Manajemen Jaringan Terpusat untuk Konfigurasi dan Otomatisasi Pemulihan menggunakan Paramiko dan Django *Framework*

http://dx.doi.org/10.28932/jutisi.v11i3.11431

Riwayat Artikel
Received: 06 Maret 2025 | *Final Revision*: 30 November 2025 Bulan 20xx | *Accepted*: 02 Desember 2025

Hillman Akhyar Damanik ✉ #1, Merry Anggraeni *2

1,2*Program Studi Teknik Informatika, Universitas Budi Luhur, Jl. Ciledug Raya, RT.10/RW.2, Petukangan Utara, Kec. Pesanggrahan, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta 12260*

[1]hillman.akhyardamanik@budiluhur.ac.id

[2]merryanggraeni@budiluhur.ac.id

✉Corresponding author: hillman.akhyardamanik@budiluhur.ac.id

Abstrak **— Pengelolaan konfigurasi dan pemulihan perangkat jaringan secara otomatis merupakan faktor penting dalam menjaga kontinuitas layanan dan ketersediaan jaringan, terutama pada perangkat *router*. Penelitian ini bertujuan mengimplementasikan sistem otomasi konfigurasi jaringan terpusat pada perangkat virtual *Cloud Host Router* (vCHR) dan RB750. Fokus konfigurasi mencakup penamaan *interface*, pengaturan VLAN mode *access* dan *trunk*, *IP Addressing*, serta konfigurasi *firewall* Network Address Translation (NAT), *mangle*, dan *filter*. Sistem ini dirancang untuk mengurangi waktu eksekusi konfigurasi dan pemulihan sebanyak 496 baris skrip yang dikirim simultan ke 10 unit *router* vCHR dan RB750. Otomasi konfigurasi dan *backup* dikembangkan menggunakan Django *Framework* sebagai antarmuka web, *library* Paramiko untuk komunikasi Secure Shell (SSH), serta protokol Secure Copy Protocol (SCP) untuk transfer *file* konfigurasi secara aman. Hasil percobaan menunjukkan peningkatan efisiensi signifikan dan penurunan waktu konfigurasi serta *backup* pada 10 *node router* vCHR dan RB750. Setiap *router* dialokasikan kecepatan transfer data 1024 Kbps, dengan kapasitas *file backup* 47 KB hingga 59 KB. Kecepatan tersebut menghasilkan estimasi waktu *backup* sekitar 7,15–7,79 detik per perangkat. Parameter *uptime* dan *downtime* yang meliputi persentase ICMP *loss*, *response time*, dan *unavailable* ICMP menunjukkan bahwa selama proses pengiriman konfigurasi, kinerja jaringan tetap optimal dengan rata-rata *availability* 100%. Hasil *throughput router* tercatat sebesar vCHR-C1 50 Kbps, vCHR-C2 49 Kbps, vCHR-C3 49 Kbps, vCHR-C4 32 Kbps, vCHR-C5 50 Kbps, RB750-C6 38 Kbps, RB750-C7 37 Kbps, RB750-C8 37 Kbps, RB750-C9 38 Kbps, dan RB750-C10 22 Kbps.**

Kata kunci— **CHR; Django; *Network Automation*; Paramiko; SCP.**

Centralized Network Management for Configuration and Recovery Automation using Paramiko and Django Framework

Abstract — ***Automatic configuration and recovery management for network devices are crucial for maintaining service continuity and network availability, especially for routers. This study aims to implement a centralized network configuration automation system on virtual Cloud Host Router (vCHR) and RB750 devices. The configuration focuses on interface naming, VLAN setup in access and trunk modes, IP Addressing, and Network Address Translation (NAT), mangle, and filter firewall configurations. This system is***

designed to reduce the execution time for configuring and recovering 496 lines of script sent simultaneously to 10 vCHR and RB750 routers. The configuration automation and backup were developed using the Django Framework for the web interface, the Paramiko library for Secure Shell (SSH) communication, and the Secure Copy Protocol (SCP) for secure configuration file transfers. Experimental results show a significant increase in efficiency and a decrease in configuration and backup time on 10 router nodes (vCHR and RB750). Each router was allocated a data transfer speed of 1024 Kbps, with backup file sizes ranging from 47 KB to 59 KB. This speed resulted in an estimated backup time of approximately 7.15–7.79 seconds per device. Uptime and downtime parameters, including ICMP loss percentage, response time, and unavailable ICMP, indicate that network performance remained optimal with an average 100% availability during the configuration delivery process. The recorded router throughput was vCHR-C1 50 Kbps, vCHR-C2 49 Kbps, vCHR-C3 49 Kbps, vCHR-C4 32 Kbps, vCHR-C5 50 Kbps, RB750-C6 38 Kbps, RB750-C7 37 Kbps, RB750-C8 37 Kbps, RB750-C9 38 Kbps, and RB750-C10 22 Kbps.

Keywords— **CHR; Django, *Network Automation*; Paramiko; SCP.**

I. PENDAHULUAN

Tantangan dalam pengelolaan infrastruktur jaringan semakin kompleks, mencakup aspek konfigurasi, pemulihan, pencegahan, dan pemeliharaan perangkat seperti *router* yang terus berkembang baik dari sisi teknologi maupun skalabilitasnya [1], [2]. Pengelolaan infrastruktur jaringan dengan cara tradisional seringkali membutuhkan waktu, rentan terhadap kesalahan manusia, dan sulit diukur dari segi efisiensi, terutama bagi perusahaan atau organisasi yang mengelola banyak perangkat dan konfigurasi yang beragam [3], [4]. Seiring dengan semakin banyaknya perangkat *router* yang terhubung dengan infrastruktur jaringan dan kebutuhan untuk mengelola berbagai konfigurasi, hal ini mendorong teknologi dan solusi yang lebih efisien, efektif, dan akurat. Menjawab tantangan ini, otomatisasi jaringan muncul sebagai solusi yang menjanjikan. Otomatisasi jaringan memungkinkan administrator jaringan untuk mengurangi beban kerja manual, meminimalkan kesalahan konfigurasi, dan mempercepat penerapan konfigurasi, sehingga perusahaan dapat mengelola jaringan yang lebih besar dan lebih kompleks tanpa meningkatkan beban kerja yang berlebihan [5], [6].

Perangkat *router* Mikrotik telah menjadi pilihan populer di banyak perusahaan di dunia termasuk Indonesia seperti Small Office Home Office dan Internet Service Provider (ISP), karena kemudahan, fleksibilitas, dan fungsionalitas untuk instalasi dan konfigurasi, terutama melalui sistem operasi RouterOS [7], [8]. Mikrotik RouterOS menyediakan berbagai fitur dan paket yang dapat disesuaikan dengan skala kebutuhan jaringan, namun konfigurasi manual sering kali menjadi kendala bagi banyak perusahaan yang ingin mengelola jaringannya secara lebih efisien. Mengingat jaringan yang semakin kompleks, maka sangat penting untuk menemukan metode yang memudahkan pengelolaan perangkat ini melalui otomatisasi [9], [10]. Salah satu pustaka *Python* yang dapat digunakan untuk mengotomatisasi konfigurasi perangkat adalah Paramiko, yaitu pustaka yang memungkinkan koneksi otomatis ke perangkat jaringan *router* menggunakan protokol SSH dan *Secure Protocol* SCP [11].

Kerangka kerja Django memiliki peran strategis dalam penelitian ini sebagai fondasi pengembangan antarmuka dan *backend* sistem otomasi konfigurasi [12]. Django menawarkan arsitektur modular yang lengkap termasuk ORM (*Object-Relational Mapping*) yang memungkinkan pengembangan perangkat monitoring infrastruktur kelas *enterprise* secara cepat dan terstruktur [13], [14], [15]. Implementasi pustaka Paramiko memungkinkan administrator jaringan untuk mengotomatisasi berbagai tugas konfigurasi pada perangkat RouterOS tanpa berinteraksi langsung dengan antarmuka *Command Line Interface* (CLI). Sementara itu, Django sebagai web *framework* berbasis Python dapat digunakan untuk membangun antarmuka pengguna berbasis web yang mudah digunakan, memungkinkan administrator untuk memantau, mengelola perangkat, dan mengawasi kinerja jaringan melalui *dashboard* yang terpusat. Dengan menggabungkan kedua alat ini, otomatisasi dan manajemen perangkat jaringan dapat dilakukan dengan lebih efisien dan terorganisir . Walaupun banyak penelitian terdahulu telah dilakukan untuk solusi otomasi perangkat jaringan, namun sebagian besar fokusnya masih terbatas pada konfigurasi konfigurasi pada perangkat *router*, bukan pada manajemen jaringan terpusat, dilengkapi fitur konfigurasi otomatis, pencadangan konfigurasi via SCP, integrasi *database*, *logging* terpusat, serta pemantauan *throughput*.

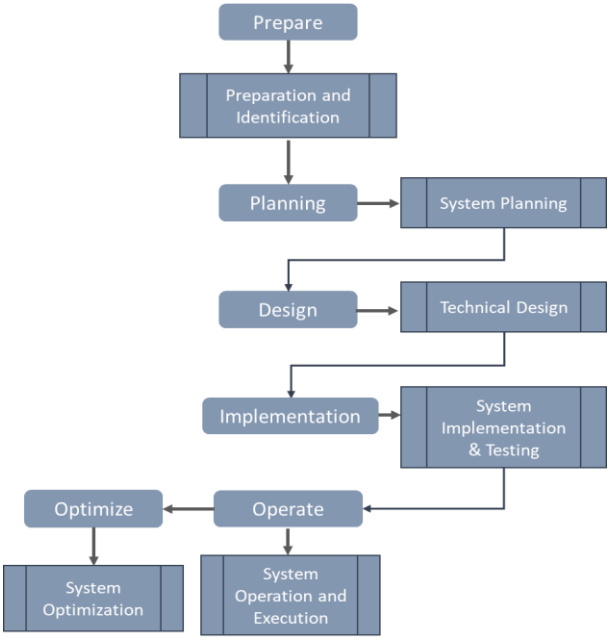
Penelitian oleh Mazin et al. [3] membahas efisiensi penggunaan *scripting* untuk mengotomatisasi konfigurasi perangkat jaringan, dengan membandingkan waktu eksekusi antara metode manual dan otomatis menggunakan emulator PNETLAB di VMware yang memuat 30 perangkat multi-vendor seperti Cisco, Juniper, dan Huawei. Hasilnya menunjukkan bahwa otomasi mampu mengonfigurasi seluruh perangkat hanya dalam 97 detik tanpa kesalahan. Penelitian yang dilakukan oleh Dumitrache et al. [16] membahas pemanfaatan Python dan *Graphical Network Simulator*-3 (GNS3) untuk menganalisis protokol *routing* OSPF dan EIGRP, di mana Python digunakan untuk mengotomatisasi konfigurasi serta *troubleshooting* perangkat jaringan. Penelitian oleh Datta et al. [17] hanya menerapkan otomasi jaringan menggunakan Ansible untuk mengkonfigurasi protokol *routing* BGP. Hasil penelitian menunjukkan bahwa Ansible mampu menerapkan konfigurasi tanpa kesalahan, mengurangi potensi *human error*, mempercepat proses, dan meningkatkan visibilitas perangkat dalam jaringan. Metode *Network Automation* (NA) diterapkan oleh [18] hanya untuk otomasi *routing*, administrasi virtual, serta pencadangan atau pemulihan konfigurasi, di lingkungan virtual berbasis GNS3 dan VirtualBox. Penelitian yang dilakukan oleh Efendi et al. [19] hanya mengusulkan infrastruktur jaringan melalui integrasi teknologi VXLAN, otomasi operasi sirkuit dengan Python, dan

konfigurasi *Network Operating System* (NOS) berbasis Ansible, dan dilengkapi dengan pencadangan konfigurasi otomatis menggunakan GitHub. Penelitian oleh Zulfikar et al. [20] menggunakan Ansible dengan metode *Network Development Operations* (NetDevOps) hanya mengotomatisasi proses *backup* dan konfigurasi *router* MikroTik dengan *Command Line Interface* (CLI). Penelitian oleh Li et al. [21] hanya mengusulkan *platform* otomasi jaringan yang bersifat *plug-in*, memiliki untuk mengatasi keterbatasan kerangka kerja otomasi jaringan tradisional yang umumnya menggunakan *pseudo-language* dan sulit diintegrasikan dengan sistem lain karena API yang kompleks.

Berdasarkan berbagai penelitian sebelumnya, otomasi jaringan telah banyak diterapkan dengan beragam pendekatan dan teknologi, seperti penggunaan Ansible untuk konfigurasi masal protokol BGP dan pencadangan konfigurasi *router* MikroTik, pemanfaatan Netmiko dalam analisis protokol *routing* di lingkungan simulasi GNS3, integrasi VXLAN dan Python untuk pengelolaan segmen jaringan berskala besar, serta penerapan REST API berbasis Django untuk perangkat Cisco CSR1000V. Beberapa studi juga mengembangkan solusi berbasis Paramiko dan Netmiko untuk administrasi jaringan virtual, maupun *platform* otomasi jaringan dengan arsitektur *plug-in* dan *high concurrency* guna meningkatkan fleksibilitas dan efisiensi. Meskipun menunjukkan peningkatan efisiensi konfigurasi dan pencadangan dibandingkan metode manual, sebagian besar penelitian tersebut berfokus pada perangkat dan lingkungan simulasi tertentu, atau memiliki ketergantungan pada ekosistem otomasi spesifik seperti Ansible atau REST API. Penelitian ini mengisi celah tersebut dengan mengimplementasikan sistem manajemen jaringan terpusat berbasis Django dan Paramiko yang diterapkan langsung pada perangkat MikroTik fisik RB750 dan virtual CHR di lingkungan nyata. Sistem ini tidak hanya mengotomatisasi konfigurasi melalui SSH, tetapi juga mengintegrasikan pencadangan konfigurasi via SCP, *logging* terpusat, pengelolaan *database* perangkat, serta pemantauan *throughput*, sehingga memberikan fleksibilitas kontrol penuh, kemudahan integrasi, dan dukungan operasional yang optimal untuk skala kecil hingga menengah.

II. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini mengadopsi *framework* PPDIOO (*Prepare, Plan, Design, Implement, Operate*, dan *Optimize*) sebagai dasar sistem manajemen jaringan terpusat. *Framework* ini dirancang untuk memastikan bahwa semua fase sistem dilakukan secara sistematis dan komprehensif, termasuk proses desain awal, implementasi, dan evaluasi kinerja sistem secara komprehensif. Gambar 1 mengilustrasikan pendekatan kerangka kerja PPDIOO yang diusulkan, di mana setiap fase dalam kerangka kerja memiliki peran penting dalam memastikan keberhasilan sistem otomasi yang dibangun.



Gambar 1. Metode Pendekatan Penelitian

Pendekatan metode, untuk setiap aspek siklus sistem dapat dipetakan dengan jelas, dari perencanaan strategis hingga optimasi, untuk memastikan efektivitas dan keberlanjutan operasi jaringan berbasis perangkat MikroTik. Tahap perencanaan dan desain berfokus pada analisis kebutuhan dan desain arsitektur sistem, sedangkan fase implementasi dan operasional

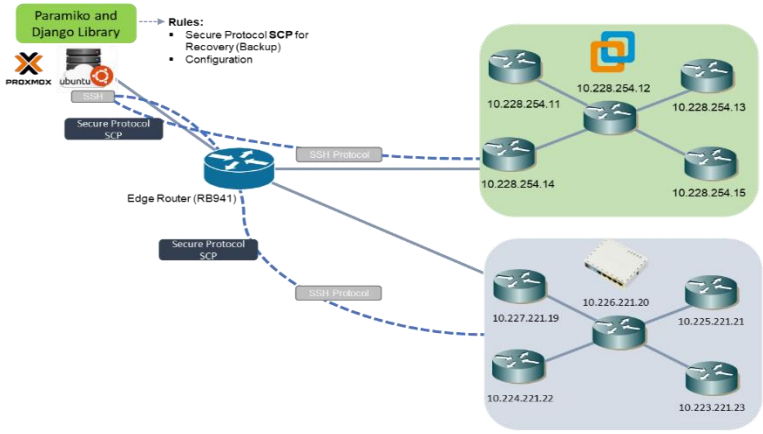
berfokus pada penerapan teknologi otomasi untuk memfasilitasi manajemen jaringan. Sementara itu, fase optimasi bertujuan untuk memastikan bahwa sistem yang dikembangkan dapat terus disesuaikan dengan kebutuhan jaringan yang berubah, sehingga tetap relevan dan efisien dalam jangka panjang. PPDIOO berperan penting dalam membangun alur kerja yang sistematis pada implementasi otomasi konfigurasi dan *backup* perangkat *router* vCHR dan RB750, dengan uraian sebagai berikut.

A. Tahap Persiapan dan Perencanaan

Pada tahap persiapan ini, langkah utama yang dilakukan adalah mengidentifikasi dan menyiapkan semua elemen dan perangkat yang dibutuhkan, baik dari segi perangkat keras maupun perangkat lunak. Sistem manajemen jaringan terpusat akan mencakup otomatisasi konfigurasi, pemantauan kinerja, dan mekanisme pemulihan yang mencakup proses pencadangan dan pemulihan data pada perangkat vCHR dan RB750 MikroTik. MikroTik RB941 berperan sebagai *gateway* utama dan pusat distribusi konektivitas, yang menghubungkan jaringan virtual Hypervisor dan MikroTik dengan *server* untuk mendukung proses pengujian dan simulasi infrastruktur jaringan. Dalam pengembangan antarmuka manajemen, digunakan Visual Studio Code sebagai *platform* utama pengembangan sistem. *Editor* ini akan diintegrasikan dengan *server* berbasis Ubuntu yang berperan sebagai penghubung antara Django dengan perangkat vCHR dan RB750 MikroTik dalam lingkungan *hypervisor*, sehingga komunikasi dan manajemen konfigurasi dapat dilakukan secara efisien dan otomatis.

B. System Planning

Desain topologi jaringan terpusat dirancang untuk mengintegrasikan berbagai komponen jaringan, baik perangkat keras maupun perangkat lunak, sehingga tercipta sistem manajemen jaringan yang efisien. Pada rancangan ini, MikroTik RB941 berperan sebagai *gateway* utama yang mengatur distribusi konektivitas antara jaringan vCHR dan RB750, *server* Ubuntu yang menjadi *platform* untuk Django dan Paramiko, serta lingkungan virtualisasi yang menggunakan Proxmox dan *hypervisor* VMware. Konfigurasi dan arsitektur pemulihan juga menjadi bagian dari desain ini. Gambar 2 menunjukkan skema topologi jaringan tersebut.



Gambar 2. *Network Architecture* dan *System Design*

Alokasi alamat IP pada arsitektur jaringan diatur sedemikian rupa agar setiap perangkat memperoleh jalur komunikasi yang aman melalui *gateway* yang telah ditentukan. *Router gateway* utama menggunakan alamat IP 10.229.254.254/24, sementara rentang IP dalam *subnet* 10.228.254.0/24 diprioritaskan khusus untuk instans vCHR yang dijalankan pada *platform* VMware. Di sisi lain, server Ubuntu, Proxmox, dan *router gateway* MikroTik lainnya ditempatkan dalam *subnet* 10.151.20.128/25. Pengelompokan ini bertujuan untuk menjaga keterpaduan infrastruktur jaringan terpusat sekaligus memastikan proses komunikasi antar sistem berlangsung secara optimal. Tabel 1 memperlihatkan rincian pembagian alamat IP tersebut.

TABEL 1
IP ADDRESS YANG DIGUNAKAN DALAM ARSITEKTUR JARINGAN

IP *Address*	***Subnet Mask***	***Hostname***
10.151.20.230	255.255.255.128	Proxmox *Hypervisor*
10.229.254.254	255.255.255.0	*Gateway*
10.151.20.143	255.255.255.128	vUbuntu Server (Django dan Paramiko)
10.223.221.23	255.255.255.248	RB750-C10

IP *Address*	***Subnet Mask***	***Hostname***
10.224.221.22	255.255.255.248	RB750-C9
10.225.221.21	255.255.255.248	RB750-C8
10.226.221.20	255.255.255.248	RB750-C7
10.227.221.19	255.255.255.248	RB750-C6
10.228.254.15	255.255.255.0	vCHR-C5
10.228.254.14	255.255.255.0	vCHR-C4
10.228.254.13	255.255.255.0	vCHR-C3
10.228.254.12	255.255.255.0	vCHR-C2
10.228.254.11	255.255.255.0	vCHR-C1

Sistem yang dikembangkan memanfaatkan *framework* Django, Paramiko dan *secure protocol* SCP sebagai solusi untuk mengotomatisasi konfigurasi *router* melalui protokol SSH. Pendekatan Paramiko memungkinkan pembentukan dan pengelolaan koneksi SSH ke perangkat *router* secara otomatis, sambil menerima dan mengeksekusi *respons* dari perangkat MikroTik yang dikendalikan dari jarak jauh. Sebanyak 10 *node* vCHR dan RB750 diinstal dan dikonfigurasi dalam lingkungan VMware *Hypervisor*, sedangkan *hypervisor* digunakan untuk mendukung pengoperasian server virtual yang menjalankan Django dan Paramiko. Seluruh sistem terintegrasi dengan Visual Studio Code sebagai alat pengembangan utama dalam manajemen infrastruktur jaringan. Semua perangkat jaringan dalam sistem ini terhubung melalui MikroTik RB941, yang juga berfungsi sebagai pusat distribusi untuk memastikan komunikasi yang optimal antara lingkungan virtual dan fisik. Manajemen jaringan terpusat untuk *router* dirancang dengan mengintegrasikan mekanisme otomatisasi dalam pencadangan konfigurasi perangkat. Proses ini menggunakan protokol aman SCP untuk memastikan keamanan dalam *transfer file* konfigurasi. Dengan menerapkan aturan SPClient, setiap perangkat dapat melakukan pencadangan konfigurasi terjadwal atau sesuai permintaan, memastikan bahwa setiap perubahan dalam sistem terdokumentasi dengan baik dan dapat diakses saat diperlukan. Selain itu, mekanisme ini memberikan keandalan dalam mengurangi gangguan jaringan dengan memungkinkan pemulihan cepat jika terjadi kegagalan pada perangkat *router*.

C. Technical Design

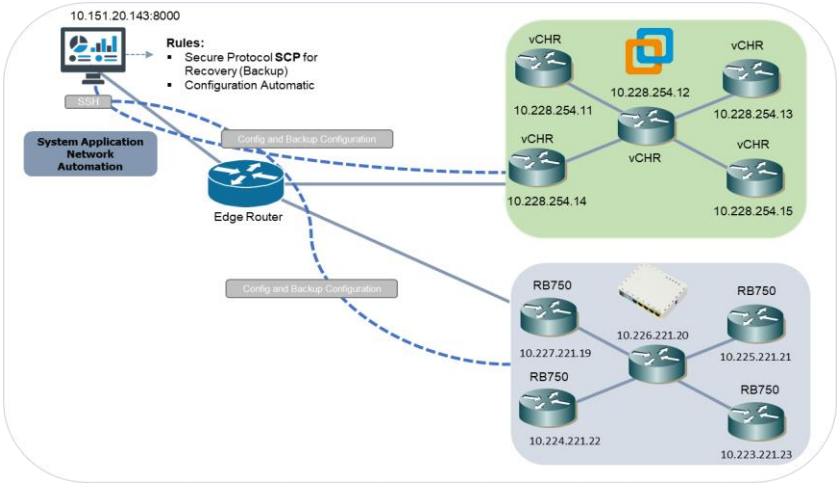
Pada tahap perancangan, dilakukan implementasi sistem manajemen jaringan terpusat dengan mengacu pada spesifikasi kebutuhan dan model arsitektur yang telah disusun pada tahap perencanaan. Proses ini meliputi perancangan mekanisme integrasi Django sebagai *framework* berbasis web yang berfungsi untuk mengelola perangkat *router* secara efisien. Sistem ini meliputi berbagai fitur utama, seperti tampilan *dashboard*, daftar perangkat, konfigurasi otomatis, verifikasi konfigurasi, *log record*, administrasi *database*, serta mekanisme *backup*, *backup* status dan *restore* berbasis protokol aman SCP untuk menjamin kelangsungan operasi jaringan. Lebih jauh, tahap perancangan ini menitikberatkan pada pengembangan arsitektur sistem teknis yang mampu mengotomatisasi proses konfigurasi perangkat *router* sekaligus mendukung pemulihan sistem dalam kondisi darurat. Implementasi skema *backup* akan memastikan setiap perubahan konfigurasi terdokumentasi dengan baik dan dapat dipulihkan dengan cepat jika terjadi gangguan. Dengan pendekatan ini, sistem manajemen jaringan yang diusulkan tidak hanya meningkatkan efisiensi administrasi perangkat tetapi juga memperkuat ketahanan jaringan terhadap potensi kegagalan atau perubahan yang tidak diinginkan.

1) Network Topology

Topologi jaringan terpusat yang dirancang dalam penelitian ini memanfaatkan MikroTik RB941 sebagai *gateway* distribusi, yang mengatur jalur konektivitas menuju server Ubuntu. Hubungan antara *router* CHR dan *router* RB750 memanfaatkan protokol SSH untuk proses konfigurasi serta pemulihan (*backup*). Fokus utama perancangan ini adalah pengujian otomatisasi konfigurasi dan proses pemulihan pada perangkat *router*, bukan pengembangan layanan interkoneksi antarperangkat. Struktur topologi dibagi menjadi dua kelompok perangkat. Kelompok pertama berisi lima unit CHR yang dijalankan di lingkungan virtualisasi Proxmox, masing-masing memperoleh alamat IP dalam *subnet* 10.228.254.0/24. Konfigurasi yang diuji mencakup penamaan *interface*, pengaturan VLAN dalam mode *access* dan *trunk*, penetapan alamat IP, hingga implementasi *firewall NAT*, *filter*, dan *IP firewall mangle*, dengan total 496 baris skrip konfigurasi per perangkat. Kelompok kedua terdiri atas lima unit *router* fisik MikroTik RB750 yang berada pada rentang *subnet* 10.223.221.0/24 hingga 10.227.221.0/24. Perangkat RB750 ini dikonfigurasi dengan prosedur yang sama seperti kelompok vCHR, termasuk pengiriman skrip sebanyak 496 baris per unit yang berisi pengaturan VLAN, IP *Addressing*, NAT, *filter*, dan *IP firewall mangle*. Gambar 3 menggambarkan susunan topologi jaringan tersebut**.** Dengan demikian, tujuan utama dari topologi ini adalah menciptakan lingkungan simulasi konfigurasi masal, di mana aplikasi yang

p-ISSN : 2443-2210
e-ISSN : 2443-2229

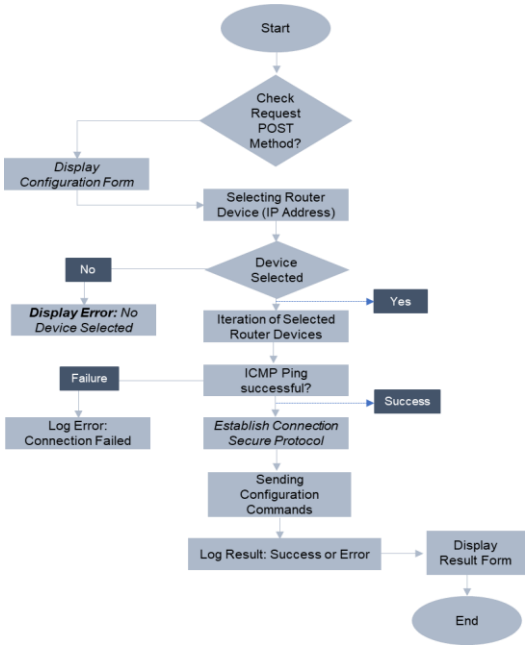
dikembangkan bertugas mengelola proses pengiriman konfigurasi secara otomatis dan *backup* konfigurasi serentak untuk semua perangkat, tanpa melibatkan interkoneksi aktif antar *router* seperti pada deployment jaringan produksi.



Gambar 3. Network Topology

2) Flow Konfigurasi Perangkat Router

Alur kerja sistem yang digunakan untuk proses konfigurasi perangkat *router* CHR dan RB750 diawali dengan pemeriksaan metode permintaan HTTP yang diterima oleh basis data Django. Apabila permintaan yang masuk menggunakan metode selain POST, aplikasi akan menampilkan formulir konfigurasi yang memungkinkan pengguna memilih perangkat *router* yang akan diatur serta mengisi parameter konfigurasi yang diperlukan. Setelah formulir dikirimkan, proses kembali ke tahap awal untuk memeriksa apakah metode permintaan adalah POST. Sistem kemudian menyaring perangkat berdasarkan alamat IP yang dipilih dan melakukan iterasi terhadap setiap perangkat guna memverifikasi konektivitas melalui ICMP ping. Jika hasil *ping* positif, koneksi SSH yang aman dibangun menggunakan pustaka Paramiko untuk mengirimkan perintah konfigurasi secara otomatis. Semua proses, baik yang berhasil maupun yang gagal, dicatat dalam *log*, sedangkan hasil akhirnya ditampilkan kepada pengguna melalui halaman hasil (*result form*). Gambar 4 menampilkan ilustrasi alur kerja sistem tersebut.



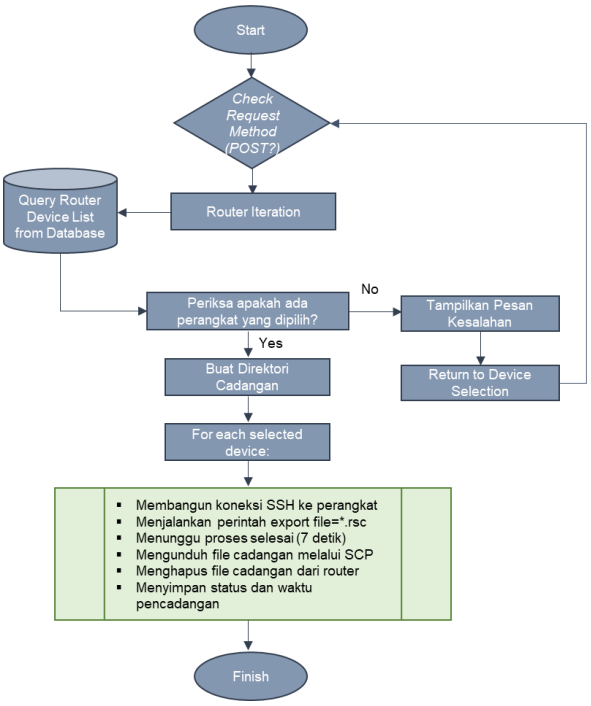
Gambar 4. Flow Konfigurasi Perangkat *Router*

3) Secure Protocol SCP (Backup and Time Backup)

Secure protocol SCP diimplementasikan sebagai metode *transfer file* yang lebih aman, menggunakan enkripsi berbasis SSH untuk melindungi *file* selama proses pencadangan untuk konfigurasi perangkat vCHR dan RB750. Proses pencadangan dilakukan dengan menyalin *file* konfigurasi *.rsc dari perangkat vCHR dan RB750 ke *server* manajemen menggunakan skrip *Python* yang mengimplementasikan pustaka Paramiko atau scpclient. Aturan tersebut melakukan pencadangan untuk semua *router* vCHR dan RB750 dengan langkah-langkah berikut:

- Menghubungkan ke setiap *router* perangkat menggunakan SSH (dengan Paramiko).
- Menjalankan perintah ex*port file*=*.rsc pada *router* vCHR dan RB750 untuk membuat *file* konfigurasi cadangan.
- Mengunduh *file* cadangan menggunakan protokol SCP yang aman.
- Menghapus *file* cadangan dari *router* vCHR dan RB750 setelah berhasil diunduh.
- Mencatat waktu pelaksanaan dan status keberhasilan pencadangan untuk setiap perangkat.

Diagram alir pada skrip pemulihan status waktu dirancang untuk menggambarkan secara terstruktur tahapan pemulihan konfigurasi perangkat *router* melalui protokol SCP. Proses dimulai dengan pengambilan daftar perangkat yang tersimpan dalam basis data Django, diikuti pemeriksaan apakah pengguna telah memilih perangkat untuk dilakukan *backup*. Jika tidak ada perangkat yang dipilih, sistem segera memberikan pesan kesalahan agar pengguna memperbaiki pilihan sebelum kembali ke tahap pemeriksaan metode permintaan. Apabila terdapat perangkat yang dipilih, skrip akan membuat direktori cadangan di sistem lokal sebagai lokasi penyimpanan *file backup*. Selanjutnya, proses pencadangan dilakukan pada setiap perangkat melalui beberapa tahapan utama: membangun koneksi SSH, menjalankan perintah export file=*.rsc pada *router* untuk menghasilkan *file* cadangan, menunggu proses selesai, lalu mengunduh *file* tersebut menggunakan SCP. Setelah proses unduh berhasil, skrip menghapus *file* cadangan dari perangkat *router* guna mengoptimalkan penggunaan ruang penyimpanan. Gambar 5 memperlihatkan alur proses pemulihan tersebut.



Gambar 5. Flow *Backup* and Time Status *Backup*

D. System Implementation and Testing

Pada tahap implementasi, seluruh elemen dan infrastruktur perangkat *router* yang telah dirancang akan diimplementasikan ke dalam suatu sistem manajemen jaringan yang terpusat. Proses ini meliputi instalasi dan konfigurasi perangkat keras dan perangkat lunak, yang dilanjutkan dengan integrasi berbagai komponen sistem. Langkah-langkah yang dilakukan meliputi konfigurasi awal, pengecekan ulang terhadap konfigurasi yang telah diimplementasikan, dan mekanisme pemulihan data melalui proses konfigurasi dan *backup*. Implementasi sistem ini menggunakan Django *framework* untuk memastikan integrasi yang optimal. *Framework* Django digunakan sebagai *framework* utama pengembangan sistem, sedangkan *library* Paramiko digunakan untuk mengelola konektivitas perangkat *router* vCHR dan RB750. Selain itu, digunakan protokol *secure* SCP

pada proses transfer data, *backup*, dan status *time backup* dengan infrastruktur berbasis *server* Ubuntu yang berjalan pada lingkungan *hypervisor* sebagai *platform* utamanya. Setelah sistem konfigurasi dan *backup* berhasil diimplementasikan, dilakukan serangkaian pengujian untuk mengevaluasi sistem yang diimplementasikan. Setiap skenario disusun untuk mengukur efektivitas sistem dalam menjalankan fungsi otomasi konfigurasi, *backup*, serta stabilitas koneksi jaringan. Berikut adalah uraian skenario yang dilakukan:

1) Pengujian otomasi pengiriman konfigurasi - Skenario ini diuji untuk mengirimkan 496 baris konfigurasi secara simultan dan bersamaan ke seluruh *node router* vCHR dan RB750. Validasi hasil konfigurasi dilakukan melalui inspeksi koneksi SSH untuk memastikan setiap baris perintah diterapkan dengan benar.
2) Pengujian pemulihan konfigurasi *router* - Pengujian ini bertujuan untuk mengukur keberhasilan sistem manajemen otomasi terpusat dalam melakukan pemulihan konfigurasi pada seluruh perangkat *router* vCHR dan RB750 *router* menggunakan protokol SCP. Selain itu, dilakukan pencatatan waktu proses *backup* serta verifikasi bahwa *file* hasil *backup* tersimpan.
3) Pengujian kinerja *uptime* dan *downtime router* selama pengiriman konfigurasi - Pengujian ini bertujuan untuk memverifikasi kestabilan operasional perangkat *router* selama proses pengiriman konfigurasi berlangsung. Parameter yang diuji meliputi *uptime* perangkat, tingkat kehilangan paket ICMP (ICMP loss), serta waktu *respons* ICMP. Monitoring dilakukan menggunakan *platform* Zabbix untuk memantau kondisi jaringan secara *real-time*.
4) Pengujian *throughput* antar *node router* vCHR dan RB750 - Pengujian ini dilakukan untuk mengetahui dampak pengiriman konfigurasi terhadap performa lalu lintas jaringan antar *node* masing-masing *router*. Masing-masing *node router* diuji untuk mengukur nilai *throughput* saat proses konfigurasi dijalankan. Parameter yang diamati mencakup besaran *throughput* (Kbps) yang diterima dan dikirimkan oleh masing-masing perangkat, serta kestabilan nilai *throughput* selama pengiriman konfigurasi berlangsung.

E. System Operation and Execution

Fokus utama pada tahapan ini dengan melakukan pengujian otomatisasi konfigurasi, pengecekan konfigurasi yang telah diimplementasikan, serta proses *backup* konfigurasi pada perangkat *router* vCHR dan RB750. Tahap ini bertujuan untuk menjalankan sistem secara langsung guna memastikan setiap fungsi bekerja sesuai dengan desain awal dan mendukung otomatisasi yang telah dirancang pada penelitian ini. Pada proses ini, sistem manajemen jaringan terpusat akan diuji dengan berbagai skenario untuk menilai efektivitas dan keandalannya. Setiap aspek terkait manajemen konfigurasi akan diperiksa secara menyeluruh guna memastikan semua parameter jaringan dapat dikontrol secara otomatis.

- Manajemen vCHR dan RB750 melalui administrasi Django - Administrator bertanggung jawab untuk mendaftarkan perangkat *router* vCHR dan RB750 melalui panel administrasi Django, yang dapat diakses menggunakan web *browser* melalui URL http://10.151.20.143:8000/admin. Melalui *interface* ini, administrator akan memasukkan berbagai informasi penting terkait perangkat, meliputi alamat IP, nama *host*, nama pengguna, kata sandi SSH, *port* SSH, nama *remote*, lokasi perangkat, dan identifikasi vendor MikroTik. Data ini digunakan sebagai acuan utama dalam proses komunikasi antara sistem manajemen jaringan dengan perangkat yang dikonfigurasi.
- Implementasi dan akses sistem manajemen jaringan terpusat - Sistem manajemen jaringan berbasis Django ini dijalankan sebagai *host* pada server Ubuntu, yang dikonfigurasi menggunakan IP *address* 10.151.20.143/25 dan berjalan pada *port* 8000. Interface aplikasi dibangun dengan integrasi antara *frontend* dan *backend*, menggunakan *Python*, HTML dan CSS untuk tampilan pengguna. Proses autentikasi perangkat *router* vCHR dan RB750 dilakukan saat administrator atau pengguna memilih perangkat yang akan dikonfigurasi. Paramiko berperan dalam membangun sesi SSH dan sesi protokol SCP antara sistem manajemen dan *router* vCHR dan RB750 yang dipilih.

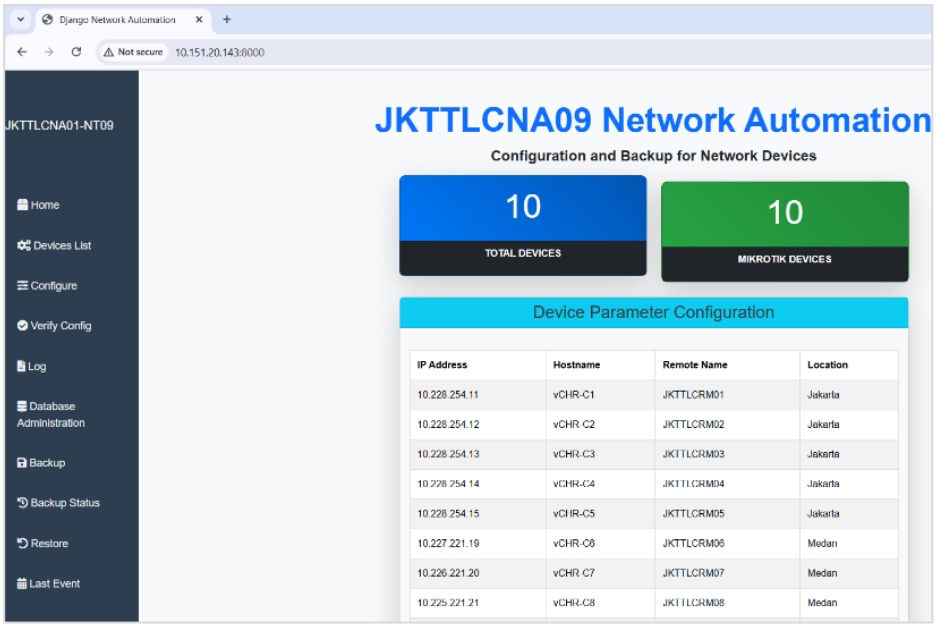
F. Optimization of Network Management System Performance and Efficiency

Pada fase optimasi, serangkaian langkah dilakukan untuk meningkatkan kinerja dan efisiensi sistem manajemen jaringan terpusat setelah fase implementasi selesai. Optimasi ini berfokus pada evaluasi menyeluruh terhadap stabilitas, dan efektivitas mekanisme otomasi yang diterapkan dalam manajemen perangkat vCHR dan RB750. Analisis kinerja dilakukan dengan menggunakan pendekatan sistematis melalui evaluasi sebagai berikut:

- Evaluasi Indikator Kinerja Utama (KPI). Pada fase awal ini, dilakukan pemantauan Indikator Kinerja Utama (KPI) yang meliputi parameter *throughput*, *uptime*, dan *downtime* saat sistem menangani konfigurasi simultan untuk 10 *node* perangkat vCHR dan RB750. Proses ini bertujuan untuk menilai sejauh mana sistem dapat mengelola beban kerja simultan tanpa mengalami penurunan kinerja yang signifikan.
- Pengujian efisiensi proses *backup* konfigurasi dan *time* status *backup*. Pengujian ini berfokus pada efektivitas mekanisme *backup* dan *time* status *backup* konfigurasi perangkat menggunakan protokol SCP untuk transfer data.

III. HASIL DAN PEMBAHASAN

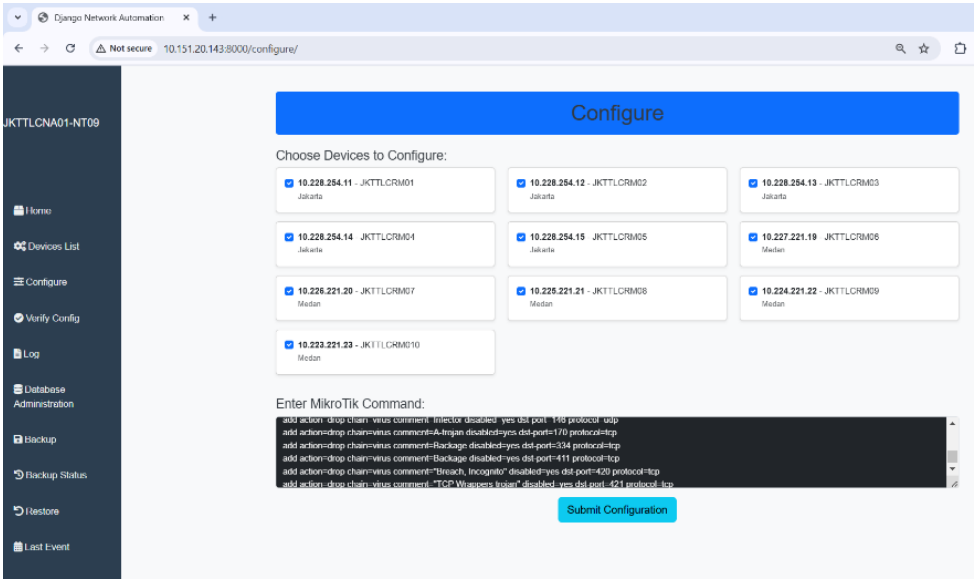
Bagian ini mengulas hasil pengujian implementasi aplikasi manajemen jaringan terpusat, untuk mendukung otomatisasi konfigurasi, dan *backup* konfigurasi untuk perangkat vCHR dan RB750. Gambar 6 menunjukkan antarmuka *dashboard* sistem aplikasi yang diimplementasikan.



Gambar 6. *Automation dashboard view* (JKTTLCNA09)

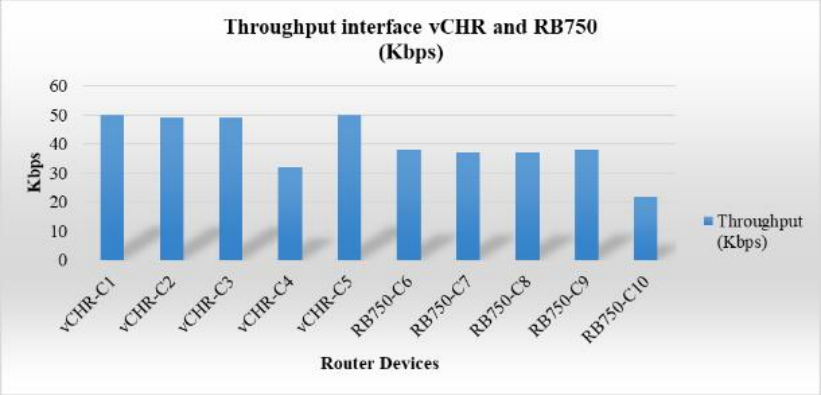
A. Evaluasi Konfigurasi Perangkat vCHR dan *Router* RB750 (*Throughput*)

Tahap pengujian sistem dilakukan dengan mengirim konfigurasi secara simultan ke 10 *node* perangkat vCHR dan RB750 untuk mengevaluasi efisiensi otomasi konfigurasi serta memastikan setiap perangkat menerima parameter yang sesuai. Proses ini digambarkan pada Gambar 7 melalui modul konfigurasi aplikasi (*configure*). Konfigurasi yang diterapkan meliputi pengaturan utama jaringan, seperti antarmuka *bridge*, manajemen *port bridge*, *trunk* VLAN-ID dan akses, alamat IP, serta *firewall* NAT, filter, dan *mangle*, dengan total 496 baris skrip. Seluruh parameter dimasukkan oleh administrator jaringan melalui antarmuka aplikasi.



Gambar 7. Konfigurasi 10 *Nodes* vCHR dan RB750 *Devices*

Pemantauan konfigurasi pada 10 *node router* vCHR dan RB750 menggunakan Zabbix sebagai *platform monitoring* jaringan, yang menyajikan visualisasi kinerja jaringan secara *real-time* sehingga memudahkan analisis dan evaluasi terhadap implementasi yang telah dilakukan. Pemantauan ini memungkinkan administrator untuk memantau status operasional setiap perangkat dan memastikan bahwa konfigurasi yang dikirim dapat diterapkan secara optimal tanpa mengalami kegagalan atau latensi yang signifikan. Pemantauan dilakukan untuk mengukur nilai *throughput* pada antarmuka setiap perangkat *router* vCHR dan RB750 selama proses pengiriman konfigurasi. Hasil pengukuran tersebut dapat dilihat pada Gambar 8.

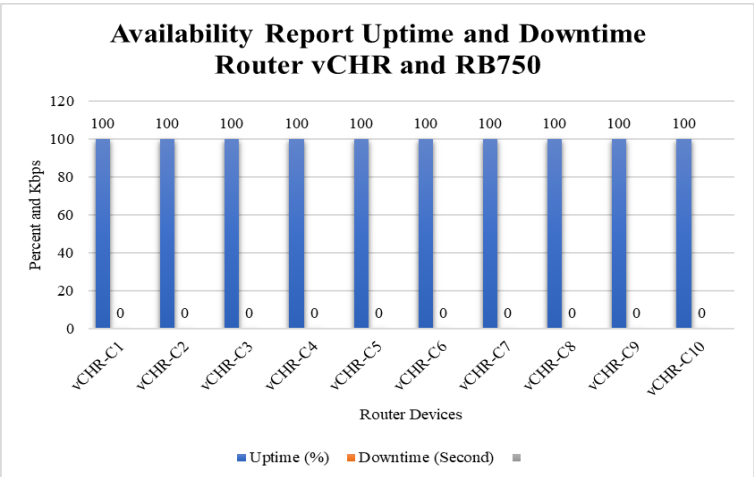


Gambar 8. Configuration of 10 *Node*s of vCHR and RB750 MikroTik Devices

Selama periode pengujian dari pukul 15:00:00 hingga 16:00:59 pada tanggal 23 Februari 2025, *throughput* setiap *router* vCHR-C1 adalah 50 Kbps, vCHR-C2 49 Kbps, vCHR-C3 49 Kbps, vCHR-C4 32 Kbps, vCHR-C5 50 Kbps dan RB750-C6 38 Kbps, RB750-C7 37 Kbps, RB750-C8 37 Kbps, RB750-C9 38 Kbps, RB750-C10 22 Kbps. Data ini merupakan salah satu indikator utama dalam mengevaluasi kinerja *router* secara keseluruhan.

B. Evaluasi Kinerja Perangkat *Router* Berdasarkan *Uptime* dan *Downtime*

Pada proses pengiriman konfigurasi, dilakukan analisis *uptime* dan *downtime* dengan mempertimbangkan tiga parameter utama, yaitu ICMP *Loss*, ICMP *Response Time*, dan *Unavailable* ICMP Ping, menggunakan Zabbix. Evaluasi ini bertujuan untuk mengukur kestabilan dan performa perangkat *router* pada infrastruktur jaringan dalam menangani pengiriman konfigurasi ke perangkat *router* vCHR dan RB750. Hasil pengujian menunjukkan bahwa perangkat *router* beroperasi sangat optimal, dengan rata-rata *uptime* mencapai 100%, yang menunjukkan bahwa sistem berjalan tanpa gangguan selama proses berlangsung. Pengujian menunjukkan bahwa selama proses berlangsung tidak ditemukan gangguan jaringan maupun *downtime* yang dapat memengaruhi komunikasi antar perangkat. Evaluasi ini menegaskan bahwa *node router* vCHR dan RB750 beroperasi secara normal, memiliki waktu *respons* yang optimal, serta mempertahankan kestabilan jaringan sepanjang proses konfigurasi. Gambar 9 menampilkan hasil pengukuran tersebut.

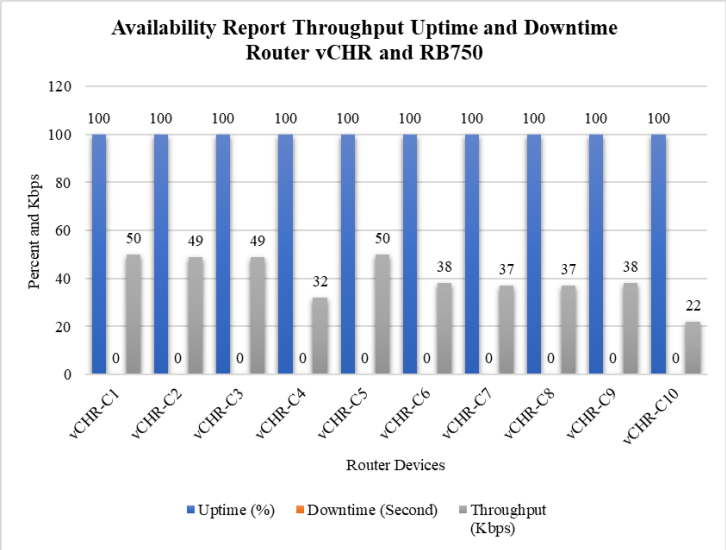


Gambar 9. *Uptime* and *Downtime* on vCHR dan RB750 *Router Device Nodes*

Salah satu parameter yang diuji adalah ICMP Ping *Loss* yang mendapatkan hasil *uptime* 100%, yang berarti tidak terjadi *packet loss* (0% *loss*) saat proses pengiriman konfigurasi ke semua perangkat *router*. Indikator ini menunjukkan bahwa setiap paket data yang dikirim pada pengujian berhasil sampai ke perangkat *router*. Parameter selanjutnya adalah ICMP *Response Time* yang menunjukkan *uptime* 100%, artinya waktu respon ping masih dalam ambang batas yang ditentukan. Hasil ini memastikan bahwa perangkat *router* vCHR dan RB750 mampu merespon setiap *request* ICMP dengan latensi normal dan tidak mengalami peningkatan waktu respon yang berlebihan. Kestabilan ini sangat penting untuk memastikan bahwa konfigurasi yang dikirimkan dapat diaplikasikan secara *real-time* tanpa *delay* yang berarti. Selain itu, parameter *unavailable* ICMP Ping juga mencatat status *uptime* 100%, yang berarti bahwa semua perangkat *router* vCHR dan RB750 tersedia dan dapat diakses tanpa adanya kegagalan konektivitas ICMP.

C. Evaluasi *Throughput* Antar *Node Router* vCHR dan RB750

Hasil pengujian yang ditunjukkan pada Gambar 10, menggambarkan perbandingan *throughput* setiap *node router* vCHR dan RB750 dalam jaringan. Berdasarkan data yang diperoleh, *node* dengan *throughput* terendah mencapai 22 Kbps, sedangkan *throughput* tertinggi tercatat sebesar 50 Kbps. Selisih *throughput* antar *node* tergolong kecil, yang menunjukkan bahwa kinerja perangkat dalam jaringan berada dalam kondisi normal dan optimal tanpa adanya lonjakan atau penurunan yang signifikan selama pengujian.



Gambar 10. *Availability Report Throughput*, *Uptime* dan *Downtime*

Nilai *throughput* yang seragam di antara *node* yang diuji juga mencerminkan bahwa proses distribusi konfigurasi berjalan secara merata dan efisien. Selain itu, hasil pengujian juga menunjukkan bahwa semua *node* memiliki *uptime* sebesar 100%, artinya selama proses pengiriman konfigurasi tidak terjadi gangguan koneksi atau kegagalan sistem. Hal ini membuktikan bahwa setiap perangkat *router* vCHR dan RB750 tetap aktif dan responsif selama pengujian. Tanpa *downtime*, semua konfigurasi dapat diterapkan secara efisien dan tanpa gangguan dalam jaringan. Secara keseluruhan, data yang diperoleh menunjukkan bahwa pengiriman konfigurasi pada perangkat *router* vCHR dan RB750 berjalan dengan baik, tanpa indikasi penurunan kinerja atau gangguan teknis yang dapat memengaruhi stabilitas sistem. Dengan *throughput* yang relatif seragam dan uptime yang maksimal, jaringan ini terbukti mampu mengakomodasi pengiriman konfigurasi dengan tingkat keandalan dan kinerja yang sangat baik.

D. Evaluasi Pemulihan Konfigurasi *Router* vCHR dan RB750

Konfigurasi *router* dan pemulihan cadangan secara bersamaan pada 10 *node router* protokol penyalinan aman (SCP) vCHR dan RB750. Langkah pertama adalah mengakses fitur cadangan. Proses ini dimulai dengan sistem yang menginisialisasi koneksi individual ke setiap perangkat menggunakan protokol aman SCP. Koneksi ICMP harus berhasil dibuat sebelum sistem dapat melanjutkan ke fase berikutnya. Setelah koneksi dibuat secara normal, sistem akan mulai membuat *file* dan mengunduh *file* konfigurasi dari setiap perangkat, yang disimpan dalam format *.rsc. Waktu yang diperlukan untuk menyelesaikan proses *backup* dihitung menggunakan rumus protokol aman SCP:

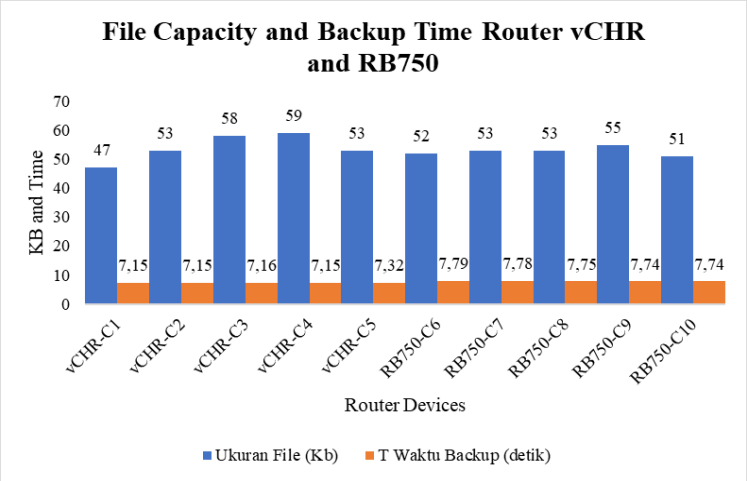
p-ISSN : 2443-2210
e-ISSN : 2443-2229

$$T = \sum_{i=1}^{N}(T_{ICMP(i)} + T_{SSH(i)} + T_{Backup(i)} + T_{Download(i)}) \quad (1)$$

Dimana:

- $T_{ICMP\ (i)}$: waktu untuk verifikasi koneksi ICMP ke *router* ke-i
- $T_{SSH(i)}$: waktu untuk inisialisasi koneksi SSH ke *router* ke-i
- $T_{Backup(i)}$: waktu untuk eksekusi perintah *backup router* ke-i
- $T_{Download\ (i)}$: proses waktu *download file* *.rsc *backup router* ke-i ke Ubuntu server SCP.

Total waktu *T* yang dibutuhkan untuk menyelesaikan proses *backup* konfigurasi pada *N router* dihitung dari akumulasi komponen waktu utama untuk setiap perangkat *router*. Dimulai dari proses verifikasi konektivitas melalui pengiriman paket ICMP, kemudian dilanjutkan dengan inisialisasi koneksi menggunakan *protocol* SSH. Setelah koneksi berhasil dibuat, sistem mengeksekusi perintah *backup* konfigurasi pada *router* untuk menghasilkan *file* konfigurasi. *File* tersebut kemudian di *download* melalui protokol SCP, dan disimpan ke dalam direktori *backup* pada Ubuntu server. Perbedaan ukuran *file backup* pada tiap perangkat (47–59 KB) menyebabkan variasi waktu *download*, tergantung kompleksitas konfigurasi dan kecepatan koneksi, seperti terlihat pada Gambar 11.



Gambar 11. *File Capacity* dan *Backup Time Router* vCHR dan RB750

Perbedaan ukuran *file backup* untuk masing-masing *router* disebabkan oleh variasi konfigurasi yang diterapkan pada masing-masing *node router*. Berdasarkan hasil pengujian, *file backup* yang dihasilkan memiliki ukuran antara 47 KB hingga 59 KB, yang menunjukkan adanya perbedaan jumlah entri konfigurasi dan jenis pengaturan jaringan yang diterapkan. *File backup* ini berisi pengaturan-pengaturan penting, seperti konfigurasi dasar sistem operasi *router*, pengaturan *interface* jaringan, pengalokasian alamat IP, aturan *firewall* NAT dan filter, *routing* statis, serta konfigurasi *interface* dan *port bridge*. Keseluruhan konfigurasi yang diterapkan tersusun dalam 496 baris skrip per perangkat, mencerminkan tingkat kompleksitas pengaturan yang diterapkan pada *router* vCHR dan RB750. Semakin kompleks konfigurasi pada perangkat, maka ukuran *file backup* dan waktu proses *backup* cenderung meningkat secara proporsional.

E. Pembahasan Komparatif terhadap Pendekatan Otomasi Jaringan Lain

Sebagai upaya memperkuat analisis terhadap sistem yang dikembangkan, diperlukan tinjauan komparatif dengan metode otomasi jaringan lain yang memiliki relevansi tinggi. Sistem dalam penelitian ini menggabungkan *library* Paramiko dengan Django *Framework* untuk membangun *platform* manajemen jaringan terpusat. Pendekatan yang digunakan bersifat imperatif, dengan memanfaatkan protokol SSH untuk mengirimkan perintah konfigurasi langsung ke perangkat *router* MikroTik, serta menggunakan protokol SCP untuk proses pencadangan (*backup*) konfigurasi. Meski efektif, metode ini bukan satu-satunya yang dapat diterapkan. Beberapa alternatif populer di lingkungan otomasi jaringan meliputi Ansible dan Netmiko. Ansible mengusung pendekatan deklaratif berbasis *state*, yang memanfaatkan bahasa YAML untuk mendefinisikan konfigurasi [18], [20], [22], [23]. Netmiko, di sisi lain, merupakan *wrapper* tingkat tinggi untuk SSH yang memudahkan pengiriman perintah CLI secara terprogram. Tabel 2 memperlihatkan detail perbandingan metode tersebut.

TABEL 2
PERBANDINGAN KONSEPTUAL KETIGA PENDEKATAN BERDASARKAN LITERATUR TERKINI

Aspek	**Paramiko + Django (Penelitian ini)**	**Ansible**	**Netmiko**
Pendekatan	Imperatif, eksekusi skrip SSH langsung	Deklaratif, berbasis *state*	Imperatif, *wrapper high-level* untuk SSH
Bahasa/ *Framework*	*Python* murni + Django	YAML + *Python*	*Python*
Antarmuka	Antarmuka *web Custom* berbasis Django	CLI / GUI (Ansible Tower / AWX)	Skrip CLI *Python*
Skalabilitas	Terbatas pada jumlah perangkat tertentu	Tinggi, cocok untuk ratusan perangkat	Sedang, cocok untuk otomasi per perangkat
Error Handling	Manual melalui *try-except*	Built-in, otomatis menangani kegagalan	*Built-in*
Kompatibilitas Vendor	Umum, selama mendukung SSH	Mendukung banyak vendor (multi-vendor)	Mendukung berbagai vendor besar (Cisco, MikroTik, Juniper, dan lainnya)
Keunggulan	Fleksibel, integrasi penuh ke aplikasi web *Python*	Efisien untuk orkestrasi konfigurasi masal	Implementasi sederhana, cepat digunakan
Kelemahan	Membutuhkan pengaturan manual per perangkat	Membutuhkan struktur inventori dan pengetahuan YAML	Kurang fleksibel untuk integrasi GUI *custom*

Berdasarkan perbandingan tersebut, pendekatan Paramiko dan Django unggul dalam fleksibilitas integrasi antarmuka dan kontrol penuh terhadap proses konfigurasi, sehingga ideal untuk lingkungan dengan jumlah perangkat terbatas atau skenario pengujian. Ansible memiliki kekuatan pada kemampuan orkestrasi skala besar yang terstruktur, namun memerlukan *learning curve* yang lebih tinggi serta ketergantungan pada ekosistemnya (misalnya Ansible Tower). Netmiko menawarkan kemudahan implementasi dengan abstraksi perintah CLI, namun kurang fleksibel jika diperlukan antarmuka kustom yang kompleks. Ke depannya, pengembangan dapat diarahkan pada pengujian komparatif kinerja antara Paramiko, Netmiko, dan Ansible dari sisi waktu eksekusi, tingkat keberhasilan penerapan konfigurasi, serta kemudahan *deployment* di berbagai skala jaringan.

IV. SIMPULAN

Hasil dari penelitian ini berupa aplikasi manajemen jaringan terpusat yang mampu mengotomatiskan konfigurasi dan *backup* perangkat *router* vCHR dan RB750 secara simultan. Berdasarkan hasil pengujian yang telah dilakukan, diperoleh beberapa kesimpulan yang menggambarkan kinerja dan efektivitas sistem yang dikembangkan. Sistem ini mampu mendistribusikan 496 baris skrip konfigurasi secara otomatis ke 10 *node router* vCHR dan RB750 dengan hasil konfigurasi yang diterapkan secara konsisten tanpa memerlukan intervensi manual. Proses konfigurasi tersebut meliputi pengaturan *interface bridge*, VLAN *access* dan trunk, IP *addressing*, serta konfigurasi *firewall* NAT, *filter*, dan *mangle*. Selama proses konfigurasi berlangsung, melalui *monitoring* Zabbix menunjukkan seluruh *router* tetap stabil dengan *uptime* 100%, tanpa terjadi *packet loss*, peningkatan *response time*, maupun *downtime*. Selain itu, eksekusi *backup file* konfigurasi *router* berjalan menggunakan protokol SCP dengan waktu *backup* per perangkat berkisar antara 7,15 hingga 7,79 detik. *File backup* yang dihasilkan memiliki ukuran bervariasi antara 47 hingga 59 KB, tergantung pada kompleksitas konfigurasi *router*, dan seluruh *file backup* berhasil tersimpan dengan baik di Ubuntu server Django tanpa terjadi kegagalan transfer. Hasil pengujian *throughput* juga menunjukkan distribusi lalu lintas data yang stabil antar *node router*, dengan rentang *throughput* antara 22 Kbps hingga 50 Kbps. Selisih *throughput* antar *node* tergolong kecil, yang menunjukkan bahwa proses konfigurasi dan *backup* tidak menyebabkan beban berlebih pada jaringan. Namun, penelitian ini masih memiliki keterbatasan yang perlu dikembangkan lebih lanjut, mengingat cakupan konfigurasi yang diuji masih terbatas pada konfigurasi dasar jaringan dan belum mencakup konfigurasi layanan jaringan yang lebih kompleks atau interkoneksi multivendor. Selain itu, pengujian masih dibatasi pada 10 perangkat *router*, sehingga skalabilitas sistem untuk jumlah perangkat yang lebih besar belum divalidasi secara menyeluruh. Secara keseluruhan, aplikasi otomasi berbasis Django *Framework*, *library* Paramiko, dan protokol SCP yang dikembangkan dalam penelitian ini terbukti efektif dalam mempercepat proses konfigurasi dan *backup router* MikroTik secara terpusat, sekaligus menjaga kestabilan jaringan selama proses berlangsung. Dengan pengembangan lebih lanjut, sistem ini berpotensi diimplementasikan pada lingkungan jaringan yang lebih besar dan heterogen.

DAFTAR PUSTAKA

[1] H. A. Damanik, M. Anggraeni, and F. A. A. Nusantari, *Konsep dan Penerapan Switching dan Routing Implementasi Jaringan Komputer Berbasis Cisco*. Mega Press Nusantara, 2023.

[2] H. A. Damanik and M. Anggraeni, "Implementation of Backoff Algorithm Bidirectional Forwarding Detection (BFD) and MPLS VRF for Fast Recovery Mechanism End-to-End Multi-Circuit (E2E)," *Jurnal Penelitian Pos dan Informatika*, vol. 11, no. 2, pp. 121–136, 2021.

[3] A. A. Mazin, H. Z. Abidin, L. Mazalan, and A. M. Mazin, "Network Automation Using Python Programming to Interact with Multiple Third-Party Network Devices," in *2023 10th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 2023, pp. 59–64.
[4] D. Chen, D. Gao, W. Quan, Q. Wang, G. Liu, and H. Zhang, "Promoting Network Automation for Heterogeneous Networks Collaboration," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020, pp. 1–5.
[5] S. A. Piskunov, A. V Mokeev, A. I. Popov, D. N. Ulyanov, and A. V Rodionov, "Technical solutions for automation of distribution networks based on SPM technology," in *2022 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*, 2022, pp. 1–6.
[6] M. Sadeghian-Jahromi, M. Lehtonen, M. Fotuhi-Firuzabad, S. Fattaheian-Dehkordi, and S. Heidary, "Investigating the Impacts of Resiliency of the Automation System on Distribution Network Resilience," in *2023 IEEE PES GTD International Conference and Exposition (GTD)*, 2023, pp. 6–10.
[7] H. A. Damanik, M. Anggraeni, and F. A. A. Nusantari, *Network Automation Pada Mikrotik RouterOS*. Deepublish, 2024.
[8] G. Song, Y. Zhou, W. Zhang, and A. Song, "A multi-interface gateway architecture for home automation networks," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1110–1113, 2008.
[9] A. Agarkar, S. Shewale, B. Shah, A. Shikalkar, and O. Sangale, "Multi-Server Management System Using Flask, SSH and Paramiko," in *2025 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, 2025, pp. 1–5.
[10] A. M. Mazin, R. A. Rahman, M. Kassim, and A. R. Mahmud, "Performance Analysis on Network Automation Interaction with Network Devices Using Python," in *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2021, pp. 360–366.
[11] Y. Alfaresa, B. Arifwidodo, and F. Khair, "Automate IGP and EGP Routing Protocol Configuration using a Network Automation Library," *Jurnal Online Informatika*, vol. 8, no. 2, pp. 222–231, 2023.
[12] M. Sharma, M. S. Khan, and J. Singh, "Python & Django the Fastest Growing Web Development Technology," in *2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)*, 2024, pp. 1–9.
[13] D. Martinez, D. Gonzalez, J. M. Sánchez, and R. M. Toasa, "Library in django framework to standardize early-stage web application development," in *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, 2023, pp. 1–4.
[14] P. V, V. P, R. K. K, S. P, O. Khan, and Ch. N. Krishna, "A Django Web Application to Promote Local Service Providers," in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, 2022, pp. 1517–1521.
[15] P. Thakur and P. Jadon, "Django: Developing web using Python," in *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2023, pp. 303–306.
[16] C.-G. Dumitrache, C. A. Ghita, G. Predusca, L. D. Circiumarescu, N. Angelescu, and D. C. Puchianu, "The Efficiency of Routing Protocols, OSPF and EIGRP, Using Graphical Network Simulator-3 and Python," in *2024 28th International Conference on System Theory, Control and Computing (ICSTCC)*, 2024, pp. 194–199.
[17] A. Datta, A. T. M. A. Imran, and C. Biswas, "Network Automation: Enhancing Operational Efficiency Across the Network Environment," *ICRRD Journal*, vol. 4, no. 1, pp. 101–111, 2023.
[18] G. S. Santyadiputra, I. M. E. Listartha, and G. A. J. Saskara, "The effectiveness of Automatic Network Administration (ANA) in network automation simulation at Universitas Pendidikan Ganesha," *J Phys Conf Ser*, vol. 1810, no. 012028, 2021.
[19] A. Efendi, D. Husna, and I. G. D. Nugraha, "Advancing Network Infrastructure: Integrating VXLAN Technology with Automated Circuit Operations and NOS Configurations," *International Journal of Electrical, Computer, and Biomedical Engineering*, vol. 1, no. 2, pp. 103–124, 2023.
[20] A. Zulfikar and Y. Akbar, "Otomasi Backup Konfigurasi Settingan Router Mikrotik Menggunakan Ansible dengan Metode Network DevOps," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 5, no. 1, pp. 57–66, 2024.
[21] Z. Li, B. Zhou, X. Xiong, and X. Xiong, "Design of a Highly Concurrent Plug-in System for Network Automation," in *2023 10th International Forum on Electrical Engineering and Automation (IFEEA)*, 2023, pp. 788–793.
[22] A.-F. Sicoe, R. Botez, I.-A. Ivanciu, and V. Dobrota, "Fully Automated Testbed of Cisco Virtual Routers in Cloud Based Environments," in *2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2022, pp. 49–53.
[23] A. N. S. Aquino and A. R. Villanueva, "Network Anomaly Detection Using NetFlow and Network Automation," in *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*, 2023, pp. 1–6.