Evaluasi Karakteristik Meter Table OpenFlow 1.3 pada Jaringan SDN Berbasis MikroTik RB750GR3

Evaluation of OpenFlow 1.3 Meter Table Characteristics in an SDN Network Based on MikroTik RB750GR3

Azlina Isnaeni Nurul'Ain[1*], Rohmat Gunawan[2], Firmansyah Maulana Sugiartana Nursuwars[3]
Informatika,Universitas Siliwangi
Mugarsari, Tasikmalaya
[1]azlinaisnaeni@gmail.com
[2]rohmatgunawan@unsil.ac.id
[3]firmansyah@unsil.ac.id

[*]Corresponding Author: azlinaisnaeni@gmail.com

Riwayat Artikel
Naskah diterima: 30/06/2026 | Naskah direvisi: 14/08/2026 | Naskah diterbitkan: 31/10/2026

Creative Commons License 4.0 (CC BY –NC)

Abstract*— The meter table is an OpenFlow 1.3 Quality of Service (QoS) mechanism designed to limit traffic rates for individual flows. However, the implementation characteristics of this feature on MikroTik RouterOS (based on OpenFlow 1.3) have not been extensively evaluated. This study analyzes the characteristics of the OpenFlow 1.3 meter table on an SDN setup using MikroTik RB750Gr3 devices and the Ryu Controller. The implementation involved a single controller, three MikroTik devices, and two hosts, utilizing TCP and UDP traffic with rates of 5 and 10 Mbps, burst sizes of 0, 250, and 500 Mbps, and packet transmission tests ranging from 2 to 20 Mbps over a 60-second duration. Evaluation metrics included the onset of packet dropping, throughput, and packet loss. The results indicate an effective throughput of 50% of the configured rate; furthermore, increasing the burst size delayed the onset of dropping and improved average throughput while burst capacity remained available, though it did not alter the effective throughput once the burst was exhausted. These findings reveal discrepancies between the meter table implementation on the MikroTik RB750Gr3 and the specifications defined in the OpenFlow 1.3 standard. Future research could evaluate meter table performance across different RouterOS versions and devices from other vendors.

Intisari— *Meter table* merupakan mekanisme *Quality of Service* (QoS) pada *OpenFlow* 1.3 yang berfungsi membatasi laju trafik (*rate limiting*) pada setiap *flow*. Namun, karakteristik implementasinya pada *MikroTik RouterOS* berbasis *OpenFlow* 1.3 belum banyak dievaluasi. Penelitian ini menganalisis karakteristik *meter table OpenFlow* 1.3 pada SDN berbasis *MikroTik* RB750Gr3 menggunakan *Ryu Controller*. Implementasi dilakukan melalui satu *controller*, tiga perangkat *MikroTik*, dan dua *host* dengan trafik TCP dan UDP, *rate* 5 dan 10 Mbps, *burst size* 0, 250, dan 500 Mbps, serta pengujian pengiriman paket 2 hingga 20 Mbps selama 60 detik. Evaluasi mencakup waktu mulai *dropping*, *throughput*, dan *packet loss*. Hasil menunjukkan *throughput* efektif 50% dari nilai *rate* yang dikonfigurasi, serta penambahan *burst size* dapat menunda *dropping* dan meningkatkan *throughput* rata-rata selama kapasitas *burst* masih tersedia, tetapi tidak mengubah *throughput* efektif setelah *burst* habis. Hasil ini menunjukkan adanya perbedaan karakteristik implementasi *meter table* pada *MikroTik* RB750Gr3 dengan *meter table* pada spesifikasi *OpenFlow* 1.3. Penelitian selanjutnya dapat mengevaluasi *meter table* pada versi *RouterOS* dan perangkat vendor yang berbeda.

Kata Kunci— Meter Table, MikroTik, OpenFlow 1.3, Packet Loss, Software Defined Network, Throughput

I. PENDAHULUAN

Software Defined Network (SDN) merupakan arsitektur jaringan yang memisahkan *control plane* dan *data plane* sehingga memungkinkan pengelolaan jaringan secara terpusat dan mendukung otomasi melalui kemampuan pemrograman jaringan [1]. Pada SDN, *control plane* berfungsi sebagai pusat pengendali berbasis perangkat lunak, sedangkan *data plane* bertugas meneruskan paket sesuai aturan yang diberikan *controller* [2]. Karakteristik tersebut menjadikan SDN banyak diterapkan pada jaringan modern dengan kebutuhan layanan yang beragam, sehingga diperlukan mekanisme pengelolaan trafik yang mampu menjaga kualitas layanan jaringan.

Quality of Service (QoS) merupakan aspek penting dalam menjaga kinerja jaringan melalui pengelolaan *bandwidth*, prioritas trafik, dan pengendalian kemacetan jaringan [3]. Namun, penerapan QoS pada jaringan SDN masih menghadapi tantangan terkait keandalan dan skalabilitas jaringan, terutama pada lingkungan dengan kebutuhan layanan yang beragam [4]. Penelitian oleh [5] menunjukkan bahwa penerapan mekanisme QoS pada

jaringan SDN mampu meningkatkan *throughput* serta menurunkan *packet loss* secara signifikan. Hasil tersebut menunjukkan pentingnya mekanisme QoS dalam menjaga performa jaringan SDN.

Salah satu mekanisme QoS yang diperkenalkan pada *OpenFlow* 1.3 adalah *meter table*. Berbeda dengan mekanisme QoS tradisional seperti *queue* dan *Hierarchical Token Bucket* (HTB) yang bekerja pada tingkat *port* atau *interface*, *meter table* bekerja pada tingkat *flow* sehingga memungkinkan pengendalian trafik secara lebih granular.

Berbagai penelitian telah menunjukkan potensi *meter table* dalam meningkatkan kualitas layanan jaringan SDN. Penelitian [6] menunjukkan bahwa penerapan *meter table* dan DSCP *remarking* mampu mempertahankan trafik prioritas tanpa *packet dropping* yang berlebihan. Selanjutnya, penelitian [5] menunjukkan bahwa penerapan manajemen QoS pada jaringan SDN mampu meningkatkan *throughput* serta menurunkan *delay* dan *packet loss*. Penelitian lain oleh [7] menerapkan *meter table* berbasis *OpenFlow* 1.3 pada lingkungan simulasi dan menunjukkan kemampuan jaringan dalam mendukung transmisi hingga 1 Gbps tanpa *packet loss*. Meskipun demikian, penelitian-penelitian tersebut masih dilakukan pada lingkungan simulasi sehingga karakteristik implementasi *meter table* pada perangkat *forwarding* nyata belum banyak dievaluasi.

MikroTik RouterOS merupakan salah satu perangkat yang mendukung *OpenFlow* 1.3 dan dapat difungsikan sebagai *OpenFlow switch* berbasis perangkat lunak. Namun, implementasi *OpenFlow* pada *MikroTik* memiliki keterbatasan karena belum seluruh fitur *OpenFlow* didukung secara penuh [8]. Selain itu, spesifikasi *OpenFlow* 1.3 menyatakan bahwa implementasi *meter table* dapat bergantung pada kemampuan perangkat yang digunakan sehingga karakteristik pengendalian trafik yang dihasilkan berpotensi berbeda antar perangkat [9].

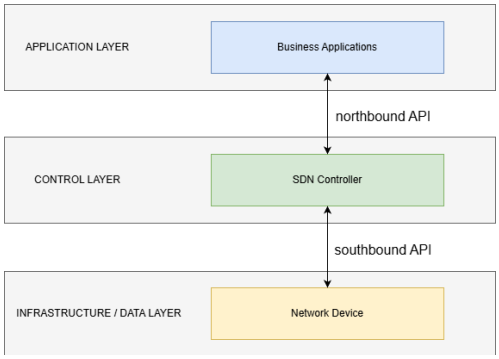
Berdasarkan kondisi tersebut, masih terdapat kesenjangan penelitian terkait implementasi *meter table OpenFlow 1*.3 pada perangkat *MikroTik RouterOS* terutama pada *MikroTik* RB750Gr3, khususnya mengenai kesesuaian perilaku mekanisme *rate limiting* dan *packet dropping* terhadap karakteristik teoritis *OpenFlow* 1.3. Oleh karena itu, diperlukan evaluasi untuk mengetahui sejauh mana implementasi *meter table* pada *MikroTik RouterOS* menghasilkan karakteristik pengendalian trafik yang konsisten dengan spesifikasi *OpenFlow* 1.3.

Penelitian ini berfokus pada evaluasi implementasi *meter table OpenFlow* 1.3 pada jaringan SDN berbasis *MikroTik*. Evaluasi dilakukan untuk menganalisis karakteristik pembatasan laju trafik pada kondisi di bawah, sama dengan, dan di atas nilai *rate* yang dikonfigurasi, serta mengukur performanya berdasarkan parameter *throughput* dan *packet loss* menggunakan Iperf3. Hasil penelitian diharapkan dapat memberikan gambaran mengenai kesesuaian karakteristik *meter table* pada *MikroTik* terhadap perilaku teoritis *OpenFlow* 1.3 serta menjadi referensi bagi pengembangan implementasi QoS pada jaringan SDN berbasis perangkat *MikroTik RouterOS*.

II. LANDASAN TEORI

A. *Software Defined Network (SDN)*

Software Defined Network (SDN) merupakan arsitektur jaringan yang memisahkan sistem pengendalian jaringan dari perangkat *forwarding*, sehingga pengelolaan jaringan dapat dilakukan secara terpusat [10]. Pemisahan antara *control plane* dan *data plane* ini memberikan fleksibilitas yang lebih tinggi dalam manajemen jaringan serta memudahkan integrasi berbagai perangkat karena seluruh kebijakan jaringan dikendalikan oleh satu entitas pusat [11]. Arsitektur SDN ditunjukkan pada Gambar 1. yang terdiri atas tiga lapisan utama, yaitu *application plane*, *control plane*, dan *data plane*.



Gambar. 1. Arsitektur Software Defined Network (SDN)

Application plane merupakan lapisan yang berisi berbagai aplikasi dan layanan jaringan untuk mendukung kebutuhan operasional, seperti *firewall*, *load balancing*, dan *intrusion detection system* [13]. *Control plane* berfungsi sebagai pusat pengendali yang bertugas mengelola kebijakan jaringan dan mengambil keputusan berdasarkan informasi global yang dimiliki mengenai perangkat *forwarding* [14]. Pada penelitian ini, *controller* yang digunakan adalah *Ryu Controller* yang dikembangkan menggunakan bahasa pemrograman *Python*.

Data plane atau *infrastructure layer* merupakan lapisan yang bertanggung jawab melakukan *forwarding* paket sesuai aturan yang diberikan oleh *controller*. Komunikasi antara *control plane* dan *data plane* dilakukan melalui *southbound* API, salah satunya menggunakan protokol *OpenFlow*. Adapun protokol yang digunakan dalam penelitian ini adalah *OpenFlow* versi 1.3. Selain itu, komunikasi antara *application plane* dan *control plane* dilakukan melalui *northbound* API, sedangkan komunikasi antara *control plane* dan *data plane* dilakukan melalui *southbound* API.

B. *OpenFlow 1.3*

OpenFlow merupakan protokol komunikasi yang berfungsi sebagai *southbound* API antara *control plane* dan *data plane*. Melalui *OpenFlow*, *controller* dapat mengelola aturan *forwarding* pada perangkat jaringan serta menentukan jalur pengiriman paket sesuai kebijakan yang ditetapkan [15]. Dalam penelitian ini, *OpenFlow* versi 1.3 digunakan untuk menghubungkan *Ryu Controller* dengan perangkat *MikroTik RouterOS* karena versi tersebut mendukung implementasi meter table sebagai mekanisme pengendalian laju trafik.

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2026.102.496

Meskipun demikian, implementasi *OpenFlow* 1.3 pada *MikroTik* masih memiliki keterbatasan karena berjalan secara *software-based*, sehingga pemrosesan paket dilakukan oleh CPU dan belum seluruh fitur *OpenFlow* dapat didukung secara optimal. Kondisi ini memungkinkan karakteristik dan performa fitur *OpenFlow*, termasuk *meter table*, berbeda bergantung pada perangkat yang digunakan. Pada *OpenFlow* 1.3, *meter table* diperkenalkan sebagai mekanisme untuk mengendalikan dan membatasi laju trafik pada setiap *flow* berdasarkan parameter yang ditentukan [9].

C. Meter Table

Meter table merupakan salah satu fitur yang diperkenalkan pada *OpenFlow* versi 1.3 sebagai mekanisme pengendalian laju trafik pada setiap *flow*. Implementasi *meter table* dilakukan melalui *flow rule* yang dikirimkan *controller* ke *switch*, yang berisi *match field* dan instruksi untuk menerapkan *meter table* sebelum paket diteruskan ke tujuan. Dengan demikian, setiap paket yang memenuhi kriteria *flow rule* akan terlebih dahulu diproses oleh *meter table*. *Meter table* terdiri dari sejumlah *meter entry* dengan komponen utama yang ditunjukkan pada Gambar 2. [9].

meter identifier	*meter bands*	*counters*

Gambar. 2. Komponen utama *meter entry* pada *meter table*

Pada setiap *meter entry*, *meter identifier* berfungsi sebagai identitas unik *meter table*, *meter bands* digunakan untuk menentukan batas laju trafik (*rate*) dan tindakan yang diterapkan ketika trafik melebihi batas tersebut, sedangkan *counters* digunakan untuk mencatat statistik paket yang diproses oleh *meter table* [9]. Struktur utama *meter bands* ditunjukkan pada Gambar 3.

band type	*rate*	*counters*	*type of specific argument*

Gambar. 3. Komponen utama *meter bands* pada *meter entry*

Menurut [9], *meter bands* terdiri atas *band type*, *rate*, *counters*, dan *type specific argument*. *Band type* menentukan tindakan yang diterapkan terhadap paket, sedangkan *rate* menentukan batas laju trafik yang menjadi acuan aktivasi suatu *band*. Selain itu, *meter band* juga mendukung parameter *burst size* sebagai toleransi terhadap lonjakan trafik sesaat [9]. Nilai *burst size* bergantung pada *meter flag* yang digunakan, yaitu dinyatakan dalam *kilobit* pada *OFPMF_KBPS* dan dalam jumlah paket pada *OFPMF_PKTPS* [9]. Dengan mekanisme ini, trafik yang sesaat melebihi nilai *rate* masih dapat diterima hingga batas *burst size* sebelum tindakan pada *meter band* diterapkan.

Karakteristik pengendalian trafik pada *meter table* ditentukan oleh *band type* yang didukung perangkat [9]. Salah satu jenis yang umum digunakan adalah *drop*, yaitu mekanisme yang membuang paket ketika laju trafik melebihi nilai *rate* yang ditetapkan [7]. Dengan demikian, paket akan diteruskan apabila laju trafik berada di bawah atau sama dengan nilai *rate*, sedangkan paket yang melebihi batas tersebut akan dibuang. Selain *drop*, *OpenFlow* 1.3 juga menyediakan DSCP *remark* yang berfungsi mengubah nilai DSCP pada *header* IP untuk mendukung mekanisme QoS berbasis *DiffServ* [9].

D. MikroTik RouterOS

MikroTik RouterOS merupakan perangkat jaringan yang dapat difungsikan sebagai *data plane* pada arsitektur SDN melalui penambahan *OpenFlow agent*. Menurut [8], *RouterOS* mendukung *OpenFlow* versi 1 dan 1.3 sehingga memungkinkan komunikasi antara *controller* dan perangkat *forwarding* dalam lingkungan SDN. Penelitian oleh [16] menunjukkan bahwa *MikroTik RouterBoard RB750* dapat beroperasi sebagai *OpenFlow software-based switch* dan menjalankan fungsi dasar perangkat *forwarding* pada jaringan SDN. Hasil penelitian tersebut menunjukkan bahwa performa *throughput* yang diperoleh relatif serupa dengan implementasi SDN pada simulator, meskipun masih terdapat perbedaan yang cukup signifikan pada parameter *latency*.

E. Quality of Service (QoS)

Quality of Service (QoS) merupakan metode untuk mengukur kemampuan jaringan dalam menyediakan tingkat layanan sesuai kebutuhan aplikasi dan pengguna [17]. Pengukuran QoS dilakukan menggunakan sejumlah parameter performa yang mencerminkan kondisi transmisi data pada jaringan. Dalam penelitian ini, parameter QoS yang digunakan meliputi *throughput* dan *packet loss* karena keduanya dapat merepresentasikan efisiensi serta keandalan jaringan.

Throughput merupakan jumlah data yang berhasil diterima dalam periode waktu tertentu [18]. Nilai *throughput* dipengaruhi oleh berbagai kondisi jaringan, seperti jumlah pengguna yang mengakses jaringan secara bersamaan, kapasitas jaringan, dan kondisi transmisi data [19],[20]. Dalam konteks penelitian ini, penerapan *meter table* berpotensi mempengaruhi *throughput* karena mekanisme *rate limiting* membatasi laju trafik sesuai nilai *rate* yang ditentukan. Nilai *throughput* dapat dihitung menggunakan Persamaan (1) [21].

$$Throughput = \frac{\text{Jumlah data yang dikirim}}{\text{Waktu pengiriman data}} \quad (1)$$

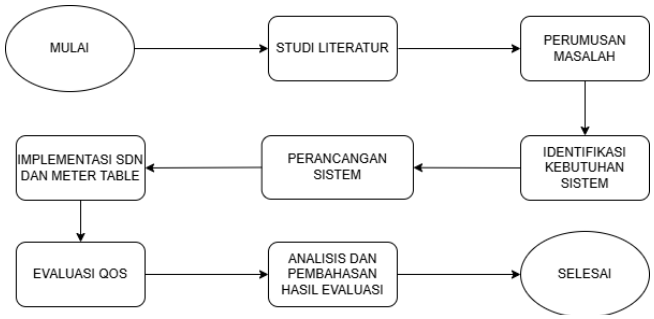
Packet loss merupakan persentase paket yang gagal mencapai tujuan selama proses transmisi data [18]. Kondisi ini umumnya terjadi akibat *congestion* atau keterbatasan kapasitas sistem sehingga paket harus dibuang oleh perangkat jaringan [22]. Pada *meter table* dengan *band type drop*, paket yang melebihi nilai *rate* akan dibuang sehingga berpotensi meningkatkan *packet loss*. Besarnya *packet loss* juga dipengaruhi oleh protokol transport yang digunakan. Pada *Transmission Control Protocol* (TCP), paket yang hilang dapat dikirim ulang melalui mekanisme *retransmission* sehingga tingkat *packet loss* cenderung lebih rendah dibandingkan protokol yang tidak memiliki mekanisme tersebut [23]. Nilai *packet loss* menunjukkan tingkat keandalan jaringan dan dapat dihitung menggunakan Persamaan (2) [21].

$$Packet\ loss = \frac{(\text{Paket dikirim} - \text{Paket diterima})}{\text{Paket yang dikirim}}\ x\ 100\% \quad (2)$$

III. METODOLOGI PENELITIAN

A. Tahapan Penelitian

Penelitian ini dilakukan untuk mengevaluasi karakteristik meter table pada jaringan *Software Defined Network* (SDN) berbasis *MikroTik OpenFlow* 1.3. Tahapan penelitian meliputi studi literatur, perancangan sistem, implementasi jaringan SDN, pengujian *Quality of Service* (QoS), serta analisis hasil pengujian. Alur penelitian secara keseluruhan ditunjukkan pada Gambar 4.



Gambar. 4. Diagram tahap penelitian

B. Studi Literatur

Tahap studi literatur dilakukan dengan menelaah berbagai sumber referensi yang berkaitan dengan penelitian, mencakup konsep *Software Defined Network* (SDN), protokol *OpenFlow* versi 1.3, *meter table*, serta implementasinya pada *MikroTik RouterOS* sebagai perangkat *forwarding*. Selain itu, kajian juga dilakukan terhadap konsep *Quality of Service* (QoS) dan parameter pengukurannya yang meliputi *throughput* serta *packet loss*.

C. Perumusan Masalah

Tahap ini dilakukan untuk mengidentifikasi peluang penelitian serta merumuskan permasalahan dan tujuan penelitian berdasarkan hasil studi literatur yang telah dilakukan. Dari kajian tersebut diketahui bahwa *Software Defined Network* (SDN) banyak digunakan pada jaringan dengan kebutuhan layanan yang beragam, namun masih menghadapi tantangan dalam pengelolaan *Quality of Service* (QoS). Berbagai penelitian sebelumnya telah menerapkan mekanisme manajemen QoS untuk menjaga kualitas layanan jaringan, salah satunya melalui penerapan *meter table* pada *OpenFlow*.

Meskipun demikian, implementasi dan evaluasi *meter table* pada jaringan SDN dengan perangkat *forwarding* berupa *MikroTik RouterOS* berbasis *OpenFlow* 1.3 masih relatif terbatas. Di sisi lain, implementasi *OpenFlow* pada *MikroTik* belum mendukung seluruh fitur secara penuh karena menggunakan arsitektur berbasis perangkat lunak, sehingga berpotensi menghasilkan karakteristik dan performa *meter table* yang berbeda dari perilaku teoritis *OpenFlow* 1.3.

Berdasarkan kondisi tersebut, penelitian ini difokuskan pada analisis karakteristik *meter table* dalam mengendalikan dan membatasi laju trafik pada jaringan SDN berbasis *MikroTik OpenFlow* 1.3. Evaluasi dilakukan melalui pengukuran parameter QoS berupa *throughput* dan *packet loss* untuk mengetahui kesesuaian implementasi *meter table* terhadap karakteristik yang didefinisikan dalam spesifikasi *OpenFlow* 1.3.

D. Identifikasi Kebutuhan Sistem

Pada tahapan ini ditentukan kebutuhan perangkat keras dan perangkat lunak yang mendukung implementasi *Software Defined Network* (SDN). Topologi yang digunakan adalah topologi *Tree* yang terdiri dari satu *controller*, tiga *switch* sebagai *data plane*, dua *host* sebagai perangkat *end device*, sehingga perangkat keras yang dibutuhkan sebagai berikut.

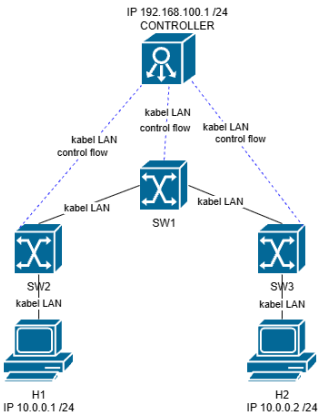
a) PC dengan OS *Ubuntu* 22.04 (*Jammy Jellyfish*) 64-bit RAM 4 GB DDR 4 2666 MHz
b) 3 *MikroTik RB750Gr3* dengan *RouterOS* versi 7.20 sebagai *switch*
c) 2 komputer sebagai *host*
d) 6 kabel LAN

Perangkat lunak yang dibutuhkan sebagai berikut:

a) *Python3* versi 3.10.12
b) *Ryu controller* versi 4.34
c) *Package OpenFlow MikroTik*
d) *Iperf3*

E. Perancangan Sistem

Arsitektur jaringan yang digunakan terdiri atas satu *controller Ryu*, tiga *switch MikroTik RB750Gr3* sebagai *data plane*, dan dua *host* sebagai perangkat *end-user*, serta protokol komunikasi antara *controller* dan *switch* menggunakan *OpenFlow* 1.3. Topologi jaringan yang digunakan ditunjukkan pada Gambar 5.



Gambar. 5. Topologi jaringan

Konfigurasi *meter table* diterapkan pada *switch2* dan *switch3* serta pada dua *flow* berbeda yaitu *flow upload* dan *download* untuk menghindari penerapan mekanisme *metering* secara berulang. *Meter* menggunakan *band type drop* dan *flag* OFPMF_KBPS sehingga nilai dinyatakan dalam satuan kbps. Untuk keperluan penyajian hasil, nilai dikonversi dari kbps menjadi Mbps. Adapun nilai *rate* yang ditetapkan yaitu sebesar 5000 kbps (5 Mbps) dan 10000 kbps (10 Mbps). Pada masing-masing nilai *rate* diterapkan tiga variasi *burst size*, yaitu tanpa *burst size* (0), *burst size* 250000 kbps (250 Mbps) dan *burst size* 500000 kbps (500 Mbps) yang ditunjukkan pada Tabel 3.

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2026.102.496

TABEL III
KONFIGURASI METER TABEL

Konfigurasi	*Flag*	*Meter Type*	*Nilai Rate*	*Nilai Burst Size*
M1	OFPMF KBPS	*Drop*	5000 kbps (5 Mbps)	0
M2	OFPMF _KBPS	*Drop*	5000 kbps (5 Mbps)	250000 kbps (250 Mbps)
M3	OFPMF _KBPS	*Drop*	5000 kbps (5 Mbps)	500000 kbps (500 Mbps)
M4	OFPMF KBPS	*Drop*	10000 kbps (10 Mbps)	0
M5	OFPMF _KBPS	*Drop*	10000 kbps (10 Mbps)	250000 kbps (250 Mbps)
M6	OFPMF _KBPS	*Drop*	10000 kbps (10 Mbps)	500000 kbps (500 Mbps)

Penetapan nilai *rate* digunakan untuk mengevaluasi karakteristik pembatasan laju trafik pada kondisi kapasitas yang berbeda. Sedangkan, penerapan *burst size* didasarkan pada spesifikasi *OpenFlow* 1.3 yang menyatakan *burst size* berfungsi sebagai kapasitas tambahan yang memungkinkan trafik melebihi nilai *rate* untuk sementara waktu sebelum dilakukan pembatasan. Sehingga penggunaan variasi *burst size* sebesar 0 Mbps, 250 Mbps dan 500 Mbps bertujuan untuk mengevaluasi apakah perubahan nilai *burst size* memengaruhi karakteristik pembatasan trafik pada implementasi *meter table OpenFlow* 1.3 di *MikroTik* RB750Gr3.

F. Implementasi Software Defined Network (SDN) dan Meter Table

Tahap implementasi diawali dengan instalasi dan konfigurasi *Ryu Controller* pada *Ubuntu* 22.04 sebagai *control plane*. Selanjutnya, *MikroTik RouterOS* dikonfigurasi agar mendukung *OpenFlow* dan terhubung dengan *controller* sebagai *data plane*. Setelah koneksi berhasil terbentuk, *controller* menginstal *flow rule* dan *meter table* sesuai kebijakan yang telah ditentukan. Tahap berikutnya dilakukan pengujian konektivitas antar *host* untuk memastikan jalur komunikasi berfungsi dengan baik, kemudian dilanjutkan dengan evaluasi performa jaringan melalui pengukuran *throughput* dan *packet loss* menggunakan *Iperf3*.

G. Evaluasi Quality of Service (QoS)

Pada tahap ini dilakukan evaluasi *Quality of Service* (QoS) menggunakan *Iperf3* melalui protokol TCP dan UDP pada kondisi jaringan terkontrol tanpa trafik tambahan. *Iperf3* digunakan untuk menghasilkan sekaligus mengukur trafik jaringan, sehingga dapat digunakan untuk mengevaluasi performa jaringan berdasarkan parameter *throughput* dan *packet loss* [24]. Pengujian dilakukan pada jaringan SDN dengan dan tanpa *meter table* menggunakan durasi pengiriman selama 60 detik untuk memperoleh data yang stabil.

Skenario pengujian dibagi menjadi tiga kondisi, yaitu trafik di bawah *rate*, sama dengan *rate*, dan di atas *rate*. Pada *meter table* dengan *rate* 5 Mbps, trafik yang dikirim meliputi 2 Mbps, 5 Mbps, 8 Mbps, 10 Mbps, 12 Mbps, 15 Mbps, dan 20 Mbps. Sementara itu, pada *meter table* dengan *rate* 10 Mbps, trafik yang dikirim meliputi 2 Mbps, 5 Mbps, 8 Mbps, 10 Mbps, 12 Mbps, 15 Mbps, dan 20 Mbps. Pada setiap skenario dilakukan pengukuran *throughput* dan *packet loss* untuk mengevaluasi karakteristik *meter table* dalam mengendalikan laju trafik.

Pengujian TCP dilakukan menggunakan perintah [25] *iperf3 -c* (IP tujuan) *-b* (bandwidth) *-t* (durasi), sedangkan pengujian UDP menggunakan perintah *iperf3 -c* (IP tujuan) *-u -b* (*bandwidth*) *-t* (durasi). Parameter *-c* digunakan untuk menentukan *host* tujuan, *-b* untuk menentukan besar bandwidth yang dikirimkan, *-t* untuk menentukan durasi pengujian, dan *-u* untuk mengaktifkan mode UDP.

H. Analisis dan Pembahasan Hasil Evaluasi

Tahap ini dilakukan untuk menganalisis dan membahas hasil evaluasi QoS pada setiap skenario pengujian. Analisis difokuskan pada karakteristik *meter table* dalam mengendalikan dan membatasi laju trafik berdasarkan hubungan antara *throughput* dan *packet loss* pada kondisi trafik di bawah *rate*, sama dengan *rate*, dan di atas *rate*.

Pada kondisi trafik di bawah *rate*, sistem diharapkan tidak melakukan pembatasan sehingga *throughput* mengikuti laju trafik yang dikirim. Ketika trafik sama dengan *rate*, *throughput* diharapkan tetap stabil mendekati nilai *rate* yang ditentukan. Sementara itu, pada kondisi trafik di atas *rate*, *meter table* diharapkan membatasi laju trafik sehingga *throughput* tetap mendekati nilai *rate*, dengan kemungkinan terjadinya *packet loss* akibat mekanisme *packet dropping*.

Selain itu, analisis juga mempertimbangkan pengaruh protokol *transport* yang digunakan. Pada pengujian TCP, *packet loss* diharapkan relatif rendah karena adanya mekanisme *retransmission*, sedangkan pada UDP *packet loss* berpotensi meningkat akibat paket yang dibuang oleh *meter table*. Sebagai pembanding, pengujian pada jaringan SDN tanpa *meter table* diharapkan tidak mengalami pembatasan laju trafik sehingga *throughput* mengikuti laju paket yang dikirim dan *packet loss* dapat diminimalkan.

IV. ANALISIS DAN HASIL

A. Implementasi Software Defined Network (SDN) dan Meter Table

Konfigurasi antara *controller* dan perangkat forwarding ditunjukkan pada Gambar 6. Menunjukkan konfigurasi *OpenFlow* pada *MikroTik RB750Gr3* telah menggunakan *OpenFlow* versi 1.3 dan berhasil terhubung dengan *Ryu Controller* melalui alamat 192.168.100.1 pada *port* 6653. Kondisi ini menunjukkan bahwa koneksi antara *control plane* dan *data plane* telah berhasil dibangun sehingga *Ryu Controller* dapat mengelola *meter table* pada perangkat *MikroTik*.

Version	Passive Port	Controllers
1.3		tcp/192.168.100.1/6653

Gambar. 6. Konfigurasi *controller* pada *data plane*

Implementasi *meter table* dilakukan pada setiap arah aliran trafik yang berbeda, yaitu *meter upload* dan *meter download*, serta hanya diterapkan pada *switch* 2 dan *switch*

3, sedangkan *switch* 1 tidak diberikan konfigurasi *meter table*. Adapun skrip *controller* yang digunakan dalam penelitian ini disajikan dalam bentuk *pseudocode* sebagai berikut.

```
Input: Switch DatapathID (DPID), Port Masuk, MAC Address
Output: Meter table terpasang pada switch

IF switch = DPID Switch2 OR switch = DPID Switch3 THEN
  IF port masuk = 1 THEN
    pilih Meter Upload
  ELSE IF port masuk = 2 THEN
    pilih Meter Download
  END IF

  Buat meter upload
    Meter Type: Drop
    Rate: 5000 kbps dan 10000 kbps
    Burst Size: 250000 kbps dan 500000 kbps
    Flags: KBPS dan BURST
  Instal meter upload ke switch

  Buat meter download
    Meter Type: Drop
    Rate: 5000 kbps dan 10000 kbps
    Burst Size: 250000 kbps dan 500000 kbps
    Flags: KBPS dan BURST
  Instal meter download ke switch

  END IF
```

Gambar 7. Menunjukkan penerapan *meter table* yang dicontohkan pada *rate* 5 Mbps sebagai batas maksimum laju trafik yang diizinkan. Parameter *burst* bernilai 0, yang menunjukkan bahwa *burst* tidak digunakan.

```
[{type=drop, rate=5000, burst=0,
[{type=drop, rate=5000, burst=0,
```

Gambar. 7. *Rate* 5 Mbps tanpa *burst size*

Gambar 8. Memperlihatkan contoh konfigurasi dengan *burst size* 250 Mbps pada *rate* 5 Mbps sebagai batas laju trafik yang diizinkan.

```
[{type=drop, rate=5000, burst=250000,
[{type=drop, rate=5000, burst=250000,
```

Gambar. 8. *Rate* 5 Mbps dengan *burst size* 250 Mbps

Gambar 9. Menunjukkan contoh hasil konfigurasi *meter table* dengan nilai *rate* sebesar 10 Mbps sebagai batas maksimum laju trafik yang diizinkan, sedangkan *burst size* sebesar 500 Mbps.

```
[{type=drop, rate=10000, burst=500000,
[{type=drop, rate=10000, burst=500000,
```

Gambar. 9. *Rate* 10 Mbps dengan *burst size* 500 Mbps

Selain itu, konektivitas antar *host* juga dipastikan menggunakan utilitas *ping* dari *host1* ke *host2*, ditunjukkan pada Gambar 10. Seluruh paket yang dikirim berhasil diterima oleh *host2* tanpa *packet loss*, yang menunjukkan bahwa jalur komunikasi antar *host* telah berfungsi dengan baik.

```
Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=34ms TTL=128

Reply from 10.0.0.2: bytes=32 time=345ms TTL=128

Reply from 10.0.0.2: bytes=32 time=1ms TTL=128

Reply from 10.0.0.2: bytes=32 time=1ms TTL=128

Ping statistics for 10.0.0.2:

  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

  Minimum = 1ms, Maximum = 345ms, Average = 95ms
```

Gambar. 10. Tes *ping host1* ke *host2*

B. *Evaluasi Quality of Service (QoS)*

1) *Hasil Pengujian Jaringan SDN Tanpa Meter Table:*

Hasil pengujian pada jaringan SDN tanpa *meter table* pada protokol TCP ditunjukkan pada Tabel 4. Menunjukkan tidak terjadinya pembatasan karena mekanisme *meter table* tidak diterapkan.

TABEL IIV
HASIL PENGUJIAN TANPA METER TCP

Sken.	***Rate* (Mbps)**	**Paket yang dikirim (Mbps)**	**Durasi (sec)**	***Throughput* (Mbps)**	***Loss* (%)**
Di bawah *rate*	0	2	60	2	0%
		5	60	5	0%
		8	60	8	0%
Sama dengan *rate*	0	10	60	10	0%
Di atas *rate*	0	12	60	12	0%
		15	60	15	0%
		20	60	20	0%

Sedangkan hasil pengujian pada jaringan SDN tanpa *meter table* pada protokol UDP ditunjukkan pada Tabel 5. Juga menunjukkan tidak terjadinya pembatasan karena mekanisme *meter table* tidak diterapkan.

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2026.102.496

TABEL V
HASIL PENGUJIAN TANPA METER UDP

Sken.	*Rate* (Mbps)	Paket yang dikirim (Mbps)	Durasi (sec)	*Throughput* (Mbps)	*Loss* (%)
Di bawah *rate*	0	2	60	2	0%
		5	60	5	0%
		8	60	8	0%
Sama dengan *rate*	0	10	60	10	0%
Di atas *rate*	0	12	60	12	0%
		15	60	15	0%
		20	60	20	0%

2) Hasil Pengujian Meter Table dengan Rate 5 Mbps:

Hasil pengujian *meter table* dengan *rate* 5 pada protokol TCP ditunjukkan pada Tabel 6. Hasil pengujian menunjukkan bahwa *meter table* berhasil melakukan pembatasan laju trafik, namun tingkat pembatasan efektif yang dihasilkan hanya mencapai sekitar 50% dari nilai *rate* yang dikonfigurasi. Serta kapasitas *burst size* yang habis meski paket yang dikirim lebih kecil dibanding nilai *burst size*.

TABEL VI
HASIL PENGUJIAN METER RATE 5 MBPS TCP

Burst Size	**Sken.**	***Rate* (Mbps)**	**Paket dikirim (Mbps)**	**Waktu mulai *drop* (s)**	***Throughput* (Mbps)**	***Loss* (%)**
Tanpa *Burst*	Di bawah *rate*	5	2	-	2.00	0
	Sama dengan *rate*	5	5	-	2.43	0
	Di atas *rate*	5	8	-	2.45	0
			10	-	2.43	0
			12	-	2.43	0
			15	-	2.43	0
			20	-	2.45	0
Burst 250	Di bawah *rate*	5	2	-	2.00	0
	Sama dengan *rate*	5	5	43	4.25	0
	Di atas *rate*	5	8	13	3.51	0
			10	10	3.44	0
			12	8	3.70	0
			15	7	3.67	0
			20	4	3.32	0
Burst 500	Di bawah *rate*	5	2	-	2.00	0
	Sama dengan *rate*	5	5	-	5.00	0
	Di atas *rate*	5	8	22	4.44	0
			10	15	4.25	0
			12	14	4.60	0
			15	10	4.40	0
			20	8	4.74	0

Sedangkan hasil pengujian *meter table* dengan *rate* 5 Mbps pada protokol UDP ditunjukkan pada Tabel 7. Hasil pengujian menunjukkan perilaku yang serupa dengan protokol TCP, yaitu *meter table* berhasil melakukan pembatasan laju trafik, namun *throughput* efektif tetap mencapai sekitar 50% dari nilai *rate* yang dikonfigurasi. Selain itu, parameter *burst size* meningkatkan *throughput* dengan menunda waktu penerapan mekanisme *packet dropping*.

TABEL VII
HASIL PENGUJIAN METER RATE 5 MBPS UDP

Burst Size	**Sken.**	***Rate* (Mbps)**	**Paket dikirim (Mbps)**	**Waktu mulai *drop* (s)**	***Throughput* (Mbps)**	***Loss* (%)**
Tanpa *Burst*	Di bawah *rate*	5	2	-	2.00	0
	Sama dengan *rate*	5	5	-	2.42	52
	Di atas *rate*		8	-	2.38	70
			10	-	2.38	76
			12	-	2.38	80
			15	-	2.38	84
			20	-	2.37	88
Burst 250	Di bawah *rate*	5	2	-	2.00	0
	Sama dengan *rate*	5	5	50	4.53	9.4
	Di atas *rate*		8	18	3.96	51
			10	13	3.94	61
			12	8	3.56	70
			15	7	3.66	76
			20	6	4.17	79
Burst 500	Di bawah *rate*	5	2	-	2.00	0

Burst Size	Sken.	*Rate* (Mbps)	Paket dikirim (Mbps)	Waktu mulai *drop* (s)	*Throughput* (Mbps)	*Loss* (%)
	Sama dengan *rate*	5	5	-	5.00	0
	Di atas *rate*	5	8	21	4.25	47
			10	25	4.90	51
			12	12	4.21	65
			15	13	4.18	72
			20	10	4.13	79

3) Hasil Pengujian Meter Table dengan Rate 10 Mbps:

Hasil pengujian *meter table* dengan *rate* 10 Mbps pada protokol TCP ditunjukkan pada Tabel 8. Hasil pengujian menunjukkan pola yang konsisten dengan konfigurasi *rate* 5 Mbps, yaitu *meter table* berhasil membatasi laju trafik, namun *throughput* efektif hanya mencapai sekitar 50% dari nilai *rate* yang dikonfigurasi, serta penambahan burst size meningkatkan *throughput* dengan menunda penerapan mekanisme *packet dropping*, sehingga *throughput* rata-rata menjadi lebih mendekati nilai *rate*.

TABEL VIII
HASIL PENGUJIAN METER RATE 10 MBPS TCP

Burst Size	Sken.	*Rate* (Mbps)	Paket dikirim (Mbps)	Waktu mulai *drop* (s)	*Throughput* (Mbps)	*Loss* (%)
Tanpa *Burst*	Di bawah *rate*	10	2	-	2.00	0
			5	-	5.00	0
			8	-	4.94	0
	Sama dengan *rate*	10	10	-	4.92	0
	Di atas *rate*	10	12	-	4.93	0
			15	-	4.95	0
			20	-	4.96	0
Burst 250	Di bawah *rate*	10	2	-	2.00	0
			5	-	5.00	0
			8	41	7.01	0
	Sama dengan *rate*	10	10	14	6.03	0
	Di atas *rate*	10	12	11	6.12	0
			15	7	5.92	0
			20	9	6.94	0
Burst 500	Di bawah *rate*	10	2	-	2.00	0
			5	-	5.00	0
			8	-	8.00	0
	Sama dengan *rate*	10	10	47	8.88	0
	Di atas *rate*	10	12	34	8.84	0
			15	24	8.84	0
			20	16	8.86	0

Sedangkan hasil pengujian *meter table* dengan *rate* 10 Mbps pada protokol UDP ditunjukkan pada Tabel 9. Hasil pengujian menunjukkan pola yang konsisten dengan pengujian TCP, yaitu *meter table* berhasil membatasi laju trafik, dengan *throughput* efektif hanya mencapai sekitar 50% dari nilai *rate* yang dikonfigurasi, serta penambahan *burst size* meningkatkan *throughput* dengan menunda penerapan mekanisme *packet dropping*.

TABEL IX
HASIL PENGUJIAN METER RATE 10 MBPS UDP

Burst Size	Sken.	*Rate* (Mbps)	Paket dikirim (Mbps)	Waktu mulai *drop* (s)	*Throughput* (Mbps)	*Loss* (%)
Tanpa *Burst*	Di bawah *rate*	10	2	-	2.00	0
			5	-	5.00	0
			8	-	4.78	40
	Sama dengan *rate*	10	10	-	4.78	52
	Di atas *rate*	10	12	-	4.77	60
			15	-	4.77	68
			20	-	4.77	76
Burst 250	Di bawah *rate*	10	2	-	2.00	0
			5	-	5.00	0
			8	41	6.94	13
	Sama dengan *rate*	10	10	17	6.11	39
	Di atas *rate*	10	12	14	6.31	47
			15	9	6.10	59
			20	9	6.75	66
Burst 500	Di bawah *rate*	10	2	-	2.00	0
			5	-	5.00	0
			8	-	8.00	0
	Sama dengan *rate*	10	10	50	9.07	9.3
	Di atas *rate*	10	12	36	9.00	25
			15	25	8.95	40
			20	17	8.91	55

C. Analisis dan Pembahasan Hasil Evaluasi

1) Analisis Jaringan SDN Tanpa Meter Table:

Berdasarkan hasil pengujian, *throughput* pada sisi penerima hampir sama dengan *throughput* pada sisi pengirim untuk seluruh skenario TCP. Pada pengiriman trafik 2 Mbps, 5 Mbps, 8 Mbps, 10 Mbps, 12 Mbps, 15 Mbps, dan 20 Mbps, *throughput* yang diterima berturut-turut sebesar 2,00 Mbps, 5,00 Mbps, 8,00 Mbps, 10,0 Mbps, 12,0 Mbps, 15,0 Mbps, dan 20,0 Mbps. Hasil ini menunjukkan bahwa tidak terdapat mekanisme pembatasan laju trafik maupun *packet dropping* pada jaringan SDN tanpa meter table, sehingga seluruh paket dapat diterima dengan baik.

Pada protokol UDP, *throughput* yang diterima juga mengikuti laju trafik yang dikirim, yaitu 2,00 Mbps, 5,00 Mbps, 8,00 Mbps, 10,0 Mbps, 12,0 Mbps, 15,0 Mbps, dan 20,0 Mbps. Perbedaan *throughput* antara sisi pengirim dan penerima relatif kecil, disertai *packet loss* sebesar 0% pada

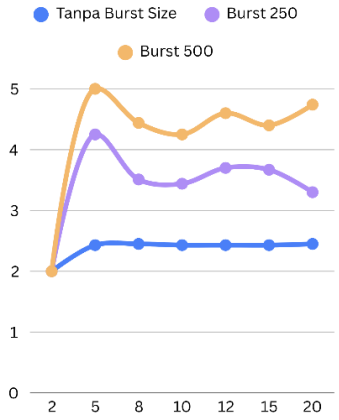
JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2026.102.496

seluruh skenario. Hal ini menunjukkan bahwa tidak terjadi *packet dropping* maupun pembatasan laju trafik pada jaringan SDN tanpa *meter table*.

2) Analisis Jaringan SDN berbasis Meter Table dengan Rate 5 Mbps pada TCP:

Hasil pengujian jaringan SDN berbasis *meter table* dengan *rate* 5 Mbps pada protokol TCP, dengan konfigurasi tanpa *burst size*, menunjukkan *meter table* melakukan pembatasan secara langsung sejak awal pengiriman. Pada pengiriman paket sama dengan dan di atas nilai *rate*, *throughput* yang dihasilkan berada pada kisaran 2,43 hingga 2,45 Mbps, di mana kondisi tersebut mengindikasi bahwa *meter table* menghasilkan *throughput* efektif sekitar 50% dari *rate* yang dikonfigurasi. Sedangkan, pada skenario pengiriman 2 Mbps, *throughput* tetap mencapai sekitar 2 Mbps karena paket yang dikirim masih berada di bawah nilai *rate* sehingga tidak terjadi mekanisme pembatasan laju trafik.

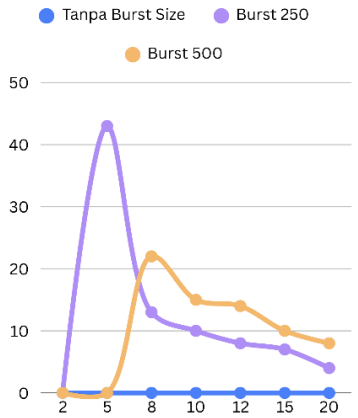
Berbeda dengan konfigurasi tanpa *burst size*, konfigurasi *burst size* memiliki karakteristik tambahan yang dihasilkan dari penerapan *burst size*, yaitu waktu mulai *dropping* setelah kapasitas *burst size* habis. Selama kapasitas *burst* masih tersedia, paket yang dikirim dapat mempertahankan *throughput* tinggi, namun setelah *burst size* habis, mekanisme *dropping* akan mulai bekerja. Kondisi ini menyebabkan nilai *throughput* pada konfigurasi dengan *burst size* menjadi lebih tinggi dibandingkan konfigurasi tanpa *burst size*, yaitu sebesar 3.32 hingga 4.25 Mbps pada *burst* 250 Mbps dan sebesar 4.25 hingga 5 Mbps pada *burst* 500 Mbps sebagaimana ditunjukkan pada Gambar 11.



Gambar. 11. *Throughput rate* 5 Mbps pada TCP

Penambahan *burst size* mempengaruhi waktu mulai *dropping*, sehingga pada konfigurasi *burst size* 250 Mbps dengan pengiriman 5 Mbps, mulai terjadi *dropping* pada detik ke-43, sedangkan pada trafik 8 Mbps, 10 Mbps, 12 Mbps, 15 Mbps, dan 20 Mbps masing-masing mulai bekerja pada detik ke-13, ke-10, ke-8, ke-7, dan ke-4.

Konfigurasi *burst size* 500 Mbps, *dropping* mulai terjadi pada waktu yang lebih lambat dibanding pada *burst* 250 Mbps. *Dropping* mulai terlihat pada detik ke-22 untuk paket 8 Mbps, detik ke-15 pada 10 Mbps, detik ke-14 pada 12 Mbps, detik ke-10 pada 15 Mbps, dan detik ke-8 pada 20 Mbps. Hasil ini menunjukkan bahwa semakin besar nilai *burst size* yang dikonfigurasi, semakin lama kapasitas *burst* dapat dipertahankan sehingga waktu mulai *dropping* menjadi lebih lambat. Sebaliknya, nilai *burst size* yang lebih kecil menyebabkan kapasitas *burst* lebih cepat habis sehingga mekanisme pembatasan mulai bekerja lebih awal seperti ditunjukkan pada Gambar 12.



Gambar. 12. Waktu mulai *dropping rate* 5 Mbps pada TCP

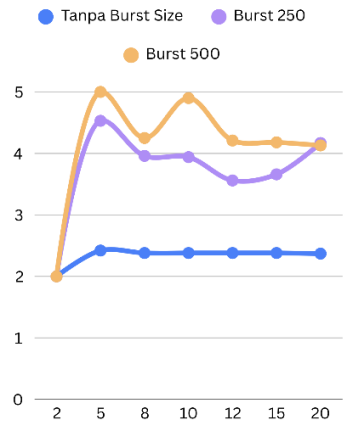
Hasil *throughput* efektif 50% dari nilai *rate* yang dikonfigurasi konsisten dengan hasil pada konfigurasi tanpa *burst size*, sehingga memperkuat indikasi bahwa implementasi *meter table* pada *MikroTik* RB750Gr3 memiliki karakteristik pembatasan efektif sekitar setengah dari nilai *rate* yang dikonfigurasi. Selain itu, parameter *burst size* mempengaruhi waktu mulai *dropping* pada mekanisme *meter table*, namun tidak mempengaruhi nilai pembatasan efektif. Di mana waktu mulai bekerjanya *meter table* dipengaruhi oleh nilai paket yang dikirim terhadap kombinasi nilai *rate* dan *burst size* yang dikonfigurasi.

Packet loss pada pengujian TCP memiliki nilai 0% karena TCP memiliki mekanisme *retransmission* dan *congestion control* memungkinkan paket yang hilang dikirim kembali serta laju pengiriman disesuaikan secara otomatis, sehingga nilai *packet loss* yang terukur cenderung sangat kecil.

3) Analisis Jaringan SDN berbasis Meter Table dengan Rate 5 Mbps pada UDP:

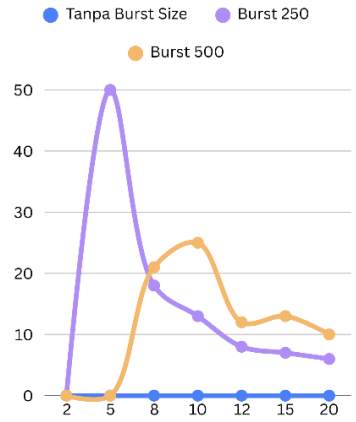
Hasil pengujian jaringan SDN berbasis *meter table* dengan *rate* 5 Mbps pada protokol UDP pada konfigurasi tanpa *burst size*, *meter table* menerapkan mekanisme *packet dropping* secara langsung sejak awal pengiriman. Pada pengiriman paket sama dengan dan di atas nilai *rate*, *throughput* yang dihasilkan berada pada kisaran 2,37 hingga 2,42 Mbps, yang menunjukkan bahwa implementasi *meter table* menghasilkan *throughput* efektif sekitar 50% dari nilai *rate* yang dikonfigurasi. Sedangkan, pada skenario pengiriman 2 Mbps, *throughput* tetap mencapai sekitar 2 Mbps tanpa *packet loss* karena trafik masih berada di bawah nilai *rate* sehingga tidak terjadi mekanisme *packet dropping*.

Sama seperti pada pengujian TCP, konfigurasi *burst size* memungkinkan pengiriman *throughput* tinggi selama kapasitas *burst size* masih tersedia dan menerapkan *meter table* setelah *burst size* habis menyebabkan *throughput* pada konfigurasi dengan *burst size* lebih tinggi dibandingkan konfigurasi tanpa *burst size*, yaitu sebesar 3,56 hingga 4,53 Mbps pada *burst size* 250 Mbps dan 4,13 hingga 5,00 Mbps pada *burst size* 500 Mbps sebagaimana ditunjukkan pada Gambar 13.



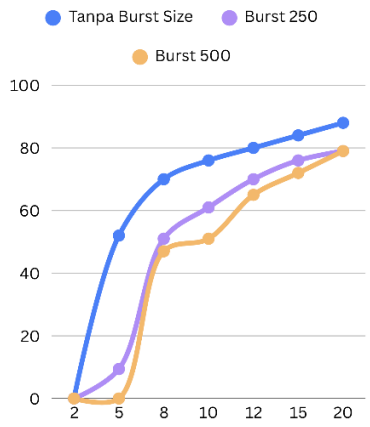
Gambar. 13. *Throughput rate* 5 Mbps pada UDP

Packet dropping pada konfigurasi *burst size* 250 Mbps mulai terlihat pada detik ke-50 untuk paket 5 Mbps, detik ke-18 pada 8 Mbps, detik ke-13 pada 10 Mbps, detik ke-8 pada 12 Mbps, detik ke-7 pada 15 Mbps dan detik ke-6 pada 20 Mbps. Sedangkan pada konfigurasi *burst size* 500 Mbps, waktu mulai *packet dropping* terjadi lebih lambat, yaitu detik ke-21 untuk paket 8 Mbps, detik ke-25 pada 10 Mbps, detik ke-12 pada 12 Mbps, detik ke-13 pada 15 Mbps, dan detik ke-10 pada 20 Mbps.



Gambar. 14. Waktu mulai *dropping rate* 5 Mbps pada UDP

Pada pengujian protokol UDP setiap paket yang dibuang oleh *meter table* secara langsung tercatat sebagai *packet loss* karena UDP tidak memiliki mekanisme retransmission maupun *congestion control*. Sehingga, peningkatan trafik yang dikirim tidak lagi meningkatkan *throughput* setelah mekanisme *packet dropping* aktif, melainkan meningkatkan nilai *packet loss*. Pada konfigurasi tanpa *burst size*, *packet loss* meningkat dari 52% menjadi 88%. Sementara itu, pada konfigurasi *burst size* 250 Mbps, *packet loss* berkisar antara 9,4% hingga 79%, sedangkan pada *burst size* 500 Mbps berada pada rentang 0% hingga 79%. Nilai *packet loss* lebih rendah daripada konfigurasi tanpa *burst size* karena sebagian trafik masih dapat diteruskan selama kapasitas *burst* tersedia sebagaimana ditunjukkan pada Gambar 15. Namun setelah *burst* habis, *packet loss* kembali meningkat seiring bertambahnya trafik yang dikirim.



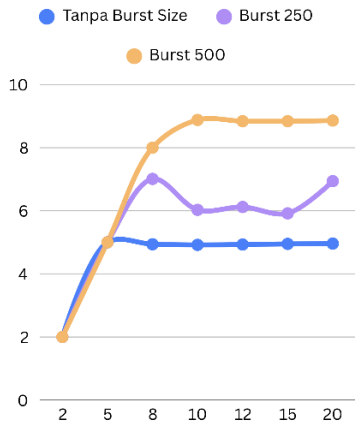
Gambar. 15. *Packet loss rate* 5 Mbps pada UDP

4) Analisis Jaringan SDN berbasis Meter Table dengan Rate 10 Mbps pada TCP:

Hasil pengujian jaringan SDN berbasis *meter table* dengan *rate* 10 Mbps pada protokol TCP, dengan konfigurasi tanpa *burst size*, *meter table* mulai menerapkan *packet dropping* segera setelah laju trafik melebihi batas efektif *rate*. Pada skenario pengiriman paket sebesar 2 Mbps dan 5 Mbps, rata-rata *throughput* akhir masing-masing sebesar 2 Mbps dan 5 Mbps karena keduanya masih berada di bawah batas efektif *meter table*, sehingga seluruh paket dapat diteruskan tanpa mengalami proses *packet dropping*.

Sementara itu, pada pengiriman paket sebesar 8 Mbps hingga 20 Mbps, *throughput* berada pada kisaran 4,92 hingga 4,96 Mbps. Hasil tersebut menunjukkan bahwa walaupun nilai konfigurasi *rate* ditetapkan sebesar 10 Mbps, mekanisme *meter table* pada *MikroTik* RB750Gr3 tetap melakukan *packet dropping* ketika *throughput* mendekati sekitar setengah dari nilai *rate*.

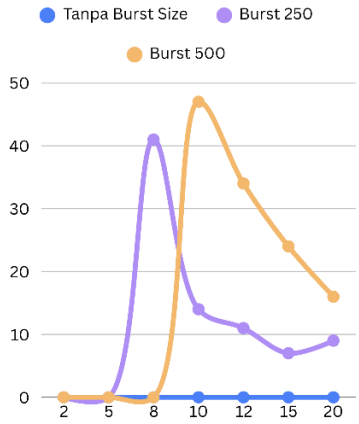
Berbeda dengan konfigurasi tanpa *burst size*, pada konfigurasi *burst size packet dropping* tidak terjadi secara langsung melainkan terjadi setelah kapasitas *burst size* habis sehingga paket yang dikirim dapat mempertahankan *throughput* yang tinggi. Sehingga nilai *throughput* pada konfigurasi dengan *burst size* menjadi lebih tinggi dibandingkan konfigurasi tanpa *burst size*, yaitu sebesar 5,92 hingga 7,01 Mbps pada *burst size* 250 Mbps dan sebesar 8,84 hingga 8,88 Mbps pada *burst size* 500 Mbps sebagaimana ditunjukkan pada Gambar 16.



Gambar. 16. *Throughput rate* 10 Mbps pada TCP

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2026.102.496

Mekanisme pembatasan pada konfigurasi *burst size* 250 Mbps mulai terlihat pada detik ke-41 untuk pengiriman 8 Mbps, detik ke-14 pada 10 Mbps, detik ke-11 pada 12 Mbps, detik ke-7 pada 15 Mbps, dan detik ke-9 pada 20 Mbps. Pada konfigurasi *burst size* 500 Mbps, *dropping* mulai bekerja lebih lambat, yaitu pada detik ke-47 untuk paket 10 Mbps, detik ke-34 pada 12 Mbps, detik ke-24 pada 15 Mbps, dan detik ke-16 pada 20 Mbps. Selain itu, pada pengiriman 8 Mbps, kapasitas *burst size* tidak habis selama durasi pengujian sehingga *throughput* tetap mencapai 8 Mbps tanpa terjadi *packet dropping*. Hasil ini menunjukkan bahwa semakin besar nilai *burst size* yang dikonfigurasi, semakin lama kapasitas burst dapat dipertahankan sehingga waktu mulai *dropping* menjadi lebih lambat seperti ditunjukkan pada Gambar 17.

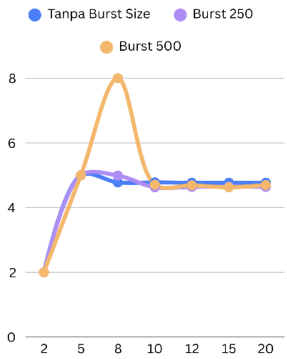


Gambar. 17. Waktu mulai *dropping rate* 10 Mbps pada TCP

5) Analisis Jaringan SDN berbasis Meter Table dengan Rate 10 Mbps pada UDP:

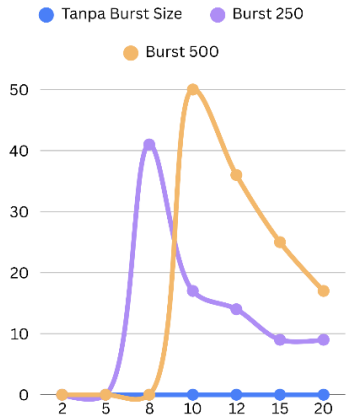
Hasil pengujian jaringan SDN berbasis *meter table* dengan *rate* 10 Mbps pada protokol UDP, dengan konfigurasi tanpa *burst size*, *meter table* menerapkan mekanisme *packet dropping* secara langsung sejak awal pengiriman. Pada pengiriman paket di bawah nilai *rate*, yaitu 2 Mbps dan 5 Mbps, *throughput* yang diperoleh tetap mencapai sekitar 2 Mbps dan 5 Mbps dengan *packet loss* sebesar 0% karena seluruh paket masih berada di bawah batas *rate* sehingga tidak mengalami *packet dropping*. Sementara itu, pada pengiriman paket sebesar 8 Mbps hingga 20 Mbps, *throughput* berada pada kisaran 4,77 hingga 4,78 Mbps.

Sama seperti pada pengujian TCP, konfigurasi *burst size* memungkinkan paket mempertahankan *throughput* yang tinggi selama kapasitas *burst size* masih tersedia. Setelah kapasitas *burst size* habis, mekanisme *dropping* baru mulai bekerja sehingga nilai akhir pada konfigurasi dengan *burst size* menjadi lebih tinggi dibandingkan konfigurasi tanpa *burst size*, yaitu sebesar 6,10 hingga 6,94 Mbps pada *burst size* 250 Mbps dan sebesar 8,91 hingga 9,07 Mbps pada *burst size* 500 Mbps sebagaimana ditunjukkan pada Gambar 18.



Gambar. 18. *Throughput rate* 10 Mbps pada UDP

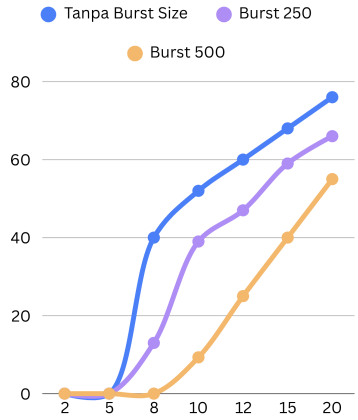
Packet dropping pada konfigurasi *burst size* 250 Mbps mekanisme *packet dropping* mulai terlihat pada detik ke-41 untuk paket 8 Mbps, detik ke-17 pada 10 Mbps, detik ke-14 pada 12 Mbps, serta detik ke-9 pada 15 Mbps dan 20 Mbps. Sementara itu, pada konfigurasi *burst size* 500 Mbps, waktu mulai *packet dropping* terjadi lebih lambat, yaitu detik ke-50 untuk paket 10 Mbps, detik ke-36 pada 12 Mbps, detik ke-25 pada 15 Mbps, dan detik ke-17 pada 20 Mbps. Pada pengiriman paket sebesar 8 Mbps, *packet dropping* tidak terjadi karena kapasitas *burst size* belum habis selama durasi pengujian. Hasil tersebut menunjukkan bahwa semakin besar nilai *burst size* yang dikonfigurasi, semakin lama kapasitas *burst size* dapat dipertahankan sehingga waktu mulai *dropping* menjadi lebih lambat. Adapun hubungan antara paket yang dikirim dengan waktu mulai *dropping* ditunjukkan pada Gambar 19.



Gambar. 19. Waktu mulai *dropping rate* 10 Mbps pada UDP

Paket yang dibuang oleh *meter table* secara langsung tercatat sebagai *packet loss* karena UDP tidak memiliki mekanisme *retransmission* maupun *congestion control*. Oleh karena itu, peningkatan trafik yang dikirim tidak lagi meningkatkan *throughput* setelah mekanisme *packet dropping* aktif, melainkan meningkatkan nilai *packet loss*. Pada konfigurasi tanpa *burst size*, *packet loss* meningkat dari 40% menjadi 76%. Sementara itu, pada konfigurasi *burst size* 250 Mbps, *packet loss* berada pada kisaran 13% hingga 66%, sedangkan pada *burst size* 500 Mbps berkisar antara 0% hingga 55%, sebagaimana ditunjukkan pada Gambar 20. Nilai *packet loss* yang lebih rendah dibandingkan konfigurasi tanpa *burst size* menunjukkan bahwa sebagian paket masih dapat diteruskan selama kapasitas *burst size* belum habis. Namun, setelah kapasitas *burst size* habis, *packet loss*

kembali meningkat seiring bertambahnya trafik yang dikirim.



Gambar. 20. *Packet loss rate* 10 Mbps pada UDP

V. KESIMPULAN

Implementasi *meter table* pada *MikroTik* RB750Gr3 berbasis *OpenFlow* 1.3 menunjukkan sebagian kesesuaian dengan karakteristik teoritis *OpenFlow* 1.3, terutama dalam kemampuan melakukan *rate limiting* melalui mekanisme *packet dropping* pada *band type drop*. Namun, terdapat perbedaan pada akurasi *meter rate*, dengan *throughput* efektif hanya mencapai sekitar 50% dari nilai *rate* yang dikonfigurasi, serta pada mekanisme *burst size* yang dapat habis meskipun trafik yang dikirim berada di bawah kapasitas *burst size*, sehingga mengindikasikan adanya perbedaan mekanisme pemrosesan *rate* dan *burst* pada implementasi *MikroTik* RB750Gr3. Pada trafik di bawah *rate* efektif, seluruh paket dapat diteruskan tanpa *packet loss*, sedangkan pada trafik yang mencapai atau melebihi *rate* efektif, *meter table* mulai melakukan *packet dropping* sehingga *throughput* terbatas dan *packet loss* meningkat. Karakteristik tersebut konsisten pada seluruh skenario, dengan semakin tinggi trafik yang melebihi batas efektif maka semakin banyak paket yang dibuang, meskipun penggunaan *burst size* memungkinkan *throughput* sesaat mendekati *rate* yang dikonfigurasi.

Meter table efektif mencegah peningkatan *throughput* meskipun laju trafik yang dikirim terus dinaikkan, dengan batas efektif sekitar 50% dari *rate* yang dikonfigurasi. Penambahan *burst size* meningkatkan rata-rata *throughput* dan memperpanjang durasi sebelum *packet dropping* dimulai, tetapi tidak mengubah batas *throughput* efektif setelah *meter table* aktif. Sementara itu, *packet loss* pada protokol UDP, mekanisme pembatasan melalui *packet dropping* terkonfirmasi dari peningkatan *packet loss* seiring bertambahnya trafik yang melebihi *rate* efektif. Penggunaan *burst size* yang lebih besar juga menurunkan *packet loss* karena memperpanjang periode sebelum proses *dropping* dimulai.

Berdasarkan hasil tersebut, penelitian selanjutnya disarankan mengevaluasi implementasi *meter table OpenFlow* pada *RouterOS* dengan versi berbeda serta pada perangkat dari vendor lain untuk membandingkan karakteristiknya. Selain itu, diperlukan analisis lebih lanjut terhadap implementasi internal *OpenFlow* pada *MikroTik RouterOS*, khususnya mekanisme *rate limiting* dan *burst size*, agar penyebab *rate limiting* efektif sekitar 50% dari nilai *rate* serta perbedaan perilaku *burst size* dapat dipahami secara lebih komprehensif.

UCAPAN TERIMA KASIH

Terima Penulis mengucapkan terima kasih kepada Universitas Siliwangi yang telah menyediakan fasilitas, perangkat pendukung dan tempat penelitian sehingga penelitian ini dapat terlaksana dengan baik.

DAFTAR PUSTAKA

[1] M. Rahouti, K. Xiong, Y. Xin, and N. Ghani, "A Priority-Based Queueing Mechanism in Software-Defined Networking Environments," pp. 4–5, 2021, doi: 10.1109/CCNC49032.2021.9369614.

[2] C. Jayawardena, J. Chen, A. Bhalla, and L. Bui, "Comparative Analysis of POX and Ryu SDN Controllers in Scalable Networks," vol. 17, no. 2, pp. 35–51, 2025, doi: 10.5121/ijcnc.2025.17203.

[3] S. Aouad, I. El Meghrouni, Y. Sabri, A. Hilmani, and A. Maizate, "Quality of services in software defined networking : challenges and controller placement problems," vol. 33, no. 2, pp. 951–959, 2024, doi: 10.11591/ijeecs.v33.i2.pp951-959.

[4] S. K. Keshari, V. Kansal, and S. Kumar, "A Systematic Review of Quality of Services (QoS) in Software Defined Networking (SDN)," *Wirel. Pers. Commun.*, no. 0123456789, 2020, doi: 10.1007/s11277-020-07812-2.

[5] A. Rozaq and I. M. Suartana, "Manajemen Quality of Service pada Jaringan Software Defined Network Menggunakan OpenDayLight Controller," vol. 03, pp. 89–94, 2021.

[6] I. Šeremet and S. Čaušević, "Advances of configuring Quality of Service (QoS) in Software Defined Networks (SDN) by using meter table," *2020 19th Int. Symp. INFOTEH-JAHORINA*, no. March, pp. 1–5, 2020.

[7] D. J. Hamad, K. G. Yalda, and N. Ţăpuş, "Performance Assessment of QoS in Software Defined Networking using Meter Table and Floodlight Controller," vol. 7, no. September, pp. 631–637, 2023.

[8] H. MikroTik and M. B., "Help MikroTik RouterOS Network Management OpenFlow." [Online]. Available: https://help.mikrotik.com/docs/spaces/ROS/pages/295239685/Openflow

[9] O. N. F. ONF, "OpenFlow Switch Specification," vol. 0, pp. 0–105, 2012.

[10] R. Deb and S. Roy, "A comprehensive survey of vulnerability and information security in SDN," *Comput. Networks*, vol. 206, no. January, p. 108802, 2022, doi: 10.1016/j.comnet.2022.108802.

[11] P. A. Khairunnisa, N. Annisa, and J. Parhusip, "Penerapan Teknologi SDN (Software-Defined Networking) untuk Meningkatkan Keamanan Jaringan Perusahaan," vol. 4, pp. 2–9, 2024.

[12] R. Nursobah, "Kenalan yuk sama SDN (Software Defined Networking)," Medium. [Online]. Available: https://medium.com/core-network-laboratory-tech-page/kenalan-yuk-sama-sdn-software-defined-networking-bacd7f9a631f

[13] M. N. Kulkarni and J. W. Bakal, "Quality of Service in Software Defined Network : Leveraging OpenFlow Protocol," vol. 10, 2025.

[14] W. Zheng *et al.*, "Application - aware QoS routing in SDNs using machine learning techniques," *Peer-to-Peer Netw. Appl.*, pp. 529–548, 2022, doi: 10.1007/s12083-021-01262-8.

[15] N. Albur, S. Handigol, S. Naik, M. M. Mulla, and D. G. Narayan, "QoS-aware Flow Management in Software Defined Network," 2020, doi: 10.1109/CICN.2020.40.

[16] R. Kartadie, "Prototipe Switch OpenFlow Software-Based MikroTik RB750 Paket OpenFlow VS OpenvSwitch Firmware," vol. 2018, no. semnasIF, pp. 58–70, 2018.

[17] Subektiningsih, Renaldi, and P. Ferdiansyah, "Analisis Perbandingan Parameter QoS Standar TIPHON Pada Jaringan Nirkabel Dalam Penerapan Metode PCQ," vol. 12, no. 1, pp. 57–63, 2022.

[18] I. Syarifatun, R. Miyarno, T. Fatwa, and A. Rizqi, "Analisis Quality of Service (QoS) Menggunakan Standar Parameter Tiphon pada Jaringan Internet Berbasis Wi-Fi Kampus 1 Unjaya," vol. 17, no. 1, pp. 1–9, 2024.

[19] T. I. Safitri *et al.*, "Analisis QoS Menggunakan Standar TIPHON

pada Jaringan Wi-Fi SMK Dharma Bahari Surabaya," vol. 4, no. 4, pp. 2433–2443, 2025.
[20] H. P. Fitrian, L. S. Irvanati, M. R. Sopyan, Firdaus, and D. Siska, "Pengaruh Jarak Terhadap Kualitas Transfer Data dan Kapasitas Bandwidth pada Jaringan Fiber Optik untuk Optimalisasi Kinerja Jaringan Komunikasi," vol. 8, no. 1, pp. 112–115, 2025.
[21] H. R. Oktaseli and A. A. Slameto, "Evaluation of Wireless LAN Quality of Service (QoS) in Primary Education Using TIPHON Standards," vol. 9, no. 2, 2025.
[22] A. Chydzinski, "Impact of the Dropping Function on Clustering of Packet Losses," 2022.
[23] K. Bazi and B. Nassereddine, "Comparative study of TCP congestion control algorithms," vol. 8, no. 6, pp. 3560–3564, 2019.
[24] D. P. Dewi, F. Maulana, T. Emaniar, and R. A. Faridz, "Simulasi Software Defined Wireless Network: Skenario Unicasting dan Bicasting," vol. 3, no. April, pp. 1–8, 2023.
[25] ESnet, "Invoking Iperf3." [Online]. Available: https://software.es.net/iperf/invoking.html

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2026.102.496