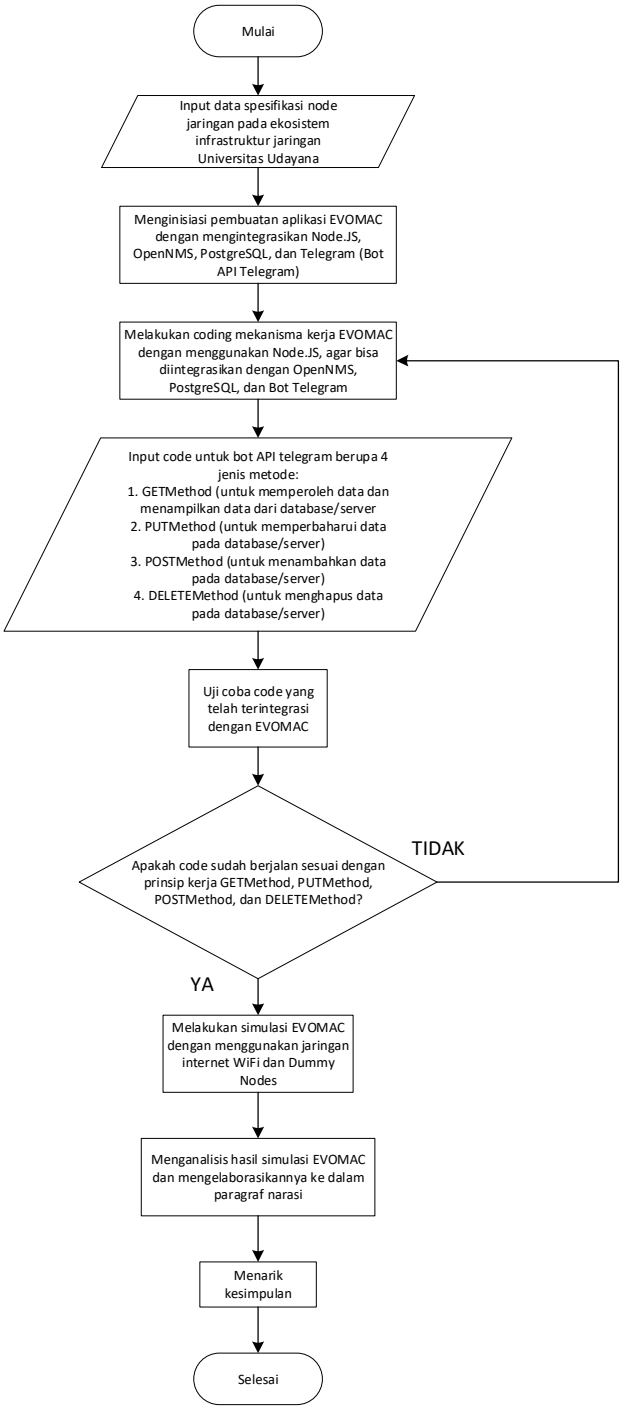
Manajemen keamanan adalah metode *safe-keeping* untuk aset-aset yang berada dalam jaringan telekomunikasi. Manajemen keamanan tidak hanya melakukan sentralisasi pada keamanan ekosistem jaringan, namun juga menganalisis secara

teratur informasi yang memiliki koneksi dengan keamanan [16]. Manajemen keamanan berfungsi untuk mengelola otentikasi, otorisasi, serta audit jaringan, dengan demikian *external user* ataupun *internal user* hanya dapat mengakses *network resource* yang proporsional sesuai dengan kebutuhan mereka [17]. *General purposes* yang lain dari manajemen keamanan adalah pengelolaan dan konfigurasi *firewall* jaringan, sistem deteksi instruksi, serta kebijakan keamanan.

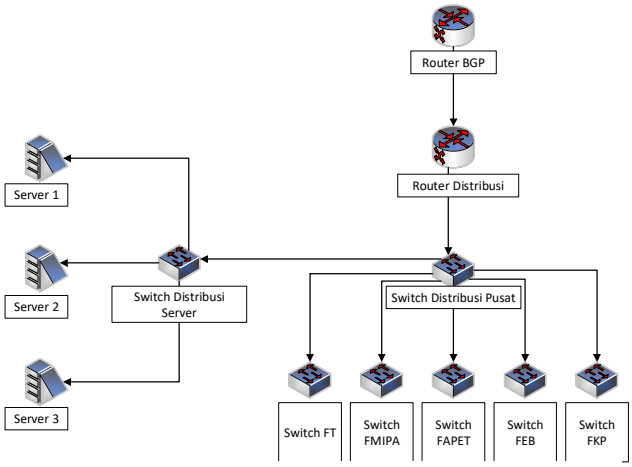
III. METODOLOGI

Penelitian ini meliputi peninjauan respons API, klasifikasi metode API, prinsip kerja EVOMAC, kecepatan respons EVOMAC, *breakdown* topologi eksisting menjadi topologi simulasi, serta simulasi pada ekosistem infrastruktur jaringan Universitas Udayana. Diagram alir penelitian ini direpresentasikan pada Gambar 1.

Langkah pertama adalah melakukan pengumpulan data spesifikasi node jaringan pada ekosistem infrastruktur jaringan Universitas Udayana. Data yang dikumpulkan berupa topologi jaringan eksisting, kemudian IP Address dari node eksisting, lalu IP Address dari *dummy node,* hingga pembuatan topologi jaringan simulasi. Langkah kedua adalah inisiasi pembuatan aplikasi EVOMAC dengan menintegrasikan Node.JS, OpenNMS, dan Bot API Telegram. Ketiga komponen tersebut dipilih sebagai elemen pendukung EVOMAC karena mudah dipelajari, banyak penelitian yang sudah menggunakan elemen tersebut, serta ketersediaan informasi untuk pedoman pengerjaan aplikasi dengan menggunakan Node.JS, OpenNMS, PostgreSQL, ataupun bot Telegram [5], [6], [7]. Langkah ketiga merupakan simulasi aplikasi EVOMAC menggunakan *dummy nodes* yang telah dibuat berdasarkan IP *Address website*, proses simulasi tersebut meliputi beberapa tahapan mulai dari penambahan node, penghapusan node, sampai memeriksa kondisi node yang telah di-*assign* pada jaringan simulasi. Langkah keempat memiliki tujuan untuk menganalisis kelebihan dan kekurangan dari EVOMAC, kelebihan didapatkan dari adanya penambahan metode API pada EVOMAC dibandingkan aplikasi pendahulu. Aplikasi pada penelitian tahun 2018 hanya berfokus menggunakan satu metode saja yaitu GETMethod [3], [4]. Sedangkan EVOMAC sudah menggunakan empat metode API sekaligus (GETMethod, PUTMethod, POSTMethod, dan DELETEMethod) yang membuat aplikasi ini memiliki lebih banyak fungsi perintah serta memiliki tingkat kompleksitas lebih tinggi. Sedangkan kekurangan didapatkan dari durasi respons EVOMAC yang memiliki ketergantungan dengan kecepatan koneksi jaringan internet terkoneksi dengan *device* milik *user*.



Gambar 1: Diagram Alir/*Flowchart* Penelitian



Gambar 2: Topologi Jaringan Eksisting Universitas Udayana

IV. HASIL DAN PEMBAHASAN

A. *Instalasi dan Konfigurasi OpenNMS*

Instalasi dan konfigurasi OpenNMS merupakan tahapan dasar dalam pembuatan aplikasi ini, hal tersebut dikarenakan peran OpenNMS sebagai *data provider* yang sangat penting dalam alur *delivery* komunikasi EVOMAC dengan *user* [18], [19]. Tahapan instalasi dan konfigurasi OpenNMS meliputi beberapa tahapan mulai dari mempersiapkan Ubuntu *server* 20.04 yang hendak digunakan sebagai *base operating system*, lalu recheck dan konfigurasi Java agar OpenNMS dapat diintegrasikan EVOMAC serta mampu memahami seluruh *command function* dari *user*, kemudian integrasi OpenNMS dengan PostgreSQL supaya mampu bersinergi dalam melakukan penyimpanan data serta *update* data dalam *database*[20], [21], [22]. Setelah seluruh tahapan tersebut sudah mampu dieksekusi secara baik dan benar, maka OpenNMS sudah siap untuk diaktivkan sebagai *complementary requirements* dalam EVOMAC.

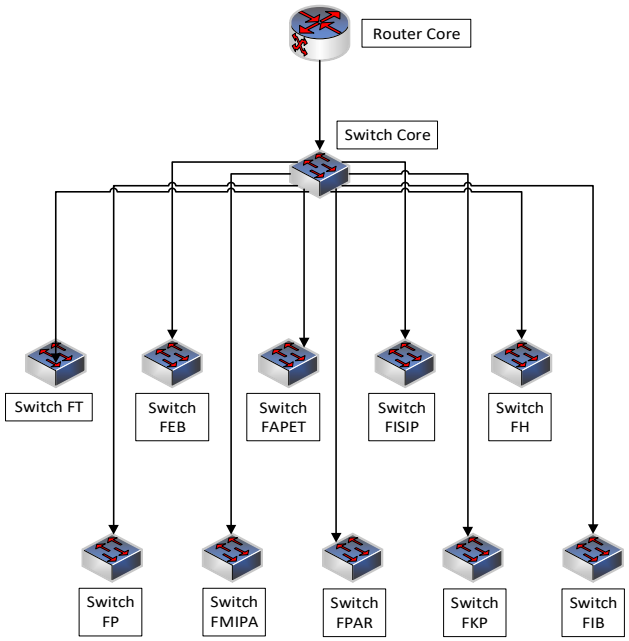
B. *Instalasi dan Konfigurasi EVOMAC*

EVOMAC memiliki beberapa tahapan untuk proses instalasi dan konfigurasi antara lain.

1) Tahap Pertama: Berdasarkan *website* OpenNMS, *user* harus mempersiapkan server dengan kapasitas minimal adalah CPU 2 Core, 4 GB RAM, 50 GB HDD, hal tersebut secara khusus diaplikasikan pada sebuah sistem uji. Sedangkan jika diaplikasikan dalam sebuah *real-life server*, kapasitas yang dibutuhkan sebesar CPU Quad core, 16 GB RAM, 1 TB HDD seperti pada Gambar 4.

2) Tahap Kedua: Pembuatan topologi jaringan simulasi EVOMAC di Gambar 3 berkaca pada topologi jaringan eksisting kampus Universitas Udayana dalam Gambar 2. Topologi jaringan simulasi merupakan hasil simplifikasi dari topologi jaringan eksisting karena proses simulasi hanya membutuhkan duabelas *dummy nodes* sesuai dalam Tabel 1.

3) Tahap Ketiga: Instalasi dan konfigurasi bot Telegram EVOMAC melalui BotFather yang dapat ditelusuri via kolom pencarian aplikasi Telegram *Messenger* seperti pada Gambar 7. Proses konfigurasi EVOMAC dengan menggunakan Visual Studio Code adalah penginputan *command functions* mulai dari /start, /nodes, /addnode, /deletenode, /alarm, /category, dan /requisitions agar EVOMAC dapat merespond perintah yang ditransmisikan oleh *user*. Pada penelitian tahun 2018, aplikasi pendahulu hanya menggunakan satu *method* API yaitu GETMethod. Sehingga fungsi perintah dari aplikasi terebut hanya dapat memvisualisasikan gangguan atau kejadian dalam suatu jaringan [3], [4]. Sedangkan pada EVOMAC, digunakan sebanyak empat *basic* API *method* antara lain POSTMethod, PUTMethod, GETMethod, dan DELETEMethod. Ke empat *basic* API *method* ini memiliki peran masing-masing dalam meningkatkan kompleksitas serta menambahkan fungsi perintah dibandingkan aplikasi pendahulu pada penelitian tahun 2018. EVOMAC tak hanya dapat menampilkan gangguan atau kejadian dalam jaringan, tetapi juga mampu melakukan modifikasi jaringan melalui penambahan node, penginputan IP *Address,* penamaan *requisition,* dan pengelompokan kategori. Seluruh metode dasar API yang digunakan oleh EVOMAC telah dielaborasi secara detail pada Tabel 5.



Gambar 3: Topologi Jaringan Simulasi Universitas Udayana

	Just testing*	Minimum server specification**
CPU	2 GHz dual core x86_64	3 GHz quad core x86_64 and above
RAM	4 GB (physical)	16 GB (physical) and above
Storage (disk space)	50 GB HDD, SSD	1 TB with SSD and above

Gambar 4: Spesifikasi Server untuk OpenNMS

C. *Simulasi EVOMAC*

Simulasi EVOMAC terbagi atas dua bagian utama yaitu, simulasi atas fungsi perintah dan simulasi atas durasi respons umpan balik. Simulasi atas fungsi perintah akan mengelaborasikan seluruh *command functions* dan bagaimana peran tiap *function* dalam EVOMAC. Kemudian untuk simulasi durasi respons *feedback* merupakan hasil pengujian respons

EVOMAC pada Cafe XYZ dengan waktu pengujian dilakukan pada Pagi Hari, Siang Hari, dan Malam Hari.

1) Simulasi Fungsi Perintah: Penelitian ini merancang agar EVOMAC mampu menjalankan beberapa fungsi perintah (*command function*) seperti /start, /nodes, /addnode, /deletenode, /alarm, /category, dan /requisitions. Seluruh fungsi perintah tersebut mendukung simplifikasi *workload* dari para *user* atau administrator yang hendak melakukan *task* melalui *dashboard* OpenNMS menjadi dapat dilakukan via *room chat* Telegram *messenger.*

TABEL I
IP *Address Dummy Nodes*

IP ADDRESS	ROLE
13.228.189.125	Switch FT
52.221.182.227	Switch FEB
3.0.184.153	Switch FAPET
13.250.41.33	Switch FISIP
3.1.203.71	Router Core
54.179.61.230	Switch Core
13.212.140.22	Switch FH
13.212.225.138	Switch FIB
18.138.19.227	Switch FKP
13.251.182.224	Switch FPAR
13.229.140.188	Switch FMIPA
18.141.88.157	Switch FP

Fungsi perintah /start memiliki peran dalam menampilkan seluruh fungsi perintah yang telah di-*assign* seperti pada Gambar 5. Fungsi perintah /nodes berperan dalam visualisasi *list node* aktif dalam jaringan terintegrasi. Tak hanya itu, fungsi perintah /nodes juga mampu memberikan informasi detail suatu *corresponding node* seperti *Node* ID, *Label*, *Source*, *Interface* dan *Category* yang dapat dilihat pada Gambar 6. Fungsi perintah /addnodes memiliki kapabilitas berupa kustomisasi *node* untuk ditambahkan pada jaringan. Kustomisasi tersebut berupa *requisition*, penambahan IP *Address*, *label*, dan *category* pada Gambar 7. Kemudian, fungsi perintah /deletenode yang merupakan *anti thesis* dari perintah /addnode memiliki peranan untuk menghapus node terpilih oleh *user* dari jaringan terintegrasi seperti pada Gambar 8. Dilanjutkan oleh fungsi perintah /alarm dengan peran sebagai *announcer* jika terdapat suatu kejadian tertentu dalam jaringan terintegrasi seperti dalam Gambar 9. Adapun tingkat kejadian dalam suatu jaringan terintegrasi dengan EVOMAC dibagi menjadi tiga tingkat antara lain *MINOR, MAJOR,* dan *CRITICAL* sesuai pada Tabel 6. Terakhir, fungsi perintah /category dan /requisition berperan dalam menampilkan kategori dan rekuisisi dari suatu node yang telah dimasukkan ke dalam jaringan seperti pada Gambar 10 serta Gambar 11 secara berturut-turut.

2) Simulasi Respons Feedback: Pengujian respons *feedback* EVOMAC dilakukan pada Cafe XYZ Denpasar yang memiliki koneksi jaringan internet dengan kecepatan *download* 20,80 Mbps, *upload* 7,62 Mbps, dan PING 4 ms dengan tiga skenario berbeda. Skenario pertama dilakukan pada waktu Pagi Hari pukul 08.00 – 09.00 WITA (*Breakfast Time*) di Tabel 2 menghasilkan *average response time* selama 0,98 s. Hal tersebut dipengaruhi oleh *occupancy rate* dari Cafe XYZ yang saat itu hanya sebesar 20%. Kemudian, skenario kedua dijalankan pada waktu Siang Hari pukul 12.00 – 13.00 WITA (*Lunch Time*) di Tabel 3 menghasilkan *average response time* selama 1,18 s. Hal tersebut dipengaruh oleh *occupancy rate* dari Cafe XYZ yang saat itu berada pada angka 70%. Lalu, skenario ketiga dilaksanakan pada waktu Malam Hari pukul 20.00 – 21.00 WITA (*Prime Time*) di Tabel 4 menghasilkan *average response time* selama 1,01 s. Hal itu dapat terjadi karena *occupancy rate* dari Cafe XYZ berada di angka 50%. *Average response time* diperoleh berdasarkan pengaplikasian persamaan (1) pada data setiap skenario [23].

$$\bar{x} = \frac{\sum fx}{n} \quad (1)$$

Keterangan:
$\bar{x}$ = *Mean* (Rerata)
f = Frekuensi dari setiap kelas
x = Nilai mid-interval dari setiap kelas
n = Total frekuensi
$\sum fx$ = Jumlah dari nilai mid-interval dan frekuensi terkait

TABEL II
Hasil Pengujian *Feedback* saat *Breakfast Time*

Command Function	Respond Durations	Unit(s)
/start	1.08	Seconds
/nodes	1.03	Seconds
/addnode	0.9	Seconds
/deletenode	0.93	Seconds
/alarms	1.02	Seconds
/categories	0.96	Seconds
/requisitions	0.93	Seconds

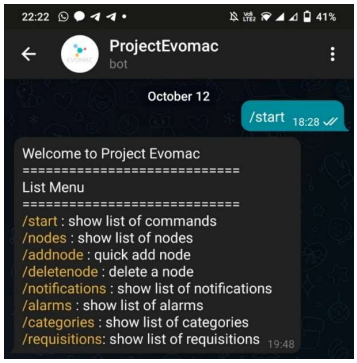
TABEL III
Hasil Pengujian *Feedback* saat *Lunch Time*

Command Function	Respond Durations	Unit(s)

/start	1.05	Seconds
/nodes	0.79	Seconds
/addnode	1.23	Seconds
/deletenode	1.56	Seconds
/alarms	1.32	Seconds
/categories	1.2	Seconds
/requisitions	1.1	Seconds

TABEL III
Hasil Pengujian *Feedback* saat *Prime Time*

Command Function	Respond Durations	Unit(s)
/start	1.05	Seconds
/nodes	1.03	Seconds
/addnode	1.09	Seconds
/deletenode	0.93	Seconds
/alarms	1.02	Seconds
/categories	1.06	Seconds
/requisitions	0.92	Seconds

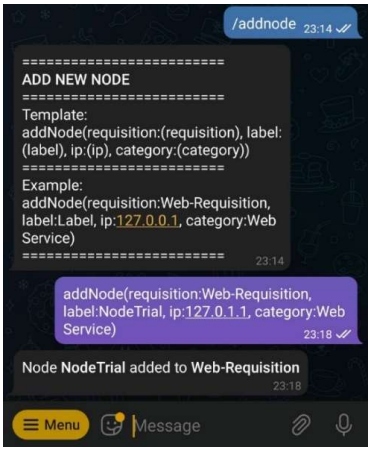


Gambar 5: Kapabilitas Fungsi Perintah /start

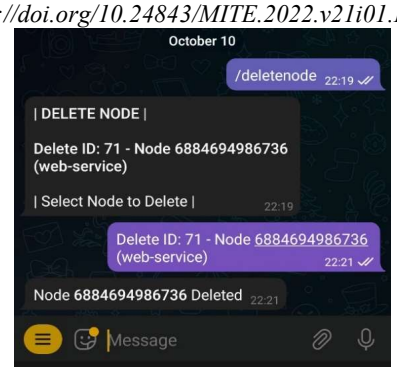
Gambar 6: Kapabilitas Fungsi Perintah /nodes

TABEL V
Basic API *Method* dalam EVOMAC

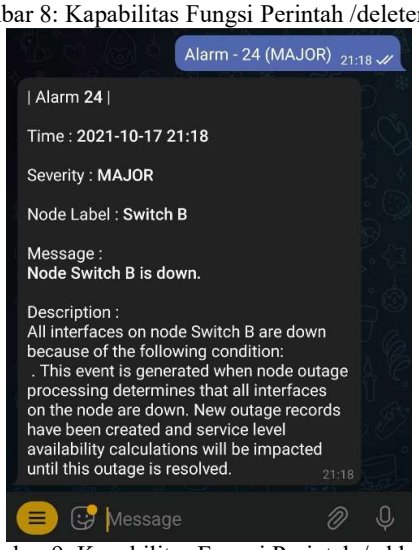
GETMethod		
No.	Fungsi Perintah	Keterangan
1	requisitionNames?limit=0	mengambil seluruh data *requisition*
2	requisitions	meng-*update* data pada semua *requisition*
3	categories?limit=0	mengambil seluruh data kategori
4	notifications?limit=0	mengambil seluruh data notifikasi
5	notifications/{id}	mengambil detail data dari notifikasi id
6	alarms?limit=0comparator=eq&severity=MAJOR	mengambil seluruh data alarm dengan *severity* MAJOR
7	alarms/{id}	mengambil detail data dari alarm id
8	nodes?limit=0	mengambil seluruh data node
9	nodes/{id}	mengambil detail data dari node id
10	nodes/{id}/ipinterfaces	mengambil seluruh data interface dari node id
POSTMethod		
No.	Fungsi Perintah	Keterangan
1	requisitions/${requisition}/nodes	menambahkan node pada *requisition*
PUTMEthod		
No.	Fungsi Perintah	Keterangan
1	requisitions/${requisition}/import	me-*refresh* data pada *requisition*
DELETEMethod		
No.	Fungsi Perintah	Keterangan
1	requisitions/${requisition}/nodes/${node}	menghapus node pada *requisition*



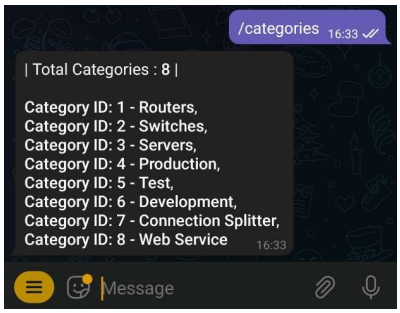
Gambar 7: Kapabilitas Fungsi Perintah /addnode



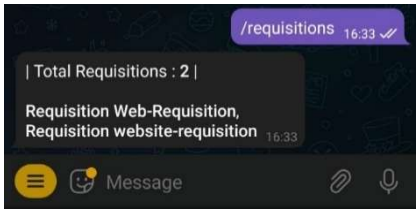
Gambar 8: Kapabilitas Fungsi Perintah /deletenode



Gambar 9: Kapabilitas Fungsi Perintah /addnode



Gambar 10: Kapabilitas Fungsi Perintah /addnode



Gambar 11: Kapabilitas Fungsi Perintah /addnode

TABEL VI
Event Severity Berdasarkan Klasifikasi EVOMAC

SEVERITY	*MEANING*	TERJEMAHAN
MINOR	*A part of a device (a service, and interface, a power supply, etc.) has stopped functioning. The device needs attention.*	Bagian dari sebuah perangkat (layanan, *interface*, catu daya, dll.) telah behenti bekerja. Perangkat tersebut membutuhkan perhatian.
MAJOR	*A device is completely down or in danger of going down. Attention needs to be paid to this problem immediately.*	Sebuah perangkat sepenuhnya telah *down* atau dalam ancaman menuju *down*. Diperlukan adanya perhatian pada masalah ini secepat mungkin.
CRITICAL	*This event means numerous devices on the network are affected by the event. Everyone who can should stop what they are doing and focus on fixing the problem.*	Alarm dengan status ini berarti banyak perangkat yang terpengaruh oleh kejadian tersebut. Siapapun yang mampu menangani permasalahan ini harus fokus memperbaiki masalah tersebut.

D. Analisis Kelebihan dan Kekurangan EVOMAC

1) *Kelebihan EVOMAC:* Dibandingkan oleh penelitian pada tahun 2018 yang hanya memberdayakan satu *method* API untuk menampilkan kejadian atau gangguan dalam suatu jaringan EVOMAC mampu menambahkan *node* ke dalam jaringan. Penambahan tersebut meliputi *input* IP *Address* serta penggolongan *requisition* dan *category*. *Node* tersebut dapat diberikan label atau nama yang sudah disepakati oleh para *administrator* sehingga akan mempermudah proses administrasi serta pengawasan jaringan.

EVOMAC memberikan informasi berupa notifikasi secara *real time* jika terdapat *node* yang ditambahkan pada jaringan melalui grup khusus notifikasi. Jadi hal tersebut bisa membantu para administrator untuk mengetahui *node* ataupun *device* terbaru yang ada dalam jaringan.

Tak seperti penelitian pendahulu pada tahun 2018 yang hanya menggunakan dua parameter alarm MINOR dan MAJOR, EVOMAC memiliki parameter alarm sebanyak tiga buah, yaitu MINOR, MAJOR, dan CRITICAL. Hal tersebut akan membantu administrator untuk mengambil tindakan pencegahan sebelum *node* benar-benar *down* serta memberikan *domino effect* negatif pada node lainnya dalam jaringan.

2) *Kekurangan EVOMAC:* Durasi *feedback* tiap *command function* sangat bergantung dengan kecepatan koneksi jaringan internet, serta jumlah pengakses jaringan internet tersebut. Hal tersebut dapat dilihat pada ketiga skenario pengujian respons *feedback* dengan menggunakan koneksi jaringan internet Cafe XYZ Denpasar yang sangat terpengaruhi oleh *occupancy rate* pada cafe tersebut. Semakin tinggi *occupancy rate* maka durasi respons *feedback* akan semakin lama. Jika *occupancy rate* semakin rendah, secara *vice versa* durasi respons *feedback* akan semakin cepat.

V. SIMPULAN

Artikel ini telah merancang dan membangun sebuah *evolved network management system application* yang terkonstruksi

dari OpenNMS, Node.JS, Bot API Telegram, serta PostgreSQL, diikuti oleh elaborasi dalam bentuk narasi terkait prinsip kerja aplikasi tersebut.

Hasil pengujian yang dilakukan menunjukkan bahwa *evolved network management system application* (EVOMAC) berhasil dibangun. Seluruh perangkat keras dan perangkat lunak pada sistem dibangun bekerja sesuai dengan rancangan yang ditentukan. Kinerja dari aplikasi ini tak hanya mampu dalam menginformasikan gangguan dalam jaringan seperti aplikasi pendahulu, namun EVOMAC mampu melakukan modifikasi jaringan via penambahan node, penginputan IP *Address,* penamaan *requisition,* pengelompokan kategori node, menampilkan kategori gangguan lebih banyak dari aplikasi pendahulu, sekaligus melakukan pengawasan jaringan.

Dibandingkan oleh penelitian pada tahun 2018 yang hanya memberdayakan 1 *method* dasar API, EVOMAC menggunakan 4 *method* dasar API antara lain, GETMethod, POSTMethod, PUTMethod, dan DELETEMethod. GETMethod berfungsi unuk mengambil data dari *server* OpenNMS lalu memberikannya pada *user* dalam bentuk informasi penting yang mudah dipahami dan diolah. POSTMethod memiliki fungsi untuk menambahkan node pada *requisition server* OpenNMS. PUTMethod berfungsi untuk memperbaharui data pada requisition server OpenNMS. Terakhir DELETEMethod memiliki fungsi untuk mengapus *node* dalam *requisition server* OpenNMS.

Pengujian kecepatan respons notifikasi EVOMAC dilakukan pada jaringan internet berkecepatan *Upload* sebesar 7,62 Mbps dan kecepatan *Download* sebesar 20,80 Mbps memperoleh tiga hasil yang berbeda dalam tiga sesi pengujian (Pagi, Siang, dan Malam) secara berturut-turut memberikan rerata kecepatan respons notifikasi sebesar 0,98s; 1,18s; 1,01s. Kecepatan dari jaringan internet terkoneksi memiliki pengaruh pada pengiriman respons EVOMAC menuju *user*. Namun, hal tersebut tak membuat EVOMAC mengalami *working failure* pada seluruh *command function* yang telah dibuktikan melalui pemberian respons sesuai ekspektasi via bot pada Telegram *messenger*. Sehingga dapat disimpulkan bahwa EVOMAC mampu dan dapat digunakan sebagai aplikasi manajemen jaringan yang optimal dengan kapasitas komputer *server* minimal CPU Quad core, 16 GB RAM, 1 TB HDD.

REFERENSI

[1] Sulhi, A., Sobri, A. M., & Alam, C. N. (2020). SELECTING NETWORK MONITORING SYSTEM SOFTWARE WITH ANALYTICAL HIERARCHY PROCESS METHOD. Jurnal Teknik Informatika Vol, 13(1), 61.

[2] Shen, G., Arnold, N., Benes, S., Jarosz, D., Johnson, A., Stasic, D., ... & Veseli, S. (2019). High-level application architecture design for the APS upgrade. In 17th Int Conf on Acc and Large Exp Physics Control Systems (pp. 1436-1440).

[3] Nugroho, Y. H., et al., 2018. *Analisis Unjuk Kerja Pemantauan Jaringan OpenNMS (Open Network Monitoring System) pada Jaringan TCP/IP*. E-Journal SPEKTRUM, Vol. 5, No. 2, pp. 158-166.

[4] Putra, R. J. et all., 2018. Pengembangan Komunikasi Multikanal untuk Monitoring Infratruktur Jaringan Berbasis Bot Telegram. E-Journal SPEKTRUM, Vol. 5, No. 2, pp. 152-157.

[5] Faisal, M., 2019. *Network Monitoring System Analysis Using OpenNMS to Analyze the Irregularities of the Internet Network*. IEEE, pp. 1-3.

[6] N. Kvartalnyi. (2020) INOXOFT. [Online]. Available: https://inoxoft.com/10-node-js-advantages/

[7] O'Donnell, S., 2018. Network Management: Open Source Solutions to Proprietary Problems. OpenNMS, pp. 1-10.

[8] Panjaitan, F. dan Rusmin, S., 2019. Pemanfaatan Notifikasi Telegram Untuk Monitoring Jaringan. Jurnal SIMETRIS, Vol. 10 No. 2, pp. 725-732.

[9] R. Christos. (2021) SNMP Center. [Online]. Available: https://www.snmpcenter.com/fcaps-network-management/

[10] Silva, J. D. C., Rodrigues, J. J. P., Saleem, K., Kozlov, S. A., & Rabêlo, R. A. (2019). M4DN. IoT-A networks and devices management platform for internet of things. IEEE Access, 7, 53305-53313.

[11] Anto, M. W., & Rizki, S. N. (2021). ANALISIS QOS JARINGAN WIRELESS LOCAL AREA NETWORKDIREKTORAT JENDRAL PAJAK BATAM. Computer and Science Industrial Engineering (COMASIE), 4(3), 87-95.

[12] Faisal, M. NETWORK MONITORING SYSTEM ANALYSIS USING OPENNMS TO ANALYZE THE IRREGULARITIES OF THE INTERNET NETWORK.

[13] J. de C. Silva, P. H. M. Pereira, L. L. de Souza, C. N. M. Marins, G. A. B. Marcondes and J. J. P. C. Rodrigues, "Performance Evaluation of IoT Network Management Platforms," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 259-265, doi: 10.1109/ICACCI.2018.8554364.

[14] Tripathi, A. (2020). TECHNOLOGY ENHANCEMENT IN IT RESOURCES MONITORING USING SMART DEVICES. Journal of Critical Reviews, 7(19), 9894-9898.

[15] M. Chuck. (2021) TechTarget. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/network-management-system.

[16] Sigalingging, Y. T. (2019). Implementasi Network Audit pada Wireless Distribution System (WDS). Jurnal Aksara Elementer, 8(2).

[17] Steinke, M., & Hommel, W. (2018, November). Overcoming Network and Security Management Platform Gaps in Federated Software Networks. In 2018 14th International Conference on Network and Service Management (CNSM) (pp. 295-299). IEEE.

[18] Chahal, D., Kharb, L., & Choudhary, D. (2019). Performance Analytics of Network Monitoring Tools. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8, 2572-2577.

[19] Surianarayanan, C., & Chelliah, P. R. (2019). Cloud Monitoring. In Essentials of Cloud Computing (pp. 241-254). Springer, Cham.

[20] Kraege, V., Vollenweider, P., Waeber, G., Sharp, S. J., Vallejo, M., Infante, O., ... & Marques-Vidal, P. (2019). Development and multi-cohort validation of a clinical score for predicting type 2 diabetes mellitus. *PloS one*, *14*(10), e0218933.

[21] Utama, I., Jaya Sasmita, I., & Jasa, L. (2020). Manajemen Jaringan Internet di Dinas Kesehatan Provinsi Bali Dengan Menggunakan Hierarchical Token Bucket. Majalah Ilmiah Teknologi Elektro, 19(2), 163-170. doi:10.24843/MITE.2020.v19i02.P07.

[22] Putra, I., Adnyana, M., & Jasa, L. (2021). Analisis Quality of Service Pada Jaringan Komputer. Majalah Ilmiah Teknologi Elektro, 20(1), 95-102. doi:10.24843/MITE.2021.v20i01.P11.

[23] Gaol, S. N. L. (2019). Implementasi Network Traffic Monitoring dengan OpenNMS pada Jaringan Dual Stack. Jurnal Aksara Komputer Terapan, 8(1).