Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)

Journal Homepage: http://journal.lembagakita.org/index.php/jtik

Analisis Performa Jaringan *Software Defined Networking* dengan Algoritma *Random Early Detection*

Yandri Y. Manuputty [1*], Indrastanti R. Widiasari [2]

[1*,2] *Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Kota Salatiga, Provinsi Jawa Tengah, Indonesia.*

a r t i c l e i n f o

Article history:
Received 5 November 2023
Received in revised form
21 December 2023
Accepted 15 March 2024
Available *online* April 2024

DOI:
https://doi.org/10.35870/jtik.v8i2.1745.

Keywords:
Software Defined Network; Random Early Detection; Ryu Controller.

Kata Kunci:
Software Defined Network; Random Early Detection; Ryu Controller.

a b s t r a c t

In the management of SDN networks, the main challenges faced are traffic congestion and network overload. One of the solutions that has been implemented is the Random Early Detection (RED) algorithm. This research aims to analyze the performance of SDN networks with the implementation of the RED algorithm compared to SDN networks without the use of the RED algorithm. The results of the study show that the use of the RED algorithm (Random Early Detection) in SDN networks has a significant impact on Quality of Service (QoS) parameters, such as packet loss, throughput, and average delay. In high load conditions, there is a controlled increase in packet loss with RED as the number of hosts increases. The average packet loss with RED is 1.57% for 4 hosts, whereas without RED, it is only 1.4%. RED also has a positive impact on throughput, with an average throughput of 243.8 kbps for 4 hosts with RED, while without RED, it only reaches 124.3 kbps. Furthermore, the use of RED reduces the average delay in packet delivery. The average delay with RED is 3.8 ms for 4 hosts, whereas without RED, it reaches 5.92 ms.

a b s t r a k

Dalam pengelolaan jaringan SDN, kendala utama yang dihadapi adalah kemacetan lalu lintas dan kelebihan beban. Salah satu solusi yang telah diterapkan adalah algoritma Random Early Detection (RED). Penelitian ini bertujuan menganalisis performa jaringan SDN dengan penerapan algoritma RED dibandingkan dengan jaringan SDN tanpa penggunaan algoritma RED. Hasil penelitian menunjukkan bahwa penggunaan algoritma RED (Random Early Detection) dalam jaringan SDN memiliki pengaruh yang signifikan terhadap parameter Quality of Service (QoS), seperti packet loss, throughput, dan rata-rata delay. Dalam kondisi beban tinggi, terjadi peningkatan packet loss yang lebih terkendali dengan RED saat jumlah host bertambah. Rata-rata packet loss dengan RED adalah 1,57% untuk 4 host, sedangkan tanpa RED hanya 1,4%. RED juga berdampak positif pada throughput, dengan rata-rata throughput 243,8 kbps untuk 4 host dengan RED, sedangkan tanpa RED hanya mencapai 124,3 kbps, selain itu penggunaan RED mengurangi rata-rata delay dalam pengiriman paket. Rata-rata delay dengan RED adalah 3,8 ms untuk 4 host, sementara tanpa RED mencapai 5,92 ms.

Corresponding Author. Email: 672019112@student.uksw.edu [1].

ACM Computing Classification System (CCS)
EBSCOhost
Communication and Mass Media Complete (CMMC)

1. Latar Belakang

Kemajuan Perkembangan teknologi jaringan telah mengarah pada kebutuhan akan solusi yang dapat mengatasi tantangan yang kompleks dalam pengelolaan jaringan yang semakin kompleks dan beragam. Salah satu solusi yang menjanjikan adalah *Software Defined Networking* (SDN), yang memisahkan lapisan kontrol dari lapisan data dalam infrastruktur jaringan. SDN menawarkan fleksibilitas, dinamisme, dan kemudahan pengendalian yang lebih tinggi dibandingkan dengan pendekatan jaringan konvensional [1]. Dalam pengelolaan jaringan SDN, terdapat tantangan dalam menghadapi situasi kelebihan beban atau kemacetan lalu lintas jaringan. Kelebihan beban ini dapat menyebabkan penurunan performa jaringan, penurunan *throughput*, peningkatan *delay*, dan peningkatan *packet loss*. Oleh karena itu, diperlukan mekanisme yang efektif untuk mengatur lalu lintas jaringan dan mengelola situasi kemacetan tersebut [2].

Salah satu solusi yang digunakan dalam pengelolaan lalu lintas jaringan adalah algoritma *Random Early Detection* (RED). Algoritma ini bekerja dengan mendeteksi tanda-tanda awal kemacetan, seperti jumlah paket yang melebihi ambang batas tertentu pada suatu antarmuka jaringan. Ketika kemacetan terdeteksi, algoritma RED akan mengurangi jumlah paket yang dikirimkan ke jaringan, sehingga menghindari penumpukan lalu lintas dan meningkatkan pengiriman paket yang lebih lancar [3]. Algoritma RED telah digunakan secara luas pada jaringan tradisional, masih perlu diketahui apakah penggunaan algoritma RED juga memberikan manfaat yang signifikan pada jaringan SDN. Oleh karena itu, perlu dilakukan penelitian untuk membandingkan performa jaringan SDN dengan penggunaan algoritma RED dan tanpa penggunaan algoritma RED. Dengan demikian, dapat diketahui apakah penggunaan algoritma RED dapat memberikan perbaikan yang signifikan dalam mengelola lalu lintas jaringan pada lingkungan SDN.

Penelitian ini dilakukan lingkungan *virtual box*, yang merupakan lingkungan simulasi yang umum digunakan untuk menguji performa jaringan. Dalam dunia jaringan, *Software Defined Networking* (SDN) telah menjadi paradigma yang *revolusioner*. SDN memisahkan lapisan kendali (*control plane*) dari lapisan penerusan (*data plane*) dalam jaringan. Ini menghasilkan kontrol yang sentralized dan terpusat, dengan perangkat jaringan fisik seperti *switch* dan router berperan sebagai elemen penerusan data yang dikendalikan oleh pengontrol. Konsep ini membawa perubahan mendasar dalam cara jaringan dikelola. Pengontrol SDN, yang berkomunikasi melalui protokol seperti OpenFlow, memberikan visibilitas dan kontrol yang lebih tinggi terhadap seluruh jaringan. Ini membuka pintu untuk pengelolaan jaringan yang lebih fleksibel, dinamis, dan terpusat [4]. Dalam upaya untuk mengoptimalkan kinerja jaringan SDN, algoritma *Random Early Detection* (RED) memainkan peran penting. RED memantau panjang antrian paket data dan, ketika antrian melebihi ambang batas tertentu, membuang paket secara acak dengan probabilitas tertentu. Tujuannya adalah mencegah kemacetan dalam jaringan dengan mengatur lalu lintas secara adaptif. Dengan penggunaan ambang batas min dan max sebagai pedoman, probabilitas pembuangan paket bervariasi sesuai dengan kondisi antrian, meningkat saat mendekati ambang batas maksimum [5].

Protokol *OpenFlow* adalah elemen kunci dalam ekosistem SDN. Berperan sebagai penghubung antara *Controller* dan *Forwarding Plane* dalam jaringan, *OpenFlow* memberikan kontrol sentral atas perangkat jaringan, termasuk switch dan router. Tradisionalnya, perangkat ini hanya meneruskan paket tanpa pemahaman tentang jenis protokol data, namun dengan *OpenFlow* dapat mengatur routing dan mengendalikan aliran paket melalui Controller. Ini membawa fleksibilitas dan adaptabilitas tinggi dalam mengatur kebijakan jaringan [6].

Sementara itu, *Ryu Controller* adalah *framework* pengontrol jaringan yang berbasis *Python*, dirancang untuk mendukung pengembangan aplikasi pengontrol jaringan yang mengintegrasikan protokol *OpenFlow*. Ryu menawarkan antarmuka yang mudah digunakan, integrasi yang kuat dengan perpustakaan *Python* yang populer, dan dukungan untuk berbagai versi protokol *OpenFlow*. Dengan fitur-fitur dan fleksibilitas yang dimilikinya, Ryu memungkinkan pengguna untuk secara efektif mengontrol jaringan SDN, mengimplementasikan algoritma dan kebijakan jaringan yang sesuai dengan kebutuhan, serta memantau dan mengelola operasi jaringan secara efisien [7].

Penelitian yang dilakukan oleh Ramadhan (2021) tentang penerapan gateway balancing pada jaringan *Software Defined Network* (SDN) dengan menggunakan controller OpenDayLight. Penelitian ini mengimplementasikan dua algoritma scheduling, yaitu *Ant Colony Optimization* dan *Random Early Detection*. Penelitian ini fokus pada pengujian performansi trafik dengan streaming UDP Flows. Hasil pengujian menunjukkan bahwa gateway balancing dengan algoritma Ant Colony Optimization memberikan kecepatan transmisi data yang lebih baik dibandingkan dengan algoritma *Random Early Detection*, dengan perbedaan *delay* sebesar 4.46% namun, algoritma *Random Early Detection* menghasilkan performansi yang lebih baik dalam hal integritas data, dengan perbedaan sebesar 3.55% pada *throughput* dan 5.85% pada *packet loss* yang dihasilkan [8].

Selanjutnya penelitian yang dilakukan oleh Bandhaso (2022) membahas penggunaan metode RED (*Random Early Detection*) pada jaringan TCP/IP untuk meningkatkan kinerja QoS (*Quality of Service*). RED adalah algoritma manajemen antrian paket data jaringan yang memantau ukuran rata-rata antrian data sebelum memasuki router dan secara probabilistik membuang paket untuk menghindari kemacetan. Metode ini menggunakan parameter min thresh dan max thresh untuk mengelola antrian dan menghindari kelebihan beban. Hasil pengujian menunjukkan bahwa metode RED efektif dalam menangani kemacetan jaringan dengan cara mendrop paket. Ketika ditambahkan lebih banyak klien ke jaringan, nilai *throughput* mengalami penurunan sekitar 1,59%, dan *delay* mengalami peningkatan sekitar 2,85% meskipun ada penurunan kinerja seiring dengan peningkatan jumlah klien, metode RED masih memberikan kinerja yang lebih baik dibandingkan dengan metode FIFO [3].

Dari penelitian-penelitian sebelumnya, dapat diketahui bahwa meskipun telah ada penelitian yang mempertimbangkan penggunaan algoritma RED dalam konteks jaringan SDN, seperti penelitian oleh Ramadhan (2021) dan Bandhaso (2022), namun keduanya memiliki cakupan yang terbatas. Penelitian Ramadhan lebih berfokus pada penerapan *gateway balancing* dengan menggunakan algoritma RED dan *Ant Colony Optimization*, sedangkan penelitian Bandhaso lebih berfokus pada penggunaan RED dalam jaringan TCP/IP untuk meningkatkan *throughput* dan mengurangi *delay*. Karena itu, masih diperlukan penelitian yang lebih mendalam untuk mengeksplorasi penggunaan algoritma RED secara menyeluruh dalam pengaturan lalu lintas jaringan SDN. Penelitian ini akan memperluas cakupan penelitian sebelumnya dengan fokus pada perbandingan performa jaringan SDN dengan dan tanpa penggunaan algoritma RED, terutama dalam hal *packet loss*, *throughput*, dan rata-rata *delay*. Harapannya, penelitian ini akan memberikan pemahaman yang lebih lengkap dan mendalam tentang manfaat dan dampak penggunaan algoritma RED dalam mengelola lalu lintas jaringan SDN.

2. Metode Penelitian

Dalam melakukan penelitian, terdapat beberapa proses dan tahapan yang dilaksanakan dari awal sampai akhir untuk mencapai tujuan penelitian dan menghasilkan kesimpulan yang akurat. Proses dan tahapan penelitian dilakukan secara berurutan dan berkesinambungan seperti ditunjukkan pada gambar 1.

Gambar 1. Tahapan Penelitian

Dalam tahap awal, studi literatur digunakan untuk mengumpulkan referensi yang relevan dengan topik tugas akhir, memperoleh pemahaman yang kuat tentang topik penelitian yang akan dirancang, dan memberikan dasar untuk perancangan dan implementasi selanjutnya. Kemudian, pada tahap perancangan dan implementasi, langkah-langkah berikutnya adalah instalasi dan konfigurasi software yang diperlukan, perancangan topologi jaringan SDN, dan implementasi algoritma RED. Semua langkah ini didasarkan pada temuan dari studi literatur sebelumnya. Setelah tahap implementasi selesai,

langkah selanjutnya adalah melakukan pengujian. Pengujian ini dilakukan untuk mengevaluasi performa jaringan SDN yang telah diterapkan algoritma RED. Parameter QoS seperti *throughput*, *delay*, dan *packet loss* diukur dan dievaluasi. Terakhir, hasil dari pengujian dianalisis untuk membandingkan kinerja jaringan SDN dengan dan tanpa penerapan algoritma RED. Kesimpulan dibuat berdasarkan temuan yang diperoleh dari analisis hasil pengujian penggunaan RED dan tanpa penggunaan RED.

3. Hasil dan Pembahasan

Alat dan Bahan Penelitian

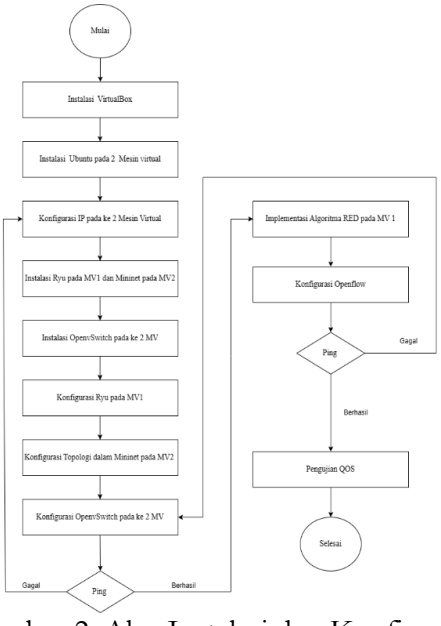
Dalam penelitian ini menggunakan perangkat keras satu buah laptop dengan RAM 8 GB dan perangkat lunak untuk mengimplementasikan dan menganalisis performa jaringan SDN. Adapun kebutuhan perangkat lunak dalam penelitian ini, seperti yang ditunjukkan pada tabel 1.

Tabel 1. Perangkat Lunak

Perangkat Lunak	Keterangan
VirtualBox	Platform virtualisasi yang digunakan untuk menjalankan mesin virtual.
Ubuntu	Sistem operasi Linux yang digunakan pada mesin virtual untuk menjalankan perangkat lunak jaringan.
RYU	Pengontrol SDN berbasis Python yang digunakan untuk mengatur operasi jaringan.
Mininet	Platform simulasi jaringan yang digunakan untuk membuat topologi jaringan virtual dan menguji performa jaringan SDN.
OpenvSwitch	Switch virtual yang mendukung protocol OpenFlow dan digunakan untuk menghubungkan dan mengatur aliran paket dalam jaringan SDN.
Wireshark	Untuk menganalisa kinerja Jaringan beserta protokol yang digunakan.
Microsoft Excel	Untuk mengolah data dan membuat grafik hasil analisis.

Alur Instalasi dan Konfigurasi

Alur instalasi dan konfigurasi sistem digunakan untuk memudahkan dalam melakukan instalasi dan konfigurasi serta implementasi algoritma RED. Pada Gambar 2 menggambarkan langkah-langkah proses instalasi dan konfigurasi jaringan SDN dengan algoritma RED. Awalnya, VirtualBox diinstal sebagai platform virtualisasi, diikuti oleh instalasi sistem operasi Ubuntu pada dua mesin virtual yang berfungsi sebagai host untuk Ryu dan Mininet. Konfigurasi alamat IP diterapkan pada kedua mesin virtual untuk memastikan konektivitas yang tepat. Setelah itu, perangkat lunak Ryu dan Mininet diinstal pada masing-masing mesin virtual, dan perangkat lunak *OpenvSwitch* di instal pada keduanya untuk menyediakan switch virtual dalam jaringan SDN. Konfigurasi Ryu dilakukan pada Mesin Virtual 1 (MV1), sementara topologi jaringan menggunakan Mininet dikonfigurasi pada Mesin Virtual 2 (MV2), termasuk konfigurasi *OpenvSwitch*. Pengujian konektivitas dilakukan dengan mengirimkan ping antara mesin virtual, dan jika berhasil, implementasi algoritma RED dimulai pada MV1.

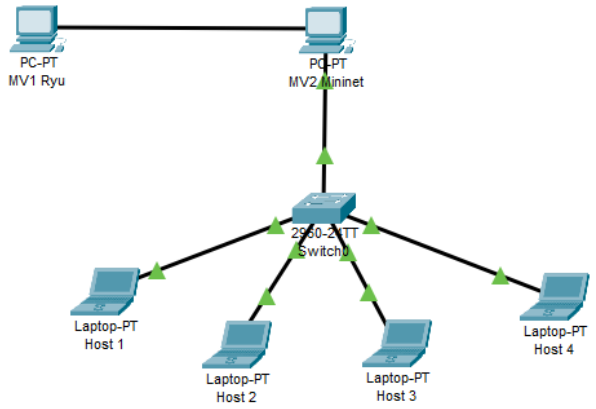


Gambar 2. Alur Instalasi dan Konfigurasi

Konfigurasi OpenFlow dilakukan untuk mengatur komunikasi antara *controller* (Ryu) dan *switch* (*OpenvSwitch*). Pengujian konektivitas diperiksa kembali setelah konfigurasi *OpenvSwitch* dan *OpenFlow* selesai. Tahapan terakhir adalah pengujian QoS untuk menilai performa jaringan SDN dengan algoritma RED, dengan parameter *throughput*, *delay*, dan *packet loss* diukur. Setelah semua tahapan tersebut selesai, penelitian ini dianggap selesai, dan proses instalasi, konfigurasi, dan pengujian telah dilakukan dengan urutan yang terstruktur, memastikan kelancaran implementasi jaringan SDN dengan algoritma RED.

Topologi Jaringan SDN

Dalam penelitian ini, sebuah topologi jaringan telah dirancang dan diimplementasikan untuk menguji performa jaringan *Software Defined Networking* (SDN) dengan penerapan algoritma *Random Early Detection* (RED). Topologi ini bertujuan untuk mensimulasikan lingkungan jaringan yang realistis, sehingga dapat dilakukan pengujian yang akurat terhadap parameter-parameter kinerja jaringan. Melalui topologi ini, berbagai elemen jaringan seperti *switch*, *host*, *ryu controller*, dan mininet dikonfigurasi secara tepat, menciptakan lingkungan yang representatif untuk mengevaluasi performa dan efektivitas penggunaan algoritma RED dalam jaringan SDN, seperti pada gambar 3.



Gambar 3. Topologi Jaringan

Pada gambar 3, terdapat dua mesin virtual yang digunakan MV1 untuk *Ryu Controller* dan MV2 untuk Mininet. Topologi Mininet terdiri dari 1 *switch* dan beberapa *host*. Pada pengujian pertama, terdapat 2 host yang terhubung ke switch. Pada pengujian kedua, terdapat 3 host yang terhubung, dan pada pengujian ketiga, terdapat 4 host yang terhubung.

Parameter Pengujian RED

Dalam penelitian ini, terdapat beberapa parameter yang digunakan untuk menguji performa jaringan SDN dengan penerapan algoritma *Random Early Detection* (RED). Parameter-parameter ini memiliki peran penting dalam mengatur lalu lintas paket dan mengukur kualitas layanan QoS (*Quality of Service*) dalam jaringan, seperti yang ditunjukkan pada tabel 2.

Tabel 2. Parameter RED

Parameter	Value
Durasi Pengujian	30 detik
Protokol	OpenFlow
Minimum Threshold	5000
Maximal Threshold	10000
Minimum Probability	0,02
Maksimal Probability	0,1

Dalam pengujian ini, terdapat beberapa parameter yang ditetapkan untuk mengukur performa jaringan SDN dengan algoritma RED. Pertama, durasi pengujian ditetapkan selama 30 detik, memberikan waktu yang cukup untuk mengamati performa jaringan dan parameter QoS yang diukur. Selanjutnya, protokol *OpenFlow* digunakan dalam pengujian ini untuk mengatur aliran paket dan menerapkan algoritma RED pada jaringan SDN yang telah dirancang.

Dalam mengatur algoritma RED, terdapat *threshold* minimal yang ditetapkan pada nilai 5000. Jika jumlah paket dalam buffer mencapai atau melebihi nilai ini, algoritma RED akan mulai mengurangi *throughput* atau menolak paket dengan probabilitas tertentu. Sebaliknya, threshold maksimal ditetapkan pada nilai 10000. Ketika jumlah paket dalam buffer melebihi nilai ini, algoritma RED akan meningkatkan probabilitas penolakan paket untuk mengurangi *congestion*.

Selanjutnya, terdapat juga parameter *probability* minimal yang ditetapkan pada nilai 0,02 atau 2%. Probabilitas ini mengindikasikan persentase paket yang akan ditolak saat ambang batas tercapai. Di sisi lain, probability maksimal ditetapkan pada nilai 0,1 atau 10%. Probabilitas ini menunjukkan persentase paket yang akan ditolak saat buffer mencapai threshold maksimal. Dengan mengatur parameter-parameter ini, pengujian dapat dilakukan untuk mengamati bagaimana performa jaringan SDN dengan penggunaan algoritma RED.

Hasil Pengujian dan Analisis

Pada bagian ini, akan dijelaskan hasil dari pengujian yang dilakukan untuk mengevaluasi performa jaringan SDN dengan penerapan algoritma RED dan tanpa algoritma RED. Berdasarkan 3 kali pengujian dengan menambahkan 1 host pada setiap pengujian dengan algoritma RED dan tanpa algoritma RED yang telah dilakukan, berikut adalah analisis dan temuan yang

didapatkan untuk parameter-parameter QoS seperti *throughput*, *delay*, dan *packet loss*. Hasil pengujian dirangkum dalam bentuk tabel seperti yang ditunjukkan pada tabel 3 dan tabel 4.

Tabel 3. Hasil Pengujian Jaringan SDN dengan RED

Jumlah Host	*Packet loss* (%)	*Throughput* (kbps)	Rata-rata *Delay* (ms)
2	1,51	277,5	3,23
3	1,62	258	3,65
4	1,57	243,8	3,8

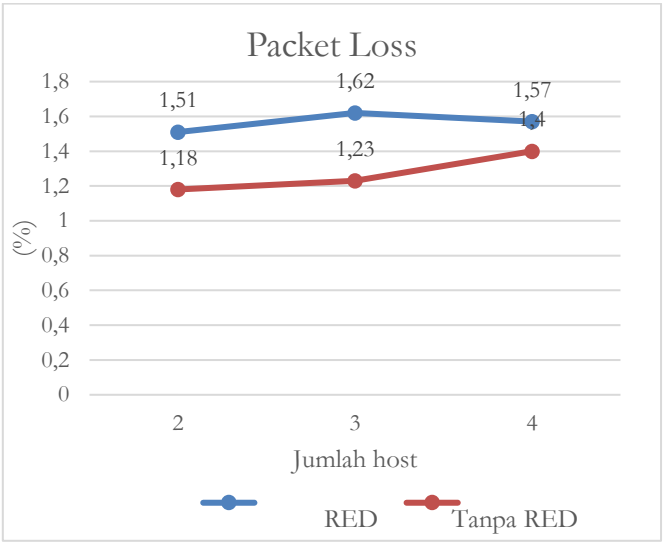
Tabel 4. Hasil Pengujian Jaringan SDN tanpa RED

Jumlah Host	*Packet loss* (%)	*Throughput* (kbps)	Rata-rata *Delay* (ms)
2	1,18	179	5,03
3	1,23	143,8	5,89
4	1,4	124,3	5,92

Hasil pengujian yang ditampilkan dalam tabel 3 dan tabel 4 berisi total perhitungan dari tiga parameter yang menjadi tolak ukur dalam menentukan performa kinerja jaringan SDN, yaitu *Packet loss* yang terdapat pada kolom kedua, *Throughput* pada kolom ketiga, dan Rata-rata *delay* yang ditunjukkan pada kolom keempat. Baris ke-1 menunjukkan *packet loss*, *throughput*, dan rata-rata *delay* pada pengujian dengan menggunakan 2 host, sedangkan baris ke-2 menunjukkan hasil pengujian yang dilakukan pada 3 host, dan baris ke-3 menampilkan hasil pengujian yang dilakukan dengan 4 host. Data pada tabel yang didapatkan dari pengujian kemudian diolah dan ditampilkan dalam bentuk grafik untuk menunjukkan secara spesifik performa dari setiap parameter.

Packet loss

Grafik pada gambar 4 menunjukkan nilai *packet loss* bahwa penggunaan RED cenderung menghasilkan tingkat *packet loss* yang lebih tinggi dibandingkan dengan penggunaan tanpa RED. Dalam pengujian dengan 2 host, tingkat *packet loss* dengan RED adalah 1,51%, sedangkan tanpa RED hanya 1,18%, meskipun penggunaan RED menyebabkan tingkat *packet loss* lebih tinggi, hal ini tidak selalu berlaku saat jumlah host bertambah. Ketika jumlah host meningkat menjadi 3, pengujian dengan RED memiliki tingkat *packet loss* sebesar 1,62%, sementara tanpa RED mencapai 1,23%. Terlihat bahwa tingkat *packet loss* tanpa RED juga meningkat, tetapi peningkatannya tidak signifikan seperti halnya penggunaan RED. Pada pengujian dengan 4 host, tingkat *packet loss* dengan RED adalah 1,57%, sedangkan tanpa RED mencapai 1,4%. Seperti sebelumnya, tingkat *packet loss* tanpa RED juga meningkat, tetapi perbedaannya tidak sebesar ketika menggunakan RED.



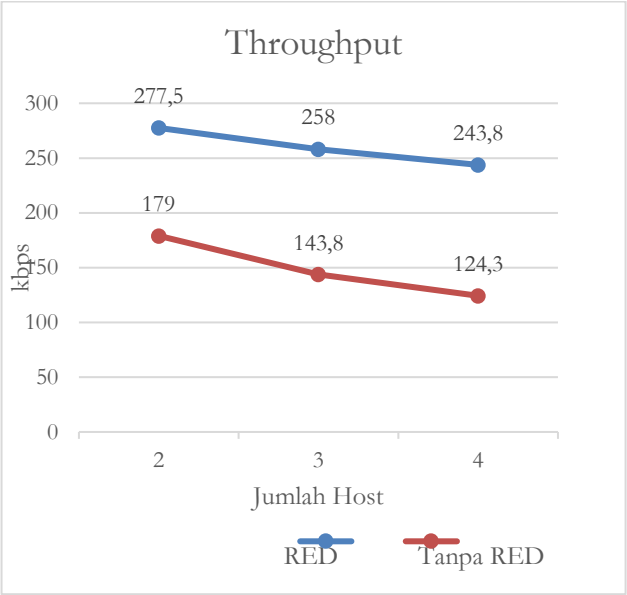
Gamber 4. Grafik Uji *Packet loss*

Penggunaan RED cenderung menghasilkan tingkat *packet loss* yang lebih tinggi dibandingkan dengan penggunaan tanpa RED, namun saat jumlah host bertambah, pengujian tanpa RED menunjukkan peningkatan yang konsisten dalam tingkat *packet loss*. Hal ini menunjukkan bahwa tanpa RED, kemungkinan kehilangan paket akan semakin tinggi seiring bertambahnya beban jaringan, namun dengan penggunaan RED, tingkat *packet loss* tidak selalu meningkat secara proporsional dengan penambahan jumlah *host*.

Throughput

Grafik pada gambar 5, terlihat bahwa penggunaan RED menghasilkan *throughput* yang lebih tinggi dibandingkan dengan penggunaan tanpa RED pada semua jumlah host yang diuji. Pada pengujian dengan 2 host, *throughput* dengan RED mencapai 277,5 kbps, sedangkan tanpa RED hanya mencapai 179 kbps. Hal ini menunjukkan bahwa penggunaan RED dapat meningkatkan *throughput* jaringan secara signifikan. Pada pengujian dengan 3 host, *throughput* dengan RED adalah 258 kbps, sedangkan tanpa RED hanya

mencapai 143,8 kbps.

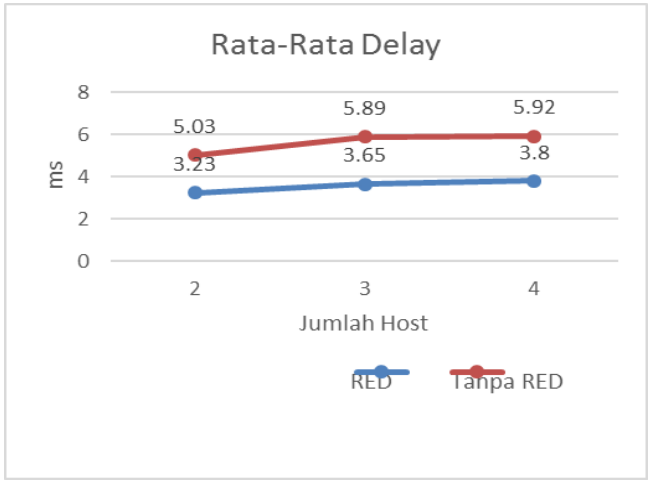


Gamber 5. Grafik Uji *Throughput*

Penggunaan RED kembali memberikan peningkatan yang cukup besar dalam *throughput*. Pada pengujian dengan 4 *host*, *throughput* dengan RED adalah 243,8 kbps, sedangkan tanpa RED hanya mencapai 124,3 kbps. Seperti sebelumnya, penggunaan RED terbukti menghasilkan *throughput* yang jauh lebih tinggi. Nilai ini menunjukkan bahwa penggunaan RED dalam jaringan SDN dapat memberikan peningkatan signifikan dalam *throughput*. RED membantu mengatur aliran paket dengan lebih efisien dan menghindari kemacetan jaringan. Dengan demikian, *throughput* jaringan dapat ditingkatkan, memungkinkan transfer data yang lebih cepat dan efektif.

Rata-Rata Delay

Dalam grafik pada gambar 6, menunjukkan bahwa penggunaan RED menghasilkan rata-rata *delay* yang lebih rendah dibandingkan dengan penggunaan tanpa RED pada semua jumlah host yang diuji. Pada pengujian dengan 2 host, rata-rata *delay* dengan RED adalah 3,23 ms, sedangkan tanpa RED mencapai 5,03 ms. Hal ini menunjukkan bahwa penggunaan RED dapat mengurangi waktu *delay* dalam pengiriman paket secara signifikan. Pada pengujian dengan 3 host, rata-rata *delay* dengan RED adalah 3,65 ms, sedangkan tanpa RED mencapai 5,89 ms.



Gamber 6. Grafik Uji Rata-Rata *Delay*

Penggunaan RED kembali memberikan penurunan yang cukup besar dalam rata-rata *delay*. Pada pengujian dengan 4 host, rata-rata *delay* dengan RED adalah 3,8 ms, sedangkan tanpa RED mencapai 5,92 ms. Seperti sebelumnya, penggunaan RED terbukti menghasilkan rata-rata *delay* yang lebih rendah. Hal ini menunjukkan bahwa penggunaan RED dalam jaringan SDN dapat membantu mengurangi rata-rata *delay* dalam pengiriman paket.

4. Kesimpulan dan Saran

Berdasarkan analisis data yang telah dilakukan, dapat disimpulkan bahwa penggunaan algoritma RED (*Random Early Detection*) dalam jaringan SDN memiliki pengaruh yang signifikan terhadap parameter performa jaringan. Dalam pengujian dengan 2 *host*, penggunaan RED menghasilkan tingkat *packet loss* sebesar 1,51%, sedangkan tanpa RED hanya 1,18%. Saat jumlah host meningkat menjadi 3, tingkat *packet loss* dengan RED adalah 1,62%, sementara tanpa RED mencapai 1,23%. Ketika jumlah *host* mencapai 4, tingkat *packet loss* dengan RED adalah 1,57%, sedangkan tanpa RED mencapai 1,4%. Hasil ini menunjukkan bahwa penggunaan RED dapat membantu mengontrol peningkatan *packet loss* yang lebih signifikan ketika jumlah host bertambah. Pentingnya penggunaan RED juga terlihat dalam peningkatan *throughput* jaringan. Pada pengujian dengan 2 *host*, *throughput* dengan RED mencapai 277,5 kbps, sementara tanpa RED hanya mencapai 179 kbps. Pada pengujian dengan 3 *host*, *throughput* dengan RED adalah 258 kbps, sedangkan tanpa RED hanya mencapai 143,8 kbps. Pada pengujian dengan 4 *host*,

throughput dengan RED adalah 243,8 kbps, sedangkan tanpa RED hanya mencapai 124,3 kbps. Hal ini menunjukkan bahwa penggunaan RED secara konsisten membantu meningkatkan efisiensi pengiriman data melalui jaringan SDN.

Penggunaan RED juga berdampak positif pada rata-rata *delay*. Pada pengujian dengan 2 *host*, rata-rata *delay* dengan RED adalah 3,23 ms, sedangkan tanpa RED mencapai 5,03 ms. Dalam pengujian dengan 3 host, rata-rata *delay* dengan RED adalah 3,65 ms, sedangkan tanpa RED mencapai 5,89 ms. Dalam pengujian dengan 4 *host*, rata-rata *delay* dengan RED adalah 3,8 ms, sedangkan tanpa RED mencapai 5,92 ms. Hasil pengujian menunjukkan bahwa penggunaan RED dapat mengurangi waktu *delay* dalam pengiriman paket, meningkatkan respons jaringan secara keseluruhan. Perlu diketahui bahwa penggunaan RED tidak selalu mengakibatkan peningkatan *packet loss* ketika penambahan jumlah host. Tingkat *packet loss* dengan RED bisa lebih tinggi daripada tanpa RED, perbedaannya tidak sebesar ketika menggunakan tanpa RED. Hal ini menunjukkan bahwa penggunaan RED secara adaptif dapat membantu menjaga tingkat *packet loss* pada tingkat yang dapat diterima meskipun ada peningkatan jumlah host dalam jaringan.

Saran untuk penelitian selanjutnya adalah untuk terus memperdalam pemahaman tentang konfigurasi dan penyesuaian parameter dalam algoritma RED. Penelitian lebih lanjut juga dapat dilakukan untuk membandingkan efektivitas algoritma RED dengan algoritma pengendalian kemacetan lainnya dalam lingkungan jaringan SDN yang lebih kompleks. Penting untuk terus menggali pengetahuan dan pemahaman tentang penggunaan algoritma RED dalam jaringan SDN agar dapat memaksimalkan manfaatnya dalam meningkatkan performa dan kualitas layanan jaringan.

5. Daftar Pustaka

[1] Rahmawan, A. D., Syaifuddin, S., & Risqiwati, D. (2020). Analisa Performansi Controller Pada Arsitektur Jaringan Software Defined Network (SDN). *Jurnal Repositor*, *2*(12). DOI: https://doi.org/10.22219/repositor.v2i12.31846.

[2] Fauzan, R., Ikhwan, S., & Ginting, J. G. A. (2020). Analysis Performance in Software Defined Network (Sdn) Using Aruba Van Controller. *Jurnal Informatika: Jurnal Pengembangan IT*, *5*(3), 64-69. DOI: https://doi.org/10.30591/jpit.v5i3.1937.

[3] Bandhaso, C. P., & Sutanto, Y. (2022). Analisis dan Implementasi Metode Random Early Detection (RED) Pada Jaringan TCP/IP. *Respati*, *17*(2), 51-57. DOI: https://doi.org/10.35842/jtir.v17i2.458.

[4] Gaur, K., Choudhary, P., Yadav, P., Jain, A., & Kumar, P. (2021, March). Software defined networking: a review on architecture, security and applications. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1099, No. 1, p. 012073). IOP Publishing.

[5] Hassan, S. O., Nwaocha, V. O., Rufai, A. U., Odule, T. J., Enem, T. A., Ogundele, L. A., & Usman, S. A. (2022). Random early detection-quadratic linear: an enhanced active queue management algorithm. *Bulletin of Electrical Engineering and Informatics*, *11*(4), 2262-2272. DOI: https://doi.org/10.11591/eei.v11i4.3875.

[6] Pradana, A. M., Purboyo, T. W., & [Last Name, Initial]. (2019). Analisis Load Balancing Pada Jaringan Software Defined Network (SDN) Menggunakan Algoritma Jaringan Syaraf Tiruan (JST). *eProceedings*, 6(1), 1393–1400. Retrieved from. https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/8553%0Ahttps://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/viewFile/8553/8421.

[7] Shodiq, F. F., & Prihanto, A. (2022). Analisis Perbandingan Performansi Kontroler RYU Dan POX Berbasis Software Defined Network (SDN) Pada Routing OSPF. *Journal of Informatics and Computer Science (JINACS)*, *3*(03), 216-223. DOI: https://doi.org/10.26740/jinacs.v3n03.p216-223.

[8] Ramadhan, A. A. (2021). Analisis Perbandingan Gateway Balancing Pada Software Defined Network Dengan Algoritma Scheduling Tambahan. *Jurnal Teknika*, *13*(2), 99-105. DOI: https://doi.org/10.30736/jt.v13i2.681.