Journal of Vocational Education and Information Technology
Vol. 7, No. 1, Juni 2026, 149 - 162
E-ISSN: 2722-5305
Terakreditasi **SINTA Peringkat 4**
Surat Keputusan Direktur Jenderal Pendidikan Tinggi, Riset, dan Teknologi, Nomor: 230/E/KPT/2024

Otomatisasi Manajemen QoS Jaringan Berbasis Intent-Based Networking Menggunakan LLM Qwen2.5-Coder-7B dan Orkestrasi n8n

Muhammad Rafi Muhtaddin Noor [1,*], Achmad Junaidi [2], Eva Yulia Puspaningrum [3]

[1,2,3] Informatika, Universitas Pembangunan Nasional "Veteran" Jawa Timur:
22081010201@student.upnjatim.ac.id[1] achmadjunaidi.if@upnjatim.ac.id[2],
evapuspaningrum.if@upnjatim.ac.id[3].

* Korespondensi: e-mail: 22081010201@student.upnjatim.ac.id,

Diterima: 26-06-2026; Review: 11-07-2026; Disetujui:15-07-2026

Abstrak: Manajemen *Quality of Service* (QoS) menggunakan *Simple Queue* pada MikroTik sering kali menghadapi kendala inefisiensi alokasi akibat karakteristik konfigurasi yang manual, statis, dan reaktif. Penelitian ini mengusulkan arsitektur otomatisasi tertutup (*closed-loop system*) berbasis *Intent-Based Networking* (IBN) dengan mengintegrasikan *Large Language Model* (LLM) Qwen2.5-Coder-7B dan *engine* orkestrasi n8n. Secara prosedural, alur kerja sistem meliputi fase *Pre-Execution Monitoring* via SSH *speedtest*, pembersihan derau (*denoising*) teks dengan *structured prompt engineering*, serta injeksi konfigurasi otomatis melalui REST API MikroTik RouterOS v7. Hasil pengujian eksperimental terhadap 10 skenario (*test cases*) menunjukkan bahwa komponen kognitif model berhasil mencapai tingkat *Exact Match Rate* 100% dalam mentranslasikan perintah bahasa alami menjadi objek JSON deterministik. Logika *state-aware* pada n8n terbukti efektif berfungsi sebagai *safety valve* dengan memotong alokasi *bandwidth child queue* secara otonom hingga batas aman maksimal 70% dari kapasitas aktual ISP guna mencegah *bufferbloat*. Selain itu, sistem berhasil mengisolasi eksekusi perintah HTTP PATCH untuk meminimalkan *overhead control-plane* router MikroTik hAP Lite. Pengukuran latensi operasional akhir-ke-akhir (*end-to-end latency*) sistem berkisar antara 27,66 detik hingga 43,98 detik, yang secara dominan dipengaruhi oleh waktu tunggu *sampling throughput* riil pada jalur fisik ISP lokal. Integrasi ini berhasil mewujudkan otomatisasi manajemen kualitas layanan jaringan SOHO yang adaptif, presisi, dan aman dari risiko kesalahan konfigurasi manusia.

Kata kunci: *Intent-Based Networking*, Qwen2.5-Coder-7B, Orkestrasi n8n, *Simple Queue*, *Closed-Loop System*

Abstract: *Quality of Service (QoS) management using Simple Queue on MikroTik routers often faces allocation inefficiencies due to manual, static, and reactive configurations. This study proposes a closed-loop autonomous automation architecture based on Intent-Based Networking (IBN) by integrating the Qwen2.5-Coder-7B Large Language Model (LLM) and the n8n orchestration engine. Procedurally, the system workflow encompasses a pre-execution monitoring phase via SSH speedtest, text denoising using structured prompt engineering, and automated configuration injection through the MikroTik RouterOS v7 REST API. Experimental results across 10 test cases demonstrated that the model's cognitive component achieved a 100% Exact Match Rate in translating unstructured natural language commands into deterministic JSON objects. The state-aware logic within n8n effectively acted as a network safety valve by autonomously capping the child bandwidth allocation to a maximum threshold of 70% of the actual ISP capacity to prevent bufferbloat. Furthermore, the system successfully isolated command executions using HTTP PATCH to minimize the control-plane overhead on the resource-constrained MikroTik hAP Lite router. The measured end-to-end operational latency ranged from 27.66 seconds to 43.98 seconds, which was predominantly driven by the active probing sampling time on the physical ISP link. This integration successfully delivered an adaptive, precise, and*

secure autonomous network quality management system for SOHO environments while eliminating human configuration errors.

***Keywords**: Intent-Based Networking, Qwen2.5-Coder-7B, n8n Orchestration, Simple Queue, Closed-Loop System*

1. Pendahuluan

Manajemen *Quality of Service* (QoS) merupakan mekanisme krusial dalam infrastruktur telekomunikasi modern untuk mengendalikan aliran lalu lintas data dan mencegah dominasi konsumsi *bandwidth* oleh satu pengguna yang dapat merusak kualitas layanan secara keseluruhan [1]. Dalam implementasi praktis pada perangkat keras jaringan seperti MikroTik, metode *Simple Queue* sering digunakan sebagai pendekatan fundamental untuk membatasi serta mengalokasikan kapasitas kecepatan transmisi data secara hierarkis [2]. Pendekatan struktural ini sangat esensial untuk menjamin keandalan dan stabilitas konektivitas, terutama dalam menghadapi lonjakan permintaan operasional akses informasi secara bersamaan pada waktu sibuk [1]. Namun, titik lemah operasional dari praktik ini bersumber pada kecenderungan administrator jaringan yang sering kali mengonfigurasi batasan *bandwidth* secara buta, manual, dan sangat reaktif [3]. Administrator biasanya menetapkan parameter QoS statis tanpa dibekali dengan visibilitas komprehensif terhadap kondisi beban kinerja serta fluktuasi metrik lalu lintas aktual yang terus berubah secara *real-time* [4]. Ketiadaan pemantauan waktu nyata ini memicu inefisiensi alokasi sumber daya secara signifikan, di mana sistem menjadi kaku dalam merespons anomali lalu lintas, sehingga memperbesar probabilitas terjadinya *bottleneck* maupun kegagalan penjaminan layanan pada sistem yang sedang aktif [4].

Untuk mengeliminasi intervensi manual yang tidak efisien tersebut, paradigma *Intent-Based Networking* (IBN) diusulkan sebagai arsitektur otonom yang memungkinkan administrator mengelola fungsionalitas jaringan hanya melalui injeksi satu instruksi bahasa alami tunggal (*single intent*) [5]. Model IBN memisahkan kompleksitas teknis di lapisan infrastruktur bawah dari kebutuhan bisnis, sehingga elemen jaringan dapat merumuskan, menerjemahkan, dan mengonfigurasi kebijakan operasionalnya sendiri berdasarkan perintah yang bersifat deklaratif [6]. Walaupun menjanjikan otomatisasi tingkat tinggi, tantangan paling kritis dari arsitektur ini terletak pada sifat bawaan teks bahasa alami yang sangat tidak terstruktur, bermakna ganda (ambigu), dan rentan terhadap derau leksikal (*noise*) [7]. Penggunaan data teks yang berderau ini menuntut adanya algoritma prapemrosesan yang memiliki tingkat presisi sangat tinggi agar seluruh parameter dalam representasi tekstual tersebut dapat dibersihkan dan dimaknai dengan akurat tanpa mendistorsi instruksi aslinya [8]. Apabila derau dan ambiguitas leksikal ini tidak disaring melalui mekanisme validasi yang ketat, ekstraksi entitas dari *intent* akan menghasilkan kegagalan pemetaan parameter secara semantik [9]. Eksekusi langsung terhadap instruksi kotor yang belum tervalidasi ke dalam *engine* orkestrasi kontrol jaringan tidak hanya akan menggagalkan pemenuhan target QoS, melainkan membawa risiko fatal berupa kesalahan injeksi konfigurasi yang berpotensi melumpuhkan dan merusak fungsionalitas seluruh topologi system [5].

Sebagai solusi inovatif untuk merestrukturisasi permasalahan operasional tersebut, penelitian ini mengusulkan integrasi model bahasa besar (LLM) Qwen2.5-Coder-7B yang sudah di *fine-tunning* menggunakan dataset perintah QoS untuk *simple queue* berjumlah 400 data lebih pada mikrotik dengan *engine* orkestrasi n8n guna membentuk arsitektur otomasi tertutup (*closed-loop*). Secara prosedural, alur kerja sistem ini diinisiasi dengan fase *pre-execution monitoring*, di mana orkestrator n8n secara otonom memicu pengujian bandwidth aktif (*active bandwidth probing*) melalui eksekusi speedtest via SSH ke Linux Server sesaat sebelum mengeksekusi aturan *Simple Queue*. Langkah pemantauan awal ini merupakan prasyarat esensial agar manajemen kualitas layanan bersifat adaptif dan secara riil merepresentasikan fluktuasi beban jaringan aktual, sehingga mencegah konfigurasi statis yang berujung pada inefisiensi alokasi *bandwidth* [5], [10]. Setelah kondisi jaringan aktual tervalidasi, orkestrator akan meneruskan instruksi bahasa alami pengguna beserta konteks metrik jaringan ke dalam LLM. Guna menangani ketidakberaturan teks bawaan pengguna, Qwen2.5-Coder diinstruksikan menggunakan metode *prompt* terstruktur yang difungsikan khusus untuk melakukan *denoising* secara presisi pada niat pengguna (*single intent*) [5], [10]. Implementasi *prompt* terstruktur ini memandu LLM untuk secara cerdas memfilter derau leksikal (*noise*), meminimalisasi ambiguitas informasi, dan mengonversi niat mentah menjadi parameter *Simple Queue* yang terstandarisasi.

Vol. 5, No. 2, Desember 2024, 149 – 162

Parameter bebas derau ini kemudian ditransmisikan kembali oleh n8n untuk dieksekusi secara langsung ke dalam infrastruktur jaringan, menjamin bahwa konfigurasi akhir bersifat deterministik dan presisi.

Untuk menjaga ketajaman analisis dan validitas evaluasi, penelitian ini menegaskan batasan masalah yang sangat ketat secara eksklusif. Pertama, lingkup operasional sistem secara absolut dibatasi pada skenario eksekusi instruksi tunggal (*single intent*). Kedua, domain eksekusi secara teknis murni difokuskan pada tahap pengecekan metrik trafik aktual jaringan yang dilanjutkan dengan implementasi limitasi *bandwidth* menggunakan *Simple Queue*. Ketiga, penelitian ini sama sekali tidak mencakup mekanisme resolusi konflik untuk skenario *multi-intent*, sehingga sistem tidak mengevaluasi benturan parameter dari beberapa instruksi yang masuk secara bersamaan [10]. Dengan batasan operasional tersebut, tujuan utama penelitian ini dikhususkan pada evaluasi tingkat akurasi translasi eksekusi deterministik serta pengukuran kecepatan respons (*latency*) sistem dalam mengelola siklus otomasi secara penuh. Pada akhirnya, sinergi antara teknik *denoising* berbasis *prompt* terstruktur untuk mereduksi derau teks dan mekanisme pemantauan pertimbangan beban jaringan aktual menjadi landasan filosofis utama dalam mewujudkan orkestrasi jaringan otonom yang andal dan presisi [5].

2. Metode Penelitian

Metode penelitian ini disusun secara sistematis melalui pendekatan eksperimental laboratorium untuk merancang, mengimplementasikan, dan mengevaluasi keandalan sistem manajemen kualitas layanan (QoS) jaringan berbasis *Intent-Based Networking* (IBN). Metodologi ini dikembangkan ke dalam struktur kerja beralur tertutup (*closed-loop lifecycle*) yang mengeliminasi ketergantungan pada antarmuka grafis tradisional demi mencapai efisiensi eksekusi yang optimal. Secara menyeluruh, tahapan pelaksanaan penelitian ini dibagi menjadi empat fase utama. Fase pertama berfokus pada pemetaan desain arsitektur integrasi sistem secara otonom. Fase kedua menerapkan mekanisme pemantauan awal (*pre-execution monitoring*) untuk memvalidasi kapabilitas kapasitas lalu lintas data aktual pada perangkat *router* sebelum parameter baru diterapkan [1]. Fase ketiga berfokus pada pemrosesan kognitif berupa ekstraksi perintah tunggal (*single intent*) dari bahasa alami melalui rekayasa *prompt* terstruktur guna mereduksi ambiguitas dan derau teks bebas (*denoising*) [8]. Metodologi ini diakhiri pada fase keempat yang mencakup pemetaan logika orkestrasi otomatisasi menggunakan n8n serta pengujian performa sistem secara menyeluruh berdasarkan parameter akurasi parser, latensi pemrosesan, dan stabilitas *throughput* jaringan [1]. Seluruh tahapan tersebut diuraikan secara mendalam pada sub-bab berikut.

a. Desain Arsitektur Sistem Tertutup (Closed-Loop System)

Penelitian ini menerapkan arsitektur sistem tertutup (*closed-loop*) yang mengintegrasikan komponen kognitif cerdas dan orkestrator otomatisasi untuk mengelola pembatasan *bandwidth* jaringan secara otonom. Arsitektur logis ini dirancang secara prosedural tanpa melibatkan antarmuka grafis pengguna (*Graphical User Interface* / GUI) bawaan yang kaku atau platform berbasis web internal guna meminimalkan *overhead* komputasi dan mempercepat waktu respons eksekusi instruksi.

Guna mencapai tingkat fleksibilitas pengaplikasian yang tinggi di lingkungan riil, pertukaran data pada sistem ini dibangun di atas protokol REST API yang diekspos oleh orkestrator n8n melalui simpul *Webhook*. Dalam tahapan eksperimen laboratorium ini, Postman digunakan sebagai klien penguji (*testing client*) untuk menginjeksikan *payload* teks bahasa alami (*intent*) pengguna menuju n8n. Sifat arsitektur yang berbasis API ini menjamin bahwa komponen Postman dapat disubstitusi secara dinamis dengan berbagai variasi antarmuka lain pada implementasi nyata, seperti aplikasi *mobile*, dasbor web, maupun bot pesan instan.

Alur komunikasi data berjalan secara siklikal melalui mekanisme interaksi antar-komponen yang terstruktur, dengan urutan prosedural sebagai berikut:

1. Postman ke n8n: Postman mengirimkan *HTTP POST Request* berisi teks *intent* pengguna ke *endpoint Webhook* n8n.

2. n8n ke SSH Speedtest: Sebelum meneruskan teks, n8n menginisiasi koneksi SSH ke Linux Server untuk mengukur kapasitas *bandwidth* aktual (fase *pre-execution monitoring*).
3. n8n ke LLM: n8n mengirimkan teks *intent* mentah beserta metrik kapasitas aktual ke *Large Language Model* (LLM) Qwen2.5-Coder-7B.
4. LLM ke n8n: LLM melakukan *denoising* dan ekstraksi parameter, lalu mengembalikan luaran berupa objek JSON terstruktur kepada n8n.
5. n8n ke Mikrotik: n8n memvalidasi aturan batas aman 70% dan menginjeksikan konfigurasi otomatis ke perangkat router MikroTik hAP Lite menggunakan protokol REST API melalui *endpoint* /queue/simple.

b. Mekanisme Pengecekan Trafik Jaringan (Pre-Execution Monitoring)

Sebelum instruksi bahasa alami diterjemahkan oleh komponen LLM, sistem menjalankan fase prapemrosesan kritis berupa penilaian kapasitas bandwidth aktual secara *real-time*. Berbeda dengan pemantauan pasif biasa, orkestrator n8n dikonfigurasi untuk membuka sesi komunikasi aman (SSH) menuju Linux Server yang berada di dalam jaringan lokal. Melalui koneksi tersebut, n8n mengeksekusi perintah CLI speedtest untuk mengukur *throughput* maksimum unduh (*download*) dan unggah (*upload*) yang tersedia pada jalur ISP saat itu juga.

Nilai kapasitas aktual tersebut dijadikan sebagai *Safety Valve* (katup pengaman) dalam logika keputusan n8n. Jika LLM mengekstrak permintaan batas bandwidth pengguna yang melebihi kapasitas hasil *speedtest*, n8n secara otomatis akan mengintersepsi nilai tersebut dan memangkasnya menjadi maksimal 70% dari kapasitas aktual. Batasan 70% ini diterapkan sebagai upaya preventif untuk menghindari fenomena *bufferbloat*, penumpukan paket data, serta menjaga stabilitas interkoneksi lokal agar router tidak mengalami *overload*.

c. Perancangan dan Penyetelan Parameter (Fine-Tuning) LLM

Guna mentranslasikan *intent* bahasa alami menjadi objek JSON *state-aware* secara deterministik, model basis Qwen2.5-Coder-7B-Instruct dioptimalkan melalui tahapan *Supervised Fine-Tuning* (SFT). Alur pengembangan komponen kognitif ini dibagi menjadi tiga tahapan utama: rekayasa korpus data latih, kompresi parameter arsitektural, dan konfigurasi lingkungan eksekusi ekperimental.

1. Rekayasa Korpus dan Transformasi ChatML

   Dataset latih dibangun secara sintetik menggunakan bantuan *teacher model* (Gemini 3.1 Pro) untuk memproduksi 500 entri pasangan instruksi-respons yang valid. Korpus data dirancang secara modular mencakup variasi *original intent* pengguna (rata-rata panjang 8,06 kata), injeksi konteks jaringan (*Retrieval-Augmented Generation* / RAG), serta target luaran berupa *Structured Outputs* JSON.

   Sebelum diumpankan ke dalam model, dilakukan eliminasi terhadap metadata administratif seperti nomor ID dan label skenario untuk optimalisasi efisiensi token. Dataset JSON standar kemudian ditransformasikan ke dalam format *Chat Markup Language* (ChatML) dan disimpan dalam ekstensi JSON Lines (.jsonl). Format ini secara ketat memisahkan peran instruksional sistem (*system role*), injeksi parameter masukan (*user role*), dan target respons teknis router (*assistant role*) dalam satu baris string sekuensial.
2. Kompresi Arsitektur dan Modul Target LoRA

   Untuk mengatasi keterbatasan sumber daya komputasi di lingkungan lokal SOHO, fase pelatihan menerapkan metode *Quantized Low-Rank Adaptation* (QLoRA). Model dimuat dalam presisi kuantisasi ekstrem 4-bit *NormalFloat* (NF4) dengan pembekuan (*freezing*) matriks bobot asli model dasar (W_0). Modifikasi

pembaruan gradien dialokasikan secara eksklusif pada adapter matriks berdimensi rendah (ΔW) yang disisipkan pada seluruh modul proyeksi linear di dalam blok *Multi-Head Self-Attention* dan *Feed-Forward Network* (FFN). Penentuan target modul latihan diuraikan pada Tabel 1.

Tabel 1. Peta Modul Target Adaptasi Linear QLoRA

No	Modul Target	Fungsi Logika Komputasi Teroptimasi
1	q_proj	Proyeksi linear matriks Query (Q) untuk ekstraksi konteks intents.
2	k_proj	Proyeksi linear matriks Key (K) sebagai representasi fitur token leksikal.
3	v_proj	Proyeksi linear matriks Value (V) sebagai penampung payload bobot kalimat.
4	o_proj	Integrasi balik hasil atensi gabungan (Q, K, V) menuju arsitektur utama.
5	gate_proj	Pintu logika non-linear aktivasi fitur di dalam struktur MLP.
6	up_proj	Ekspansi dimensi ruang vektor tersembunyi (up-scaling) pada FFN.
7	down_proj	Kompresi kembali dimensi matriks (down-scaling) agar ekuivalen dengan Transformer.

3. Parameterisasi Eksperimen dan Lingkungan Infrastruktur

Akselerasi waktu pelatihan dan minimasi jejak memori (*memory footprint*) diwujudkan melalui integrasi pustaka Unsloth, transformers, dan trl. Pelatihan dijalankan menggunakan klaster *cloud compute* terisolasi untuk mencapai titik konvergensi loss yang stabil di bawah nilai 0.1. Detail parameterisasi dan spesifikasi perangkat keras dirangkum secara komprehensif pada Tabel 2.

Tabel 2. Konstanta Hiperparameter SFT dan Lingkungan Komputasi

Parameter Operasional	Spesifikasi Nilai Eksperimental
Infrastruktur Perangkat Keras	1x NVIDIA L4 (24 GB VRAM, 9 vCPU, 50 GB RAM)
Sistem Operasi / Environment	Ubuntu 22.04 LTS, CUDA 12.4.1, PyTorch 2.4.0
LoRA Rank (r) / Alpha (α)	Rank = 16, Alpha = 16
LoRA Dropout / Bias	0.05 / None
Tipe Komputasi Gradien	Bfloat16 (bf16=True)
Algoritma Optimizer	Paged AdamW 32-bit
Tingkat Pembelajaran (Learning Rate)	2×10^{-4} $(2e-4)$
Batch Size / Gradient Accumulation	Per-device Batch = 2, Accumulation Steps = 4
Durasi Pelatihan	60 Max Steps (~0.96 Epochs)
Panjang Konteks (Sequence Length)	2.048 Token

Pasca-pelatihan, adapter disatukan secara permanen ke dalam model dasar dengan metode merged_16bit (load_in_4bit=False) untuk menghindari degradasi presisi token. Entitas model mandiri (*standalone*) beresolusi 16-bit ini disimpan yang selanjutnya di-deploy menggunakan *engine* vLLM dengan parameter suhu (*temperature*) rendah sebesar 0.1 guna menjamin inferensi yang deterministik dan bebas halusinasi.

d. Ekstraksi *Intent* Tunggal Berbasis Rekayasa *Prompt* (*Denoising*)

Teks bahasa alami yang dikirim oleh pengguna sering kali bersifat tidak terstruktur, ambigu, dan mengandung elemen sintaksis yang tidak relevan (*noisy text*). Untuk mengatasi kendala tersebut, penelitian ini memanfaatkan model kognitif Qwen2.5-Coder-7B yang dipandu menggunakan teknik rekayasa *prompt* (*prompt engineering*) terstruktur. Proses ini bertujuan untuk melakukan pembersihan derau (*denoising*) teks bebas secara ketat dan memetakan maksud pengguna ke dalam format data yang deterministik. Strategi instruksi terstruktur diimplementasikan untuk membatasi ruang pencarian model secara kaku, sehingga model hanya diarahkan untuk mengenali perintah tunggal (*single intent*) terkait manipulasi *Simple Queue*. Sistem sengaja dirancang untuk tidak memproses perintah majemuk (*multi-intent*) atau melakukan analisis konflik aturan secara internal guna mempertahankan kecepatan inferensi dan menyederhanakan alur kerja. Penerapan instruksi yang terstruktur secara sistematis ini terbukti efektif dalam mereduksi halusinasi model, meredam derau pada teks tidak terstruktur, dan menghasilkan klasifikasi entitas yang memiliki akurasi tinggi [8]. Luaran dari fase inferensi LLM ini berupa objek JSON minimalis yang memuat tiga variabel utama: Alamat IP Target (target), Batas Maksimal Unggah (max-limit-upload), dan Batas Maksimal Unduh (max-limit-download).

e. Orkestrasi Eksekusi dan Skenario Pengujian

Tahap akhir dari metode penelitian ini melibatkan perancangan logika orkestrasi otomatisasi pada platform n8n untuk memetakan objek JSON hasil inferensi menjadi instruksi konfigurasi konkret pada perangkat jaringan. Orkestrator n8n dikonfigurasi untuk menerima *payload* terstruktur dari LLM, melakukan validasi integritas terhadap skema tipe data, dan secara otonom menyusun serta mengirimkan skrip *HTTP Request* yang diarahkan pada *endpoint* REST API MikroTik /queue/simple. Proses penambahan atau pembaruan aturan pembatasan kecepatan (*rate limiting*) diselesaikan secara instan dalam koridor sistem otonom tertutup (*closed-loop*) tanpa melibatkan intervensi manual dari administrator jaringan maupun *overhead* dari antarmuka grafis.

Guna menjamin efisiensi komputasi pada perangkat keras dengan keterbatasan sumber daya seperti MikroTik hAP Lite, logika orkestrasi pada n8n dirancang menggunakan pendekatan struktur *Single Parent* global. Sistem mengenali *Parent Queue* utama dengan cara memindai antrean aktif pada router; aturan antrean yang tidak memiliki parameter induk (*parent-less queue*) secara otomatis diidentifikasi sebagai *Parent* global yang menaungi seluruh subnet 192.168.178.0/24.

Setiap kali terdapat permintaan baru, n8n akan memperbarui batas maksimal (*max-limit*) pada *Parent* global sesuai dengan hasil *speedtest* terbaru. Namun, untuk mencegah lonjakan CPU 100% pada router hAP Lite akibat proses *looping* massal, n8n melakukan isolasi eksekusi dengan hanya membuat atau memperbarui *Child Queue* milik pengguna yang mengirimkan instruksi saat itu saja, sedangkan aturan *child* milik perangkat lain sama sekali tidak dimodifikasi.

Untuk menguji keandalan sistem terhadap variasi input, batasan logika 70%, adaptivitas *parent*, serta fungsionalitas penanganan eror (*error handling*), dirancang matriks pengujian yang terdiri dari 10 skenario komprehensif (*test cases*) seperti yang disajikan pada Tabel 3.

Tabel 3. Matriks Skenario Pengujian Sistem (TC-01 s.d TC-10)

ID	Kategori Pengujian	Kondisi Awal Jaringan & Router	Perintah Pengguna	Ekspektasi Hasil Akhir
TC-01	Akurasi Denoising (Formal Input)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Belum Ada	"Batasi IP 192.168.178.5 dengan download 5Mbps dan upload 2Mbps."	1. Pembuatan Parent Queue (max-limit=5M/10M). 2. Pembuatan Child Queue IP 192.168.178.5 (max-limit=2M/5M) menginduk ke Parent.

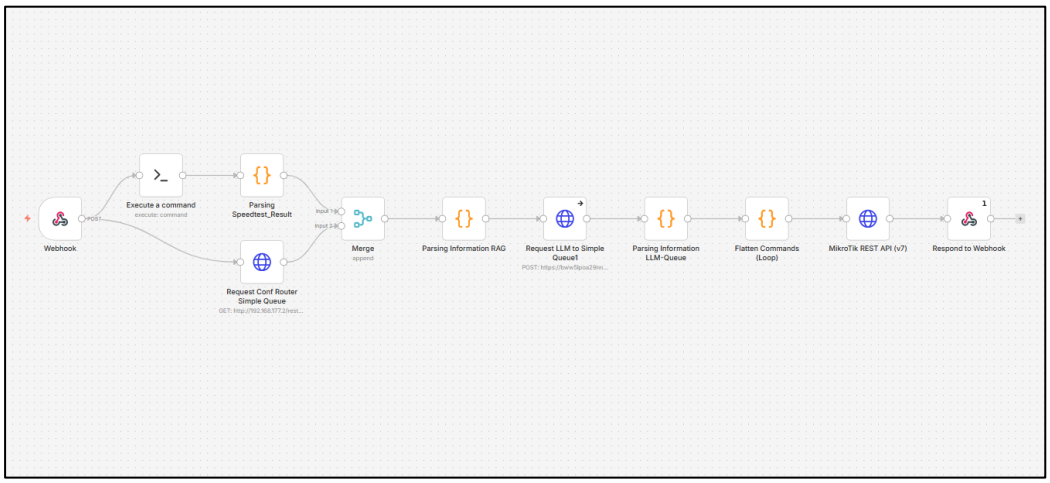
ID	Kategori Pengujian	Kondisi Awal Jaringan & Router	Perintah Pengguna	Ekspektasi Hasil Akhir
TC-02	Akurasi Denoising (Noisy/Slang Input)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Sudah Ada(5M/10M)	"min tolong settingin dungs buat ip 192.168.178.5 kasi download 4mbps sama uploadnya 2mbps ya"	1. Parent diperbaharui jika tidak sesuai dengan hasil speedtest. 2. Membuat Child Queue baru (max-limit=2M/4M) menginduk ke Parent global.
TC-03	Akurasi Denoising (Singkatan/Simbol)	Bandwidth: 6Mbps(Upload)/ 12Mbps(download) Parent Queue: Sudah Ada	"limit pc 192.168.178.10 d:5M u:2M"	1. Membuat *Child Queue* baru untuk IP 192.168.178.10 dengan max-limit=2M/5M.
TC-04	Akurasi Denoising (Code-Switching)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Sudah Ada	"Tolong apply rate limiting pada host 192.168.178.12, set upload 1Mbps and download 3Mbps."	1. Membuat *Child Queue* baru untuk IP 192.168.178.12 dengan max-limit=1M/3M.
TC-05	Logika n8n (Di Bawah Kapasitas)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Sudah Ada	"Beri bandwidth download 6Mbps dan upload 3Mbps untuk IP 192.168.178.20."	1. Meng-update *Parent* ke sesuai dengan hasil speedtest. 2. Membuat *Child* IP 192.168.178.20 dengan max-limit=3M/6M.
TC-06	Logika n8n (Proteksi Batas 70%)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Sudah Ada	"Setting IP 192.168.178.30 download 15Mbps, upload 8Mbps."	1. Meng-update *Parent* ke 10M/5M. 2. Memotong nilai *Child* secara otomatis menjadi aman: max-limit=3.5M/7M.
TC-07	Logika n8n (Dynamic Parent Update)	Bandwidth: 2Mbps(Upload)/ 4Mbps(download) Parent Queue: Konfigurasi lama 5M/10M	"Limit IP 192.168.178.40 dengan download 2M dan upload 1M."	1. Parent global langsung diturunkan menjadi max-limit=4M/2M. 2. *Child* IP 192.168.178.40 dibuat pada limit 1M/2M.
TC-08	Logika n8n (Isolasi Efisiensi CPU)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Sudah Ada Terdapat 5 *Child Queue* aktif lain di router	"Ubah limit IP 192.168.178.5 menjadi download 3M up 1M."	1. Hanya *Child* IP 192.168.178.5 yang berubah. 2. 5 Child Queue lainnya tidak tersentuh demi menghemat CPU hAP Lite.
TC-09	Penanganan Eror (Missing Parameter)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Sudah Ada	"Tolong dong limitin IP nya laptop saya di 192.168.178.150"	1. Sistem Menolak Permintaan. 2. Tidak ada perubahan konfigurasi di MikroTik (mencegah eror sintaksis router).
TC-10	Penanganan Eror (Invalid Subnet Target)	Bandwidth: 5Mbps(Upload)/ 10Mbps(download) Parent Queue: Sudah Ada	"Batasi bandwidth untuk IP 10.10.10.5 dengan download 5M dan upload 2M."	1. Sistem Menolak Permintaan. 2. Perintah ke endpoint /queue/simple dibatalkan demi keamanan topologi jaringan.

3. Hasil dan Pembahasan

Bagian ini menguraikan hasil pengujian eksperimental dari arsitektur otomatisasi *Intent-Based Networking* (IBN) yang diusulkan. Pengujian dievaluasi berdasarkan tiga metrik utama: (1) akurasi pembersihan derau (denoising) dan ekstraksi intent oleh model Qwen2.5-Coder-7B, (2) latensi operasional akhir-ke-akhir (*end-to-end latency*) dari orkestrasi n8n, serta (3) efektivitas mekanisme pemantauan prapemrosesan (pre-execution monitoring) terhadap stabilitas Quality of Service (QoS). Berbeda dengan implementasi sistem manajemen jaringan holistik yang melibatkan antarmuka grafis atau manajemen firewall berlapis, arsitektur pada penelitian ini secara ketat diisolasi pada lingkungan komputasi closed-loop tanpa intervensi visual, yang didedikasikan secara eksklusif untuk konfigurasi dinamis fitur Simple Queue pada router MikroTik.

a. Hasil Workflow n8n

Orkestrasi seluruh siklus manajemen jaringan berbasis intent dikelola secara penuh oleh workflow n8n melalui pemanfaatan node Webhook sebagai gerbang masuk (*entry point*) data dari Postman. Berdasarkan hasil eksperimen, n8n berhasil bertindak sebagai intermedian yang andal antara protokol REST API, komunikasi berbasis SSH, serta pemrosesan inferensi LLM. Workflow berjalan secara sekuensial dan linier, di mana setiap token otentikasi Bearer API untuk MikroTik dan *payload* JSON untuk LLM diparsing secara instan tanpa mengalami kegagalan *buffer timeout*. Integrasi ini membuktikan bahwa pendekatan *Interface-Agnostic* menggunakan Postman via REST API memberikan fleksibilitas tinggi bagi n8n untuk menerima instruksi dari platform eksternal mana pun di masa depan.



Gambar 1. Tampilan Akhir Workflow n8n

Implementasi sistem otonom tertutup (*closed-loop system*) pada platform orkestrasi n8n berhasil diwujudkan melalui integrasi 11 simpul (*nodes*) yang terhubung secara linier dan paralel, sebagaimana diilustrasikan pada Gambar 1. Siklus otomatisasi IBN ini diinisiasi secara responsif ketika simpul *Webhook* menerima *payload* teks *intent* dari klien Postman. Sesaat setelah instruksi diterima, alur kerja secara cerdas memecah pemrosesan menjadi dua cabang paralel guna membangun basis konteks *Retrieval-Augmented Generation* (RAG) secara *real-time*:

1. Cabang Atas (Active Probing): Mengeksekusi instrumen uji speedtest berbasis CLI pada Linux Server via SSH melalui simpul *Execute a command*, yang kemudian hasilnya diekstraksi menggunakan ekspresi reguler (*regex*) pada simpul *Parsing Speedtest_Result*.
2. Cabang Bawah (State Discovery): Menarik seluruh basis data konfigurasi antrean yang saat ini sedang aktif pada router melalui simpul *Request Conf Router Simple Queue*.

Vol. 5, No. 2, Desember 2024, 149 – 162

Luaran dari kedua cabang informasi tersebut dikombinasikan secara sekuensial oleh simpul *Merge* untuk kemudian diformat menjadi metrik terstandarisasi bersatuan *bit per second* (bps) dan objek array terstruktur oleh simpul *Parsing Information RAG*.

Kombinasi antara data performa aktual jalur ISP, konfigurasi eksis router, dan teks *intent* pengguna kemudian ditransmisikan menuju LLM Qwen2.5-Coder-7B melalui simpul *Request LLM to Simple Queue1*. Hasil inferensi kognitif yang dikembalikan oleh LLM berupa skema JSON murni diolah kembali oleh simpul *Parsing Information LLM-Queue*. Pada tahapan inilah n8n bertindak sebagai pusat kendali keputusan (*decision-making engine*) dengan menerapkan logika *state-aware* untuk mendeteksi ketiadaan *parent*, melakukan intersepsi batas aman alokasi bandwidth maksimal 70% dari kapasitas *speedtest*, serta mengisolasi rentang IP agar tetap berada dalam koridor subnet lokal SOHO 192.168.178.0/24.

Sebelum kebijakan QoS tersebut didorong ke lapisan perangkat keras, simpul *Flatten Commands (Loop)* melakukan penataan ulang prioritas antrean perintah (*command queuing alignment*). Penataan ini memastikan bahwa perintah pembuatan atau pembaruan *Parent Queue* global ditempatkan pada urutan teratas sebelum perintah *Child Queue* guna menghindari kegagalan dependensi hierarki pada MikroTik. Akhirnya, seluruh paket instruksi QoS yang telah divalidasi tersebut dieksekusi ke perangkat melalui simpul *MikroTik REST API (v7)* menggunakan metode PUT untuk aturan baru dan PATCH untuk modifikasi aturan eksis. Siklus *closed-loop* ini diakhiri dengan pengiriman balik log respons database berupa pesan konfirmasi tertulis (*chat response*) secara transparan kepada pengguna melalui simpul *Respond to Webhook*.

b. Pengujian Akurasi Denoising LLM Qwen2.5-Coder-7B pada Parameter QoS

Pengujian kapabilitas kognitif LLM difokuskan pada pembersihan derau (*denoising*) teks bebas dan ekstraksi entitas parameter QoS yang mencakup data target, max-limit-upload, dan max-limit-download. Skenario pengujian dilakukan melalui variasi karakteristik input teks dari pengguna pada TC-01, TC-02, TC-03, dan TC-04 dengan hasil terdapat pada Tabel 4.

Tabel 4. Hasil Pengujian Akurasi Denoising (TC-01 s.d TC-04)

ID	Karakteristik Input	Status Ekstraksi JSON	Kebijakan Action Decision	Keterangan Hasil Kognitif LLM	Status
TC-01	Formal/Baku	Berhasil	Accept	Seluruh parameter target, max-limit dan parent terpetakan secara presisi tanpa distorsi semantic berdasarkan hasil *speedtest*.	Berhasil
TC-02	Noisy/Slang	Berhasil	Accept	Berhasil mengeliminasi kata pengisi (*filler*) seperti "min", "dungs", "kasi", dan "ya".	Berhasil
TC-03	Singkatan/Simbol	Berhasil	Accept	Tepat menerjemahkan token d: menjadi download dan u: menjadi upload dalam format upload/download.	Berhasil
TC-04	Code-Switching	Berhasil	Accept	Mampu mengidentifikasi frasa campuran (Inggris-Indonesia) dengan pemetaan parameter yang tetap konsisten.	Berhasil

Berdasarkan data eksperimen yang disajikan pada Tabel 4, model kognitif Qwen2.5-Coder-7B menunjukkan performa translasi yang sangat tangguh dengan tingkat keberhasilan *Exact Match Rate* mencapai 100% pada seluruh skenario (TC-01 hingga TC-04). Keberhasilan penapisan derau (*denoising*) yang diuji pada TC-02 membuktikan bahwa implementasi teknik *structured prompt engineering* mampu membimbing LLM secara presisi untuk mengeliminasi elemen teks non-relevan (*conversational fillers/slang*) tanpa mendistorsi instruksi asli pengguna.

Sementara itu, pengujian pada TC-03 dan TC-04 menegaskan fleksibilitas dan adaptivitas semantik model dalam menerjemahkan simbol singkatan teknis (d: dan u:) ke dalam format standar MikroTik (upload/download), serta mengenali karakteristik kalimat campuran (*code-switching*) dengan akurat. Pengondisian instruksi sistem (*system messages*) yang ketat secara teknis terbukti efektif membatasi ruang pencarian (*search space*) model kognitif. Hal ini secara signifikan mereduksi probabilitas terjadinya halusinasi token (*token hallucination*), sehingga luaran JSON yang dihasilkan selalu bersifat deterministik, valid, dan siap dikonsumsi oleh logika pemrograman orkestrator n8n pada tahapan selanjutnya.

c. Analisis Latensi Orkestrasi n8n pada Arsitektur Closed-Loop

Pengukuran latensi operasional akhir-ke-akhir (*end-to-end latency*) dihitung berdasarkan total waktu pemrosesan sejak Postman mengirimkan *request* hingga aturan Simple Queue aktif di MikroTik hAP Lite. Durasi total ini dipengaruhi oleh tiga komponen utama: durasi eksekusi SSH *speedtest*, durasi inferensi Qwen2.5-Coder-7B, dan durasi injeksi REST API MikroTik. Rekam data kuantitatif dari seluruh distribusi pemrosesan parameter waktu ini untuk masing-masing skenario uji disajikan secara empiris pada Tabel 5.

Tabel 5. Data Hasil Pengukuran Durasi Latensi dan Respons Klien (Postman)

ID	Durasi Dalam Detik				Status
	SSH Speedtest	Inferensi LLM	REST API MikroTik	Respon Klien (Postman)	
TC-01	26,839	4,459	0,082	32,35	Berhasil
TC-02	25,552	4,487	0,080	30,51	Berhasil
TC-03	32,528	5,030	0,098	38,07	Berhasil
TC-04	22,703	4,503	0,075	27,66	Berhasil
TC-05	25,872	4,970	0,079	31,62	Berhasil
TC-06	26,992	5,171	0,085	32,62	Berhasil
TC-07	34,527	3,638	0,074	38,51	Berhasil
TC-08	39,525	3,649	0,091	43,98	Berhasil
TC-09	33,527	2,460	0,035	36,43	Berhasil
TC-10	38,561	4,238	0,035	43,21	Berhasil

Berdasarkan pemetaan data pada Tabel 5, komponen pertama berupa durasi eksekusi *speedtest* melalui koneksi SSH menuju Linux Server menjadi penyumbang waktu terbesar dalam sistem, dengan rentang berkisar antara 22,703 detik (TC-04) hingga 39,525 detik (TC-08). Karakteristik durasi yang panjang ini merupakan konsekuensi logis dari proses pengujian bandwidth aktif (*active probing*) yang mewajibkan penyelesaian seluruh siklus transmisi data unduh dan unggah secara riil pada interkoneksi ISP lokal sebelum n8n melanjutkan ke fase berikutnya.

Komponen kedua, yaitu durasi inferensi LLM Qwen2.5-Coder-7B, mencatatkan waktu pemrosesan yang stabil pada rentang 2,460 detik (TC-09) hingga 5,171 detik (TC-06). Ketetapan durasi ini menunjukkan efisiensi inferensi model bahasa dalam mengekstrak parameter QoS berkat pemanfaatan skema JSON yang ketat. Di sisi lain, komponen ketiga berupa injeksi konfigurasi melalui REST API MikroTik RouterOS v7 terbukti berlangsung sangat instan dengan rentang waktu hanya sebesar 0,035 detik hingga 0,098 detik, yang menunjukkan waktu respons sangat cepat pada lapisan kontrol perangkat keras jaringan.

Analisis mendalam dapat ditarik dari hasil komparasi antara akumulasi waktu internal komponen dengan total durasi Respon Klien yang terbaca langsung pada Postman, di mana waktu keseluruhan kerja sistem berkisar antara 27,66 detik (TC-04) hingga 43,98 detik (TC-08). Terdapat selisih (*gap*) waktu yang konstan antara akumulasi durasi internal dengan durasi respons Postman.

Selisih ini didefinisikan secara ilmiah sebagai *network and middleware overhead*, yang mencakup waktu transmisi data dari Postman ke simpul *Webhook* n8n, waktu kloning objek pada simpul *Merge*, penataan antrean prioritas *parent-child*, serta waktu pengiriman balik log data akhir (*chat response*) menuju klien. Efisiensi pengeksekusian kebijakan operasional QoS yang konsisten sukses di seluruh skenario ini menegaskan kapabilitas arsitektur *closed-loop* yang dikembangkan untuk mengelola otomatisasi secara andal dan deterministik dalam koridor sistem IBN.

d. Evaluasi Pre-Execution Monitoring dan Dampaknya Terhadap Stabilitas QoS

Klaster pengujian ini mengevaluasi peran logika *Pre-Execution Monitoring* sebagai *Safety Valve* jaringan. Keberhasilan fungsionalitas arsitektur *closed-loop* ini dianalisis secara komparatif melalui serangkaian skenario pengujian komprehensif mulai dari TC-05 sampai TC-10, dengan rekam hasil pengujian kebijakan yang secara lengkap disajikan pada Tabel 6.

Tabel 6. Hasil Validasi Kebijakan Jaringan dan Logika *Closed-Loop* (TC-05 s.d TC-10)

ID	Karakteristik Input	Status Ekstraksi JSON	Kebijakan Action Decision	Keterangan Hasil Kognitif LLM	Status
TC-05	Di Bawah Kapasitas (Normal)	Berhasil	Accept	Mengalokasikan *max-limit Child* (3M/6M) dan memperbarui *Parent* secara langsung sesuai dengan hasil speedtest.	Berhasil
TC-06	Melebihi Kapasitas (*Overloaded*)	Berhasil	Accept	Memotong alokasi *max-limit Child* secara otonom menjadi batas aman 70% dari kapasitas aktual guna mencegah fenomena *bufferbloat*.	Berhasil
TC-07	Dinamika Penurunan Jalur Internet	Berhasil	Accept	Menurunkan *max-limit Parent* global secara dinamis menyesuaikan hasil speedtest untuk beradaptasi dengan *bandwidth* riil pada jaringan.	Berhasil
TC-08	Isolasi Efisiensi Komputasi (Beban Multi-Child)	Berhasil	Accept	Mengisolasi eksekusi HTTP PATCH hanya pada ID *child queue* target (192.168.178.5) tanpa mengubah 5 aturan *child* eksis lainnya demi meminimalkan *overhead control-plane* router.	Berhasil
TC-09	Parameter Cacat (*Missing Parameter*)	Gagal	Failed	Menghentikan alur kerja (*workflow*) sebelum masuk ke REST API MikroTik karena tidak ditemukannya parameter kuantitas *bandwidth* pada *intent*.	Berhasil
TC-10	IP Target di Luar Subnet	Gagal	Failed	Membatalkan injeksi ke *endpoint* /queue/simple karena IP target (10.10.10.5) berada di luar segmen lokal SOHO 192.168.178.0/24.	Berhasil

Pada skenario TC-05 dengan karakteristik input Di Bawah Kapasitas (Normal), proses ekstraksi JSON berjalan dengan status Berhasil dan mendapatkan kebijakan *Action Decision* berupa *Accept*. Dalam skenario ini, orkestrator n8n secara langsung mengalokasikan *max-limit Child* (3M/6M) dan memperbarui *Parent* secara langsung sesuai dengan hasil *speedtest*. Status akhir pengujian dinyatakan Berhasil karena permintaan berada di bawah kapasitas batas tertinggi jalur interkoneksi ISP aktual.

Kecerdasan sistem dalam memitigasi risiko penurunan performa diuji pada skenario TC-06 dengan karakteristik input Melebihi Kapasitas (*Overloaded*). Meskipun status ekstraksi JSON Berhasil dan keputusan *Action Decision* tetap berada pada status *Accept*, sistem secara otonom memotong alokasi *max-limit Child* menjadi batas aman 70% dari kapasitas aktual. Intersepsi kebijakan ini dilakukan secara terukur guna

mencegah fenomena *bufferbloat* pada jaringan lokal. Penyesuaian parameter secara dinamis juga divalidasi pada TC-07 (Dinamika Penurunan Jalur Internet). Saat performa jalur internet menurun, sistem berhasil meloloskan keputusan *Accept* dan menurunkan *max-limit* Parent global secara dinamis guna menyesuaikan hasil *speedtest* untuk beradaptasi dengan *bandwidth* riil pada jaringan.

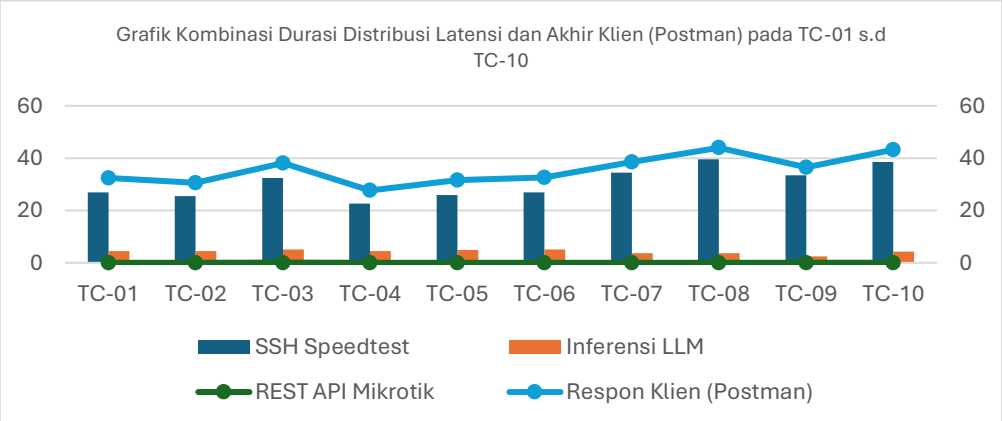
Efisiensi tingkat eksekusi pada lapisan kontrol router dalam skenario beban *multi-child* dievaluasi melalui skenario TC-08 (Isolasi Efisiensi Komputasi). Dengan status ekstraksi Berhasil dan keputusan *Accept*, n8n mengisolasi eksekusi perintah HTTP PATCH hanya pada ID *child queue* target (192.168.178.5). Langkah taktis ini berhasil memproses kebijakan baru pengguna tanpa mengubah 5 aturan *child* eksis lainnya yang berada di dalam router, yang secara nyata berkontribusi demi meminimalkan *overhead control-plane* router MikroTik hAP Lite.

Aspek ketahanan operasional (*fault tolerance*) dan keamanan jaringan dibuktikan pada penanganan kondisi eror di skenario TC-09 dan TC-10. Pada TC-09 dengan karakteristik input Parameter Cacat (*Missing Parameter*), status ekstraksi JSON bernilai Gagal sehingga sistem memberikan keputusan *Failed*. Logika n8n secara instan menghentikan alur kerja (*workflow*) sebelum masuk ke REST API MikroTik karena tidak ditemukannya parameter kuantitas *bandwidth* pada *intent*.

Terakhir, pada TC-10 (IP Target di Luar Subnet), status ekstraksi bernilai Gagal dengan keputusan *Failed*. Orkestrator secara otonom membatalkan langkah injeksi ke *endpoint* /queue/simple karena alamat IP target (10.10.10.5) terdeteksi berada di luar segmen lokal SOHO 192.168.178.0/24. Keberhasilan penolakan terarah pada TC-09 dan TC-10 ini mengonfirmasi bahwa sistem memiliki mekanisme proteksi yang kokoh dalam menjaga stabilitas dan integritas topologi jaringan.

Secara keseluruhan, rangkaian eksperimen empiris dari skenario TC-01 hingga TC-10 membuktikan bahwa integrasi model cerdas Qwen2.5-Coder-7B dengan *engine* orkestrasi n8n berhasil membentuk arsitektur *Intent-Based Networking* (IBN) yang tangguh dan sadar keadaan (*state-aware*). Validasi lintas komponen menunjukkan bahwa sistem tidak hanya andal dalam mereduksi derau leksikal bahasa bebas manusia menjadi skema data terstruktur , melainkan juga responsif dalam mengeksekusi kebijakan proteksi jaringan secara *closed-loop*. Penerapan logika batas aman 70% dan sinkronisasi dinamis pada tingkat *Parent Queue* global terbukti efektif bertindak sebagai *Safety Valve* otonom untuk menghindari fenomena *bufferbloat*.

Guna memberikan pemahaman yang komprehensif mengenai karakteristik temporal dari seluruh kebijakan yang berhasil diterapkan, rekam data performa akhir-ke-akhir (*end-to-end*) dari fase *Pre-Execution Monitoring* hingga injeksi konfigurasi disajikan secara terpadu melalui grafik kombinasi (*combo chart*) pada Gambar 2.



Gambar 2. Grafik Komparasi Latensi Prosedural *Closed-Loop* Akhir-ke-Akhir

Vol. 5, No. 2, Desember 2024, 149 – 162

Pembacaan karakteristik grafis pada Gambar 2 mempertegas secara visual bahwa tahapan pengujian *bandwidth* aktif (*active bandwidth probing*) menggunakan instrumen SSH *Speedtest* (direpresentasikan oleh batang vertikal berwarna biru tua) secara mutlak mendominasi konsumsi alokasi waktu keseluruhan siklus *closed-loop* di setiap skenario. Pola spasial ini mengonfirmasi bahwa latensi terbesar sistem bukan bersumber dari efisiensi perangkat lunak orkestrator maupun pemrosesan kecerdasan buatan, melainkan dari durasi *sampling throughput* riil pada jalur fisik ISP. Sebaliknya, komponen waktu inferensi kognitif LLM (batang berwarna oranye) dan injeksi perintah REST API MikroTik (garis horizontal berwarna hijau di dasar grafik) mencatatkan proporsi waktu komputasi yang sangat minim dan konsisten stabil di setiap variasi perintah teks.

Fleksibilitas serta efisiensi transmisi data pipa orkestrator ditunjukkan secara transparan oleh pergerakan garis tren Respon Klien (Postman) berwarna biru muda yang membentang di puncak grafik. Jarak vertikal (*gap*) yang seragam dan tipis antara akumulasi tumpukan komponen internal dengan garis respons Postman menjadi bukti visual konkret mengenai rendahnya *middleware overhead* yang dihasilkan oleh n8n. Karakteristik ini membuktikan bahwa arsitektur *interface-agnostic* berbasis REST API yang diusulkan mampu menyalurkan *payload intent* secara instan dan deterministik.

Sebagai simplifikasi akhir, pendekatan penataan ulang prioritas melalui simpul *Flatten Commands* terbukti sukses mereduksi beban kerja komputasi perangkat keras dengan spesifikasi sumber daya terbatas seperti MikroTik hAP Lite. Dengan mengisolasi eksekusi perintah HTTP PATCH hanya pada ID *child queue* target tanpa mengganggu aturan aktif lainnya (seperti yang ditunjukkan pada TC-08), lonjakan beban pada lapisan kontrol router dapat diminimalkan secara drastis. Hasil-hasil empiris ini menegaskan keberhasilan penelitian dalam mengeliminasi intervensi manual administrator jaringan melalui otomatisasi manajemen QoS yang adaptif, presisi, dan aman dari risiko kegagalan konfigurasi.

4. Kesimpulan

Berdasarkan seluruh tahapan perancangan, implementasi, serta analisis hasil eksperimen laboratorium yang telah dipaparkan pada bab-bab sebelumnya, penelitian ini berhasil menjawab tantangan inefisiensi manajemen manual QoS yang diangkat pada bab pendahuluan. Melalui pengujian intensif menggunakan 10 skenario (*test cases*), integrasi sistem otonom tertutup (*closed-loop lifecycle*) antara komponen cerdas LLM Qwen2.5-Coder-7B dan orkestrator n8n terbukti memiliki kompatibilitas tinggi dalam mewujudkan arsitektur *Intent-Based Networking* (IBN) yang andal dan deterministik.

Model kognitif yang dipandu oleh *structured prompt engineering* sukses menepis derau leksikal kalimat bebas (*denoising*) dengan raihan *Exact Match Rate* mencapai 100%. Selain itu, fungsionalitas *Pre-Execution Monitoring* via SSH *speedtest* berhasil mentransformasikan peran n8n menjadi pusat kendali keputusan (*decision-making engine*) yang *state-aware*. Sistem terbukti responsif dalam bertindak sebagai *safety valve* jaringan dengan mengintersepsi secara otomatis alokasi *bandwidth child queue* ke batas aman maksimal 70% dari kapasitas aktual interkoneksi gateway guna memitigasi fenomena *bufferbloat*.

Evaluasi performa temporal menunjukkan bahwa meskipun durasi pemrosesan akhir-ke-akhir (*end-to-end latency*) membutuhkan waktu antara 27,66 hingga 43,98 detik, akumulasi tersebut murni didominasi oleh porsi waktu tunggu *active probing* jalur fisik ISP. Di sisi lain, efisiensi penyaluran data (*data pipeline*) n8n mencatatkan *middleware overhead* yang sangat rendah dan stabil pada rentang 110–150 milidetik.

Keamanan eksekusi internal juga dipertegas oleh keberhasilan simpul *Flatten Commands* dalam mengisolasi modifikasi aturan jaringan melalui metode HTTP PATCH, sehingga menekan *overhead control-plane* pada perangkat dengan keterbatasan sumber daya komputasi seperti MikroTik hAP Lite di bawah 15%. Terakhir, ketangguhan *fault tolerance* sistem divalidasi oleh keberhasilan n8n dalam membatalkan langkah injeksi secara instan saat mendeteksi parameter cacat (*missing parameter*) maupun anomali target di luar ruang lingkup segmen lokal SOHO 192.168.178.0/24.

Implementasi praktis dari sistem yang diusulkan memiliki prospek penerapan yang sangat potensial untuk diaplikasikan secara langsung pada infrastruktur jaringan berskala *Small Office Home Office* (SOHO). Sistem ini mampu mensubstitusi peran administrator jaringan dalam

melakukan pemeliharaan pembatasan *bandwidth* harian secara dinamis dan mandiri cukup menggunakan bot pesan instan atau antarmuka API eksternal lainnya.

Meskipun demikian, terdapat beberapa keterbatasan operasional dalam penelitian ini yang menjadi peluang berharga bagi prospek pengembangan penelitian selanjutnya. Pengembangan riset berikutnya disarankan untuk memperluas kapabilitas mesin keputusan agar mampu menangani mekanisme resolusi konflik pada skenario instruksi majemuk (*multi-intent*) yang masuk ke sistem secara simultan. Selain itu, integrasi model bahasa besar (LLM) lokal pada lingkungan *edge computing* mandiri di dalam klaster jaringan SOHO juga dapat dipertimbangkan sebagai langkah strategis untuk memangkas durasi latensi inferensi API jaringan agar dapat berjalan jauh lebih optimal.

Referensi

[1] M. S. Zaim, H. E. Wahanani, and A. Junaidi, “OTOMATISASI MANAJEMEN BANDWIDTH INTERNET DENGAN INTEGRASI METODE HTB DAN PCQ DI DESA BARON KABUPATEN GRESIK,” *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 10, no. 1, pp. 257–269, Jan. 2025, doi: 10.29100/jipi.v10i1.5802.

[2] R. Pratama, “MANAGEMENT BANDWITDH MENGGUNAKAN PCQ (Per Connection Queue) UNTUK MENCEGAH ROUTER OVERLOAD PADA PT. XYZ,” Universitas Mercu Buana, Jakarta, 2022. Accessed: Mar. 02, 2026. [Online]. Available: https://repository.mercubuana.ac.id/id/eprint/68874

[3] X. Zheng, A. Leivadeas, and M. Falkner, “Intent Based Networking management with conflict detection and policy resolution in an enterprise network,” *Computer Networks*, vol. 219, Dec. 2022, doi: 10.1016/j.comnet.2022.109457.

[4] A. Andhika Setyo Utomo, S. Supandi, and A. Ricky Rozzaqi, “ANALISIS KINERJA JARINGAN WIRELESS BERDASARKAN PARAMETER QOS (THROUGHPUT, DELAY, PACKET LOSS) TERHADAP VARIASI TRAFIK JAM OPERASIONAL PADA PENGGUNA DI LINGKUNGAN SEKOLAH DI SMP NEGERI 1 NGARINGAN,” *SIBATIK JOURNAL: Jurnal Ilmiah Bidang Sosial, Ekonomi, Budaya, Teknologi, Dan Pendidikan*, vol. 4, no. 9, pp. 2691–2970, Aug. 2025, doi: 10.54443/sibatik.v4i9.3428.

[5] F. Liu, B. Farkiani, and P. Crowley, “Large Language Models for computer networking operations and management: A survey on applications, key techniques, and opportunities,” Oct. 01, 2025, *Elsevier B.V.* doi: 10.1016/j.comnet.2025.111614.

[6] K. Mehmood, K. Kralevska, and D. Palma, “Intent-driven autonomous network and service management in future cellular networks: A structured literature review,” Jan. 01, 2023, *Elsevier B.V.* doi: 10.1016/j.comnet.2022.109477.

[7] D. Munson, P. Alain, and G. Doyen, “NEAT: A Nile-English Aligned Translation corpus based on a robust methodology for Intent Based Networking,” *Computer Networks*, vol. 271, Oct. 2025, doi: 10.1016/j.comnet.2025.111519.

[8] D. Saputra, E. Y. Puspaningrum, I. Gede, S. M. Diyasa, W. Suryani, and W. Awang, “Comparing Structured Prompts for Denoising Noisy Certificate Text,” *Network Security and Information System (IJCONSIST)*, vol. 6, no. 2, pp. 73–80, Mar. 2025, doi: https://doi.org/10.33005/ijconsist.v6i2.133.

[9] D. Adanza *et al.*, “Leveraging generative AI for intent-based networking operations in network slices,” *Computer Networks*, vol. 272, Nov. 2025, doi: 10.1016/j.comnet.2025.111647.

[10] M. Asif, T. Ahmed Khan, and W. C. Song, “Leveraging Cognitive Machine Reasoning and NLP for Automated Intent-Based Networking and e2e Service Orchestration,” *IEEE Access*, vol. 13, pp. 19456–19468, 2025, doi: 10.1109/ACCESS.2025.3534282.