

Monitoring Router Mikrotik CHR Secara Otomatis Menggunakan Telegram Bot Berbasis Python

Automatic Monitoring of Mikrotik CHR Router Using Python-Based Telegram Bot

Dea Anggraini¹, Dedy Kiswanto², Natasha Patricia Nainggolan*³, Zevan Irfandi Surbakti⁴

^{1,2,3,4}Ilmu Komputer, Universitas Negeri Medan

E-mail: ¹anggrainidea2006@gmail.com, ²dedykiswanto@unimed.ac.id,

³nattadsah@gmail.com, ⁴zevanirfandisurbakti@gmail.com

Abstrak

Pemantauan kondisi router secara manual terbukti tidak efisien karena administrator baru mengetahui gangguan setelah laporan dari pengguna, bukan dari deteksi sistem. Penelitian terdahulu menunjukkan integrasi Telegram Bot dengan sistem monitoring jaringan mampu meningkatkan respons terhadap gangguan secara nyata. Penelitian ini bertujuan membangun sistem monitoring Router Mikrotik CHR berbasis Telegram Bot menggunakan bahasa pemrograman Python yang dapat mendeteksi gangguan dan mengirim notifikasi otomatis melalui pemantauan secara periodik. Sistem dikembangkan dengan pendekatan prototyping dan terdiri dari enam modul Python yang berjalan paralel menggunakan mekanisme threading. Pemantauan dilakukan menggunakan protokol ICMP setiap 30 detik, sedangkan pengambilan informasi perangkat dilakukan via protokol SSH menggunakan library Netmiko. Pengujian dilakukan melalui dua belas skenario uji dengan simulasi router down menggunakan perintah `/disable_interface` melalui bot Telegram. Sistem juga dilengkapi dua fitur otomatisasi jaringan yaitu auto restart interface yang mengirim perintah `/interface enable` ke router secara otomatis saat gangguan terdeteksi, dan auto backup konfigurasi yang menyimpan konfigurasi router ke file lokal setiap kali router pulih. Hasil pengujian menunjukkan notifikasi gangguan terkirim rata-rata 51 detik setelah interface dinonaktifkan pada tiga pengujian lengkap. Router terbaca pulih 35 sampai 44 detik setelah interface diaktifkan pada sebagian besar kejadian, alert duplikat tercegah, dan 691 baris log tercatat otomatis. Hasil ini menunjukkan sistem berpotensi diterapkan pada lingkungan laboratorium atau jaringan skala kecil.

Kata kunci: ICMP, Mikrotik CHR, Monitoring jaringan, Python, Telegram Bot

Abstract

Manual router monitoring is inefficient because network administrators only learn about failures through user complaints rather than automatic system detection. This study aimed to build an automated monitoring system for Mikrotik CHR Router using a Python-based Telegram Bot that can detect failures and send automated notifications through periodic monitoring. The system was developed using a prototyping approach and consists of six Python modules running in parallel via threading. Monitoring was performed using the ICMP protocol every 30 seconds, while device information retrieval was done via SSH using the Netmiko library. Testing was conducted through twelve test scenarios simulating router failure using the `/disable_interface` command through the Telegram bot. The system also includes two network automation features: auto restart interface, which automatically sends the `/interface enable` command to the router when a failure is detected, and auto backup configuration, which saves the router configuration to a local file each time the router recovers. Test results showed that the failure notification was sent an average of 51 seconds after the interface was disabled in three complete tests. The router was detected as recovered 35 to 44 seconds after the interface was enabled in most events, duplicate alerts were prevented, and 691 log lines were

recorded automatically. The results indicate that the system has potential for use in laboratory or small-scale networks.

Keywords: ICMP, Mikrotik CHR, *network monitoring*, Python, Telegram Bot

1. PENDAHULUAN

Perkembangan teknologi informasi yang pesat menjadikan jaringan komputer sebagai infrastruktur yang sangat kritis dalam mendukung aktivitas operasional berbagai organisasi, baik di sektor pendidikan, pemerintahan, maupun industri. Jaringan komputer berfungsi sebagai tulang punggung komunikasi data yang menghubungkan berbagai perangkat dan layanan digital. Di dalam jaringan tersebut, *router* memegang peranan sentral sebagai perangkat yang bertanggung jawab atas proses penerusan paket data antar segmen jaringan. Apabila kinerja *router* terganggu, maka seluruh layanan yang bergantung pada konektivitas jaringan berisiko mengalami gangguan serius, sehingga pemantauan kondisi *router* secara *real-time* menjadi suatu kebutuhan agar ketersediaan (*availability*) layanan jaringan dapat terjaga [1].

Pemantauan jaringan secara manual terbukti tidak efisien karena administrator baru mengetahui gangguan setelah laporan dari pengguna akhir, bukan dari deteksi sistem sendiri [2]. Waktu respons yang lambat ini kerap mengakibatkan *downtime* lebih lama dan berdampak besar bagi pengguna jaringan. Kondisi serupa juga ditemukan pada penyedia layanan internet, di mana Zabbix diimplementasikan justru karena sistem *monitoring* yang ada belum mampu memberikan notifikasi secara *real-time* saat gangguan terjadi [3].

Penelitian-penelitian sebelumnya sudah mencoba berbagai solusi. Sebagian menggunakan *tools monitoring* seperti Uptime Kuma yang diintegrasikan dengan Telegram Bot untuk memantau jaringan *Wi-Fi* berbasis Mikrotik [4], namun pendekatan ini tetap memerlukan *server* terpisah dan konfigurasi *platform monitoring* tambahan.

Penelitian lain memilih Zabbix di lingkungan *cloud* untuk mendeteksi serangan jaringan, tapi pendekatannya jauh dari skenario *monitoring router* sederhana di lingkungan lokal [5]. Penelitian yang membangun sistem *monitoring* langsung berbasis Python tanpa *platform monitoring* tambahan, dijalankan sepenuhnya pada satu laptop dengan *router* virtual Mikrotik CHR sebagai pengganti *router* fisik, masih terbatas. Penelitian ini mengisi celah tersebut dengan pendekatan yang lebih ringan, terbuka, dan dapat diimplementasikan tanpa infrastruktur khusus.

Mikrotik CHR (*Cloud Hosted Router*) merupakan versi virtual dari RouterOS Mikrotik yang dapat dijalankan di atas *platform* virtualisasi seperti VirtualBox dan GNS3. Penelitian sebelumnya sudah membuktikan bahwa Mikrotik CHR bisa menjalankan fungsi *routing* dengan stabil, dengan rata-rata *delay* di bawah 20 ms dan *packet loss* 0% di sebagian besar skenario pengujian [6]. Ini membuktikan Mikrotik CHR cukup layak dipakai sebagai pengganti *router* fisik dalam penelitian berbasis simulasi, termasuk untuk keperluan pengujian sistem *monitoring* jaringan.

Penelitian yang secara spesifik mengintegrasikan Telegram Bot dengan metode ICMP untuk *monitoring router* Mikrotik telah dilakukan oleh Afiq et al. dengan hasil deteksi gangguan jaringan secara *real-time* yang akurat, di mana notifikasi dapat terkirim ke operator jaringan dalam rata-rata waktu 10,20 detik setelah gangguan

terdeteksi [7]. Pendekatan lain menggunakan fitur Netwatch bawaan RouterOS dengan metode PPDIIO untuk memantau koneksi antar cabang perusahaan, dan sistem berhasil mendeteksi gangguan *tunnel* secara otomatis [8].

Dari sisi keamanan jaringan, implementasi *Intrusion Detection System* (IDS) berbasis Snort yang diintegrasikan dengan *chatbot* Telegram berhasil memberikan notifikasi serangan jaringan secara otomatis, dengan waktu deteksi serangan *Ping of Death* dalam 4 detik, *Nmap scanning* dalam 3 detik, dan *DDoS* dalam 2 detik [9]. Hasil ini menunjukkan kombinasi protokol jaringan dengan Telegram Bot bisa bekerja *real-time* bahkan untuk kebutuhan respons cepat terhadap ancaman keamanan.

Dari sisi pendekatan berbasis bahasa pemrograman, Hermawan et al. membuktikan bahwa Python bersama *library* Paramiko mampu mengotomasi konfigurasi *router* via SSH untuk protokol *routing* OSPF dan EIGRP, dengan waktu konfigurasi rata-rata 3,4 detik per *router* dan performa QoS yang masuk kategori sangat baik [10]. Penelitian lain juga membuktikan bahwa Grafana yang diintegrasikan dengan Prometheus dan Telegram mampu memberikan notifikasi *down* maupun *up* secara otomatis ketika konektivitas antar *router* terganggu [11]. Dibandingkan penelitian tersebut, sistem *monitoring* yang dibangun langsung dengan Python tanpa *platform* perantara pada *router* virtual Mikrotik CHR masih jarang dikaji. Penelitian ini berkontribusi pada rancangan sistem *monitoring* ringan berbasis Python yang menggabungkan *polling* ICMP, Telegram Bot, pengelolaan perangkat lewat SSH, pemulihan *interface* otomatis, dan *backup* konfigurasi dalam satu lingkungan virtual Mikrotik CHR. Kinerjanya dievaluasi lewat waktu deteksi dan keandalan fungsi pada dua belas skenario pengujian.

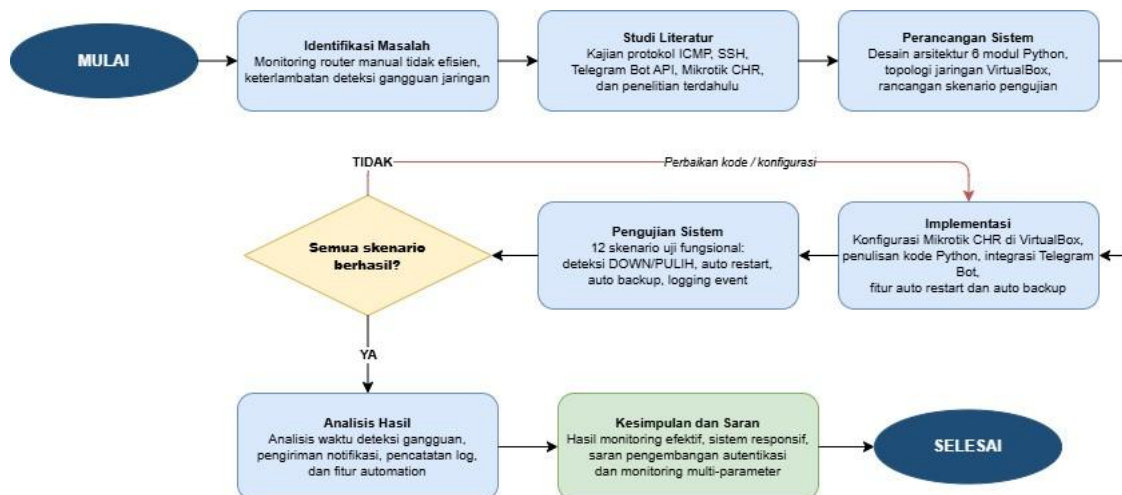
Berdasarkan permasalahan tersebut, penelitian ini bertujuan mengembangkan sistem *monitoring Router* Mikrotik CHR berbasis Telegram Bot menggunakan bahasa pemrograman Python. Sistem yang dibangun menggunakan protokol ICMP untuk mendeteksi ketersediaan *router* secara periodik setiap 30 detik dan protokol SSH melalui *library* Netmiko dan Paramiko untuk mengambil informasi detail perangkat. Ketika *router* terdeteksi mengalami gangguan, sistem langsung mengirimkan peringatan ke Telegram, dan ketika *router* kembali normal, notifikasi pemulihan dikirim ke admin tanpa perlu intervensi apapun.

2. METODOLOGI PENELITIAN

Penelitian ini menggunakan metode eksperimental dengan pendekatan *prototyping* untuk membangun dan menguji sistem *monitoring router* berbasis Telegram Bot. Sistem dirancang terdiri dari enam modul Python yang diintegrasikan dengan *Router* Mikrotik CHR pada lingkungan virtual, lalu diuji melalui dua belas skenario pengujian fungsional untuk mengukur waktu deteksi gangguan, keberhasilan pengiriman notifikasi, dan kelengkapan pencatatan *log*.

Penelitian ini dijalankan melalui lima tahap yang berurutan. Tahap pertama adalah identifikasi masalah, yaitu merumuskan permasalahan *monitoring* jaringan manual yang lambat dan tidak efisien. Tahap kedua adalah studi literatur, di mana peneliti mengkaji penelitian terdahulu terkait *monitoring* jaringan, Telegram Bot, dan Mikrotik CHR sebagai dasar perancangan sistem. Tahap ketiga adalah perancangan sistem, mencakup desain arsitektur modul Python, topologi jaringan virtual, dan rancangan skenario pengujian. Tahap keempat adalah implementasi, yaitu membangun keenam modul Python dan mengonfigurasi *Router* Mikrotik CHR di VirtualBox. Tahap kelima adalah pengujian dan analisis, di mana sistem diuji

melalui dua belas skenario fungsional dan hasilnya dianalisis secara deskriptif kualitatif. Alur penelitian secara keseluruhan dapat dilihat pada Gambar 1.



Gambar 1. Alur Penelitian

2.1. Metode dan Jenis Data Penelitian

Penelitian ini termasuk penelitian terapan dengan pendekatan eksperimental. Pengembangan sistem menggunakan pendekatan *prototyping*, yaitu sistem dibangun secara bertahap mulai dari konfigurasi jaringan virtual, penulisan kode program, sampai pengujian fungsional. Pendekatan ini dipilih karena setiap tahap bisa langsung dievaluasi dan diperbaiki tanpa mengulang dari awal.

Data penelitian berupa data hasil eksperimen yang diperoleh dari pengujian langsung terhadap sistem *monitoring*. Pengumpulan data dilakukan melalui observasi langsung terhadap respons sistem saat setiap skenario uji dijalankan, seperti mencatat waktu deteksi, memeriksa pesan yang masuk ke Telegram, dan melihat isi *file log*. Waktu deteksi dihitung dari selisih *timestamp* perintah `/disable_interface` sampai *alert* terkirim pada *log*. Analisis data dilakukan secara deskriptif kualitatif dengan membandingkan hasil pengujian terhadap indikator keberhasilan (pada Tabel 2).

Variabel yang diukur dalam penelitian ini meliputi waktu deteksi gangguan *router*, waktu pengiriman notifikasi pemulihan, jumlah *alert* yang terkirim selama kondisi *down*, dan jumlah entri *log* yang tercatat selama sesi pengujian. Pendekatan serupa juga diterapkan oleh Prabowo dan Ariyanto yang membangun program otomasi jaringan berbasis Python untuk konfigurasi *router* Mikrotik dan mengujinya langsung bersama administrator jaringan, dengan hasil sistem terbukti mempercepat proses konfigurasi dan mengurangi kemungkinan kesalahan manual [12].

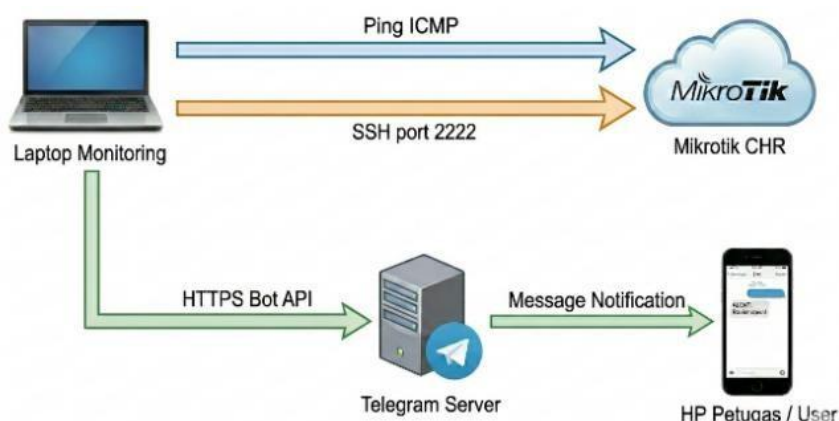
2.2. Spesifikasi Perangkat dan Topologi Jaringan

Sistem dibangun dan diuji pada satu unit laptop dengan sistem operasi Windows 10 Build 19045. Spesifikasi lengkap perangkat yang digunakan dalam penelitian ini dapat dilihat pada Tabel 1.

Tabel 1. Spesifikasi Lingkungan Pengembangan

Komponen	Spesifikasi
Sistem Operasi	Windows 10 Build 19045
Prosesor	Intel Core i5
RAM	8 GB
Penyimpanan	256 GB SSD
Python	3.13
VirtualBox	7.2.8
Mikrotik CHR	RouterOS 7.21.4 (long-term)
python-telegram-bot	21.6
Netmiko	4.x
Paramiko	3.x

Router yang dipantau adalah Mikrotik CHR versi 7.21.4 yang berjalan sebagai mesin virtual di dalam VirtualBox pada laptop yang sama. Untuk menghubungkan laptop *host* dengan *router* virtual, dipakai dua adapter jaringan VirtualBox secara bersamaan. Adapter pertama menggunakan mode NAT dengan konfigurasi *port forwarding* dari port 2222 pada *host* ke port 22 pada *router* virtual, dan dipakai khusus untuk koneksi SSH. Adapter kedua menggunakan mode *Host-Only* dengan alamat IP 192.168.56.10 pada antarmuka ether2 *router*, dan dipakai sebagai target *ping monitoring*. Notifikasi dikirim dari program Python di laptop *host*, melewati internet ke Telegram Bot API, lalu diteruskan ke akun Telegram administrator. Jalur ini terpisah dari jaringan virtual tempat *router* berada, sehingga laptop *host* harus tersambung ke internet agar notifikasi sampai. Topologi hubungan antara laptop *host*, *router* virtual, dan *server* Telegram dapat dilihat pada Gambar 2.

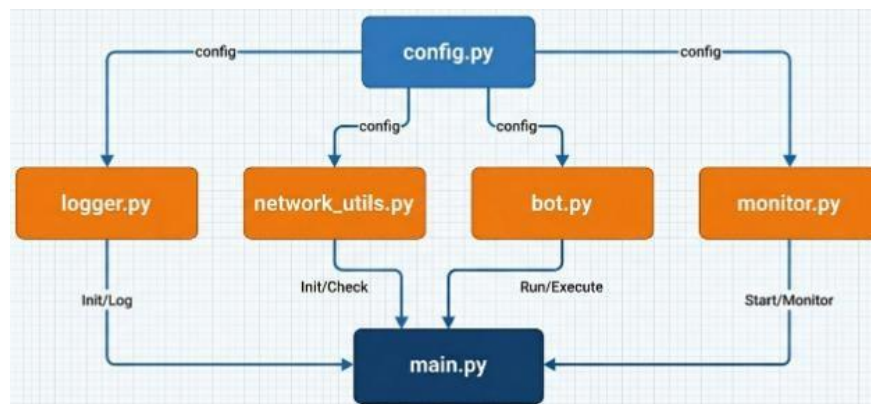


Gambar 2. Topologi Jaringan Sistem *Monitoring Router*

2.3. Implementasi Sistem

Sistem dibangun menggunakan Python 3.13 dan terdiri dari enam modul yang masing-masing punya peran berbeda. *File config.py* menyimpan seluruh konfigurasi seperti token *bot*, alamat IP *router*, dan interval *monitoring*. *File logger.py* menangani pencatatan *event* ke file logs/events.txt dengan format *timestamp*. *File network_utils.py* berisi fungsi *ping* yang memanggil perintah sistem operasi melalui *subprocess.run()* dengan parameter *-n* untuk jumlah paket dan *-w* untuk batas waktu dalam milidetik, serta fungsi SSH via *library* Netmiko. Hasil *ping*

dibaca dari *return code* proses, di mana nilai 0 menandakan *router* merespons. File *bot.py* menangani enam perintah Telegram yaitu */start*, */ping*, */status*, */interfaces*, */disable_interface*, dan */enable_interface*. File *monitor.py* berisi loop *monitoring* yang berjalan setiap 30 detik. File *main.py* menjalankan *monitor.py* dan *bot.py* secara bersamaan menggunakan mekanisme *threading* Python, di mana *monitor* berjalan pada thread terpisah sementara *bot* berjalan pada main thread. Arsitektur hubungan antar modul ditampilkan pada Gambar 3.



Gambar 3. Arsitektur Modul Program

Untuk mencegah pengiriman *alert* yang berulang ketika *router* masih dalam kondisi *down*, digunakan sebuah variabel *flag* bernama *router_was_down* bertipe *boolean*. *Alert* hanya dikirim satu kali saat status berubah dari *UP* ke *DOWN*, dan notifikasi pemulihan hanya dikirim satu kali saat status kembali *UP*. Pengiriman pesan ke Telegram dilakukan langsung via HTTP POST ke endpoint *Bot API* menggunakan *library* *requests*, terpisah dari *library* *Python-Telegram-bot* yang dipakai untuk menangani perintah masuk. Koneksi SSH ke *router* Mikrotik menggunakan *library* *Netmiko* dengan *device_type* "*Mikrotik_routeros*". Perintah yang dikirim ke *router* adalah */interface* print terse untuk mengambil daftar dan status seluruh antarmuka jaringan, serta */system identity* print dan */system resource* print untuk mengambil informasi sistem *router*.

Sistem juga dilengkapi dua fitur otomasi jaringan. Fitur pertama adalah *auto restart interface* yang berjalan saat sistem pertama kali mendeteksi *router DOWN*. Sistem secara otomatis mengirim perintah */interface enable ether2* ke *router* Mikrotik via SSH tanpa perlu ada yang mengetiknya secara manual. Fitur kedua adalah *auto backup konfigurasi* yang berjalan saat *router* kembali *UP* setelah kondisi *down*. Sistem mengirim perintah */export* ke *router* dan menyimpan seluruh konfigurasi ke *folder* backups dengan nama *file* bertimestamp. Kedua fitur ini masuk kategori *network automation* karena sistem mengeksekusi perintah manajemen perangkat jaringan secara mandiri berdasarkan kondisi jaringan yang terdeteksi. Perintah */disable_interface* akan mengirim *command /interface disable ether2* ke *router* via SSH, sedangkan */enable_interface* mengirim */interface enable ether2*. Kedua perintah ini memungkinkan administrator mengelola status *interface router* langsung dari chat Telegram tanpa harus masuk ke konsol Mikrotik secara manual.

2.4. Pengujian Sistem

Pengujian dilakukan secara langsung dengan menjalankan dua belas skenario uji. Simulasi kondisi *router down* dikerjakan dengan mengirim perintah */interface*

disable ether2 ke *router* Mikrotik via konsol atau via *command* `/disable_interface` di Telegram, lalu diamati apakah *alert* masuk ke Telegram dalam rentang waktu yang wajar. Pemulihan *router* disimulasikan dengan mengirim perintah `/interface` enable ether2. Seluruh skenario diuji satu per satu dan hasilnya dicatat secara langsung. Kedua belas skenario pengujian beserta indikator keberhasilannya dapat dilihat pada Tabel 2.

Tabel 2. Skenario Pengujian Sistem

No	Skenario Uji	Cara Pelaksanaan	Indikator Keberhasilan
1	<i>Bot</i> merespons <code>/start</code>	Kirim perintah via Telegram	<i>Bot</i> membalas menu perintah lengkap
2	<i>Ping</i> manual <code>/ping</code>	Kirim perintah via Telegram	<i>Bot</i> menampilkan hasil <i>ping</i> beserta output
3	Cek <i>interface</i> <code>/interfaces</code>	Kirim perintah via Telegram	Daftar <i>interface</i> dan status muncul
4	Status <i>router</i> <code>/status</code>	Kirim perintah via Telegram	Info sistem <i>router</i> dan <i>log</i> terakhir muncul
5	Deteksi <i>router</i> DOWN	Kirim <code>/disable_interface</code> via Telegram atau jalankan <code>/interface</code> disable ether2 di konsol Mikrotik	<i>Alert</i> terkirim ke Telegram dalam kurang dari 65 detik
6	Deteksi <i>router</i> PULIH	Kirim <code>/enable_interface</code> via Telegram atau jalankan <code>/interface</code> enable ether2 di konsol Mikrotik	Notifikasi pemulihan masuk ke Telegram
7	Tidak ada <i>alert</i> duplikat	<i>Interface</i> ether2 tetap dalam kondisi disable beberapa siklus monitoring	Hanya satu <i>alert</i> yang terkirim
8	Pencatatan <i>log</i>	Cek file logs/events.txt	Semua <i>event</i> tercatat lengkap dengan <i>timestamp</i>
9	Disable <i>interface</i> via Telegram	Kirim <code>/disable_interface</code> via Telegram	<i>Interface</i> ether2 mati, <i>alert</i> DOWN masuk
10	Enable <i>interface</i> via Telegram	Kirim <code>/enable_interface</code> via Telegram	<i>Interface</i> ether2 aktif, notifikasi PULIH masuk
11	Auto <i>restart</i> <i>interface</i>	Sistem deteksi DOWN	Perintah <code>/interface</code> enable terkirim otomatis ke <i>router</i>
12	Auto <i>backup</i> konfigurasi	<i>Router</i> kembali UP	<i>File</i> .txt tersimpan di <i>folder</i> backups dengan <i>timestamp</i>

3. HASIL DAN PEMBAHASAN

3.1. Hasil Implementasi dan Konfigurasi Jaringan

Sistem berhasil dibangun dan dijalankan pada laptop dengan sistem operasi Windows 10. *Router* virtual Mikrotik CHR berjalan stabil di dalam VirtualBox dengan dua antarmuka jaringan aktif. Antarmuka ether1 mendapat alamat IP 10.0.2.15/24 melalui mode NAT, sedangkan ether2 dikonfigurasi dengan alamat

192.168.56.10/24 melalui mode *Host-Only*. Konfigurasi ini bisa dilihat dari *output* perintah */ip address print* di konsol Mikrotik seperti pada Gambar 4.

```

MMM      MMM      KKK      TTTTTTTTTT      KKK
MMM      MMM      KKK      TTTTTTTTTT      KKK
MMM      MMM      III      KKK      RRRRRR      000000      TTT      III      KKK      KKK
MMM      MM      MMM      III      KKKKK      RRR      RRR      000      000      TTT      III      KKKKK
MMM      MMM      III      KKK      KKK      RRRRRR      000      000      TTT      III      KKK      KKK
MMM      MMM      III      KKK      KKK      RRR      RRR      000000      TTT      III      KKK      KKK

MikroTik RouterOS 7.21.4 (c) 1999-2026      https://www.mikrotik.com/

Press F1 for help

2026-05-17 00:11:05 system,error,critical login failure for user \ladmin via lo
cal

Change your password (Ctrl-C to skip)
new password>
[admin@RouterLab] > /ip address print
Flags: D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE, VRF
# ADDRESS NETWORK INTERFACE VRF
0 192.168.56.10/24 192.168.56.0 ether2 main
1 10.0.2.15/24 10.0.2.0 ether1 main
[admin@RouterLab] >

```

Gambar 4. *Output* */ip address print* pada Konsol Mikrotik CHR

Koneksi antara laptop dan *router* diuji dengan perintah *ping* dari CMD Windows. Hasil *ping* menunjukkan empat paket terkirim, empat paket diterima, dan tidak ada paket yang hilang atau 0% *loss*, dengan waktu respons rata-rata di bawah 1 milidetik. Koneksi SSH juga berhasil dibangun dari laptop ke *router* melalui port 2222 dengan perintah *ssh admin@127.0.0.1 -p 2222*. Setelah login, *prompt* [admin@RouterLab] muncul yang membuktikan koneksi berjalan normal.

```

Command Prompt - ssh admin@127.0.0.1 -p 2222
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ThinkPad>ping 192.168.56.10

Pinging 192.168.56.10 with 32 bytes of data:
Reply from 192.168.56.10: bytes=32 time<1ms TTL=64
Reply from 192.168.56.10: bytes=32 time<1ms TTL=64
Reply from 192.168.56.10: bytes=32 time<1ms TTL=64
Reply from 192.168.56.10: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\ThinkPad>ssh admin@127.0.0.1 -p 2222

[admin@RouterLab] >

```

Gambar 5. Hasil *Ping* dan Koneksi SSH dari CMD Windows

3.2. Hasil Pengujian *Command Bot Telegram*

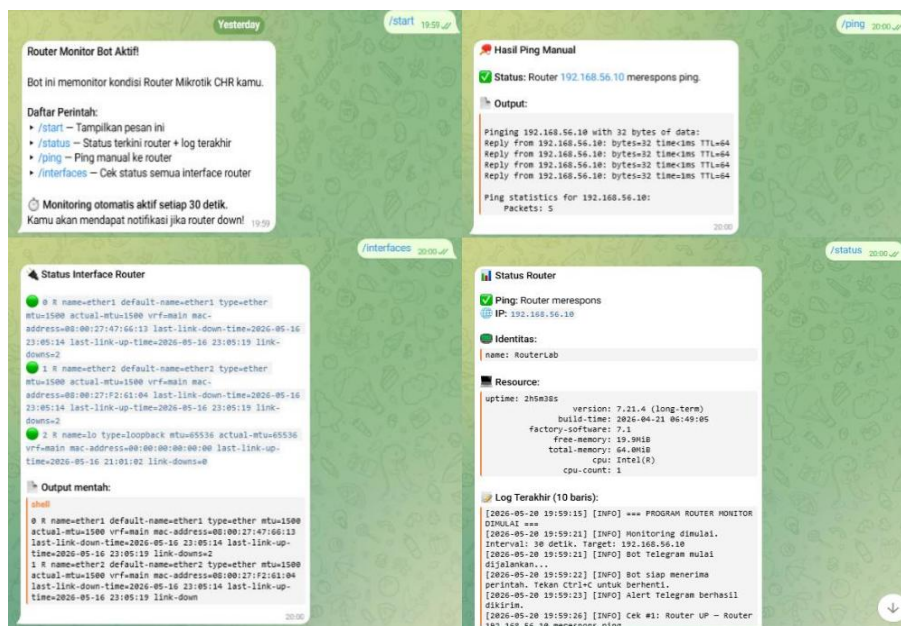
Setelah program dijalankan dengan perintah python main.py, *bot* langsung bisa menerima perintah dari Telegram. Keenam *command* yang dibuat semuanya berfungsi sesuai rancangan. Hasil pengujian masing-masing *command* dicatat dalam Tabel 3.

Tabel 3. Hasil Pengujian *Command Bot Telegram*

No	Command	Waktu Respons	Hasil
1	/start	kurang dari 1 detik	Menu perintah lengkap tampil
2	/ping	2 hingga 3 detik	<i>Output ping</i> 4 paket ditampilkan
3	/interfaces	3 hingga 5 detik	Daftar ether1, ether2, lo dengan status Running
4	/status	4 hingga 6 detik	Info sistem <i>router</i> dan 10 <i>log</i> terakhir tampil
5	/disable_interface	3 hingga 5 detik	<i>Interface</i> ether2 berhasil dinonaktifkan, konfirmasi muncul di Telegram

6 /enable_interface 3 hingga 5 detik Interface ether2 berhasil diaktifkan kembali, konfirmasi muncul di Telegram

Command /ping dan /start merespons lebih cepat karena tidak butuh koneksi SSH. Sementara /interfaces dan /status butuh waktu lebih lama karena harus membuka sesi SSH ke *router* terlebih dahulu, menjalankan perintah, lalu menutup koneksi. Selisih waktu ini wajar dan tidak mengganggu penggunaan secara praktek. *Screenshot* hasil masing-masing *command* dapat dilihat pada Gambar 6.



Gambar 6. Tampilan Balasan Bot Telegram untuk *Command* /start, /ping, /interfaces, dan /status

3.3. Hasil Pengujian *Alert* Otomatis dan Pencatatan *Log*

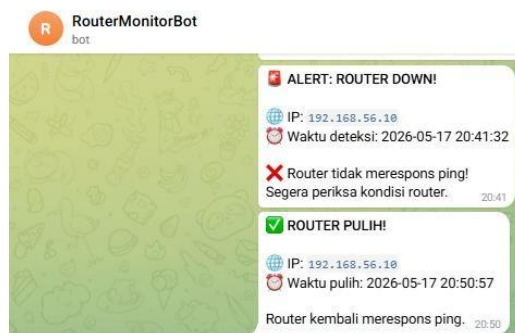
Pengujian deteksi *router down* dilakukan dengan mengirim perintah /disable_interface lewat *bot* Telegram, yang menjalankan /interface disable ether2 pada *router* Mikrotik. Perintah ini memutus konektivitas pada *interface* yang dipantau tanpa mematikan *router*. Setelah *interface* dinonaktifkan, sistem mendeteksi kegagalan *ping* pada siklus pemeriksaan berikutnya dan langsung mengirim pesan *alert* ke Telegram.

Waktu dari *interface* dinonaktifkan sampai *alert* terkirim dicatat pada tiga pengujian yang lengkap, yaitu 44, 50, dan 58 detik, dengan rata-rata 51 detik. Satu pengujian lain tidak dihitung karena status down baru terbaca setelah *interface* diaktifkan kembali. Waktu ini lebih lama dari interval *polling* 30 detik. Satu siklus pemeriksaan memakan sekitar 33 detik saat *router* normal dan 53 sampai 65 detik saat *router down*, kemungkinan karena *ping* yang gagal menunggu *timeout*. Karena itu, indikator awal 35 detik pada Tabel 2 diubah menjadi 65 detik sesuai siklus terpanjang pada *log*.

Setelah *interface* diaktifkan kembali, *router* terbaca UP lagi dalam 35 sampai 44 detik pada tujuh dari sembilan kejadian. Dua kejadian lain memakan 88 detik karena satu siklus tambahan masih terbaca down.

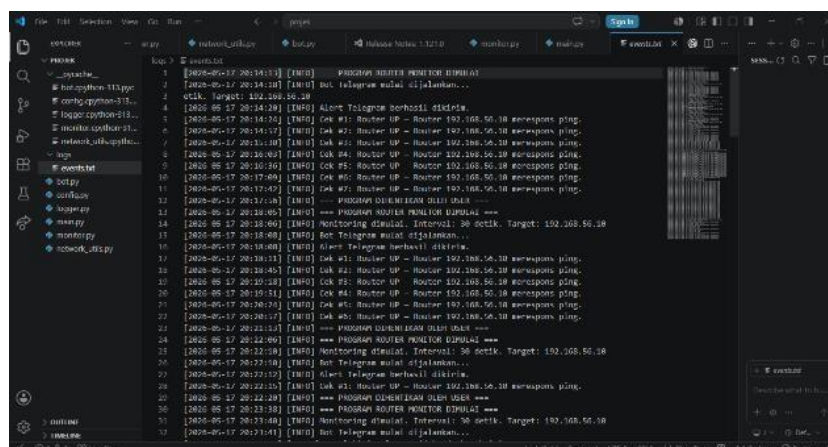
Selama *router* dalam kondisi *down*, hanya satu pesan *alert* yang dikirim

meskipun *monitoring* terus berjalan dengan interval 30 detik. Ini karena variabel *router_was_down* pada *monitor.py* mencegah pengiriman *alert* berulang selama status belum berubah. *Alert* baru dikirim lagi hanya ketika status berubah dari *down* kembali ke *up*. Tampilan kedua pesan tersebut di Telegram dapat dilihat pada Gambar 7.



Gambar 7. Pesan *Alert Router DOWN* dan Notifikasi *Router PULIH* di Telegram

Semua kejadian selama pengujian tercatat di *file logs/events.txt* dengan format *timestamp* yang lengkap. Setiap baris *log* memuat waktu kejadian, jenis *event* dalam tanda kurung siku seperti INFO, WARNING, atau ALERT, dan keterangan singkat. *File log* mencatat 691 baris selama masa pengujian, mencakup 303 siklus *monitoring*, 20 *alert down*, dan 26 kejadian pencegahan *alert* duplikat. Isi *file log* tersebut dapat dilihat pada Gambar 8.



Gambar 8. Isi *File logs/events.txt* yang Terbuka di VS Code

Fitur *auto restart interface* berjalan saat sistem pertama kali mendeteksi *router DOWN*. Sistem langsung mengirim perintah */interface enable ether2* ke *router* via SSH dan notifikasi konfirmasi terkirim ke Telegram. Fitur ini berjalan dalam waktu kurang dari 5 detik setelah *alert DOWN* dikirim. Dari 11 percobaan yang tercatat di *log*, *auto restart* berhasil pada 10 percobaan. Satu percobaan gagal karena koneksi SSH *timeout*.

Fitur *auto backup* konfigurasi berjalan setiap kali *router* terdeteksi kembali *UP*. Sistem mengirim perintah */export* ke *router* dan hasilnya disimpan ke *folder* backups dengan nama *file* bertimestamp seperti *router_config_20260608_155411.txt*. Notifikasi keberhasilan *backup* beserta nama *file* juga dikirim ke Telegram. Selama fitur ini aktif, *log* mencatat 13 *file backup* tersimpan pada 13 kejadian pemulihan *router*.

3.4. Pembahasan

Waktu deteksi rata-rata 51 detik lebih lama dari interval *polling* 30 detik. *Log* menunjukkan satu siklus pemeriksaan memakan 53 sampai 65 detik saat *router down*, sehingga lamanya *ping* yang gagal ikut menambah waktu deteksi. Penelitian tentang *monitoring* berbasis ICMP di lingkungan ISP mencatat notifikasi bisa masuk dalam kurang dari 2 detik ketika sistem berjalan di atas *server* fisik, tetapi *downtime* rata-rata tetap bisa dipangkas dari 30 menit menjadi 3 menit meski ada jeda *polling* [13]. Artinya interval deteksi bukan satu-satunya faktor yang menentukan efektivitas sistem, yang lebih utama adalah apakah sistem bisa mengirim peringatan sebelum *downtime* terasa oleh pengguna.

Dua fitur otomasi yang ditambahkan cukup berguna dalam praktiknya. Fitur *auto restart interface* bekerja dengan mengirim perintah */interface enable* ke *router* via SSH saat gangguan pertama kali terdeteksi, dan fitur *auto backup* konfigurasi menyimpan hasil */export router* ke *file* lokal setiap kali *router* pulih. *Backup* terjadwal otomatis seperti ini memang dianjurkan karena bisa mencegah kehilangan konfigurasi saat *router* mengalami kerusakan atau perlu direset [14].

Dari keseluruhan pengujian, sistem mampu mendeteksi gangguan dan mengirim notifikasi ke Telegram tanpa perlu ada yang mengoperasikan program secara manual. Waktu sekitar 51 detik masih dapat dipertimbangkan untuk *monitoring* skala kecil seperti laboratorium atau kantor, dengan catatan pengujian waktu yang lengkap baru tiga kali. Waktu ini mungkin bisa dipersingkat dengan memperpendek interval atau membatasi *timeout ping*, dan hal itu perlu diuji lebih lanjut. Penggunaan Mikrotik CHR sebagai *router* virtual cukup memadai untuk keperluan ini.

Satu keterbatasan yang ditemukan selama pengerjaan adalah sistem masih bergantung penuh pada koneksi internet untuk bisa mengirim notifikasi ke Telegram. Kalau koneksi internet laptop terputus saat *router down*, *alert* tidak akan sampai ke pengguna meski *monitoring* tetap berjalan dan mencatat *log*. *Log* mencatat 10 kegagalan pengiriman notifikasi karena koneksi internet terputus, dan *alert* yang gagal tidak dikirim ulang karena *flag* sudah terpasang. Pada satu pengujian, *interface* dinonaktifkan lalu diaktifkan kembali dalam 28 detik sehingga gangguan tidak terdeteksi oleh siklus pemeriksaan. Catatan ini perlu diperhatikan jika sistem akan dikembangkan lebih lanjut untuk lingkungan produksi. *Bot* juga belum punya mekanisme autentikasi pengguna, jadi siapa saja yang tahu *username bot* bisa mengirim perintah dan mendapat info *router*. Untuk pengembangan ke depan, perlu ditambah pengecekan *chat ID* supaya hanya administrator yang bisa mengakses [15].

KESIMPULAN

Penelitian ini berhasil mengembangkan sistem *monitoring Router* Mikrotik CHR berbasis Telegram Bot menggunakan Python yang mampu mendeteksi gangguan jaringan dan mengirim notifikasi secara otomatis tanpa infrastruktur *monitoring* tambahan. Pada tiga pengujian yang lengkap, notifikasi gangguan terkirim rata-rata 51 detik setelah *interface* dinonaktifkan. Sistem juga mencegah *alert* duplikat, mencatat 691 baris *log*, menjalankan *auto restart interface* pada 10 dari 11 percobaan, dan menyimpan 13 *file backup* konfigurasi setiap kali *router* pulih. Hasil ini menunjukkan potensi penerapan pada lingkungan laboratorium atau jaringan skala kecil, karena *monitoring* yang responsif dapat dibangun dengan perangkat lunak gratis pada satu *laptop*. Keterbatasannya, jumlah pengujian waktu

masih sedikit, pengujian hanya pada satu *laptop* dengan satu *router* virtual, notifikasi yang gagal akibat internet terputus tidak dikirim ulang, dan *bot* belum punya autentikasi pengguna. Penelitian selanjutnya disarankan menambahkan autentikasi berbasis *chat ID*, pengiriman ulang notifikasi, *log* lokal sebagai cadangan saat internet terputus, serta pemantauan CPU dan memori *router*.

DAFTAR PUSTAKA

- [1] M. I. Baihaqi and S. Raharjo, "Sistem Monitoring Jaringan Dengan Metode Network Development Life Cycle Di PT Lintas Teknologi Solusindo," JUPITER Teknol. Inform. Komput., vol. 6, no. 2, pp. 183–194, 2025, doi: 10.53990/jupiter.v6i2.516.
- [2] A. G. Wahyuriyanto and H. Yulianton, "Implementasi Sistem Monitoring Jaringan Berbasis Simple Network Management Protocol Dengan Notifikasi Telegram," Progresif J. Ilm. Komput., vol. 20, no. 2, pp. 632–647, 2024, doi: 10.35889/progresif.v20i2.1939.
- [3] M. Y. Ishaq and Firmansyah, "Implementasi Sistem Monitoring Menggunakan Zabbix Dan Notifikasi Realtime Telegram," J. Insa. (Journal Inf. Syst. Manag. Innov.), vol. 3, no. 1, pp. 72–77, 2023.
- [4] S. Nurjanah, F. Sembiring, and R. R. Ayuningsih, "Integration of Telegram Bot and Uptime Kuma for Wi-Fi Network Monitoring Using Mikrotik," Pilar Nusa Mandiri J. Comput. Inf. Syst., vol. 20, no. 2, pp. 118–126, 2024, doi: 10.33480/pilar.v20i2.5535.
- [5] M. F. M. Fuzi, N. F. M. Ashraf, and M. N. F. Jamaluddin, "Integrated Network Monitoring using Zabbix with Push Notification via Telegram," J. Comput. Res. Innov., vol. 7, no. 1, pp. 155–163, 2022, doi: 10.24191/jcrinn.v7i1.282.
- [6] F. G. Sembiring, S. Ashillah, A. K. Nasution, and D. Kiswanto, "Analisis Kinerja Routing Dinamis Pada Jaringan Virtual Menggunakan Mikrotik CHR," J. Inform. dan Tek. Elektro Terap., vol. 13, no. 3, pp. 1545–1553, 2025, doi: 10.23960/jitet.v13i3.6516.
- [7] M. P. Afiq, Sudarmaji, and D. Irawan, "Implementasi Sistem Monitoring Jaringan Berbasis Bot Telegram Menggunakan Metode ICMP Monitoring Koperasi Mulia Sejahtera," JMIK (JURNAL Mhs. ILMU KOMPUTER), vol. 6, no. 2, pp. 155–166, 2025.
- [8] W. Baydowi, R. Ester, R. Rinaldi, and R. A. Najmi, "Implementasi Sistem Monitoring Jaringan Terpusat Menggunakan Mikrotik Dan Bot Telegram Di PT. Mitra Bersama Jaya," JRIIN J. Ris. Inform. dan Inov., vol. 3, no. 3, pp. 650–657, 2025.
- [9] F. Ardiyansyah, K. Setiawan, and N. Sutisna, "Implementasi IDS pada Jaringan Komputer Menggunakan Snort Berbasis Chatbot Telegram," MALCOM Indones. J. Mach. Learn. Comput. Sci., vol. 4, no. 4, pp. 1614–1623, 2024.
- [10] Y. Hermawan, E. Wahyudi, and J. G. A. Ginting, "Analisis Performansi Otomasi Jaringan Pada Routing OSPF dan EIGRP Berbasis Web Menggunakan Bahasa Pemrograman Python," J. LITEK J. List. Telekomun. Elektron., vol. 21, no. 2, pp. 59–68, 2024.
- [11] N. F. A. Hakim, S. Elvyanti, and Y. Ardiyansyah, "Design of a Telegram- Integrated Grafana Network Monitoring System for Enhanced Fault Detection and Real-time Notification," J. Ilm. Teknol. dan Rekayasa, vol. 30, no. 1, pp. 95–107, 2025, doi: 10.35760/tr.2025.v30i1.14058.

-
- [12] W. Prabowo and R. Ariyanto, "Implementation of Python-Based Network automation Technology for Computer Infrastructure Optimization," *IMPACT Multidiscip. J.*, vol. 1, no. 01, pp. 34–40, 2025.
 - [13] Febriana, R. Waluyo, and D. Mustofa, "Network Monitoring Using Zabbix ICMP Ping and Telegram Notifications Using the Network Development Life Cycle Model," *J. Technol. Informatics*, vol. 8, no. 1, pp. 62–72, 2026.
 - [14] A. H. Prayoga, K. Insani, and D. A. Farizal, "Implementasi Otomatisasi Backup Pada Router Mikrotik Server Jaringan Internet Universitas Majalengka," *INFOTECH J.*, vol. 10, no. 2, pp. 228–233, 2024, doi: 10.31949/infotech.v10i2.10851.
 - [15] R. Andriani and A. Sa'di, "Implementasi Notifikasi Bot Telegram pada Sistem Monitoring Perangkat Jaringan," *Sist. J. Sist. Inf.*, vol. 13, no. 1, pp. 74–82, 2024, [Online]. Available: <http://sistemasi.ftik.unisi.ac.id>