Journal of Artificial Intelligence and Software Engineering
Vol. 5, No. 1, Maret 2025, pp. 283-295
E-ISSN: 2777-001X, DOI: 10.30811/jaise.v5i1.6516

Implementation of Network Performance Monitoring (NPM) in Nagios for Sending Alert Notifications Via Telegram

Munawir [1], Khairul Muttaqin[2], Weldyan Rezeki[3]
[1,2,3] Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Samudra , Kota Langsa, 24416, Indonesia

Informasi Artikel

Diterima : 21 Februari 2025
Revisi : 5 Maret 2025
Publikasi : 20 Maret 2025

Kata Kunci:

Nagios
Network Performance *Monitoring*
Telegram
Pemantauan Jaringan

ABSTRAK

Stabilitas dan kinerja jaringan komputer merupakan faktor krusial dalam mendukung berbagai aktivitas. Pemantauan kinerja jaringan (*Network Performance Monitoring* (NPM) diperlukan untuk mendeteksi dan menangani gangguan secara efektif. Penelitian ini bertujuan untuk menerapkan NPM menggunakan Nagios yang terintegrasi dengan Telegram sebagai media notifikasi *real-time*. Metode yang digunakan adalah pendekatan eksperimen dengan pengamatan langsung pada tiga server Linux yang dijalankan melalui *virtual machine*. Implementasi mencakup instalasi dan konfigurasi Nagios untuk memantau *host* serta *service*, serta integrasi dengan bot Telegram guna mengirimkan notifikasi otomatis. Hasil penelitian menunjukkan bahwa perbedaan waktu deteksi antara server 1, 2, dan 3 dapat disebabkan oleh keterlambatan jaringan pada *Virtual Machine* yang mempengaruhi kecepatan komunikasi antar sistem. Sementara itu, variasi signifikan dalam waktu pengiriman notifikasi ke Telegram terjadi akibat lambatnya koneksi jaringan, yang menyebabkan delay dalam proses pengiriman pesan. Sistem dapat mengirim notifikasi peringatan ke grup Telegram dalam rata-rata waktu 51 detik setelah perubahan status terdeteksi. Nagios terbukti efektif dalam memantau kondisi *host* dan *service*, sementara Telegram memungkinkan penyampaian informasi secara cepat dan sehinga dapat meningkatkan respons tim dalam menangani gangguan jaringan.

ABSTRACT

The stability and performance of computer networks are crucial factors in supporting various activities. Network Performance Monitoring (NPM) is essential for detecting and addressing network issues effectively. This study aims to implement NPM using Nagios, integrated with Telegram as a real-time notification medium. The method used is an experimental approach with direct observation on three Linux servers running on a virtual machine. The implementation includes the installation and configuration of Nagios to monitor hosts and services, as well as the integration with a Telegram bot for automatic notifications. The research results indicate that the difference in detection time between servers 1, 2, and 3 is caused by network delays in VirtualBox, which affect communication speed between systems. Meanwhile, the significant variation in notification delivery time to Telegram is due to slow network connections, causing delays in message transmission. Furthermore, the system can send warning notifications to a Telegram group within an average time of 51 seconds after a status change is detected. Nagios has proven effective in monitoring host and service conditions, while Telegram enables fast information delivery and enhances the team's response in handling network disruptions.

***Munawir**

Email: munawir@unsam.ac.id

Cara sitasi IEEE:

Munawir, K. Muttaqin, dan W. Rezeki, "Implementation of Network Performance Monitoring (NPM) in Nagios for Sending Alert Notifications Via Telegram," *Journal of Artificial Intelligence and Software Engineering (J-AISE)*, vol. 5, no. 1, pp. 283-295, Maret 2025. doi:10.30811/jaise.v5i1.6516

1. PENDAHULUAN

Dalam era digital saat ini, jaringan komputer menjadi tulang punggung utama bagi berbagai jenis layanan dan aktivitas di banyak organisasi. Stabilitas dan kinerja jaringan yang optimal sangat penting untuk mendukung keberlangsungan bisnis, pendidikan, ma*up*un sektor lain yang bergantung pada teknologi informasi. Oleh karena itu, pemantauan kinerja jaringan atau *Network Performance Monitoring* (NPM) menjadi aspek yang krusial untuk mengidentifikasi dan mengatasi potensi masalah yang dapat mempengaruhi performa jaringan.

Network *Monitoring System* adalah sebuah *tool* yang digunakan untuk melakukan *Monitoring* ataupun pengawasan pada hal-hal tertentu didalam jaringan komputer, *Monitoring* dilakukan untuk mengetahui suatu masalah sedini mungkin agar dapat langsung ditangani dengan maksimal sehingga permasalahan tersebut tidak akan meluas, serta untuk mengetahui performa dari suatu perangkat ata*up*un jaringan yang nantinya akan dianalisa oleh seorang administrator[1]. Penggunaan metode *Network Performance Monitoring* (NPM) dalam penelitian ini didasarkan pada beberapa alasan yang kuat. Pertama, NPM memungkinkan pemantauan performa jaringan secara real-time, sehingga masalah yang muncul dapat segera terdeteksi dan ditangani sebelum berdampak signifikan pada operasional. Dengan menggunakan platform Nagios, sistem dapat memonitor berbagai aspek jaringan, termasuk status *host* dan layanan, sehingga memberikan gambaran yang jelas mengenai kondisi jaringan. Kedua, metode ini mendukung otomatisasi notifikasi melalui integrasi dengan Telegram, di mana peringatan akan dikirim secara langsung saat terjadi perubahan status, seperti ketika *host* atau layanan mengalami gangguan.

Salah satu perangkat lunak yang sering digunakan untuk pemantauan jaringan adalah Nagios. Nagios merupakan salah satu aplikasi opensource untuk memonitor dan memiliki banyak plugin, salah satunya melalui Telegram [2]. Dan Nagios juga mampu memberikan notifikasi ketika ada gangguan atau penurunan kinerja pada jaringan, memungkinkan *Network* administrator untuk merespons dengan cepat dan mencegah kerugian yang lebih besar. Nagios dipilih karena merupakan perangkat lunak *open-source* yang andal dalam memantau jaringan dan server secara komprehensif, termasuk status *host* dan layanan. Nagios mendukung pengiriman notifikasi otomatis melalui Telegram, memudahkan administrator menerima peringatan secara real-time. Fleksibilitasnya memungkinkan penyesuaian sesuai kebutuhan penelitian, seperti integrasi dengan plugin khusus. Selain itu, dukungan komunitas yang luas dan dokumentasi yang lengkap mempermudah implementasi. Dengan efisiensi sumber daya dan relevansi dengan konsep *Network Performance Monitoring* (NPM), Nagios menjadi pilihan tepat untuk penelitian ini, terutama dalam aplikasi praktis di dunia industri.

Namun, salah satu tantangan dalam sistem pemantauan ini adalah cara penyampaian notifikasi yang efektif dan efisien kepada tim yang bertugas. Sistem notifikasi adalah sebuah sistem yang mampu memberikan pesan secara realtime dalam bentuk laporan[3]. Mengirim notifikasi melalui email terkadang kurang responsif karena tidak semua anggota tim dapat memantau email secara *real-time*. Oleh karena itu, penggunaan aplikasi perpesanan instan seperti Telegram sebagai media notifikasi semakin diminati karena Telegram menawarkan fitur notifikasi yang lebih cepat, fleksibel.

Telegram adalah aplikasi pesan instan berbasis cloud yang fokus pada kecepatan dan keamanan Telegram dirancang untuk memudahkan pengguna saling berkirim pesan teks, audio, video, gambar dan sticker dengan secara default, seluruh konten yang ditransfer akan dienskripsi berstandar internasional[4]. Telegram dipilih dalam penelitian ini karena memiliki API yang komprehensif dan mudah digunakan, memungkinkan integrasi dengan Nagios tanpa pengaturan yang rumit. Telegram juga dikenal dengan kecepatan dan keandalannya dalam mengirim notifikasi, bahkan pada koneksi internet yang tidak stabil. Selain itu, fitur dukungan pesan multimedia memungkinkan notifikasi dikustomisasi dengan teks, gambar, atau dokumen yang lebih informatif. Telegram mendukung pengiriman ke gr*up* atau channel, memudahkan tim administrator untuk menerima notifikasi secara bersamaan dan meningkatkan koordinasi. Aksesibilitasnya yang independen pada berbagai perangkat, seperti *smartphone*, *desktop*, dan web browser, memastikan notifikasi dapat diterima di mana saja dan kapan saja. Dengan tambahan fitur keamanan, privasi, serta tanpa biaya penggunaan, Telegram menjadi solusi yang efektif dan efisien untuk mendukung sistem *Network Performance Monitoring* (NPM) yang digunakan dalam penelitian ini.

Integrasi Nagios dengan Telegram mer*up*akan langkah penting dalam meningkatkan efisiensi dan responsivitas dalam memantau jaringan dan sistem. Nagios memiliki banyak fitur seperti laporan, event handler, *Monitoring* sumber daya (beban CPU, memori, status *up*/*down*, *up*time, lalu lintas data, bandwidth)

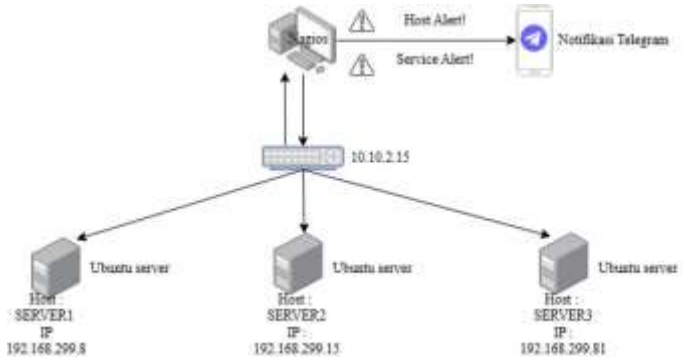
dan lain-lain. Salah satu fitur penting yang dimiliki oleh Nagios adalah peringatan notifikasi peringatan. Merupakan fitur yang akan berfungsi ketika salah satu perangkat mengalami masalah. Fitur ini akan menginformasikan kepada administrator jaringan atau orang yang berwenang di divisi tertentu mengenai jaringan yang error[5]. Untuk melakukan integrasi, harus membuat bot Telegram dan mendapatkan token API-nya. Bot ini akan bertindak sebagai perantara antara Nagios dan Telegram untuk mengirim notifikasi. Selanjutnya, perlu men*down*load PHP-Script Telegram notifikasi pada *library* Nagios yang memungkinkan Nagios mengirimkan notifikasi ke bot Telegram dengan mudah. PHP-Script ini memungkinkan Nagios untuk mengirim pesan yang dikustomisasi sesuai dengan perubahan status *host* atau layanan yang dipantau. Oleh karena itu, dengan memanfaatkan fitur telegram Bot pada Telegram Messenger maka informasi tentang kondisi jaringan dapat diketahui oleh *Network administrator* melalui aplikasi Telegram Messenger yang terpasang pada smartphone milik *Network administrator*[6]. Penggunaan notifikasi *alert* Nagios dengan Telegram Messenger dapat mengirimkan notifikasi kepada *Network* Administrator ber*up*a chat pada gr*up* Telegram apabila *host* dan *service* pada suatu jaringan dalam kondisi mati (*down*) atau mengalami masalah dengan waktu rata-rata pengiriman notifikasi dari waktu Nagios mendeteksi perubahan status yaitu 5-10 menit[7].

Tujuan Penelitian ini adalah menerapkan *Network Performance Monitoring* (NPM) menggunakan Nagios untuk memantau kinerja jaringan dan mengintegrasikan Nagios dengan Telegram untuk mengirim notifikasi *alert* secara real-time. Penelitian ini akan mengkaji penerapan Nagios untuk (NPM) *Network Performance Monitoring* dengan fokus pada implementasi sistem notifikasi alert melalui Telegram. Dengan mengintegrasikan Nagios dan Telegram, diharapkan proses pengiriman notifikasi menjadi lebih cepat dan efisien, sehingga gangguan pada jaringan dapat segera diatasi oleh pengelola jaringan.

2. METODE

Penelitian ini menggunakan metode eksperimen untuk menerapkan dan menguji sistem *Network Performance Monitoring* (NPM) menggunakan Nagios yang diintegrasikan dengan Telegram untuk mengirim notifikasi alert secara *real-time*. Menurut Kerlinger (1986) eksperimen adalah suatu penelitian ilmiah dimana peneliti memanipulasi dan mengontrol satu atau lebih variabel bebas dan melakukan pengamatan terhadap variabel-variabel terikat untuk menemukan variasi yang muncul bersamaan dengan manipulasi terhadap variabel bebas tersebut[8]. Eksperimen dilakukan dengan menguji kinerja dan efektivitas dari Nagios dalam memantau server, serta bagaimana notifikasi dikirimkan melalui Telegram ketika terjadi perubahan status pada jaringan. Penelitian dilakukan menggunakan *virtual machine* dengan 3 server sebagai bahan uji coba Nagios dan notifikasi status *host* "*up*" dan "*down*".

2.1. Topologi Jaringan

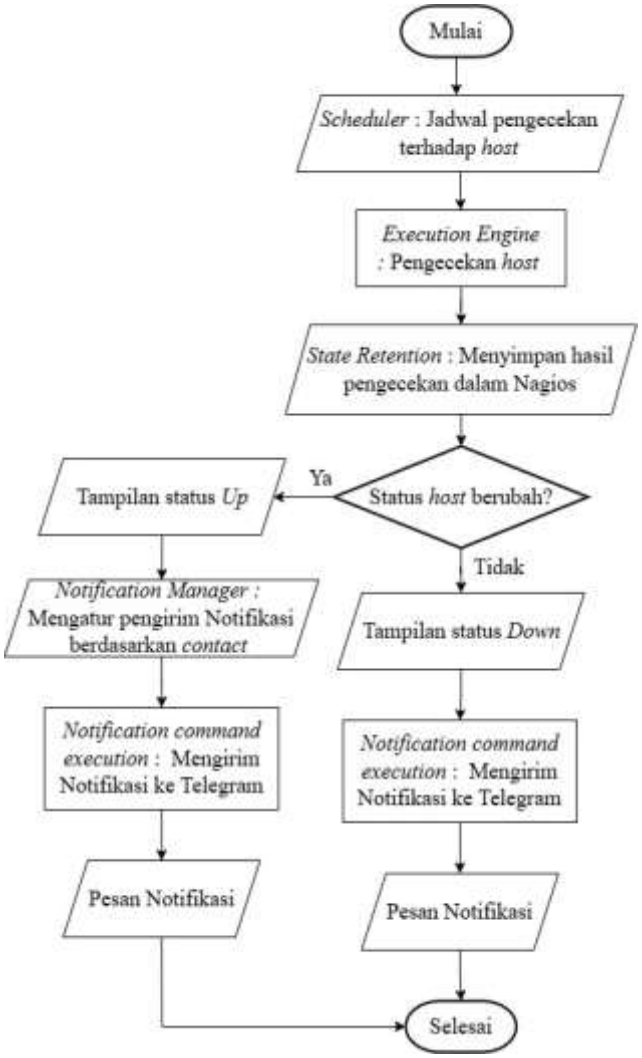


Gambar 1. Topologi Jaringan

Pada Gambar 1 menampilkan topologi jaringan yang digunakan dalam penelitian ini yaitu topologi *star*. Dalam topologi Star, komputer-komputer terhubung melalui kabel ke sebuah komponen secara terpusat yang disebut dengan hub/switch[9]. Dalam topologi Penelitian ini terdiri dari beberapa komponen utama yaitu komputer yang berperan sebagai pusat pemantauan yang telah diinstal dan dikonfigurasi dengan perangkat lunak Nagios. Selain itu, Nagios juga telah diintegrasikan dengan bot Telegram untuk mendukung pengiriman notifikasi secara otomatis. Topologi ini mencak*up* lima server yaitu SERVER1, SERVER2, dan SERVER3 yang terhubung ke komputer pusat, di mana status dan kondisi dari masing-masing server akan dipantau secara real-time oleh Nagios. Setiap perubahan atau kejadian penting pada server, seperti *down time* atau masalah kinerja, akan segera dikirimkan oleh Nagios ke bot Telegram, sehingga pengguna dapat menerima notifikasi langsung melalui aplikasi Telegram. Dengan integrasi ini, proses pemantauan jaringan menjadi lebih efektif dan responsif terhadap setiap insiden yang terjadi di server.

2.2. Proses Notifikasi Host Down ke Up

Seluruh proses notifikasi pada nagios dibuat dalam bentuk *flowchart* yang bertujuan untuk menggambarkan suatu tahapan penyelesaian masalah sederhana, terurai, rapi, dan jelas menggunakan simbol-simbol yang telah ditentukan. *Flowchart* merupakan gambaran berbentuk suatu grafik yang disertai langkah-langkah dan urutan suatu prosedur dari suatu program[10].



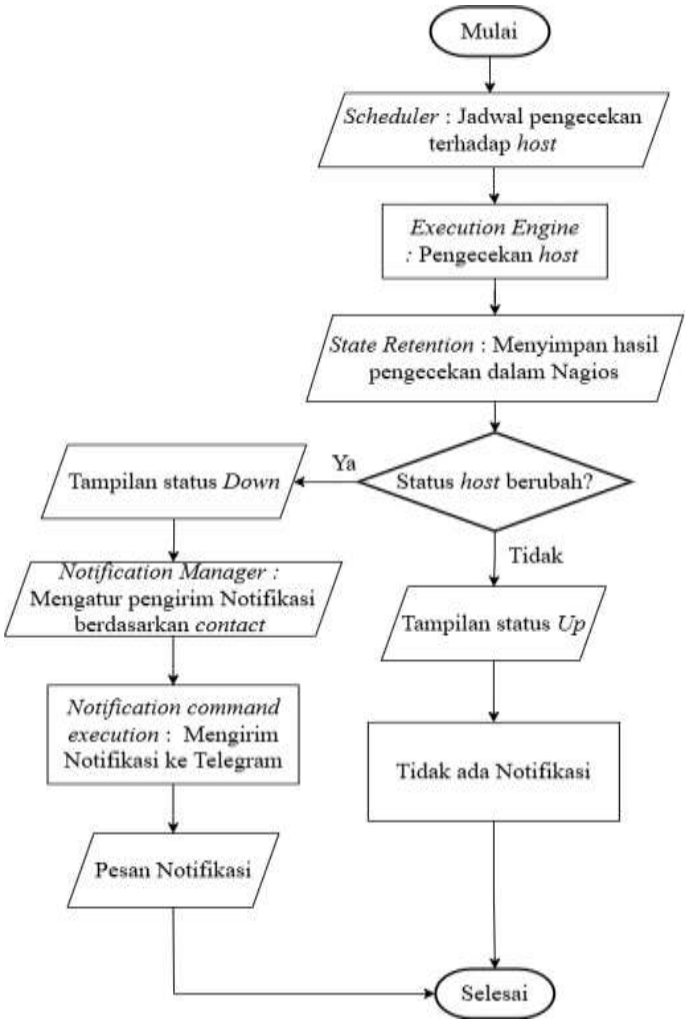
Gambar 2. Alur status notifikasi *host* ketika *up*

Pada Gambar 2 menampilkan proses notifikasi *host UP* di mulai dari *scheduler* yaitu komponen nagios yang bertugas untuk menjadwalkan pengecekan *host* berdasarkan waktu yang sudah ditentukan dalam konfigurasi, lalu *Execution Engine* yang melakukan pengecekan pada *host* sesuai dengan jadwal yang sudah di konfigurasi, lalu *State retention* yaitu proses penyimpanan hasil pengecekan kedalam *database* internal milik Nagios. Kemudian *Event Manager* yang akan memproses perubahan pada *host* (dari *DOWN* ke *UP*), dan *event manager* yang akan memicu *notification manager*. Kemudian notification *command execution* yang akan menjalankan *notification command* untuk mengirim notifikasi. Jika *host* tetap status *DOWN* maka *notification command execution* akan mengirim notifikasi *alert* ke Telegram sesuai dengan interval waktu yang sudah di konfigurasikan. Dan jika status *host* berubah menjadi *UP* maka *notification command execution* juga akan mengirim notifikasi peringatan ke Telegram sesuai dengan konfigurasi pada Nagios.

2.3. Proses Notifikasi Host Up ke Down

Gambar 3 menampilkan proses notifikasi *host DOWN* di mulai dari *scheduler* yaitu komponen nagios yang bertugas untuk menjadwalkan pengecekan *host* berdasarkan waktu yang sudah ditentukan dalam konfigurasi, lalu *Execution Engine* yang melakukan pengecekan pada *host* sesuai dengan jadwal yang sudah di konfigurasi, lalu *State retention* yaitu proses penyimpanan hasil pengecekan kedalam *database* internal milik nagios. Kemudian *Event Manager* yang akan memproses perubahan pada *host* (dari *DOWN* ke *UP*), dan event

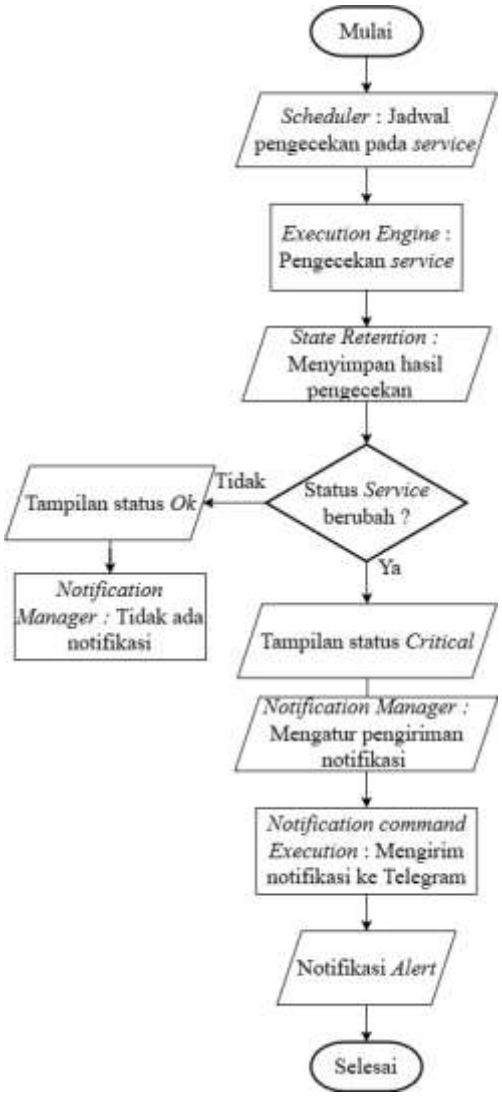
manager yang akan memicu notification manager. Kemudian *notification command execution* yang akan menjalankan *notification command* untuk mengirim notifikasi. Jika *host* tetap dalam status *UP* maka *notification command execution* tidak akan mengirim pesan ke Telegram karena dianggap tidak ada masalah, tetapi jika *host* berubah status ke *down* maka *notification command execution* akan mengirim notifikasi ke Telegram.



Gambar 3. Alur notifikasi *host* ketika *down*

2.4. Proses Notifikasi *Service* pada *Host*

Gambar 4 menampilkan proses notifikasi *service* Ok atau *Critical* di mulai dari *scheduler* yaitu komponen nagios yang bertugas untuk menjadwalkan pengecekan *service* berdasarkan waktu yang sudah ditentukan dalam konfigurasi, lalu *Execution Engine* yang melakukan pengecekan pada *service* sesuai dengan jadwal yang sudah di konfigurasi, lalu *State retention* yaitu proses penyimpanan hasil pengecekan kedalam *database* internal milik Nagios. Kemudian *Event Manager* yang akan memproses perubahan pada *service* (dari *Ok* ke *Critical* atau *Critical* ke *Ok*), dan *event manager* yang akan memicu *notification manager*. Kemudian *notification command execution* yang akan menjalankan *notification command* untuk mengirim notifikasi. Jika ada *service* yang tetap dalam status OK maka *notification command execution* tidak akan mengirim pesan ke Telegram karena dianggap tidak ada masalah, tetapi jika ada *service* yang berubah status ke *Critical* maka *notification command execution* akan mengirim notifikasi ke Telegram.



Gambar 4. Alur notifikasi *service* ketika status *host* nya *up*

3. HASIL DAN PEMBAHASAN

Setelah melakukan pengujian yang bertujuan untuk memastikan notifikasi agar dapat berjalan dengan semestinya. Pengujian ini akan dilakukan pengujian terhadap beberapa performa dan layanan *service* yang terdapat pada sistem. Terdapat beberapa komponen utama yang diuji diantaranya adalah pemantauan *host*, layanan (*Service*) (HTTP, PING dan SSH), dan notifikasi alert Telegram. Untuk pengujian notifikasi di uji sebanyak 3 kali untuk mendapatkan hasil yang akurat. Adapun pengujian yang akan dilakukan adalah :

1 Pengujian pemantauan status *host* "*up* "dan "*down*".
2 Pengujian pemantauan status *service* ketika *host* "*up*".
3 Pengujian notifikasi telegram ketika status *host* "*up*" dan "*down*".
4 Pengujian notifikasi *service* ke telegram ketika status *host up.*

3.1. Pemantauan Durasi Perubahan Status pada Host

Tabel 1. Hasil uji coba pemantauan perubahan status *host* pada saat status *Up*

No	*Host*	Status	Jam aktif server	Jam terdeteksi Nagios	Durasi
1	Server1	*Up*	16 : 22 : 13	16 : 22 : 17	4 detik
	Server1	*Up*	16 : 34 : 31	16 : 34 : 35	1 detik
	Server1	*Up*	16 : 56 : 32	16 : 56 : 36	4 detik
2	Server2	*Up*	17 : 17 : 08	17 : 17 : 40	32 detik
	Server2	*Up*	17 : 30 : 46	17 : 30 : 50	19 detik
	Server2	*Up*	17 : 43 : 30	17 : 45 : 55	145 detik
3	Server3	*Up*	18 : 03 : 50	18 : 03 : 55	5 detik
	Server3	*Up*	18 : 21 : 29	18 : 21 : 30	1 detik
	Server3	*Up*	18 : 40 : 57	18 : 42 : 04	173 detik

Tabel 1 terdapat hasil uji coba pemantauan *host* Server1, Server2 dan Server3 masing-masing sebanyak 3 kali menghid*up*kan *host* untuk mendapatkan waktu yang bervariasi. Pada Server1 durasi terdeteksi tercepat adalah 1 detik, Server2 19 detik, dan Server3 1 detik.

Tabel 2. Hasil uji coba pemantauan perubahan status *host* pada saat status *down*

No	*Host*	Status	Jam nonaktif Server	Jam terdeteksi Nagios	Durasi (Detik)
1	Server1	*Down*	16 : 32 : 05	16 : 32 : 17	12 detik
	Server1	*Down*	16 : 39 : 05	16 : 39 : 35	30 detik
	Server1	*Down*	17 : 02 : 32	17 : 06 : 36	244 detik
2	Server2	*Down*	17 : 25 : 28	17 : 27 : 44	136 detik
	Server2	*Down*	17 : 39 : 52	17 : 40 : 20	62 detik
	Server2	*Down*	17 : 55 : 29	17 : 56 : 29	30 detik
3	Server3	*Down*	18 : 15 : 06	18 : 19 : 02	244 detik
	Server3	*Down*	18 : 30 : 03	18 : 31 : 34	91 detik
	Server3	*Down*	18 : 47 : 06	18 : 47 : 08	2 detik

Tabel 2 terdapat hasil uji coba dari Server1, Server2, dan Server3 yang masing-masing sebanyak 3 kali mematikan *host* untuk mendapatkan waktu yang bervariasi. Pada Server1 durasi tercepat adalah 12 detik, pada Server2 30 detik, dan pada Server3 2 detik.

Tabel 3. Hasil uji coba pemantauan status *service* pada server1

No	*Service*	Status	Jam Server1 *Up*	Jam *Service* terdeteksi	Durasi (Detik)
1	HTTP	*Ok*	16 : 22 : 13	16 : 28 : 11	358 detik
	PING	*Ok*	16 : 22 : 13	16 : 30 : 16	483 detik
	SSH	*Ok*	16 : 22 : 13	16 : 22 : 23	10 detik
2	HTTP	*Ok*	16 : 34 : 30	16 : 36 : 31	121 detik
	PING	*Ok*	16 : 34 : 30	16 : 36 : 19	109 detik
	SSH	*Ok*	16 : 34 : 30	16 : 36 : 16	106 detik
3	HTTP	*Ok*	16 : 56 : 26	16 : 58 : 19	113 detik
	PING	*Ok*	16 : 56 : 26	17 : 00 : 16	230 detik
	SSH	*Ok*	16 : 56 : 26	16 : 56 : 31	5 detik

Tabel 3 terdapat hasil uji coba pemantauan pada *host* Server1, Server2, dan Server3 masing-masing sebanyak 3 kali mematikan *host* untuk mendapatkan waktu yang bervariasi. Pada Server1 durasi tercepat adalah 12 detik, pada Server2 30 detik, dan pada Server3 2 detik. Rata – rata durasi pada ketiga server saat di uji coba adalah 94 detik.

Tabel 4. Hasil uji coba pemantauan status *service* pada server2

No	*Service*	Status	Jam Server2 *Up*	Jam *Service* terdeteksi	Durasi (detik)
1	HTTP	*Ok*	17 : 17 : 08	17 : 24 : 10	422 detik
	PING	*Ok*	17 : 17 : 08	17 : 18 : 48	100 detik
	SSH	*Ok*	17 : 17 : 08	17 : 20 : 45	217 detik
2	HTTP	*Ok*	17 : 30 : 27	17 : 34 : 10	257 detik
	PING	*Ok*	17 : 30 : 27	17 : 38 : 48	501 detik
	SSH	*Ok*	17 : 30 : 27	17 : 30 : 45	18 detik
3	HTTP	*Ok*	17 : 43 : 30	17 : 54 : 10	680 detik
	PING	*Ok*	17 : 43 : 30	17 : 48 : 48	318 detik
	SSH	*Ok*	17 : 43 : 30	17 : 52 : 55	569 detik

Tabel 4 terdapat hasil uji coba pemantauan layanan (*service*) pada Server2 pada saat status nya *UP* sebanyak 3 kali menghasilkan waktu yang bervariasi yang tercepat pada uji coba ke-2 SSH dengan 18 detik saja dan Rata-rata waktu deteksi adalah 5 menit 9 detik. Rata – rata durasi pada *service* Server1 saat di uji coba adalah 342 detik.

Tabel 5. Hasil uji coba pemantauan status *service* pada server3

No	*Service*	Status	Jam Server3 *Up*	Jam *Service* terdeteksi	Durasi (Detik)
1	HTTP	*Ok*	18 : 03 : 50	18 : 12 : 43	547 detik
	PING	*Ok*	18 : 03 : 50	18 : 04 : 40	70 detik
	SSH	*Ok*	18 : 03 : 50	18 : 09 : 17	393 detik
2	HTTP	*Ok*	18 :21 : 29	18 : 22 : 43	74 detik
	PING	*Ok*	18 :21 : 29	18 : 24 : 40	191 detik
	SSH	*Ok*	18 :21 : 29	18 : 23 : 30	121 detik
3	HTTP	*Ok*	18 : 40 : 57	18 : 42 : 43	134 detik
	PING	*Ok*	18 : 40 : 57	18 : 44 : 40	257 detik
	SSH	*Ok*	18 : 40 : 57	18 : 43 : 30	207 detik

Tabel 5 terdapat hasil uji coba pemantauan layanan (*service*) pada Server3 pada saat status *UP* sebanyak 3 kali yang mengahasilkan waktu yang bervariasi yaitu yang tercepat pada uji coba ke-1 dengan waktu 1 menit 10 detik. Dan rata-rata waktu durasinya adalah 221 detik.

3.2. Uji Coba Notifikasi pada Host dan Service

Tabel 6. Hasil uji coba notifikasi *host* Server1 saat status *up*

No	Server	Status	Waktu Deteksi	Waktu Notifikasi
1	Server1	*Up*	4 detik	1 detik
2	Server1	*Up*	1 detik	4 detik
3	Server1	*Up*	4 detik	4 detik

Tabel 6 terdapat hasil uji coba pertama pada server1 dengan cara menghid*up*kan Server1 sebanyak 3 kali menunjukan variasi waktu deteksi dan notifikasi pada Server1. Waktu deteksi tercepat adalah 1 detik, dan notifikasi tercepat adalah 1 detik juga. Pada uji coba notifikasi status *up* pada server1 memiliki rata-rata waktu 3 detik.

Gambar 5. notifikasi server1 pada saat status *down*

Tabel 7. Uji coba notifikasi *host* Server1 "*Down*"

No	Server	Status	Waktu Deteksi	Waktu Notifikasi
1	Server1	*Down*	12 detik	30 detik
2	Server1	*Down*	30 detik	651 detik
3	Server1	*Down*	4 menit 4 detik	30 detik

Tabel 7 terdapat hasil uji coba kedua Server1 dengan cara mematikan Server1 sebanyak 3 kali menunjukan variasi waktu deteksi dan waktu notifikasi pada Server1. Waktu deteksi tercepat adalah 12 detik dan waktu notifikasi tercepat adalah 30 detik. Uji coba kedua ini menunjukan variasi waktu deteksi dan notifikasi yang besar yang menunjukan ketidakstabilan kinerja Server1. Variasi signifikan dalam waktu pengiriman notifikasi ke Telegram dapat terjadi akibat lambatnya koneksi jaringan, yang menyebabkan *delay* dalam proses pengiriman pesan. Pada uji coba notifikasi pada saat server1 *down* rata-rata waktu notifikasi adalah 237 detik.

Gambar 6. Notifikasi server1 pada saat status nya *down*

Tabel 8. Uji coba notifikasi *service* Server1

No	*Service*	Status	Waktu Deteksi	Waktu Notifikasi
1	HTTP	*Ok*	5 menit 58 detik	Tidak ada notifikasi
	PING	*Ok*	8 menit 3 detik	Tidak ada notifikasi
	SSH	*Ok*	10 detik	Tidak ada notifikasi
2	HTTP	*Ok*	2 menit 1 detik	Tidak ada notifikasi
	PING	*Ok*	1 menit 49 detik	Tidak ada notifikasi
	SSH	*Ok*	1 menit 46 detik	Tidak ada notifikasi
3	HTTP	*Ok*	1 menit 53 detik	Tidak ada notifikasi
	PING	*Ok*	3 menit 50 detik	Tidak ada notifikasi
	SSH	*Ok*	5 detik	Tidak ada notifikasi

Pada Tabel 8 terdapat hasil uji coba ketiga dengan cara mengecek status dan notifikasi *service* pada saat Server1 dalam keadaan *Up* sebanyak 3 kali menunjukan variasi waktu deteksi dan waktu notifikasi pada *service* Server1. Waktu deteksi tercepat adalah 5 detik. Dan dalam 3 kali pengujian tidak ada notifikasi yang masuk ke telegram karena semua status *service* nya Ok. Rata-rata waktu deteksi adalah 171 detik.

Tabel 9. Uji coba notifikasi *host* Server2 "*Up*"

No	Server	Status	Waktu Deteksi	Waktu Notifikasi
1	Server2	*Up*	32 detik	4 detik
2	Server2	*Up*	19 detik	4 detik
3	Server2	*Up*	2 menit 25 detik	4 detik

Pada Tabel 9 terdapat hasil uji coba pertama pada Server2 dengan cara menghid*up*kan Server2 sebanyak 3 kali menunjukan variasi waktu deteksi dan notifikasi pada Server2. Waktu deteksi tercepat adalah 19 detik, dan semua waktu notifikasi adalah 4 detik. Pada waktu deteksi bervariasi mulai dari 19 detik hingga 2 menit 25 detik, dan pada waktu pengiriman notifikasi memiliki waktu yang konsisten yaitu 4 detik.

Gambar 7. Notifikasi server2 pada saat status *up*

Pada Tabel 10 terdapat hasil uji coba kedua pada Server2 dengan cara mematikan Server2 sebanyak 3 kali menunjukan variasi waktu deteksi dan notifikasi pada Server2. Waktu deteksi tercepat adalah 30 detik dan waktu notifikasi tercepat adalah 1 menit 19 detik. Waktu deteksi relative singkat dengan rata-rata waktu notifikasi 90 detik, kemudian waktu notifikasi yang konsisten menunjukan efesiensi dalam pengiriman pemberitahuan.

Tabel 10. Uji coba notifikasi *host* Server2 "*Down*"

No	Server	Status	Waktu Deteksi	Waktu Notifikasi
1	Server2	*Down*	2 menit 16 detik	89 detik
2	Server2	*Down*	1 menit 2 detik	90 detik
3	Server2	*Down*	30 detik	90 detik

Gambar 8. Notifikasi *host* server2 pada saat status *down*

Tabel 11. Uji coba notifikasi *service* Server2

No	*Service*	Status	Waktu Deteksi	Waktu Notifikasi
1	HTTP	*Critical*	7 menit 2 detik	2 detik
	PING	*Ok*	1 menit 40 detik	Tidak ada notifikasi
	SSH	*Ok*	3 menit 37 detik	Tidak ada notifikasi
2	HTTP	*Critical*	4 menit 17 detik	2 detik
	PING	*Ok*	8 menit 21 detik	Tidak ada notifikasi
	SSH	*Ok*	18 detik	Tidak ada notifikasi
3	HTTP	*Critical*	11 menit 20 detik	2 detik
	PING	*Ok*	5 menit 18 detik	Tidak ada notifikasi
	SSH	*Ok*	9 menit 29 detik	Tidak ada notifikasi

Pada Tabel 11 terdapat hasil uji coba ketiga dengan cara mengecek status dan notifikasi *service* pada saat Server2 dalam keadaan *Up* sebanyak 3 kali menunjukan variasi waktu deteksi dan waktu notifikasi pada *service* Server2. Waktu deteksi tercepat adalah 1 menit 40 detik. Rata-rata waktu deteksi adalah 5 menit 9 detik. Dan dalam 3 kali pengujian hanya ada 1 notifikasi yang masuk ke telegram yaitu notifikasi HTTP karena status *service* nya Critical. Waktu notifikasi nya konsisten yaitu 2 detik.

Gambar 9. Notifikasi *service* pada *host* server2

Tabel 12. Uji coba notifikasi *host* Server3 "*Up*"

No	Server	Status	Waktu Deteksi	Waktu Notifikasi
1	Server3	*Up*	5 detik	113 detik
2	Server3	*Up*	1 detik	4 detik
3	Server3	*Up*	2 menit 53 detik	3 detik

Pada Tabel 12 terdapat hasil uji coba pertama pada Server3 dengan cara menghid*up*kan Server3 sebanyak 3 kali menunjukan variasi waktu deteksi dan notifikasi pada Server3. Waktu deteksi tercepat adalah 1 detik, sedangkan waktu notifikasi tercepat adalah 3 detik. Rata-rata waktu dari uji coba notifikasi pada server 3 adalah 40 detik.

Gambar 10. Notifikasi *host* server3 pada saat status *up*

Tabel 13. Uji coba notifikasi *host* Server3 "*Down*"

No	Server	Status	Waktu Deteksi	Waktu Notifikasi
1	Server3	*Down*	4 menit 4 detik	30 detik
2	Server3	*Down*	1 menit 31 detik	90 detik
3	Server3	*Down*	2 detik	30 detik

Pada Tabel 13 terdapat hasil uji coba kedua pada Server3 dengan cara mematikan Server3 sebanyak 3 kali untuk menunjukan variasi waktu yang berbeda pada Server3. Waktu deteksi tercepat adalah 2 detik dan waktu notifikasi tercepat 30 detik. Rata-rata waktu notifikasi pada saat status server *down* adalah 50 detik.

Gambar 11. Notifikasi *host* server3 pada saat status nya *down*

Tabel 14. Uji coba notifikasi *service* pada *host* Server3

No	*Service*	Status	Waktu Deteksi	Waktu Notifikasi
1	HTTP	*Critical*	9 menit 7 detik	2 detik
	PING	*Ok*	1 menit 10 detik	Tidak ada notifikasi
	SSH	*Ok*	6 menit 33 detik	Tidak ada notifikasi
2	HTTP	*Critical*	1 menit 14 detik	2 detik
	PING	*Ok*	3 menit 11 detik	Tidak ada notifikasi
	SSH	*Ok*	2 menit 1 detik	Tidak ada notifikasi
3	HTTP	*Critical*	2 menit 14 detik	1 detik
	PING	*Ok*	4 menit 17 detik	Tidak ada notifikasi
	SSH	*Ok*	3 menit 27 detik	Tidak ada notifikasi

Pada Tabel 14 terdapat hasil uji coba ketiga dengan cara mengecek status dan notifikasi *service* pada saat Server3 dalam keadaan *Up* sebanyak 3 kali menunjukan variasi waktu deteksi dan waktu notifikasi pada *service* Server3. Waktu deteksi tercepat adalah 1 menit 10 detik. Rata-rata waktu deteksi adalah 3 menit 42 detik. Dan dalam 3 kali pengujian hanya ada 1 notifikasi yang masuk ke Telegram yaitu notifikasi HTTP karena status *service* nya Critical, dan waktu notifikasi nya konsisten yaitu 2 detik.

3.3. Pembahasan

Berdasarkan hasil uji coba yang dilakukan, implementasi *Network Performance Monitoring* (NPM) dengan Nagios menunjukkan bahwa waktu deteksi dan pengiriman notifikasi ke Telegram bervariasi tergantung pada kondisi server dan layanan yang dipantau. Secara umum, hasil uji coba menunjukkan bahwa Nagios mampu mendeteksi perubahan status host maupun layanan dengan tingkat kecepatan yang cukup baik, meskipun terdapat perbedaan durasi deteksi pada masing-masing server.

Hasil pengujian menunjukkan bahwa waktu deteksi status *UP* pada Server1, Server2, dan Server3 memiliki rata-rata 43 detik, sedangkan waktu deteksi status *DOWN* rata-rata 94 detik. Variasi waktu ini dapat disebabkan oleh beberapa faktor, termasuk konfigurasi check interval dan retry interval dalam Nagios, serta kondisi jaringan yang mempengaruhi responsivitas pemantauan. Menurut dokumentasi resmi Nagios, pengaturan interval pemeriksaan dan interval percobaan ulang sangat mempengaruhi kecepatan deteksi perubahan status pada *host* dan layanan.

Dalam konteks pemantauan layanan, hasil menunjukkan bahwa waktu deteksi layanan (*service*) di setiap server memiliki durasi yang bervariasi, dengan rata-rata deteksi pada Server1 sebesar 170 detik, Server2 sebesar 342 detik, dan Server3 sebesar 221 detik. Perbedaan ini terkait dengan jenis layanan yang dipantau , beban kerja masing-masing server dan kondisi koneksi jaringan.

Kecepatan pengiriman notifikasi ke Telegram juga menunjukkan performa yang cukup baik. Notifikasi status *UP* pada server memiliki waktu pengiriman rata-rata 3-40 detik, sementara notifikasi status *DOWN* berkisar antara 50-237 detik. Variasi waktu pengiriman notifikasi ini dapat dipengaruhi oleh stabilitas jaringan dan efisiensi integrasi antara Nagios dan Telegram.

Hasil uji coba juga menemukan bahwa notifikasi pada layanan (*service*) lebih jarang terkirim dibandingkan pemantauan host. Dalam beberapa pengujian, meskipun layanan mengalami perubahan status, Nagios tidak selalu mengirim notifikasi ke Telegram, kecuali jika status layanan berubah menjadi Critical.
Berdasarkan hasil ini, beberapa aspek yang dapat ditingkatkan dalam implementasi Nagios untuk pemantauan jaringan dan notifikasi Telegram adalah:

1 Pengoptimalan interval pemantauan – Penyesuaian *check_interval* dan *retry_interval* agar deteksi lebih cepat dan stabil.
2 Peningkatan efisiensi pengiriman notifikasi – Menggunakan fitur *escalation notification* di Nagios agar notifikasi lebih adaptif terhadap kondisi layanan.
3 Peningkatan keandalan deteksi layanan – Menggunakan metode *multi-check* agar Nagios dapat lebih akurat dalam memantau kondisi layanan sebelum mengirimkan notifikasi.

Dengan optimalisasi lebih lanjut, implementasi Nagios sebagai NPM dengan notifikasi Telegram dapat memberikan manfaat yang lebih maksimal dalam manajemen insiden jaringan, meningkatkan efisiensi pemantauan, serta mengurangi *downtime* layanan secara signifikan.

4. KESIMPULAN

Berdasarkan hasil penelitian mengenai *penerapan Network Performance Monitoring* (NPM) dalam Nagios untuk mengirim notifikasi alert melalui Telegram, beberapa kesimpulan yang dapat diambil adalah penerapan *Network Performance Monitoring* (NPM) menggunakan Nagios terbukti berjalan efektif dalam mengirimkan notifikasi alert melalui Telegram untuk kondisi "*Host Down*", "*Host Up*", dan "*Service Warning*". Hasil uji coba menunjukkan bahwa rata-rata waktu penerimaan notifikasi hanya membutuhkan waktu sekitar 51 detik, yang menandakan efisiensi sistem ini. Meskipun demikian, perbedaan waktu deteksi antara server1, server2, dan server3 tidak disebabkan oleh perbedaan spesifikasi atau konfigurasi server, melainkan lebih dipengaruhi oleh kondisi jaringan yang ada pada saat itu. Variasi waktu pengiriman notifikasi ke Telegram juga tidak terkait dengan delay dari pihak Telegram, tetapi lebih kepada dampak kondisi jaringan lokal yang mempengaruhi kecepatan pengiriman notifikasi tersebut. Secara keseluruhan, Nagios berhasil memberikan peringatan secara *real-time*, khususnya untuk kondisi-kondisi kritis seperti server yang down atau *service* yang mengalami masalah, sehingga memungkinkan administrator jaringan untuk segera mengambil tindakan yang diperlukan. Secara keseluruhan, sistem NPM dengan Nagios ini dapat diandalkan dalam memantau performa jaringan dan mengirimkan notifikasi yang tepat waktu melalui Telegram, meskipun terdapat beberapa area yang perlu dioptimalkan.

REFERENSI

[1] C. Ayu and H. Saptono, "Perancangan Dan Implementasi Network Monitoring System Telegram," vol. 4, no. 1, pp. 7–18, 2018.
[2] I. Vingestin, T. U. Kalsum, and Y. Mardiana, "The Design Of Network Monitoring System Using SNMP Protocol With Telegram Notification," *J. Media Comput. Sci.*, vol. 2, no. 1, pp. 93–100, 2022.
[3] M. Alhady, E. Supratman, F. I. Komputer, and U. B. Darma, "725-Article Text-1651-1-10-20200115 Alhady," pp. 2113–2119, 2022.

[4] A. Fitria Sari, "Telegram Sebagai Media Internal (Studi Kebutuhan Informasi Karyawan Pt.Telkom Witel Sulteng)," vol. 5, no. 3, p. 33, 2018.
[5] F. Fahreza and M. Rifqi, "Nagios Core Optimization By Utilizing Telegram as Notification of Disturbance," *J. Appl. Sci. Eng. Technol. Educ.*, vol. 2, no. 2, pp. 121–135, 2020, doi: 10.35877/454ri.asci2259.
[6] M. Mukmin, P. Purnawansyah, and M. Hasnawi, "Notifikasi Bot Telegram Untuk Monitoring Jaringan Pada Kementrian Kelautan Dan Perikanan Untia," *Bul. Sist. Inf. dan Teknol. Islam*, vol. 3, no. 2, pp. 127–133, 2022, doi: 10.33096/busiti.v3i2.1162.
[7] M. H. Fikri and I. Nurhaida, "Pemantauan Jaringan Menggunakan Nagios Dan Zabbix Dengan Notifikasi Telegram Messenger Dan Google Mail," *Simetris J. Tek. Mesin, Elektro dan Ilmu Komput.*, vol. 11, no. 2, pp. 578–593, 2021, doi: 10.24176/simet.v11i2.5320.
[8] Safrin, "Pendekatan Eksperimental dalam Penelitian Komunikasi," *Talent. Conf. Ser. Local Wisdom, Soc. Arts*, vol. 3, no. 1, 2020, doi: 10.32734/lwsa.v3i1.810.
[9] A. Subki, M. N. Karim, and J. Juhartini, "Pengembangan Jaringan Hotspot Menggunakan Mikrotik Routerboard RB951UI-2HND pada SMKN 2 Selong," *Explore*, vol. 10, no. 1, p. 14, 2020, doi: 10.35200/explore.v10i1.43.
[10] Malabay, "Pemanfaatan Flowchart Untuk Kebutuhan Deskripsi Proses Bisnis," *J. Ilmu Komput.*, vol. 12, no. 1, pp. 21–26, 2016, [Online]. Available: https://digilib.esaunggul.ac.id/pemanfaatan-flowchart-untuk-kebutuhan-deskripsi-proses-bisnis-9347.html
[11] W. L. Ogogo, "Real-Time Monitoring of Network Devices: Its Effectiveness in Enhancing Network Security," *East African J. Inf. Technol.*, vol. 3, no. 1, pp. 1–6, 2021, doi: 10.37284/eajit.3.1.153.