Implementasi *Network Monitoring System* Nagios dengan *Event Handler* dan Notifikasi Telegram *Messenger*

Tubagus Habibullah, Defiana Arnaldy
Program Studi Teknik Informatika
Jurusan Teknik Informatika dan Komputer
Politeknik Negeri Jakarta
tubagus.habibullah@gmail.com, defiana.arnaldy@tik.pnj.ac.id

Diterima: 9 Maret 2016. Disetujui: 16 April 2016. Dipublikasikan: Mei 2016

Abstrak* - Jaringan komputer sekarang ini sudah seperti sebuah kebutuhan, sehingga diperlukan adanya pemeliharaan dan kontrol teratur untuk menjamin agar jaringan dapat berjalan sebagaimana mestinya. Oleh sebab itu diperlukan sebuah fasilitas pendukung yaitu network monitoring system agar network administrator dapat memonitor suatu perangkat ataupun service pada suatu jaringan meskipun tidak di depan komputer secara langsung. Nagios merupakan salah satu aplikasi open source untuk network monitoring dan memiliki banyak fitur seperti monitoring, notifikasi alert, report, Event Handler dan lain-lain. Dalam penelitian ini diimplementasikan network monitoring system menggunakan Nagios dengan Event Handler dan notifikasi Telegram Messenger yang dapat melakukan monitoring perangkat dan services pada jaringan, dapat mengirimkan notifikasi melalui Telegram Messenger jika host atau service mengalami gangguan, serta dapat melakukan restart terhadap service yang gagal fungsi secara otomatis. Pengujian sistem dilakukan dengan menguji fungsionalitas kinerja sistem Nagios. Dari hasil pengujian didapatkan bahwa Nagios dapat mendeteksi perubahan status dengan selang waktu interval 5 menit untuk hosts dan 10 menit untuk services, notifikasi alert dapat terkirim kepada Group Telegram dengan selang waktu rata-rata 5-10 menit setelah Nagios mendeteksi adanya perubahan status down/recovery, dan Event Handler dapat melakukan restart terhadap services yang gagal fungsi dengan rata-rata waktu aksi 2 menit. Hal ini membuktikan bahwa Nagios dapat memonitoring perangkat dan services pada jaringan dengan baik sesuai dengan konfigurasi yang tel*ah dilakukan.

Kata Kunci: event handler; Nagios; network monitoring system; notifikasi alert; telegram messenger.

I. PENDAHULUAN

Jaringan komputer bukanlah sesuatu yang baru dalam teknologi informasi dan komunikasi. Pada umumnya di setiap perusahaan, instansi, ataupun institusi pendidikan saat ini sudah menggunakan jaringan komputer untuk memperlancar arus informasi ataupun pertukaran datanya. Jaringan komputer saat ini menjadi sebuah kebutuhan, sehingga diperlukan adanya pemeliharaan dan kontrol yang teratur untuk menjamin agar jaringan dapat berjalan sebagaimana mestinya.

Untuk mencegah terjadinya gangguan yang terlalu lama umumnya seorang *network administrator* selalu mengawasi dan memantau kinerja jaringan komputer. *Network administrator* tidak mungkin terus menerus mengamati dan mengawasi di depan komputer secara langsung suatu perangkat ataupun *service* yang berjalan pada sebuah jaringan selama 24 jam. Oleh sebab itu diperlukan sebuah fasilitas pendukung yaitu *netwok monitoring system* agar *network administrator* dapat memonitor suatu perangkat ataupun *service* pada suatu jaringan meskipun tidak di depan komputer secara langsung.

Nagios merupakan salah satu aplikasi *open source* untuk *network monitoring* yang memiliki banyak *plugins*dan dapat digunakan untuk memaksimalkan proses *monitoring* perangkat jaringan yang ada. Beberapa fitur yang dimiliki Nagios seperti *monitoring,* notifikasi *alert, report, event handler, monitoring resource (CPU load, memory, status up/down, up time, data traffic, bandwidth)* dan lain-lain. Nagios sendiri dapat dikonfigurasi sesuai dengan kebutuhan.

Salah satu fitur penting dalam *Network Monitoring System* adalah notifikasi *alert*, yaitu sistem pemberitahuan *notifikasi* kepada kontak *network administrator* jika sistem mendeteksi adanya permasalahan pada *host* maupun *service*. Pemberitahuan *notifikasi* dapat melalui berbagai media seperti *email*, sms, Telegram *Messenger*, dan lainnya. Selain fitur notifikasi *alert, Network Monitoring System* Nagios mempunyai fitur yang dapat melakukan *restart* terhadap *service* yang

mengalami gangguan (*error)* secara otomatis. Berdasarkan latar belakang tersebut, maka dilakukan penelitian tentang **"Implementasi *Network Monitoring System* Nagios dengan *Event Handler* dan Notifikasi Telegram *Messenger*".**

Dalam penelitian ini yang menjadi fokus permasalahan adalah bagaimana mengimplementasikan penggunaan notifikasi *status (up* dan *down)* untuk *host*, status *(unknown, critical, warning,* dan *recovery)* untuk *service* melalui Telegram pada Nagios dan bagaimana memanfaatkan *Event Handler* untuk penanganan *error pada service* di Nagios.

Dalam penelitian ini terdapat beberapa batasan masalah, yaitu:

a. Perangkat yang dimonitoring adalah Mikrotik RouterBoard RB2011, Windows *machine*, dan Linux Ubuntu.
b. *Monitoring* pada Windows menggunakan *add-ons* Nagios yaitu NSClient++ yang harus diinstall.
c. *Monitoring* pada Linux Ubuntu menggunakan *add-ons* Nagios yaitu NRPE yang harus diinstall.
d. Notifikasi *alert* dikirim melalui Telegram ketika *host* dan *service* mengalami gangguan pada jaringan.
e. *Event Handler* menangani *service* Printer Spooler pada *host* Windows dan menangani *service* FTP dan SSH pada *host* Ubuntu.

II. TINJAUAN PUSTAKA

A. Network Monitoring System

NMS (*Network Monitoring System*) adalah kumpulan sistem yang memiliki tugas mengamati atau memonitor sistem-sistem di dalam jaringan terhadap kemungkinan terjadinya masalah-masalah pada sistem tersebut, untuk dapat dideteksi secara dini. Sebagai contoh, suatu *monitoring* sistem secara berkala menghubungi sebuah *server* untuk menjamin adanya respons dari *server*, jika tidak ada respons maka sistem *monitoring* akan mengirimkan pesan atau notifikasi kepada teknisi jaringan [1].

Pengiriman notifikasi pada sistem *monitoring* jaringan tentunya ditentukan oleh nilai ambang tertentu yang dapat ditentukan /*setting* nilainya pada setiap objek yang akan dimonitor sesuai dengan kebutuhan. Penentuan media notifikasi yang akan dipakai juga dapat ditentukan /*setting* dalam sebuah system *monitoring* jaringan, seperti melalui *email* ataupun media SMS [1].

Beberapa hal yang menyebabkan *monitoring* wajib dilakukan untuk sebuah network:

a. Untuk *monitoring performance* dari suatu jaringan.
b. Mengetahui status (*up*/*down*) *service* dari *host* yang dimonitor secara *realtime* dengan sistem *alert*/alarm.
c. Media *log* dari *service* sehingga segala aktivitas *host* yang dimonitor akan tercatat.

B. Nagios

Nagios adalah sebuah *tools* untuk *monitoring* sistem. Yang berarti Nagios akan mengawasi komputer atau *device* di dalam jaringan dan memastikan bahwa komputer atau *device* bekerja dengan semestinya. Nagios secara teratur mengecek apakah mesin dalam keadaan baik atau tidak. Dan juga melakukan verifikasi berbagai macam *service* didalamnya dalam keadaan baik. [2]

Monitoring sistem di Nagios dibedakan menjadi dua kategori, yaitu *hosts* dan *service*. Hosts mewakilkan sebuah perangkat fisik atau *virtual* di dalam jaringan seperti server, router, workstation, printer, dll. *Service*s adalah sebuah fungsi partikular, sebagai contoh, sebuah Secure Shell (SSH) server (*sshd process* pada perangkat) bisa didefinisikan sebaga *service* yang akan dimonitor. Setiap *service* pasti berhubungan dengan sebuah host yang menjalankannya [2].

Gambar 1. Logo Nagios [3]

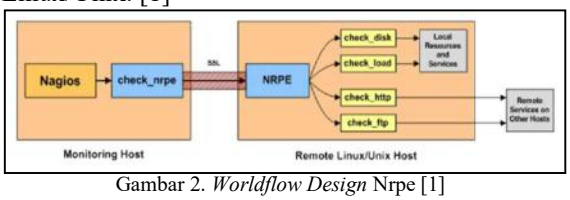
Berikut adalah fitur-fitur yang tersedia pada Nagios, antara lain:

a. Memonitor *network services* (SMTP, POP3, HTTP, PING, dan lain-lain)
b. Memonitor *host resources* (*processor load, disk usage*, dan lain-lain)
c. Desain *plugins* yang sederhana, yang memungkinkan pengguna untuk mengembangkan sendiri pemeriksaan terhadap servisnya.
d. *Service checks* yang paralel
e. *Web interface* yang fakultatif untuk melihat status *network*, urutan masalah dan notifikasi, *log file*, dan sebagainya.
f. Kemampuan untuk mendefinisikan kejadian yang ditangani selama servis atau *host* berlangsung untuk mempermudah pemecahan masalah
g. Perputaran *log file* yang otomatis
h. Notifikasi kontak ketika servis atau host terjadi masalah dan mendapat penanganan (via email, pager, sms, whatsapp, telegram atau *method* yang didefinisikan user)

C. NRPE

Nagios Remote Plugin Executor (NRPE) adalah sebuah *add-on* yang dirancang untuk mengizinkan *plugins* Nagios dapat berjalan pada

remote machines (Linux/Unix). NRPE digunakan agar Nagios dapat memantau *local resources* (seperti *CPU load, memory usage, etc.*) pada *remote machines*. Oleh karena *public resources* pada suatu mesin tidak dipublikasikan kepada mesin lain, maka harus ada sebuah *agent* NRPE yang harus di-*install* pada *remote machines* Linux/Unix. [1]



Gambar 2. *Worldflow Design* Nrpe [1]

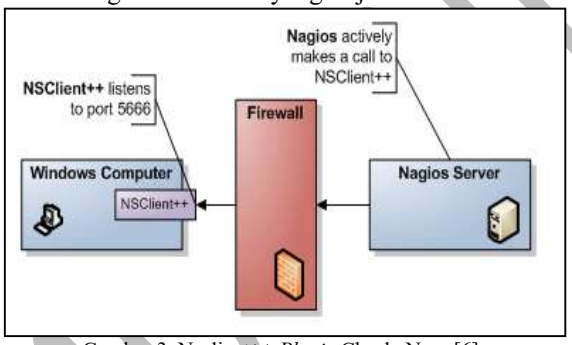
Add-ons NRPE terdiri dari dua bagian:

a. *Plugin* check_nrpe, yang berada pada *local monitoring machine*.
b. NRPE *Daemon*, yang berjalan pada *remote machines* (Linux/Unix).

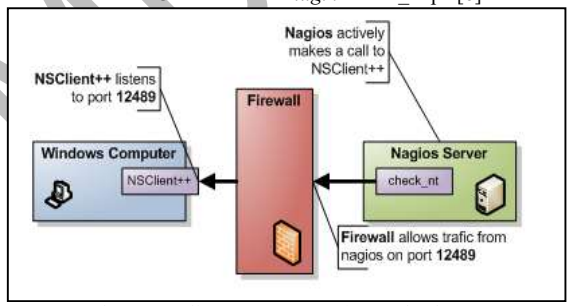
D. NSClient++

NSClient++ adalah sebuah aplikasi sederhana yang *powerfull* dan merupakan *daemon* yang fleksibel untuk *monitoring* system [4]. NSClient++ merupakan jembatan penghubung antara server *monitoring* dengan *host* yang akan dimonitor. Pada dasarnya NSClient++ bekerja pada tiga hal [5]:

a. Memungkinkan melakukan *remote checks,*
b. Memantau sistem secara *realtime,*
c. Mengatasi masalah yang terjadi.



Gambar 3. Nsclient++ *Plugin* Check_Nrpe [6]



Gambar 4. Nsclient++ Dengan *Plugin* Check_Nt [6]

E. Telegram Messenger

Telegram *Messenger* adalah aplikasi *messaging* yang berfokus pada kecepatan, keamanan, sederhana, dan dapat diunduh secara gratis. Telegram dapat digunakan di semua perangkat dalam satu akun dalam waktu yang sama. Telegram juga dapat mengirim pesan, foto, video, dan beberapa jenis file (doc, zip, mp3, dll), serta dapat membuat group hingga 5000 orang atau *channel* untuk broadcasting untuk khalayak terbatas. Beberapa fitur lain yang terdapat dalam Telegram *Messenger*: [7]

a. Enkripsi pesan, personal dan *bussiness secret.*
b. Menghapus pesan secara otomatis dengan *timer*.
c. Menyimpan media di dalam *cloud.*
d. Membangun sistem/*tools* sendiri dengan menggunakan API telegram.

Gambar 5. Logo Telegram Messenger [7]

Salah satu fitur telegram yang berbeda dengan aplikasi *messenger* lainnya adalah Telegram menyediakan API (*Application Programming Interface*) yang terbuka 100% untuk publik yang ingin mengembangkan aplikasi menggunakan API Telegram.

F. Telegram APIs

Telegram mempunyai dua jenis APIs untuk *developer,* yaitu [7]:

a. Bot API
   Bot API memungkinkan developer untuk menghubungkan Bot dengan sistem Telegram. Telegram Bots adalah sebuah akun khusus yang tidak memerlukan nomor telepon tambahan dalam pengaturannya. Akun ini berfungsi sebagai antarmuka untuk tempat berjalannya kode di pada suatu server.
b. Telegram API
   Telegram API memungkinkan developer untuk membangun sendiri Telegram *clients* yang diinginkan. Telegram API terbuka 100% untuk semua developer yang ingin membuat aplikasi dengan platform Telegram.

Gambar 6. The Botfather Telegram [7]

G. Telegram CLI

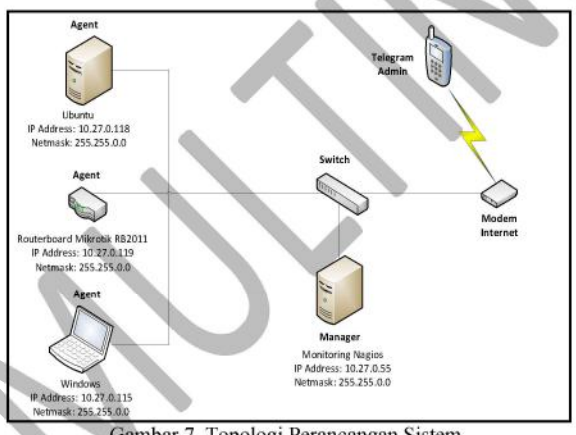
Telegram CLI adalah sebuah aplikasi *command-lineinterface* yang digunakan untuk menjalankan Telegram. Telegram CLI memanfaatkan Telegram API dan Protokol Telegram (Mtproto) dalam penggunaannya. Daftar seluruh *command* yang ada pada Telegram CLI dapat dilihat dengan mengetik *commandhelp* di dalam tg-cli-session.

III. PERANCANGAN DAN REALISASI

A. Perancangan Sistem

Sistem yang dibangun adalah implementasi *Network MonitoringSystem* Nagios dengan *Event Handler* dan Notifikasi Telegram *Messenger*. Dengan sistem *monitoring* jaringan ini, *monitoring machine* dapat memonitor objek *monitoring* di dalam jaringan secara *real-time*, dapat mendeteksi status (*up* atau *down*) serta dapat mendeteksi adanya *error* atau masalah yang terjadi pada jaringan. *Monitoring machine* akan mengirimkan notifikasi *alert* melalui Telegram *Messenger* apabila *host* maupun *service* yang berada pada jaringan dalam keadaan down/*error*. *Monitoring machine* pun dapat menangani *error* pada *service* secara otomatis dengan *Event Handler*.

Gambar 7 menjelaskan bagaimana topologi perancangan sistem terdiri dari *agent* dan *manager*. *Monitoring agent* atau *host* disini adalah Windows, Linux dan Router. Sedangkan yang bertindak sebagai *manager* adalah PC server yang terintegrasi dengan NMS Nagios.



Gambar 7. Topologi Perancangan Sistem

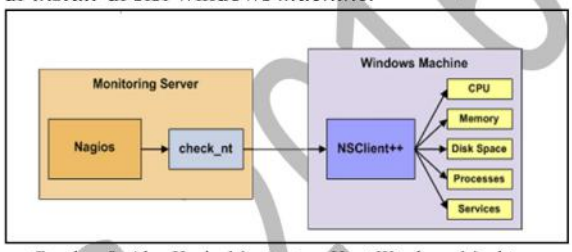
B. Cara kerja system

Cara kerja *Network Monitoring System* Nagios diuraikan menjadi 3 bagian, yaitu: Sistem *monitoring host* (Windows, Linux Ubuntu, dan Router), Sistem notifikasi *alert* dengan Telegram *Messenger,* dan Penanganan *errorservice* dengan *Event Handler.*

Terdapat beberapa *host* yang dimonitor oleh *Network Monitoring System* Nagios, yaitu:

a. *Monitoring host* Windows

Gambar 8 menjelaskan alur kerja *monitoring host* agar Nagios dapat memonitor *privateservice* dan *local resources* (CPU, *Memory*, *Disk Space*, *Process*, dan *Service*s) pada windows *machine*. *Windows machine* membutuhkan suatu *agent/daemon* yang betindak seperti proxy antara *plugin* Nagios (check_nt) dan *service* yang akan dimonitor. *Agent* tersebut adalah NSClient++ yang di-*install* di sisi windows *machine.*

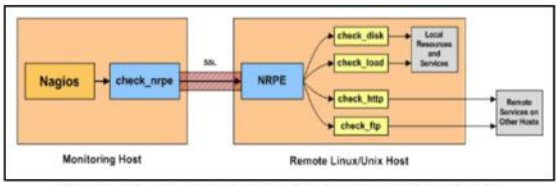


Gambar 8. Alur Kerja *Monitoring Host Windows Machine*

Cara kerja sistem *monitoringhost* windows:

a. Nagios mengeksekusi *plugin* check_nt dan akan menginformasikan *service* yang perlu diperiksa.
b. *Plugin* check_nt memanggil *daemon* NSClient++ pada *remote host* melalui *SSL-protected connection* (opsional).
c. *Daemon* NSClient++ menjalankan *plugin* Nagios (check_nt) yang sesuai untuk dilakukan pengecekan *service* atau *resource.*
d. Hasil pengecekan *service* dari *daemon* NSClient++ akan dikembalikan ke *plugin* check_nrpe, dimana hasil pengecekan tersebut akan diteruskan untuk diproses oleh Nagios.

b. *Monitoring host* Linux Ubuntu

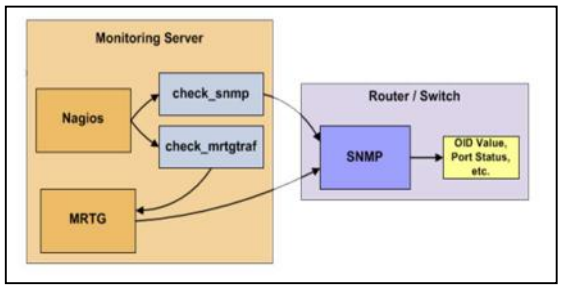


Gambar 9. Alur Kerja Monitoring Host Linux/Unix

Ubuntu merupakan salah satu dari distro linux/unix. Gambar 9 menjelaskan alur kerja *monitoringhost* agar Nagios dapat memonitor *privateservice* dan local *resource*s pada linux/unix *host*. Sama seperti *monitoringhost* Windows, *monitoring* Linux/Unix pun membutuhkan suatu *agent/daemon* yang betindak seperti proxy antara *plugin* Nagios (check_nrpe) dan *service* yang akan dimonitor. *Daemon* tersebut adalah NRPE yang harus di-*install* di sisi linux/unix *host*. [7]

Cara kerja sistem *monitoring* pada linux/unix *host*:

a. Nagios mengeksekusi *plugin* check_nrpe dan akan menginformasikan *service* yang perlu diperiksa.
b. *Plugin* check_nrpe memanggil NRPE *Daemon* pada *remote host* melalui *SSL-protected connection* (opsional).
c. *NRPE Daemon* menjalankan *plugin* Nagios (check_nrpe) yang sesuai untuk dilakukan pengecekan *service* atau *resource.*
d. Hasil pengecekan *service* dari *NRPE Daemon* akan dikembalikan ke *plugin* check_nrpe, dimana hasil pengecekan tersebut akan diteruskan untuk diproses oleh Nagios.
c. *Monitoring* Router



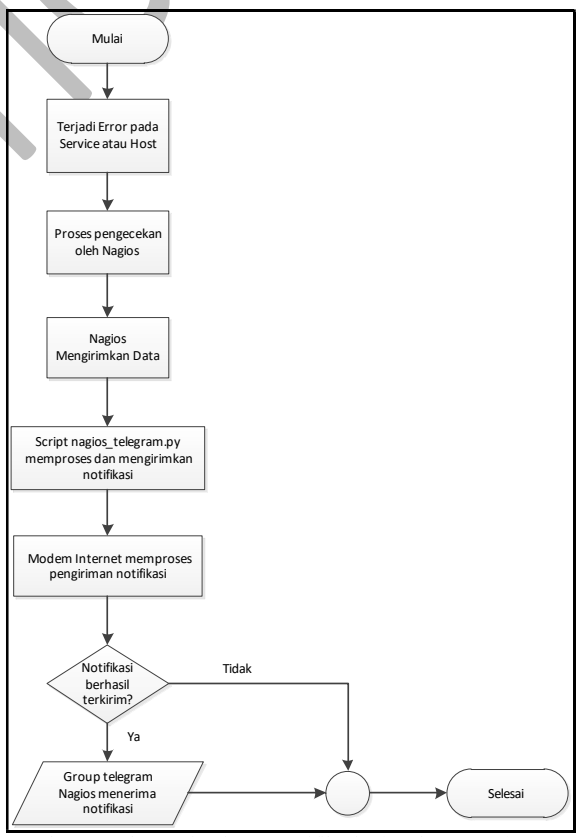
Gambar 10. Alur Kerja Sistem *Monitoring Router/Switch*

Gambar 10 menjelaskan alur kerja *monitoring host* agar Nagios dapat memonitor *service*/*resource* pada *host* router/switch. Nagios dapat memonitor *resource*/*service* dengan mengaktifkan dan melakukan konfigurasi SNMP pada *host* router/switch. Cara kerja sistem *monitoring* pada router *host*:

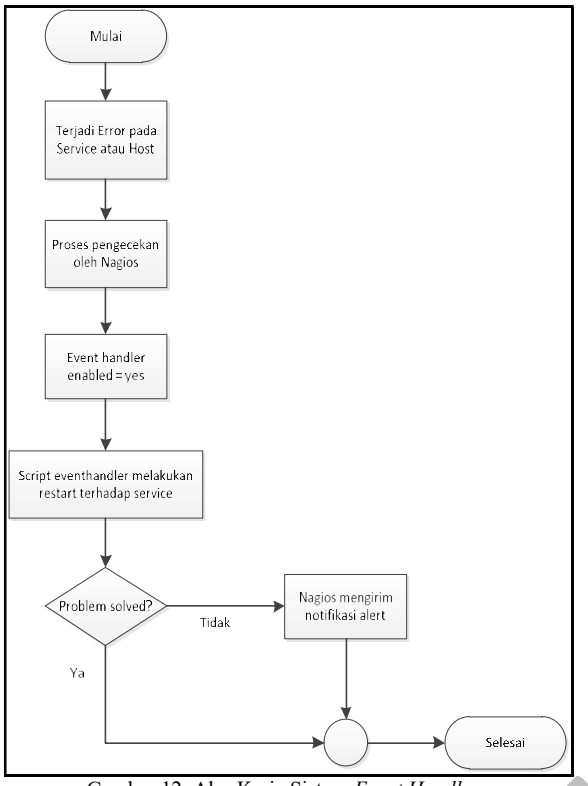
a. Nagios mengeksekusi *plugin* check_snmp dan check_mrtgtraf, kemudian menginformasikan *service* yang perlu diperiksa
b. *Plugin* check_snmp akan mengambil data dari OID *value* pada router/switch yang sesuai dengan OID *value resource*/*service* yang perlu diperiksa.
c. Router/switch akan menginformasikan OID *value* yang sesuai untuk dilakukan pengecekan *service*/*resource*.
d. Hasil pengecekan *service/resource* dari router/switch akan dikembalikan ke *plugin* check_snmp, dimana hasil pengecekan tersebut akan diteruskan untuk diproses oleh Nagios.
e. *Plugin*s check_mrtg akan mengambil data dari MRTG (Multi Router Traffic Grapher) dimana MRTG mendapatkan data dari port status yang sesuai pada router.
f. Router/switch akan menginformasikan *port status* yang sesuai untuk dilakukan pengecekan *traffic*.
g. Hasil pengecekan *traffic* dari router/switch akan dikembalikan ke MRTG, lalu diteruskan ke *plugin* check_mrtg, dan hasil pengecekan tersebut akan diterukan untuk diproses oleh Nagios.

Cara kerja sistem notifikasi *alert* dengan Telegram *Messenger* pada Gambar 11:

a. Nagios mendeteksi adanya masalah/*error* pada *service* atau *host*.
b. Nagios melakukan pengecekan pada *service* atau *host* yang mengalami masalah/*error*.
c. Nagios mengirim data *host* atau *service* yang mengalami masalah/*error* berupa *output* pada script nagios_telegram.py
d. *Script* nagios_telegram.py menerima *output* dari Nagios berupa hasil *error host* atau *service*, yang selanjutnya akan dikirim melalui *Bot* Telegram dalam bentuk *chat text* yang berisi *alertservice* yang bermasalah ke *Group* Telegram Nagios.
e. Modem internet meneruskan proses untuk mengirimkan *alert* tersebut.
f. Notifikasi *alert* dikirim oleh sript nagios_telegram.py melalui modem internet.
g. *Group* telegram Nagios akan menerima *chat text* berupa notifikasi *alert* informasi host yang mengalami masalah/*error.*



Gambar 11. Alur Kerja Sistem Notifikasi *Alert* dengan *Telegram Messenger*



Gambar 12. Alur Kerja Sistem *Event Handler*

Gambar 12 menunjukkan cara kerja penanganan *error* pada *service* dengan *Event Handler*:

a. Nagios mendeteksi adanya masalah/*error* pada *service* atau *host*.
b. Nagios melakukan pengecekan pada *service* atau *host* yang mengalami masalah/*error*.
c. Nagios mendapatkan *Event Handler* diaktifkan di dalam *service*.
d. *ScriptEvent Handler* melakukan *restart* terhadap *service* yang mengalami masalah/*error*.
e. Ketika *Event Handler* tidak dapat melakukan tugasnya, dan *service* masih dalam keadaan *error*/bermasalah, maka Nagios akan mengirim notifikasi *alert*.

IV. PEMBAHASAN

A. Pengujian

Pengujian bertujuan untuk memastikan sistem agar dapat berjalan dengan semestinya. Pengujian dilakukan dengan metode *Black Box*, dimana dilakukan pengujian terhadap beberapa fungsi yang terdapat pada sistem. Terdapat 4 komponen utama yang diuji diantaranya adalah *monitoringagent* (NSClient++, NRPE, dan SNMP), *web interface*, notifikasi Telegram, dan *Event Handler*.

B. Prosedur pengujian

a. Pengujian *monitoring agent* terhadap fungsionalitas *plugins* Nagios:
   1. Pengujian NSClient++ terhadap *plugin* check_nt
   2. Pengujian NRPE terhadap *plugin* check_nrpe
   3. Pengujian SNMP terhadap *plugin* check_snmp
b. Pengujian terhadap fungsionalitas web interface dilakukan pada autentikasi:
   1. Pengujian *hosts*
   2. Pengujian *services*
c. Pengujian terhadap fungsionalitas notifikasi Telegram
   1. Pengujian notifikasi Telegram terhadap *services*
   2. Pengujian notifikasi Telegram terhadap *hosts*
d. Pengujian terhadap fungsionalitas Event Handler

C. Data hasil pengujian

Hasil pengujian *monitoring agent* terhadap fungsionalitas *plugins* Nagios (NSClient++, NRPE, dan SNMP) akan dijelaskan pada Tabel 1.

TABEL 1. PENGUJIAN *MONITORING AGENT*

Pengujian	*Plugins*	*Service*	Output	Kesimpulan
NSClient++	Check_nt	Check_load	CPU Load 4%	Berhasil
NRPE	Check_nrpe	Check_load	*OK* – load	Berhasil
SNMP	Check_snmp	Uptime	SNMP *OK* - Timeticks 1:31:02	Berhasil

Hasil pengujian autentikasi web interface dilakukan dengan pengujian benar dan salah yang dapat dilihat pada Tabel 2.

TABEL 2. PENGUJIAN AUTENTIKASI *WEB INTEFACE*

Hasil Uji Benar			
Input	Hasil yang Diharapkan	*Output*	Kesimpulan
Data login: Username: Nagiosadmin Password: Admin	Data login berupa username dan password yang valid dapat masuk ke dalam Web Interface Nagios	Input data berhasil dilakukan dan masuk ke dalam web interface	Berhasil diuji
Hasil Uji Salah			
Mengosongkan username dan password	Tidak dapat Login dan kembali pada window autentikasi	Tidak dapat login dan diarahkan kembali pada window autentikasi	Berhasil diuji

Pengujian *hosts* diuji pada *host* Ubuntu yang dilakukan sebanyak 5 kali dengan kondisi saat *host* "*DOWN*"dan kondisi saat *host* dilakukan *recovery*. Hasil pengujian *service* saat *host* Ubuntu dalam keadaan "*DOWN*" dapat dilihat pada Tabel 3 dan "*UP*" pada Tabel 4.

TABEL 3 DATA PENGUJIAN *HOSTS* UBUNTU "*DOWN*"

Pengujian	Waktu Penonaktifan	Waktu Deteksi Nagios	Durasi*
1	11:33:03	11:33:29	0:00:26
2	12:01:34	12:01:51	0:00:17
3	12:17:14	12:17:22	0:00:08
4	12:32:35	12:33:22	0:00:47
5	12:48:16	12:48:53	0:00:37

**Durasi = Selisih Waktu Deteksi Nagios dengan Waktu Penonaktifan*

TABEL 4. DATA PENGUJIAN *HOSTS* UBUNTU "*UP*"

Pengujian	Waktu *Recovery*	Waktu Deteksi Nagios	Durasi*
1	11:48:23	11:48:24	0:00:01
2	12:13:25	12:13:26	0:00:01
3	12:29:34	12:29:47	0:00:13
4	12:44:10	12:44:11	0:00:01
5	13:01:32	13:03:27	0:01:30

**Durasi = Selisih Waktu Deteksi Nagios dengan Waktu Recovery*

Pengujian *services* diuji pada FTP dan HTTP yang dilakukan sebanyak 5 kali pengujian untuk masing-masing *service* dengan kondisi *service* mengalami gangguan/*error* dan kondisi *service* dilakukan *recovery*. Hasil pengujian *service* saat FTP dalam keadaan *error*/dinonaktifkan dapat dilihat pada Tabel 5.

TABEL 5. DATA PENGUJIAN *SERVICE* FTP "*CRITICAL*"

Pengujian	Waktu Penonaktifan	Waktu Deteksi Nagios	Durasi*
1	13:16:47	13:24:22	0:07:35
2	13:49:08	13:58:22	0:09:14
3	14:21:08	14: 22:22	0:01:14
4	16:08:14	16:16:34	0:08:20
5	16:34:27	16:42:34	0:08:07

**Durasi = Selisih Waktu Deteksi Nagios dengan Waktu Penonaktifan*

Selanjutnya *service* FTP dilakukan *recovery*/diaktifkan kembali. Hasil pengujian *service* saat *service* FTP diaktifkan kembali dijelaskan pada Tabel 6.

TABEL 6. DATA PENGUJIAN *SERVICE* FTP "*OK*"

Pengujian	Waktu *Recovery*	Waktu Deteksi Nagios	Durasi*
1	13:35:23	13:38:22	0:02:59
2	14:06:35	14:12:22	0:05:47
3	14:35:33	14:36:22	0:00:49
4	16:23:39	16:30:22	0:06:43
5	16:51:33	16:54:22	0:02:49

**Durasi= Selisih Waktu Deteksi Nagios dengan Waktu Recovery*

Pengujian selanjutnya yaitu pada *service* HTTP. Hasil pengujian *service* saat HTTP mengalami dalam keadaan *error*/dinonaktifkan dijelaskan pada Tabel 7.

TABEL 7. DATA PENGUJIAN *SERVICE* HTTP "*CRITICAL*"

Pengujian	Waktu Penonaktifan	Waktu Deteksi Nagios	Durasi*
1	13:16:47	13:25:56	0:09:09
2	13:49:08	13:49:55	0:00:47
3	14:21:08	14:23:55	0:02:47
4	16:08:14	16:17:42	0:09:28
5	16:34:27	16:41:42	0:07:15

**Durasi = Selisih Waktu Deteksi Nagios dengan Waktu Penonaktifan*

Selanjutnya *service* HTTP dilakukan *recovery*/ diaktifkan kembali. Hasil pengujian *service* saat *service* HTTP diaktifkan kembali dijelaskan pada Tabel 8.

TABEL 8. DATA PENGUJIAN *SERVICE* HTTP "*OK*"

Pengujian	Waktu *Recovery*	Waktu Deteksi Nagios	Durasi*
1	13:35:23	13:39:55	0:04:32
2	14:06:35	14:13:55	0:07:20
3	14:35:33	14:37:55	0:02:22
4	16:23:39	16:31:42	0:08:03
5	16:51:33	16:55:42	0:04:09

**Durasi = Selisih Waktu Deteksi Nagios dengan Waktu Recovery*

Hasil pengujian notifikasi terhadap *hosts* merupakan lanjutan dari hasil pengujian *hosts* pada Ubuntu.Hasil pengujian *host* saat Nagios mengirim notifikasi pada perubahan status *host* Ubuntu "*DOWN*" dapat dilihat pada Tabel 9.

TABEL 9. DATA PENGUJIAN NOTIFIKASI TERHADAP *HOST* UBUNTU "*DOWN*"

Pengujian	Waktu Deteksi Nagios	Waktu Pengiriman Notifikasi	Durasi*
1	11:33:29	11:40:56	0:07:27
2	12:01:51	12:07:53	0:06:02
3	12:17:22	12:25:54	0:08:32
4	12:33:22	12:41:53	0:08:31
5	12:48:53	12:58:17	0:09:24

**Durasi = Selisih Waktu Pengiriman Notifikasi dengan Waktu Deteksi Nagios*

Selanjutnya saat *host* Ubuntu dilakukan *recovery*/diaktifkan kembali dan Nagios mendeteksi perubahan status "*UP*". Hasil pengujian *host* saat Nagios mengirim notifikasi pada perubahan status "*UP*" *host* Ubuntu dijelaskan pada Tabel 10. Hasil pengujian notifikasi terhadap *services* merupakan lanjutan dari hasil pengujian *services* pada FTP dan HTTP. Hasil pengujian *service* FTP saat Nagios mengirim notifikasi pada perubahan status "*CRITICAL*" *service* FTP dapat dilihat pada Tabel 11.

TABEL 10. DATA PENGUJIAN NOTIFIKASI TERHADAP *HOSTS* UBUNTU "*UP*"

Pengujian	Waktu Deteksi Nagios	Waktu Pengiriman Notifikasi	Durasi *
1	11:48:24	11:48:24	0:00:00
2	12:13:25	12:13:22	0:00:00
3	12:29:51	12:29:51	0:00:00
4	12:44:15	12:44:15	0:00:00
5	13:03:27	13:03:27	0:00:00

**Durasi = Selisih Waktu Pengiriman Notifikasi dengan Waktu Deteksi Nagios*

TABEL 11. DATA PENGUJIAN NOTIFIKASI TERHADAP *SERVICE* FTP "*CRITICAL*"

Pengujian	Waktu Deteksi Nagios	Waktu Pengiriman Notifikasi	Durasi*
1	13:24:22	13:28:22	00:04:00
2	13:58:22	14:02:22	00:04:00
3	14:22:22	14:26:22	00:04:00
4	16:16:34	16:20:34	00:04:00
5	16:42:34	16:44:34	00:04:00

**Durasi = Selisih Waktu Pengiriman Notifikasi dengan Waktu Deteksi Nagios*

Selanjutnya saat *service* FTP dilakukan *recovery*/diaktifkan kembali dan Nagios mendeteksi perubahan status "*OK*". Hasil pengujian *service* saat Nagios mengirim notifikasi pada perubahan status *service* FTP "*OK*" dapat dilihat pada Tabel 12.

TABEL 12. DATA PENGUJIAN NOTIFIKASI TERHADAP *SERVICE* FTP "*OK*"

Pengujian	Waktu Deteksi Nagios	Waktu Pengiriman Notifikasi	Durasi*
1	13:38:22	13:38:22	00:00:00
2	14:12:22	14:12:22	00:00:00
3	14:36:22	14:36:22	00:00:00
4	16:30:22	16:30:22	00:00:00
5	16:54:22	16:54:22	00:00:00

**Durasi = Selisih Waktu Pengiriman Notifikasi dengan Waktu Deteksi Nagios*

Pengujian selanjutnya yaitu pada *service* HTTP. Hasil pengujian *service* HTTP saat Nagios mengirim notifikasi pada perubahan status "*CRITICAL*" *service* HTTP dijelaskan pada Tabel 13.

TABEL 13. DATA PENGUJIAN NOTIFIKASI TERHADAP *SERVICE* HTTP "*CRITICAL*"

Pengujian	Waktu Deteksi Nagios	Waktu Pengiriman Notifikasi	Durasi*
1	13:25:56	13:29:55	00:03:59
2	13:49:55	13:53:55	00:04:00
3	14:23:55	14:27:55	00:04:00
4	16:17:42	16:21:42	00:04:00
5	16:41:42	16:45:42	00:04:00

**Durasi = Selisih Waktu Pengiriman Notifikasi dengan Waktu Deteksi Nagios*

Selanjutnya saat *service* HTTP dilakukan *recovery*/diaktifkan kembali dan Nagios mendeteksi perubahan status "*OK*". Hasil pengujian *service* saat Nagios mengirim notifikasi pada perubahan status *service* HTTP "*OK*" dijelaskan pada Tabel 14.

TABEL 14. DATA PENGUJIAN NOTIFIKASI TERHADAP *SERVICE* HTTP "*OK*"

Pengujian	Waktu Deteksi Nagios	Waktu Pengiriman Notifikasi	Durasi*
1	13:39:55	13:39:55	00:00:00
2	14:13:55	14:13:55	00:00:00
3	14:37:55	14:37:55	00:00:00
4	16:31:42	16:31:42	00:00:00
5	16:55:42	16:55:42	00:00:00

**Durasi = Selisih Waktu Pengiriman Notifikasi dengan Waktu Deteksi Nagios*

Setelah dilakukan pengujian dari *Event Handler*, didapatkan hasil pengujian terhadap fungsionalitas *Event Handler* yang dilakukan melalui 5 kali pengujian terhadap *service* Printer Spooler pada *host* Windows-PC. Hasil pengujian dijelaskan pada Tabel 15.

TABEL 15. DATA PENGUJIAN *EVENT HANDLER* TERHADAP *SERVICE* PRINTER SPOOLER

Pengujian	Waktu Penonaktifan	Waktu Deteksi Nagios	Waktu Aksi *Event Handler*	Durasi*
1	13:10:57	13:13:45	13:15:45	00:02:00
2	13:32:13	13:35:45	13:37:45	00:02:00
3	14:23:41	13:27:45	14:29:45	00:02:00
4	16:34:40	16:38:35	-	-
5	16:43:31	16:47:34	-	-

**Durasi = Selisih Waktu Aksi Event Handler dengan Waktu Deteksi Nagios*

D. Analisis Data/Evaluasi

a. Analisis pengujian *monitoring* agent terhadap fungsionalitas *plugins* Nagios

Berdasarkan hasil pengujian yang telah dilakukan, komunikasi *plugins* Nagios dengan *monitoringagent* NSClient++ pada Windows, NRPE pada Ubuntu, dan SNMP pada Router berhasil diuji. Dalam pengujian dapat dilihat bahwa masing-masing pengujian berhasil mendapatkan *output* berupa informasi *service/resource* sesuai dengan *commmand plugins*. Dengan demikian NSClient++, NRPE, dan SNMP pada *monitoring host* berfungsi dengan baik untuk memberikan informasi *service*/*resource* kepada Nagios untuk dilakukan *monitoring*.

b. Analisis pengujian terhadap fungsionalitas *web interface*

1) Analisis pengujian autentikasi web interface

Dari pengujian autentikasi *web interface* yang telah dilakukan didapatkan hasil yaitu autentikasi berhasil diuji. Hal ini didapat ketika melakukan *input user name* dan *password*

dengan benar, *user* dapat masuk ke dalam halaman *web interface* Nagios, sedangkan ketika salah dalam *inputusename* dan *password, user* tidak dapat masuk ke halaman *web interface* Nagios dan dialihkan kembali pada *window* autentikasi.

2) Analisis pengujian hosts

Dari pengujian *hosts* yang telah dilakukan, didapatkan hasil yaitu Nagios berhasil diuji dengan mendeteksi perubahan keadaan status *hosts* secara *real-time* dengan hasil durasi deteksi Nagios. Durasi Nagios dalam mendeteksi perubahan status *host* tergantung pada konfigurasi *check_interval*.

Dari data pengujian *hostUbuntu* pada Tabel 3 dan Tabel 4 didapatkan durasi Nagios dalam mendeteksi perubahan status *hosts* dalam kondisi "*DOWN*" dan "*UP*" terjadi selama selang waktu kurang dari 5 menit. Kondisi tersebut sesuai dengan konfigurasi yang telah didefinisikan, yaitu *check_interval* pada *host* Ubuntu yang diatur 5 menit.

3) Analisis pengujian services

Dari pengujian *services* yang telah dilakukan, didapatkan hasil yaitu Nagios berhasil diuji dengan mendeteksi perubahan keadaan status *services* secara *real-time* dengan hasil durasi deteksi Nagios. Durasi Nagios dalam mendeteksi perubahan status *services* tergantung pada konfigurasi *normal_check_interval*.

Dari data pengujian *services* FTP pada Tabel 5 dan tabel 6 didapatkan durasi Nagios dalam mendeteksi perubahan status kedua *services* dalam kondisi "*CRITICAL*" dan "*OK*" terjadi selama selang waktu kurang dari 10 menit. Kondisi tersebut sesuai dengan konfigurasi yang telah didefinisikan, yaitu *normal_check_interval* pada *service* FTP dan HTTP yang diatur 10 menit.

c. Analisis pengujian terhadap fungsionalitas notifikasi Telegram

1) Analisis pengujian notifikasi hosts

Dari pengujian fungsionalitas notifikasi Telegram terhadap *hosts* yang telah dilakukan, didapatkan hasil yaitu Nagios berhasil diuji karena dapat mengirimkan pesan notifikasi *alert* kepada admin dalam *Group* Telegram "Nagios *Alerts*". Durasi saat Nagios melakukan pengiriman notifikasi tergantung pada konfigurasi *retry_interval* dan *max_check_attempts.*

Dari data pengujian notifikasi *host* Ubuntu dalam kondisi "*DOWN*" pada Tabel 10 didapat durasi saat Nagios mendeteksi perubahan status "*DOWN"* sampai Nagios melakukan pengiriman notifikasi terjadi selama selang waktu kurang dari 10 menit. Kondisi tersebut sesuai dengan konfigurasi yang telah didefinisikan, yaitu *max_check_attempts* yang diatur 10 kali, dan *retry_interval* yang diatur 1 menit. Jadi, ketika Nagios mendeteksi perubahan status pada *host* Ubuntu(dalam keadaan *soft state* 1), Nagios melakukan pengecekan ulang dalam selang *interval* 1 menit yang dilakukan selama 10 kali sampai *host* dalam keadaan *hard state*.

Sedangkan dari data pengujian notifikasi *host* Ubuntu ketika dilakukan *recovery* sampai kondisi *host* menjadi "*UP*" pada Tabel 11 didapat hasil yaitu tidak ada selang waktu diantaranya. Kondisi tersebut dikarenakan ketika Nagios melakukan pengecekan pertama untuk memastikan perubahan status menjadi "*UP*", Nagios langsung mendeteksi bahwa keadaan tersebut sudah dalam keadaan *hard state*.

Notifikasi Nagios terhadap *hosts* melalui Telegram akan dikirim setelah Nagios mendeteksi kondisi *hard state* setelah dilakukan pengecekan berulang, hal ini dilakukan untuk memastikan kembali apakah kondisi *hosts* benar dalam keadaan bermasalah atau tidak sebelum dikirimkan kepada Group Telegram Admin. Pengiriman notifikasi Telegram pun bergantung pada koneksi internet di sisi server Nagios. Jika tidak ada koneksi internet/koneksi internet terputus, maka Nagios tidak akan mengirimkan notifikasi kepada Group Telegram Admin.

2) Analisis pengujian notifikasi services

Dari pengujian fungsionalitas notifikasi Telegram terhadap *services* yang telah dilakukan, didapatkan hasil yaitu Nagios berhasil diuji karena dapat mengirimkan pesan notifikasi *alert* kepada admin dalam *Group* Telegram "Nagios *Alerts*". Durasi saat Nagios melakukan pengiriman notifikasi tergantung pada konfigurasi *retry_interval* dan *max_check_attempts.*

Dari data pengujian notifikasi *services* FTP dalam kondisi "*CRITICAL*" pada Tabel 11 dan data pengujian notifikasi *services* HTTP dalam kondisi "*CRITICAL*" pada Tabel 13 didapat durasi saat Nagios mendeteksi perubahan status "*CRITICAL"* sampai Nagios

melakukan pengiriman notifikasi terjadi selama selang waktu kurang dari 6 menit. Kondisi tersebut sesuai dengan konfigurasi yang telah didefinisikan, yaitu *max_check_attempts* yang diatur 3 kali, dan *retry_interval* yang diatur 2 menit. Jadi, ketika Nagios mendeteksi perubahan status "*CRITICAL*"pada *services* FTP dan HTTP(dalam keadaan *soft state* 1), Nagios melakukan pengecekan ulang dalam selang *interval* 2 menit yang dilakukan selama 3 kali sampai *services* dalam keadaan *hard state*. Setelah *services* dalam keadaan *hard state*, pada saat itulah Nagios mengirimkan notifikasi melalui Telegram *Messenger.*

Sedangkan dari data pengujian notifikasi *service* FTP dan HTTP ketika dilakukan *recovery* sampai kondisi *services* menjadi "*OK*" pada Tabel 12 dan 14 didapat hasil yaitu tidak ada selang waktu diantaranya. Kondisi tersebut dikarenakan ketika Nagios melakukan pengecekan pertama untuk memastikan perubahan status *services* menjadi "*OK*", Nagios langsung mendeteksi bahwa keadaan tersebut sudah dalam keadan *hard state*.

Notifikasi Nagios terhadap *services* melalui Telegram akan dikirim setelah Nagios mendeteksi kondisi *hard state* setelah dilakukan pengecekan berulang, hal ini dilakukan untuk memastikan kembali apakah kondisi *services* benar dalam keadaan bermasalah atau tidak sebelum dikirimkan kepada Group Telegram Admin. Pengiriman notifikasi Telegram pun bergantung pada koneksi internet di sisi server Nagios. Jika tidak ada koneksi internet/koneksi internet terputus, maka Nagios tidak akan mengirimkan notifikasi kepada Group Telegram Admin.

d. Analisis pengujian terhadap fungsionalitas *Event Handler*

Dari pengujian *Event Handler* yang telah dilakukan, didapatkan hasil yaitu Nagios berhasil diuji karena dapat melakukan *restart* terhadap *service* yang mengalami masalah/*error*. Durasi saat eksekusi *Event Handler* terhadap *service* tergantung pada konfigurasi *retry_interval* dan *max_check_attempts.*

Dari data pengujian *Event Handler* pada Tabel 15 didapat durasi saat Nagios mendeteksi perubahan status *service* Printer Spooler "*CRITICAL*" sampai *Event Handler* melakukan aksi terhadap *service* terjadi selama selang waktu 2-6 menit. Kondisi tersebut sesuai dengan konfigurasi yang telah didefinisikan, yaitu *max_check_attempts* yang diatur adalah 4 kali, dan *retry_interval* yang diatur adalah 2 menit. Jadi, ketika Nagios mendeteksi adanya perubahan status pada *service* Printer Spooler, dalam interval 1 menit *Event Handler* melakukan eksekusi dan mengulangnya selama 4 kali sampai berhasil melakukan *restart* terhadap *service.*

Pada data hasil ke 3 dan 4 diketahui bahwa aksi *Event Handler* tidak dapat melakukan *restart service* setelah dilakukan pengulangan selama 4 kali. Hal tersebut dikarenakan *remote hosts* dalam keadaan mati yang menyebabkan remote agent (NSClient++) tidak dapat menjalankan *service* untuk melakukan *restart service*. Jika *Event Handler* tidak dapat melakukan aksinya untuk *restart service* bermasalah, maka Nagios akan meneruskannya agar dapat diproses untuk pengiriman notifikasi.

V. PENUTUP

A. Kesimpulan

a. Implementasi dari *Network Monitoring System* Nagios dapat memonitoring perangkat dan *services* yang ada pada jaringan dengan interval waktu pengecekan status pada *host* setiap 5 menit dan interval waktu pengecekan status pada *service* setiap10 menit.
b. Pemanfaatan *Event Handler* dengan melakukan *restart service* secara otomatis dapat menangani masalahyang terjadi pada *service* yang mengalami gangguan dengan waktu aksi *Event Handler* rata-rata 2 menit.
c. Penggunaan notifikasi *alert* Nagios dengan Telegram *Messenger* dapat mengirimkan notifikasi kepada *network administrator* berupa *chat* pada *Group* 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. Pengiriman notifikasi *alert* dengan Telegram *Messenger* bergantung pada koneksi internet di sisi server Nagios.

B. Saran

a. Menambahkan *target monitoring* untuk melakukan monitor terhadap perangkat yang lain seperti printer, IP *camera*, dan lain-lain.
b. Menambahkan *alert* notifikasi lain, seperti *audio notification alarm*.
c. Menambahkan fitur Nagios lain seperti, nagmap untuk mapping menggunakan API Google *Maps,* NagiosQL, dan lain-lain.

REFERENSI

[1] Prasetia, WC. 2012. "Implementasi Sistem Monitoring Jaringan Menggunakan Nagios dengan SMS *Alert* Menggunakan Ozeki NG *SMS Gateway*". Laporan Karya Ilmiah, Telkom University. Bandung

[2] Kocjan, Wojciech. 2014. *Learning Nagios 4.* Birmingham: Packt Publishing Ltd.

[3] Nagios. *Nagios Core Documentation*. 2016. https://www.nagios.org [30 Oktober 2015]

[4] Medin, Michael. 2016. *Using NSClient++ from Nagios with check_nt*. https://docs.nsclient.org/tutorial/nagios/check_nt.html [29 April 2016]

[5] Ding Mei-Zhen, Huang Chen. 2015. Design And Implementation Of Network Monitoring System Based On Nagios. Information Technology And Informatization

[6] Galstad, Ethan. 2007. *NRPE Documentation.* https://assets.nagios.com/downloads/nagioscore/docs/nrpe/NRPE.pdf [29 April 2016]

[7] SK. 2015. *How to Install Nagios 4.1 in Ubuntu 15.04.* Tamilnadu, India. http://www.unixmen.com/how-to-install-nagios-4-1-in-ubuntu-15-04/ [30 Oktober 2015]