

Self-Healing Network System with Adaptive Quality of Service on MikroTik RouterOS Using Proximal Policy Optimization

Daffa Faris Abqari Ramdhani ^{1*}, Marisa Premitasari ^{2*}

* Informatika, Fakultas Teknologi Industri, Institut Teknologi Nasional
abqari.ramdhani@gmail.com ¹, marisa@itenas.ac.id ²

Article Info

Article history:

Received 2026-08-12

Revised 2026-09-11

Accepted 2026-10-18

Keyword:

*Closed-Loop Control,
Proximal Policy Optimization,
Quality of Service,
MikroTik RouterOS,
Self-Healing Networks*

ABSTRACT

The limited adaptivity of Quality of Service (QoS) systems on MikroTik RouterOS, which still rely on static configurations and manual intervention, demands a network management mechanism capable of automatically adapting to traffic fluctuations. This study designs a self-healing system using Proximal Policy Optimization (PPO) integrated directly into MikroTik RouterOS via API within a closed-loop control architecture to realize adaptive QoS management. The agent is trained using a 12 feature state space that combines QoS parameters (latency, jitter, packet loss) and queue tree statistics (queued bytes, queued packets, dropped) for each traffic type (bulk, game, streaming), as well as a multidiscrete action space to adapt the queue type, limit-at, and max-limit parameters. The reward function is composed of anomaly penalties and a health bonus with clipping values in the range of [-5.0, 2.0] for training stability. Evaluation is conducted under three load scenarios (normal, static QoS, adaptive QoS) using a combination of Python generator scripts for streaming and game traffic, as well as real downloads via IDM for bulk traffic, with primary performance metrics including response time, recovery time, and recovery quality. Test results show that the self-healing system is capable of adjusting QoS configurations with an average response time of 22.11 ms, recovering network conditions with an average recovery time of 5.65 seconds, and achieving a recovery quality of 100%. These findings indicate that integrating PPO via the RouterOS API can realize an adaptive QoS self-healing system capable of responding to disruptions and restoring network conditions automatically on MikroTik infrastructure.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. PENDAHULUAN

Penggunaan kecerdasan buatan dalam operasi jaringan semakin berkembang untuk mendukung otomatisasi dan menjaga keandalan layanan jaringan [1]. Rata-rata penggunaan data jaringan per rumah tangga diproyeksikan meningkat delapan kali lipat hingga mencapai 1,3 TB per bulan pada tahun 2030. Seiring dengan peningkatan skala jaringan, kompleksitas pengelolaannya juga meningkat dan mendorong kebutuhan terhadap kemampuan *self-configuring*, *self-healing*, dan *self-optimizing* [2]. Sejalan dengan kebutuhan tersebut, dua pertiga tim jaringan perusahaan menempatkan kemampuan *Artificial Intelligence for IT Operations* (AIOps) sebagai elemen esensial dalam strategi otomatisasi jaringan *data center* [3]. Fenomena

tersebut menegaskan perlunya kemampuan pengelolaan jaringan yang lebih adaptif dan mandiri, termasuk pada perangkat operasional RouterOS MikroTik.

Permasalahan utama pada penelitian ini adalah keterbatasan adaptivitas sistem QoS pada RouterOS MikroTik yang masih menggunakan konfigurasi statis dan manual dengan intervensi administrator [4]. Akibatnya, sistem belum mampu menyesuaikan konfigurasi QoS secara otomatis terhadap fluktuasi trafik. Pengelolaan QoS secara statis dan manual tersebut dinilai tidak lagi memadai untuk mengakomodasi kompleksitas jaringan modern yang dinamis dan heterogen [5], [6]. Kondisi ini menuntut adanya sistem pengelolaan QoS yang mampu beradaptasi secara otomatis terhadap perubahan beban jaringan tanpa intervensi manual.

Penelitian terdahulu telah menerapkan *Simple Queue*, *Queue Tree*, dan *Per Connection Queue* (PCQ) dalam pengelolaan QoS konvensional. *Simple Queue* digunakan untuk membatasi layanan pada IP tertentu secara statis [7], sedangkan *Queue Tree* berbasis PCQ digunakan untuk mengklasifikasikan trafik sekaligus mendistribusikan kapasitas jaringan secara adil antarpengguna aktif [8]. Seiring perkembangan pendekatan pengelolaan jaringan, *Deep Reinforcement Learning* (DRL) mulai dimanfaatkan untuk optimasi *routing* dan pengelolaan QoS yang lebih adaptif pada lingkungan *Software Defined Network* (SDN) [9], [10].

Meskipun penelitian terdahulu telah mengeksplorasi pengelolaan QoS pada RouterOS MikroTik serta pemanfaatan DRL untuk optimasi jaringan. Penerapan DRL untuk mengadaptasi konfigurasi QoS secara langsung pada RouterOS MikroTik masih sangat terbatas, sebagian besar studi masih berfokus pada implementasi skema QoS konvensional seperti *Simple Queue* dan *Queue Tree*, atau menerapkan DRL pada lingkungan SDN. Sementara itu, RouterOS MikroTik digunakan pada lingkungan jaringan non-SDN, sehingga kondisi ini meninggalkan celah penelitian mengenai belum tersedianya sistem *self-healing* menggunakan DRL yang terintegrasi langsung pada RouterOS MikroTik untuk pengelolaan QoS yang adaptif.

Guna mengisi celah penelitian tersebut, penelitian ini bertujuan untuk merancang dan mengintegrasikan algoritma *Proximal Policy Optimization* (PPO) secara langsung pada RouterOS MikroTik sebagai bagian dari sistem *self-healing*, serta mengevaluasi efektivitasnya dalam mengadaptasi konfigurasi QoS secara otomatis terhadap fluktuasi trafik. Kontribusi penelitian ini terletak pada integrasi PPO dengan mekanisme *closed-loop* dan RouterOS API untuk mewujudkan pengelolaan QoS yang adaptif dan otonom pada infrastruktur jaringan nyata.

II. METODE

Bagian ini menguraikan metodologi penelitian, yang mencakup perancangan sistem *self-healing*, lingkungan jaringan, pemodelan agen PPO, evaluasi kinerja, dan skenario pengujian.

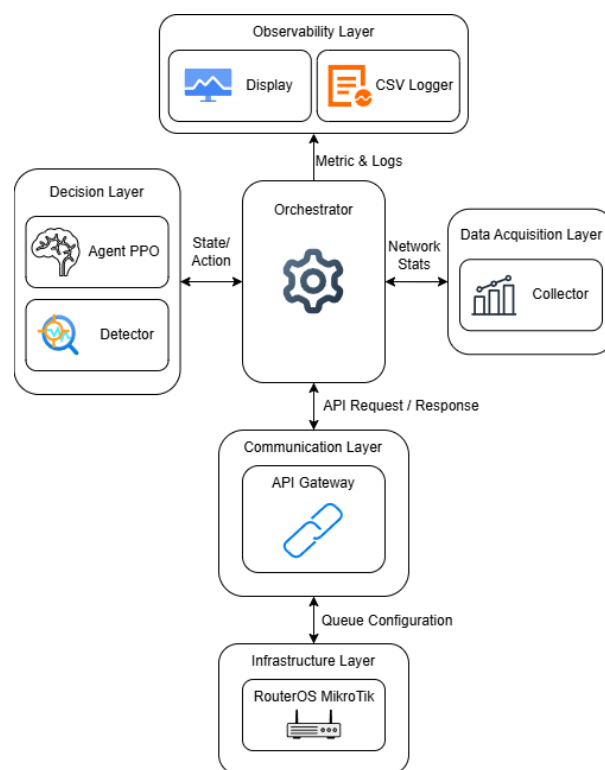
A. Perancangan Sistem Self-Healing

Perancangan sistem *self-healing* mencakup dua aspek, yaitu arsitektur dan siklus interaksi. Arsitektur memaparkan struktur dan peran tiap komponen, siklus interaksi menjelaskan alur interaksi antar komponen dalam operasi nyata.

1) Arsitektur

Arsitektur sistem dirancang dalam lima *layer* operasional yang memiliki tanggung jawab berbeda dan saling mendukung dalam menjalankan sistem *self-healing*. Kelima *layer* tersebut terdiri atas *Infrastructure Layer* yang menyediakan perangkat dan lingkungan jaringan sebagai

tempat sistem diterapkan, *Communication Layer* yang mengelola komunikasi dengan RouterOS MikroTik melalui RouterOS API [11], *Data Acquisition Layer* yang mengumpulkan data kondisi jaringan, *Decision Layer* yang menjalankan proses deteksi anomali dan pengambilan keputusan menggunakan agen PPO, serta *Observability Layer* yang mencatat kondisi dan aktivitas sistem untuk mendukung pemantauan dan analisis. Pembagian arsitektur ke dalam beberapa *layer* bertujuan untuk memisahkan fungsi setiap bagian sistem sehingga proses pemantauan, pengambilan keputusan, penerapan tindakan, dan pencatatan kondisi jaringan dapat dilakukan secara terstruktur. Arsitektur sistem tersebut dapat dilihat pada Gambar 1.

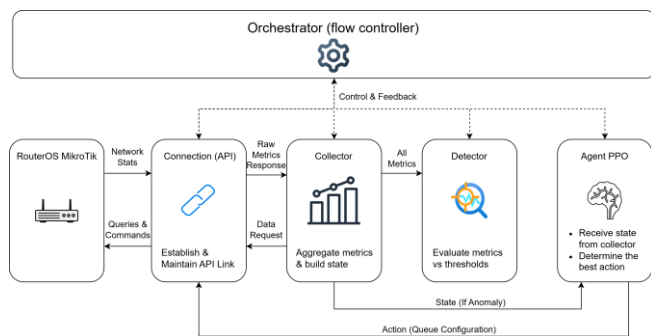


Gambar 1. Arsitektur Sistem

2) Siklus Interaksi dan Integrasi Sistem

Sistem *self-healing* menerapkan mekanisme kendali *closed-loop* melalui interaksi dan integrasi antarkomponen sistem [12], [13]. Dalam arsitektur ini, *Orchestrator* berperan sebagai pengendali alur sekaligus mengintegrasikan komponen-komponen sistem untuk menjalankan proses secara berurutan, sementara setiap komponen menjalankan fungsi masing-masing secara independen. Siklus diawali dengan *Orchestrator* yang menginisialisasi *Connection* untuk berkomunikasi dengan RouterOS MikroTik melalui RouterOS API. Setelah koneksi berhasil dibuat, *Collector* mengumpulkan data jaringan secara berkala setiap 4 detik. Selanjutnya, *Detector* mengevaluasi data dengan membandingkan nilai setiap parameter QoS (*max latency*,

jitter, dan *packet loss*) serta statistik antrean (*queued bytes*, *queued packets*, dan *dropped*) terhadap *threshold* yang telah ditetapkan untuk menentukan kondisi normal atau anomali. Jika terdeteksi anomali, *Orchestrator* meneruskan *state* jaringan kepada agen PPO untuk menentukan konfigurasi QoS yang sesuai, sedangkan jika jaringan berada dalam status normal, agen PPO tidak dijalankan sehingga konfigurasi QoS tetap dipertahankan. Hasil keputusan agen PPO kemudian dikirim melalui *Connection* dan diterapkan pada RouterOS MikroTik. Setelah konfigurasi diterapkan, kondisi jaringan dipantau kembali sehingga membentuk mekanisme *closed-loop*. Alur interaksi sistem dapat dilihat pada Gambar 2.



Gambar 2. Siklus Interaksi dan Integrasi Sistem

B. Lingkungan Jaringan

Lingkungan jaringan diterapkan sebagai media implementasi, pelatihan, dan pengujian sistem. Lingkungan ini dibangun dengan merancang topologi jaringan dan sumber trafik yang diimplementasikan menggunakan infrastruktur nyata untuk merepresentasikan kondisi jaringan yang menjadi objek penelitian.

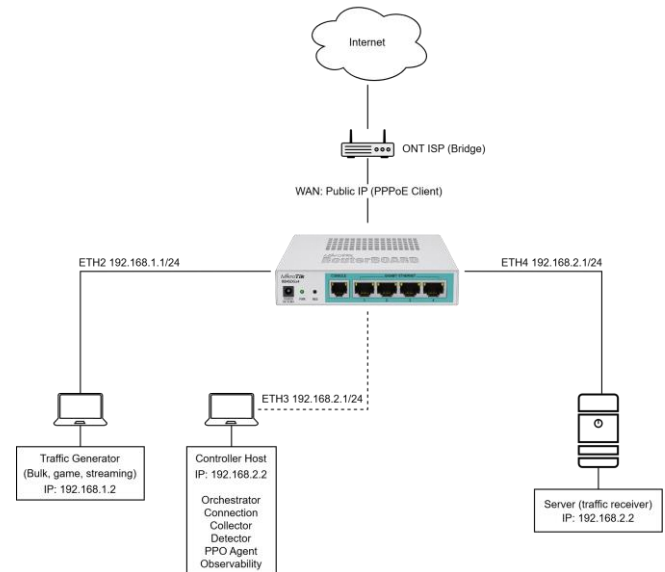
1) Sumber Trafik

Tiga kategori trafik, yaitu *bulk*, *game*, dan *streaming*, digunakan untuk merepresentasikan variasi beban jaringan selama proses pengujian. Trafik *streaming* dan *game* dibangkitkan menggunakan skrip *Python* dengan parameter teknis berupa *bitrate*, *packet size*, *inter-packet arrival time*, dan pola *burstiness* yang disesuaikan dengan karakteristik masing-masing jenis trafik. Sementara itu, trafik *bulk* dihasilkan melalui proses unduhan nyata menggunakan IDM untuk merepresentasikan aktivitas transfer data dalam jumlah besar. Ketiga kategori trafik dipilih berdasarkan karakteristiknya, yaitu pola *bursty* pada *bulk*, sensitivitas terhadap *jitter* pada *streaming*, dan kebutuhan *latency* rendah pada *game*. Perbedaan karakteristik tersebut digunakan untuk menguji respons agen PPO terhadap berbagai kondisi beban jaringan dengan kondisi pengujian yang dapat dikendalikan dan diulangi.

2) Topologi

Topologi jaringan mengatur koneksi antarperangkat untuk mendukung implementasi sistem *self-healing*. Konfigurasi

ini memastikan seluruh trafik melewati RouterOS MikroTik sehingga sistem dapat memantau kondisi jaringan dan melakukan penyesuaian konfigurasi QoS. Rincian hubungan antarperangkat dapat dilihat pada Gambar 3.



Gambar 3. Topologi Jaringan

C. Pemodelan Agen PPO

Agen PPO diimplementasikan sebagai inti pengambil keputusan dalam sistem *self-healing*. Model ini dibangun melalui integrasi tiga elemen fundamental *reinforcement learning*, yaitu *state space*, *action space*, dan *reward function* [14]. Penggunaan PPO dipilih karena sifat *on-policy*-nya yang sesuai untuk *online learning* di lingkungan jaringan yang dinamis, serta mekanisme pembaruan kebijakan (*policy update*) yang menjaga kestabilan pelatihan sehingga agen mampu beradaptasi secara aman terhadap perubahan beban trafik secara *real-time*.

1) State Space

State space merupakan representasi kondisi lingkungan yang diamati oleh agen PPO. Dalam implementasinya, *state* menggunakan tipe *Box* yang memuat 12 fitur yang terdiri dari parameter QoS (*maximum latency*, *jitter*, *packet loss*) dan statistik antrean (*queued bytes*, *queued packets*, *dropped*) dari setiap *child queue* (*bulk*, *game*, *streaming*). Parameter QoS dipilih karena merupakan indikator langsung yang umum digunakan untuk mengukur kualitas layanan jaringan. Sementara itu, statistik antrean ditambahkan agar agen tidak hanya mengandalkan informasi QoS secara global, melainkan juga memperoleh gambaran yang lebih lengkap mengenai kondisi internal setiap *child queue* secara individual. Seluruh fitur yang digunakan dinormalisasi secara otomatis melalui modul *vecnormalize* yang tersedia dalam *library Stable-Baselines3* selama proses pelatihan, sehingga setiap fitur memiliki skala yang sebanding tanpa mengubah distribusi data asli secara signifikan.

2) Action Space

Action space merupakan ruang aksi yang digunakan agen PPO dalam menentukan tindakan. Implementasinya menggunakan tipe *MultiDiscrete*, sehingga dalam satu *step* keputusan, agen dapat menentukan tiga parameter, yaitu *queue type*, *limit-at*, dan *max-limit*, untuk setiap *child queue* (*bulk*, *game*, dan *streaming*) secara bersamaan. Dengan demikian, terdapat 9 keputusan diskrit yang dihasilkan dalam setiap *step*. Masing-masing parameter memiliki tiga pilihan nilai, sehingga setiap *child queue* memiliki $3 \times 3 \times 3 = 27$ kemungkinan kombinasi perubahan parameter. Kombinasi tersebut diterapkan pada konfigurasi *child queue* dengan tetap mempertimbangkan kapasitas *parent queue* sebesar 60 Mbps. Untuk mencegah konfigurasi yang tidak valid dan menjaga stabilitas antrean, diterapkan batasan logis berupa $\text{limit-at} \leq \text{max-limit}$ pada setiap *child queue*. Apabila agen menghasilkan kombinasi yang melanggar batasan tersebut, aksi ditolak dan sistem mempertahankan konfigurasi sebelumnya. Rincian pilihan nilai dan kombinasi *action space* dapat dilihat pada Tabel I.

Tabel I. ACTION SPACE

Trafik	Queue Type	Limit-At (mbps)	Max-Limit (mbps)
Bulk	pfifo, cake, fq_codel	10, 15, 20	25, 30, 35
Game	pfifo, cake, fq_codel	2, 4, 6	8, 10, 12
Streaming	pfifo, cake, fq_codel	5, 10, 15	20, 25, 30

3) Reward Function

Reward function berfungsi sebagai sinyal *feedback* yang digunakan agen PPO untuk mengevaluasi kualitas setiap tindakan selama proses pelatihan. Nilai *reward* dihitung berdasarkan hasil evaluasi kondisi jaringan dari modul *Detector*, sehingga setiap tindakan yang dipilih agen memperoleh *feedback* terhadap kondisi jaringan setelah tindakan tersebut diterapkan. Formulasi matematis perhitungan *reward* dirumuskan pada Persamaan berikut:

$$R = \text{clip}(e^{-\lambda W} + B_{\text{health}} - (\beta_c C + \beta_w N), R_{\min}, R_{\max})$$

Keterangan:

R = nilai *reward*

λ = parameter pengali eksponensial

W = total *weight* hasil deteksi

B_{health} = *health bonus*

β_c = koefisien penalti *critical*

C = jumlah parameter berstatus *critical*

β_w = koefisien penalti *warning*

N = jumlah parameter berstatus *warning*

$R_{\min}$ = batas bawah fungsi *clipping*

$R_{\max}$ = batas atas fungsi *clipping*

Berdasarkan Persamaan tersebut, nilai *reward* dihitung menggunakan fungsi eksponensial yang bergantung pada

total weight, kemudian dikombinasikan dengan *health bonus* serta penalti berdasarkan jumlah parameter berstatus *warning* maupun *critical*. Penalti digunakan untuk memberikan sinyal negatif terhadap tindakan yang masih menghasilkan kondisi jaringan kurang optimal. Pemberian penalti tersebut diperlukan agar agen dapat membedakan tindakan yang menghasilkan kondisi buruk dan secara bertahap mempelajari tindakan yang lebih sesuai. Sebaliknya, *health bonus* digunakan untuk memberikan sinyal positif ketika seluruh parameter berada dalam kondisi sehat. Komponen ini diperlukan agar agen tidak hanya menghindari kondisi buruk, tetapi juga memperoleh dorongan untuk mempertahankan kondisi jaringan yang optimal. Selanjutnya, nilai *reward* dibatasi menggunakan fungsi *clipping* untuk mencegah terbentuknya nilai *reward* yang terlalu ekstrem. Pembatasan tersebut diperlukan agar skala sinyal pembelajaran tetap terkendali sehingga proses optimasi kebijakan dapat berlangsung secara lebih stabil.

4) Skema Pelatihan Agen

Pelatihan agen PPO menerapkan skema *online learning* dengan *environment* sebagai media interaksi agen. Parameter pelatihan menggunakan *total timesteps* sebesar 10.240 dengan panjang setiap *episode* sebesar 200 *step*. Pembaruan *policy* dilakukan menggunakan parameter *n_steps* sebesar 512 dan *batch size* sebesar 64. Parameter PPO lainnya menggunakan nilai *default*, yaitu *learning rate* sebesar 0,0003, *discount factor* sebesar 0,99, *clipping parameter* sebesar 0,2, dan *entropy coefficient* sebesar 0,0. Selama proses pelatihan, performa agen dievaluasi menggunakan metrik *rollout/ep_reward_mean* dan *eval/mean_reward* untuk memantau perkembangan kemampuan agen dalam memperoleh *reward*. Sementara itu, perubahan *policy* dievaluasi menggunakan metrik *train/approx_kl* untuk memantau perubahan *policy* selama pelatihan. *Training* dihentikan secara otomatis melalui *early stopping* apabila nilai *eval/mean_reward* tidak menunjukkan peningkatan yang signifikan selama 3 kali evaluasi berturut-turut.

Environment pelatihan dibangun menggunakan infrastruktur jaringan nyata yang mencakup topologi jaringan dan sumber trafik yang telah dijelaskan pada subbab sebelumnya. Dalam proses *online learning*, agen PPO berinteraksi secara langsung dengan lingkungan tersebut dengan memperoleh *state* berdasarkan kondisi jaringan, menghasilkan *action* berupa perubahan parameter antrean, dan menerapkan tindakan tersebut pada RouterOS MikroTik melalui RouterOS API. Kondisi jaringan setelah tindakan diterapkan kemudian digunakan untuk menghitung *reward* sebagai umpan balik bagi agen. Dengan demikian, proses pelatihan agen PPO dilakukan secara langsung pada jaringan nyata dan tidak menggunakan simulator.

D. Evaluasi Kinerja

Evaluasi kinerja sistem *self-healing* dilakukan menggunakan tiga metrik yang mencakup *response time*,

recovery time, dan *recovery quality* pada setiap siklus pengujian [15]. Data *response time* dan *recovery time* yang diperoleh dari setiap siklus pada skenario pengujian kemudian dirangkum menggunakan nilai minimum, maksimum, dan rata-rata. Nilai minimum dan maksimum digunakan untuk menunjukkan waktu tercepat dan terlama yang diperlukan sistem, sedangkan nilai rata-rata digunakan untuk menggambarkan kecenderungan waktu respons dan waktu pemulihan. Sementara itu, *recovery quality* dievaluasi berdasarkan jumlah keberhasilan pemulihan dari seluruh siklus pengujian pada setiap skenario.

1) Response Time

Response Time menunjukkan waktu yang dibutuhkan sistem sejak anomali terdeteksi hingga konfigurasi QoS diterapkan. Nilainya dihitung dari selisih waktu antara deteksi anomali dan penerapan konfigurasi QoS. Metrik ini digunakan untuk mengukur kecepatan respons sistem dalam menerapkan konfigurasi QoS setelah anomali terdeteksi, tanpa mempertimbangkan kondisi status jaringan pada siklus pengukuran berikutnya. Perhitungannya dirumuskan pada Persamaan berikut:

$$T_{\text{response}} = T_{\text{apply}} - T_{\text{detection}}$$

2) Recovery Time

Recovery Time mengukur waktu yang diperlukan sistem untuk mengembalikan status jaringan ke kondisi normal setelah konfigurasi QoS diterapkan sebagai tindakan *self-healing*. Nilai *recovery time* diperoleh dari selisih waktu antara penerapan konfigurasi QoS dan waktu saat status jaringan pertama kali terdeteksi normal pada salah satu dari maksimal tiga proses *collection* setelah tindakan *recovery*. Apabila status jaringan tetap anomali setelah *collection* ketiga, *recovery time* dinyatakan N/A (FAILED). Metrik ini digunakan untuk menilai kecepatan sistem dalam menyelesaikan proses pemulihan jaringan. Perhitungannya dirumuskan pada Persamaan berikut:

$$T_{\text{recovery}} = T_{\text{normal}} - T_{\text{apply}}$$

3) Recovery Quality

Recovery Quality menggambarkan tingkat keberhasilan sistem dalam memulihkan kondisi jaringan setelah anomali terdeteksi dan tindakan *recovery* diterapkan. Penilaian dilakukan berdasarkan persentase percobaan *recovery* yang berhasil mengembalikan status jaringan dari kondisi anomali menjadi normal pada salah satu dari maksimal tiga proses *collection* setelah tindakan *recovery*. Percobaan *recovery* dinyatakan berhasil apabila status normal tercapai dalam batas tersebut, sedangkan percobaan dinyatakan gagal apabila status jaringan tetap anomali setelah *collection* ketiga. Metrik ini digunakan untuk mengukur tingkat keberhasilan sistem *self-healing* dalam memulihkan kondisi

jaringan. Perhitungannya dirumuskan pada Persamaan berikut:

$$\frac{\text{Jumlah recovery berhasil}}{\text{Jumlah seluruh percobaan recovery}} \times 100\%$$

E. Rancangan Skenario Pengujian

Pengujian pada setiap skenario dilakukan dalam 10 siklus untuk memperoleh data yang merepresentasikan kondisi jaringan dan respons sistem selama pengujian. Pada setiap siklus, pengukuran dilakukan terhadap tiga parameter QoS, yaitu *max latency*, *jitter*, dan *packet loss*, pada kondisi sebelum dan sesudah sistem melakukan tindakan *self-healing*. Nilai hasil pengukuran sebelum dan sesudah tindakan tersebut kemudian dibandingkan untuk mengetahui perubahan kondisi jaringan serta mengevaluasi kemampuan sistem dalam memulihkan kualitas layanan pada masing-masing skenario.

1) Trafik Normal

Trafik normal merupakan kondisi ketika beban jaringan berada di bawah *threshold*. Kondisi tersebut dibentuk dengan membangkitkan trafik *bulk*, *game*, dan *streaming* menggunakan parameter yang telah ditentukan, kemudian dipertahankan di bawah *threshold*. Selama pengujian, sistem *self-healing* tidak mengaktifkan agen PPO untuk melakukan penyesuaian konfigurasi QoS. Pengujian ini digunakan untuk memastikan sistem tetap beroperasi secara normal tanpa intervensi yang tidak diperlukan.

2) Trafik Overload dengan QoS Statis

Pada kondisi *overload* dengan QoS statis, konfigurasi QoS dipertahankan tanpa penyesuaian oleh agen PPO. Kondisi tersebut dibentuk dengan membangkitkan trafik *bulk*, *game*, dan *streaming* secara bertahap menggunakan parameter yang telah ditentukan hingga *parent queue* mencapai kapasitas maksimum. Pengujian ini digunakan sebagai *baseline* untuk membandingkan kinerja sistem dengan kondisi *overload* menggunakan QoS adaptif.

3) Trafik Overload dengan QoS Adaptif

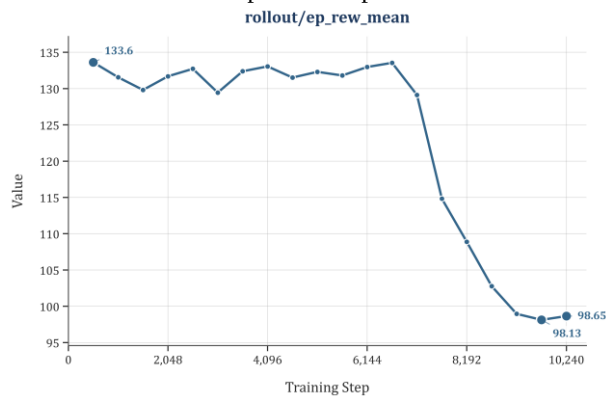
Pada kondisi *overload* dengan QoS adaptif, konfigurasi QoS disesuaikan berdasarkan keputusan yang dihasilkan oleh agen PPO. Kondisi tersebut dibentuk dengan membangkitkan trafik *bulk*, *game*, dan *streaming* secara bertahap menggunakan parameter yang telah ditentukan hingga *parent queue* mencapai kapasitas maksimum. Kondisi ini memicu sistem mendeteksi anomali dan mengaktifkan agen PPO untuk menyesuaikan konfigurasi QoS. Pengujian ini digunakan untuk mengamati respons sistem *self-healing* dalam menangani kondisi kongesti.

III. HASIL DAN PEMBAHASAN

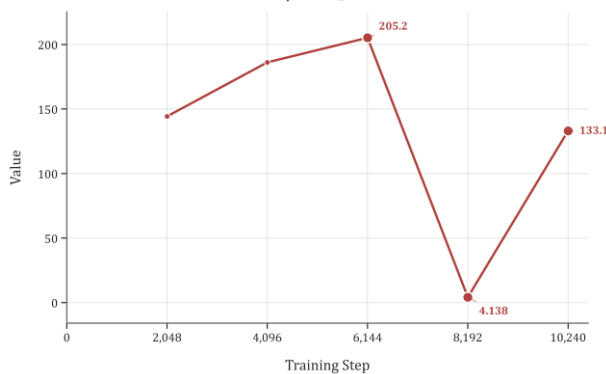
A. Performa Pelatihan Agen PPO

1) Perkembangan Nilai Reward

Selama proses *online learning*, nilai *rollout/ep_reward_mean* diawali pada nilai 133,6 dan cenderung stabil hingga *step* 6.144, sebelum mengalami penurunan hingga mencapai nilai terendah sebesar 98,13 pada *step* 9.600 dan berakhir pada nilai 98,65 di akhir pelatihan. Sementara itu, nilai *eval/mean_reward* meningkat secara bertahap hingga mencapai nilai tertinggi sebesar 205,2 pada *step* 6.144, kemudian menurun hingga mencapai nilai terendah sebesar 4,138 pada *step* 8.192, dan kembali meningkat hingga mencapai nilai 133,1 pada akhir pelatihan. Perkembangan kedua metrik tersebut dapat dilihat pada Gambar 4 dan 5.



Gambar 4. Nilai rollout/ep-reward-mean
eval/mean_reward



Gambar 5. Nilai eval/mean-reward

2) Perubahan Policy selama Pelatihan

Pada proses pelatihan yang sama, nilai *train/approx_kl* menunjukkan pola fluktuatif sepanjang pelatihan, dengan nilai tertinggi sebesar 0.01556 pada *step* 4.096, nilai terendah sebesar 0.006055 pada *step* 5.120, dan nilai akhir sebesar 0.01281 pada *step* 10.240. Perkembangan nilai *train/approx_kl* dapat dilihat pada Gambar 6.

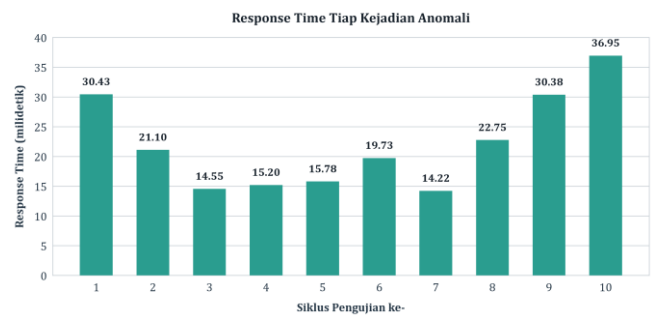


Gambar 6. Nilai Approx_Kl

B. Kinerja Sistem

1) Response Time

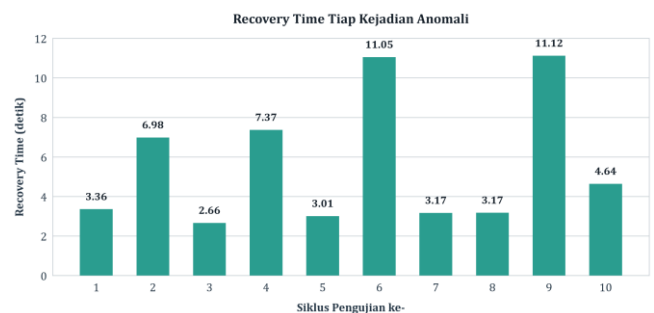
Evaluasi *response time* menghasilkan waktu penyelesaian dari anomali terdeteksi hingga konfigurasi QoS diterapkan tercepat 14.22 ms dan terlama 39.95 ms, dengan rata-rata *response time* 22.11 ms (95% CI: 16.07-28.15 ms). Distribusi hasil pengujian tersebut dapat dilihat pada Gambar 7.



Gambar 7. Hasil Response Time

2) Recovery Time

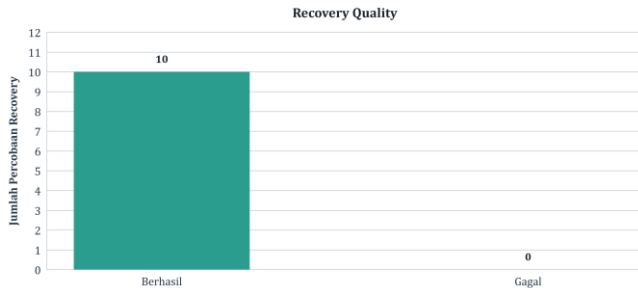
Berdasarkan evaluasi *recovery time*, diperoleh durasi pemulihan tercepat 2.66 detik dan terlama 11.12 detik, dengan rata-rata *recovery time* 5.65 detik (95% CI: 3.30–8.00 detik). Distribusi hasil pengujian tersebut dapat dilihat pada Gambar 8.



Gambar 8. Hasil Recovery Time

3) Recovery Quality

Dalam evaluasi *recovery quality*, tercatat pengembalian kondisi jaringan dari anomali ke normal sebanyak 10 dari 10 siklus percobaan, dengan nilai *recovery quality* sebesar 100%. Distribusi hasil pengujian tersebut dapat dilihat pada Gambar 9.

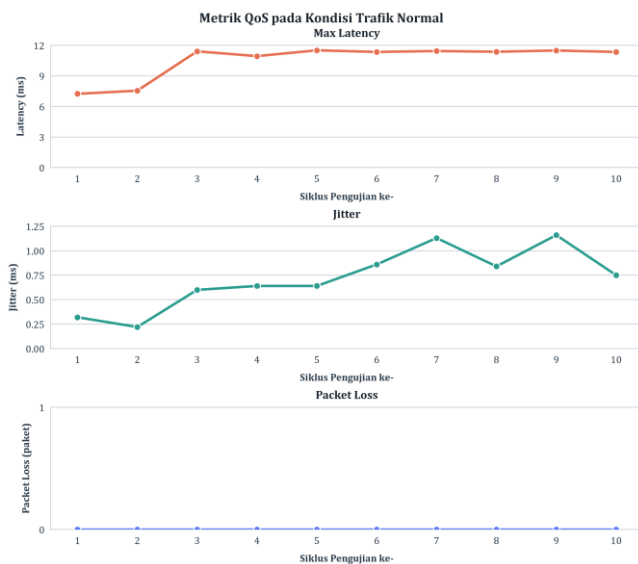


Gambar 9. Hasil *Recovery Quality*

C. Perbandingan setiap Skenario

1) Trafik Normal

Pada kondisi trafik normal, nilai *max latency* tercatat sebesar 7.24-7.55 ms pada dua siklus pertama, lalu meningkat dan relatif stabil pada rentang 10.94-11.51 ms hingga siklus ke-10. Sementara itu, *jitter* bergerak fluktuatif pada rentang 0.26-1.16 ms, sedangkan *packet loss* konsisten bernilai 0 paket di seluruh siklus pengujian. Hasil pengujian juga menunjukkan status sistem berada pada kondisi normal tanpa adanya *warning count* maupun *critical count*. Perkembangan nilai QoS tersebut dapat dilihat pada Gambar 10.

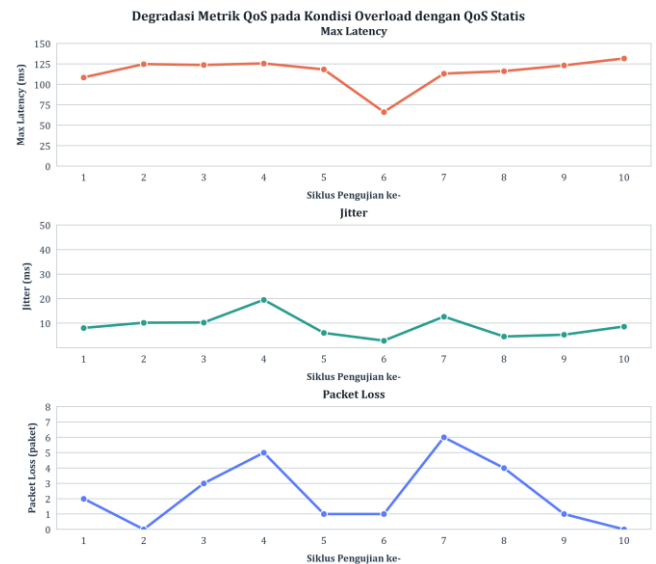


Gambar 10. Nilai Parameter QoS pada Kondisi Trafik Normal

2) Trafik Overload dengan QoS Statis

Pada kondisi trafik *overload* dengan QoS statis dan agen PPO nonaktif, nilai *max latency* mayoritas berada pada tingkat yang tinggi dalam rentang 108.47-131.68 ms dengan

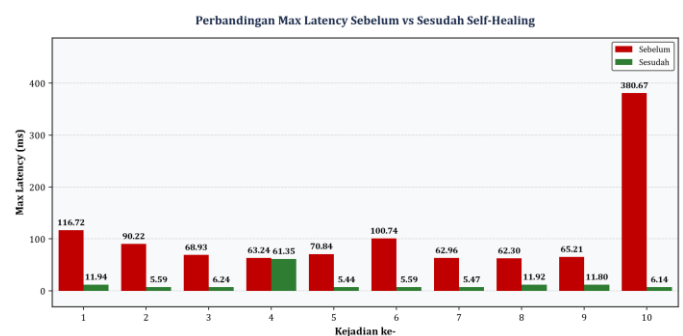
titik tertinggi dicapai pada siklus ke-10, meskipun sempat mengalami penurunan signifikan hingga 66.25 ms pada siklus ke-6. Nilai *jitter* bergerak fluktuatif pada rentang 2.90-19.54 ms dengan lonjakan tertinggi terjadi pada siklus ke-4. Sementara itu, *packet loss* berfluktuasi pada rentang 0-6 paket sepanjang pengujian dan mencapai jumlah terbanyak sebesar 6 paket pada siklus ke-7. Perkembangan seluruh nilai parameter QoS tersebut dapat dilihat pada Gambar 11.



Gambar 11. Nilai Parameter QoS pada Kondisi *Overload* dengan QoS Statis

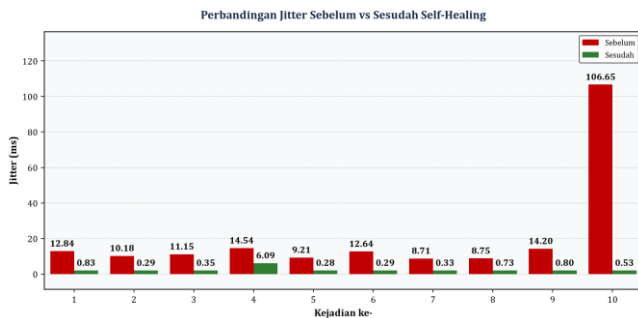
3) Trafik Overload dengan QoS Adaptif

Pada kondisi trafik *overload* dengan QoS adaptif, nilai *max latency* sebelum penerapan *self-healing* berada pada rentang 62.30-380.67 ms. Nilai tertinggi terjadi pada kejadian ke-10 sebesar 380.67 ms, diikuti kejadian ke-1 sebesar 116.72 ms dan kejadian ke-6 sebesar 100.74 ms. Setelah *self-healing* diterapkan, nilai *max latency* menurun secara signifikan dan berada pada rentang 5.44-11.94 ms. Adapun nilai tertinggi setelah pemulihan tercatat sebesar 61.35 ms pada kejadian ke-4. Pergerakan nilai *max latency* sebelum dan sesudah penerapan *self-healing* dapat dilihat pada Gambar 12



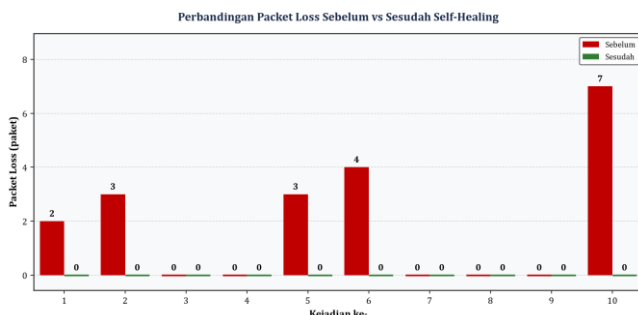
Gambar 12. *Max Latency* Sebelum dan Sesudah *Self-Healing*
Sementara itu, nilai *jitter* sebelum penerapan *self-healing* menunjukkan fluktuasi pada rentang 8.71-106.65 ms. Nilai

tertinggi terjadi pada kejadian ke-10, yaitu sebesar 106.65 ms, sedangkan sembilan kejadian lainnya berada pada kisaran 8.71-14.54 ms. Setelah mekanisme *self-healing* diterapkan, nilai *jitter* mengalami penurunan yang signifikan dan berada pada rentang 0.28-0.83 ms pada sebagian besar kejadian, sedangkan kejadian ke-4 mencatat nilai sebesar 6.09 ms. Pergerakan nilai *jitter* sebelum dan sesudah penerapan *self-healing* dapat dilihat pada Gambar 13



Gambar 13. *Jitter* Sebelum dan Sesudah *Self-Healing*

Sedangkan pada metrik *packet loss*, sebelum penerapan *self-healing*, kehilangan paket tercatat dalam rentang 0-7 paket selama pengujian. Jumlah *packet loss* tertinggi terjadi pada kejadian ke-10 dengan 7 paket, diikuti kejadian ke-6 dengan 4 paket, serta kejadian ke-2 dan ke-5 yang masing-masing mencatat 3 paket. Setelah mekanisme *self-healing* diterapkan, jumlah *packet loss* menurun hingga mencapai 0 paket pada seluruh 10 kejadian pengujian. Pergerakan nilai *packet loss* sebelum dan sesudah penerapan *self-healing* dapat dilihat pada Gambar 14..



Gambar 14. *Packet Loss* Sebelum dan Sesudah *Self-Healing*

D. Pembahasan

Hasil evaluasi proses pelatihan agen PPO memperlihatkan pola pembelajaran yang tercermin pada nilai *reward* dan perubahan *policy* selama pelatihan. Pada metrik *reward*, lonjakan nilai hingga mencapai puncak terjadi saat agen PPO berhasil menerapkan kombinasi konfigurasi QoS (*queue type*, *limit-at*, dan *max-limit*) yang tepat, sehingga metrik jaringan berada pada kondisi sehat dan agen memperoleh *health bonus* secara maksimal. Sebaliknya, penurunan tajam pada *step* tertentu disebabkan oleh fase *re-eksplorasi*, di mana agen menguji kombinasi aksi baru yang belum teruji hingga memicu akumulasi penalti *warning* dan *critical*

secara bersamaan pada metrik jaringan. Pola naik-turun ini mencerminkan proses *trial and error* yang wajar sekaligus esensial dalam *reinforcement learning* dengan dampak positif berupa terbentuknya sistem yang *robust* dalam menghadapi berbagai skenario gangguan ekstrem dan tidak sekadar menghafal satu kondisi jaringan ideal saja. Sementara itu, fluktuasi *approximate KL* menunjukkan bahwa besarnya perubahan *policy* pada setiap pembaruan tidak selalu sama. Perubahan tersebut mencerminkan penyesuaian *policy* dari *policy* lama menuju *policy* baru selama pelatihan, sehingga agen tidak mempertahankan satu pola pengambilan tindakan secara tetap dan terus memperbarui *policy* berdasarkan pengalaman yang diperoleh dari lingkungan.

Hasil pengujian terhadap *response time*, *recovery time*, dan *recovery quality* memperlihatkan kemampuan mekanisme *closed-loop* dalam merespons anomali dan memulihkan kondisi jaringan. Rendahnya *response time* dengan rata-rata 22.11 ms pada rentang 14.22-39.95 ms menunjukkan bahwa rekonfigurasi QoS dapat segera diterapkan setelah anomali terdeteksi sehingga tidak menghambat alur kerja *orchestrator*. Setelah konfigurasi diterapkan, proses *recovery time* membutuhkan waktu 2.66-11.12 detik dengan rata-rata 5.65 detik. Durasi *recovery* yang lebih panjang tersebut dipengaruhi oleh interval *Collector* yang mengumpulkan sampel data setiap 4 detik serta waktu yang dibutuhkan hingga kondisi jaringan kembali normal. Nilai *recovery time* tercepat sebesar 2.66 detik terjadi ketika penerapan konfigurasi berada di tengah siklus *polling* sehingga kondisi jaringan telah pulih saat pengambilan sampel berikutnya. Sementara itu, keberhasilan proses pemulihan tercermin dari *recovery quality* sebesar 100% pada seluruh percobaan yang menunjukkan bahwa seluruh tindakan pemulihan berhasil mengembalikan status jaringan dari anomali ke kondisi normal.

Hasil pengujian terhadap setiap skenario menunjukkan adanya perbedaan kondisi QoS antara trafik normal, trafik *overload* dengan QoS statis, dan trafik *overload* dengan QoS adaptif. Pada trafik normal, sistem mampu mempertahankan kualitas QoS tanpa melakukan penyesuaian konfigurasi oleh agen PPO. Pada QoS statis, peningkatan beban trafik memicu kondisi kongesti yang berdampak pada penurunan kualitas QoS dengan *max latency* mencapai 131.68 ms, *jitter* 19.54 ms, dan *packet loss* 6 paket. Kondisi tersebut berkaitan dengan penggunaan konfigurasi QoS statis yang tidak dapat menyesuaikan konfigurasi terhadap perubahan beban jaringan. Sebaliknya, pada skenario trafik *overload* dengan QoS adaptif, penyesuaian konfigurasi QoS berdasarkan kondisi jaringan mampu memperbaiki kualitas QoS, dengan *max latency* menurun menjadi 5.44-11.94 ms, *jitter* menjadi 0.28-0.83 ms, dan *packet loss* mencapai 0 paket. Perbedaan tersebut memperlihatkan bahwa kemampuan sistem dalam menyesuaikan konfigurasi QoS terhadap perubahan beban berperan penting dalam mempertahankan kualitas QoS, sehingga sistem *self-healing* menggunakan PPO memberikan

keunggulan dibandingkan konfigurasi QoS statis ketika jaringan mengalami kongesti.

Penelitian ini memiliki beberapa keterbatasan. Implementasi sistem bergantung pada perangkat MikroTik dan RouterOS API yang digunakan, sehingga penerapannya pada perangkat atau platform jaringan lain memerlukan penyesuaian. Selain itu, kelas trafik yang digunakan dalam pengujian terbatas pada *bulk*, *game*, dan *streaming*. Sistem juga belum diuji pada jaringan produksi dengan karakteristik trafik dan kondisi operasional yang lebih beragam, sehingga efektivitas penerapan agen PPO pada kondisi tersebut belum dapat dipastikan.

IV. KESIMPULAN

Integrasi algoritma PPO melalui RouterOS API berhasil mewujudkan sistem *self-healing* dengan QoS adaptif pada infrastruktur MikroTik. Sistem mampu mendeteksi anomali dan merekonfigurasi parameter antrean (*queue type*, *limit-at*, dan *max-limit*) secara *real-time* dengan rata-rata *response time* 22.11 ms, serta memulihkan kondisi jaringan dengan rata-rata *recovery time* 5.65 detik dan *recovery quality* sebesar 100%. Pada kondisi trafik *overload* dengan QoS adaptif, penerapan PPO mampu menurunkan *max latency* dan *jitter* serta menekan *packet loss* hingga 0 paket setelah proses *self-healing* dibandingkan dengan hasil pengujian pada kondisi trafik *overload* dengan QoS statis. Hasil tersebut menunjukkan bahwa sistem yang diusulkan mampu mengatasi keterbatasan pengelolaan QoS melalui penyesuaian konfigurasi QoS secara otomatis terhadap perubahan kondisi jaringan. Dengan demikian, penelitian ini memberikan kontribusi berupa implementasi sistem *self-healing* berbasis *closed-loop control* yang terintegrasi langsung pada RouterOS MikroTik menggunakan PPO untuk mendukung pengelolaan QoS secara adaptif.

DAFTAR PUSTAKA

- [1] Cisco, "Global Networking Trends Report," 2024.
- [2] Huawei, "Communications Network 2030," 2024.
- [3] Juniper Networks, "5 Key Data Center Trends for 2024 The currents driving change in data center networking," 2024.
- [4] MikroTik, "RouterOS Documentation," 2026. Accessed: Jul. 29, 2026. [Online]. Available: <https://help.mikrotik.com/docs/>
- [5] K. Santosh and D. Garry, "Self-Healing Networks: Implementing AI-Powered Mechanisms to Automatically Detect and Resolve Network Issues with Minimal Human Intervention," *International Journal of Scientific Research and Engineering Development*, vol. 6, 2023, [Online]. Available: www.ijssred.com
- [6] S. Tummalpalli, "Self-Healing Network Infrastructure: The Future Of Autonomous Network Management," *International Journal Of Research In Computer Applications And Information Technology*, vol. 8, no. 1, pp. 470–484, Jan. 2025, doi: 10.34218/IJRCAIT_08_01_039.
- [7] T. Rahman and A. S. Ranindrianto, "Analisis Dan Implementasi Qos Dengan Simple Queue Serta Firewall Layer 7 Pada Mikrotik," *Jurnal Mnemonic*, vol. 9, no. 1, pp. 19–27, Feb. 2026, doi: 10.36040/mnemonic.v9i1.16298.
- [8] Fikri Dwilaksono, Henni Endah Wahanani, and Mohammad Idhom, "Implementasi Manajemen Bandwidth Menggunakan Metode Queue Tree dengan PCQ di SMK Negeri 1 Surabaya," *Jurnal Ilmiah Teknik Informatika dan Komunikasi*, vol. 5, no. 2, pp. 58–72, May 2025, doi: 10.55606/juitik.v5i2.1029.
- [9] G. Kim, Y. Kim, and H. Lim, "Deep Reinforcement Learning-Based Routing on Software-Defined Networks," *IEEE Access*, vol. 10, pp. 18121–18133, 2022, doi: 10.1109/ACCESS.2022.3151081.
- [10] F. Naaz, A. Chinmay, and A. Pattanaik, "QoS aware deep reinforcement learning technique for task scheduling in cloud computing," *Discover Computing*, vol. 29, no. 1, p. 185, Mar. 2026, doi: 10.1007/s10791-026-09996-w.
- [11] Y. Effendi, H. Yutanto, and G. Supriyanto, "Development of a Mikrotik API-Based Leaderboard System for Monitoring ISP User Activity," *Jurnal Sistem Informasi Bisnis*, vol. 15, no. 3, pp. 300–309, Jul. 2025, doi: 10.14710/vol15iss3pp300-309.
- [12] A. Kamaldeen, "Closed-Loop Autonomous Network Healing Using Digital Twins and Predictive Reinforcement Learning," 2026. doi: 10.2139/ssrn.5989154.
- [13] N. F. S. de Sousa, M. T. Islam, R. U. Mustafa, D. A. L. Perez, C. E. Rothenberg, and P. H. Gomes, "Machine Learning-Assisted Closed-Control Loops for Beyond 5G Multi-Domain Zero-Touch Networks," *Journal of Network and Systems Management*, vol. 30, no. 3, p. 46, Jul. 2022, doi: 10.1007/s10922-022-09651-x.
- [14] A. E. Kyzrkanov, Y. S. Nurakhov, Z. Otarbay, and D. V. Lebedev, "Log-Driven Proximal Policy Optimization for Adaptive Traffic Control in Software-Defined Networks," *Applied Sciences*, vol. 16, no. 9, p. 4424, May 2026, doi: 10.3390/app16094424.
- [15] R. Zhang, L. Fan, and D. Lu, "Research on Self-Diagnosis and Self-Healing Technologies for Intelligent Fiber Optic Sensing Networks," *Sensors*, vol. 25, no. 6, p. 1641, Mar. 2025, doi: 10.3390/s25061641.