Department of Digital Business
Journal of Artificial Intelligence and Digital Business (RIGGS)
Homepage: https://journal.ilmudata.co.id/index.php/RIGGS
Vol. 5 No. 2 (2026) pp: 21911-21926
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Analisis QoS Jaringan pada Server Debian Menggunakan Algoritma Hierarchical Token Bucket untuk Manajemen Bandwidth

Andre Septianus Nababan[1], Raymon Simamora[2], Cristina Situmorang[3], Lotar Mateus Sinaga[4]
[1,2,3,4]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Katolik Santo Thomas
[1]andre.nababan2019@gmail.com , [2] raysimamora26@gmail.com , [3]cristinasitumorang26@gmail.com*,
[4]lotarmateus88@gmail.com

Abstrak

Pertumbuhan jumlah pengguna dan volume lalu lintas data menuntut jaringan komputer yang mampu menyediakan koneksi stabil, cepat, dan terkontrol. Penelitian ini bertujuan menganalisis kualitas layanan jaringan pada server Debian 12 melalui penerapan algoritma Hierarchical Token Bucket sebagai metode manajemen bandwidth. Penelitian menggunakan metode eksperimen dengan membandingkan kondisi jaringan sebelum dan sesudah penerapan HTB pada topologi bintang yang melibatkan satu server Debian, perangkat MikroTik sebagai switch atau bridge, serta dua client pengujian. Konfigurasi HTB membagi bandwidth sebesar 10 Mbps untuk Client 1 dan 5 Mbps untuk Client 2. Pengukuran dilakukan terhadap parameter Quality of Service yang meliputi throughput, delay, jitter, dan packet loss menggunakan iperf3 dan ping. Hasil penelitian menunjukkan bahwa sebelum penerapan HTB, throughput kedua client relatif sama, yaitu 94,7 Mbps pada Client 1 dan 94,8 Mbps pada Client 2. Setelah HTB diterapkan, throughput menjadi 8,81 Mbps dan 4,71 Mbps, sehingga pembatasan bandwidth berjalan sesuai kebijakan. Nilai delay rata-rata tetap rendah, yaitu 3 ms pada Client 1 dan 5 ms pada Client 2, sedangkan packet loss tetap 0% pada kedua client. Nilai jitter sesudah HTB sebesar 0,292 ms dan 5,862 ms, yang masih termasuk kategori sangat baik berdasarkan standar TIPHON. Temuan ini menunjukkan bahwa HTB efektif mengatur distribusi bandwidth secara adil, terstruktur, dan efisien tanpa menurunkan stabilitas layanan jaringan secara signifikan.

Kata kunci: Quality of Service (QoS), Hierarchical Token Bucket (HTB), Manajemen Bandwidth, Debian Server, Throughput, Delay.

1. Latar Belakang

Perkembangan teknologi informasi dan komunikasi yang semakin pesat menyebabkan kebutuhan terhadap jaringan komputer yang stabil, cepat, dan andal menjadi semakin penting. Seiring dengan meningkatnya jumlah pengguna dan volume lalu lintas data, sering terjadi permasalahan berupa kemacetan trafik (*congestion*), pembagian bandwidth yang tidak merata, serta penurunan kualitas layanan jaringan.

Kualitas layanan jaringan atau *Quality of Service* (QoS) merupakan salah satu aspek penting dalam mengukur performa suatu jaringan. QoS merupakan metode untuk mengukur dan mengelola kinerja jaringan berdasarkan parameter seperti *throughput*, *delay*, *jitter*, dan *packet loss* guna mengetahui kualitas layanan jaringan yang sedang berjalan. Nilai QoS yang baik akan memberikan pengalaman penggunaan jaringan yang lebih optimal, terutama pada layanan yang membutuhkan kestabilan koneksi seperti *video conference*, *streaming*, dan akses aplikasi berbasis jaringan.[1]

Salah satu penyebab utama menurunnya kualitas layanan jaringan adalah banyaknya pengguna yang menggunakan bandwidth secara bersamaan tanpa adanya mekanisme manajemen bandwidth. Kondisi tersebut menyebabkan akses internet menjadi lambat sehingga diperlukan penerapan metode manajemen bandwidth, seperti Hierarchical Token Bucket (HTB), untuk mengalokasikan bandwidth sesuai kebutuhan dan prioritas pengguna.[2]

Salah satu solusi yang dapat diterapkan untuk mengatasi permasalahan tersebut adalah melalui manajemen bandwidth. Manajemen bandwidth bertujuan untuk mengatur alokasi bandwidth agar penggunaan jaringan dapat dilakukan secara adil dan efisien. Sistem operasi Debian Linux merupakan salah satu sistem operasi yang banyak digunakan sebagai server jaringan karena memiliki tingkat stabilitas yang tinggi serta menyediakan berbagai utilitas jaringan untuk mendukung implementasi manajemen bandwidth.

Metode Hierarchical Token Bucket (HTB) merupakan mekanisme manajemen trafik yang mampu menjamin alokasi bandwidth pada setiap kelas pengguna sesuai kebutuhan. HTB juga menerapkan mekanisme *borrowing bandwidth*, yaitu bandwidth yang tidak digunakan oleh suatu kelas dapat dimanfaatkan oleh kelas lain sehingga pemanfaatan sumber daya jaringan menjadi lebih efisien.[3]

Salah satu metode *traffic shaping* yang dapat diterapkan pada server Debian adalah algoritma *Hierarchical Token Bucket* (HTB). HTB merupakan algoritma manajemen bandwidth yang memungkinkan pemberian jaminan laju bit (*bitrate*) pada setiap aliran trafik, mendukung pembagian bandwidth berlebih (*excess bandwidth sharing*), serta menyediakan mekanisme prioritas trafik sehingga mampu memenuhi kebutuhan QoS secara hierarkis.[4] Selain itu, HTB merupakan metode *traffic shaping* yang mampu mengimplementasikan pembagian trafik secara lebih akurat, di mana bandwidth yang tidak digunakan oleh suatu kelas dapat dimanfaatkan oleh kelas lain sehingga penggunaan bandwidth menjadi lebih optimal.[5]

Penggunaan fitur QoS dengan algoritma HTB pada perangkat jaringan mampu meningkatkan efisiensi distribusi bandwidth.[6]

2. Metode Penelitian

Penelitian ini menggunakan metode eksperimental untuk menganalisis pengaruh penerapan algoritma Hierarchical Token Bucket (HTB) terhadap kualitas layanan jaringan (Quality of Service/QoS) pada server Debian. Metode eksperimen dipilih karena memungkinkan peneliti melakukan perlakuan tertentu terhadap objek penelitian, yaitu penerapan algoritma HTB, kemudian mengamati perubahan yang terjadi pada perameter QoS jaringan sebelum dan sesudah penerapan algoritma tersebut.

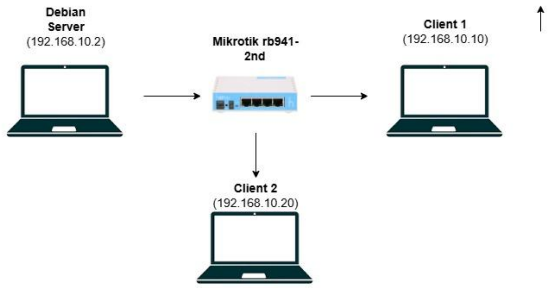
Penelitian ini menggunakan metode eksperimen. Penelitian eksperimen bertujuan untuk menguji hubungan sebab-akibat melalui pemberian perlakuan pada objek yang diteliti.[7]

Penelitian ini menggunakan pendekatan eksperimen melalui implementasi algoritma HTB pada server Debian, kemudian dilakukan pengukuran parameter Quality of Service berupa throughput, delay, jitter, dan packet loss sebelum dan sesudah penerapan HTB. Pendekatan seperti ini banyak digunakan dalam penelitian implementasi manajemen bandwidth karena mampu menunjukkan pengaruh langsung algoritma terhadap performa jaringan.[8]

2.1. Topologi Jaringan

Penelitian ini menggunakan topologi jaringan berbentuk star (bintang) yang terdiri atas satu server Debian sebagai server pengujian sekaligus tempat implementasi algoritma HTB, dua client sebagai endpoint pengujian, dan perangkat MikroTik RB941 yang berfungsi sebagai switch/bridge.

Berdasarkan deskripsi penelitian, topologi umum penelitian adalah sebagai berikut.



Gambar 1. Topologi lingkungan pengujian

Keterangan:

1. Server Debian 12 (192.168.10.2) berfungsi sebagai server pengujian sekaligus implementasi HTB.
2. MikroTik RB941 berfungsi sebagai switch/bridge.
3. Client 1 (192.168.10.10) dan Client 2 (192.168.10.20) berfungsi sebagai endpoint pengujian QoS.

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

2.2 Analisis Kebutuhan

Tabel 1. Perangkat Keras

Perangkat	Peran dalam penelitian
Server Debian 12	Gateway jaringan dan titik implementasi HTB
Router	Distribusi koneksi dari gateway ke client
Client 1	Endpoint pengujian QoS
Client 2	Endpoint pengujian QoS

Tabel 2. Perangkat lunak

Perangkat lunak	Fungsi
Debian GNU/Linux 12 CLI	Sistem operasi server gateway
Linux Traffic Control tc	Implementasi policy bandwidth berbasis kernel
iperf3	Pengujian throughput TCP dan jitter UDP
ping	Pengujian delay RTT dan packet loss
Command Prompt Windows pada client	Antarmuka eksekusi iperf3 dan ping pada client

Pengujian kinerja jaringan menggunakan *iperf3* untuk menganalisis *throughput, jitter*, dan *packet loss*, dengan menerapkan mode *reverse* (-R) serta protokol UDP (-u). Sementara itu, parameter *delay* diukur menggunakan *ping* sebanyak 100 kali (*-n 100*) untuk memastikan akurasi data. Seluruh detail sintaks dan parameter konfigurasi yang digunakan dapat dilihat pada dokumentasi *screenshot* pengujian.

2.3 Konfigurasi Hierarchical Token Bucket (HTB)

Secara teoritis, implementasi Hierarchical Token Bucket (HTB) pada Linux dilakukan menggunakan *traffic control* (tc) dengan struktur *root qdisc*, kelas induk, dan kelas anak. Setiap kelas diberikan parameter *rate* dan *ceil* untuk mengatur alokasi bandwidth. Pada penelitian ini, penerapan HTB diduga telah dilakukan karena hasil pengujian menunjukkan adanya pembatasan bandwidth yang stabil pada masing-masing client. Namun, karena dokumentasi konfigurasi *tc* tidak tersedia, detail seperti *classid*, *rate*, dan *ceil* tidak dapat diverifikasi secara langsung. Oleh karena itu, analisis konfigurasi hanya didasarkan pada hasil pengujian yang diperoleh.

Implementasi HTB pada sistem operasi Linux memungkinkan administrator mengatur prioritas trafik dan pembagian bandwidth secara lebih fleksibel melalui mekanisme queue sehingga penggunaan bandwidth menjadi lebih terkontrol sesuai kebutuhan setiap pengguna.[9]

Algoritma Hierarchical Token Bucket (HTB) memungkinkan administrator jaringan melakukan pembagian bandwidth secara otomatis dan terstruktur berdasarkan kelas trafik. Mekanisme tersebut menjadikan distribusi bandwidth lebih adil serta mempermudah pengelolaan jaringan ketika jumlah pengguna meningkat.[10]

Algoritma Hierarchical Token Bucket (HTB) mampu mengalokasikan bandwidth secara adil kepada setiap pengguna melalui mekanisme pembagian kelas (class) dan prioritas trafik. Pendekatan tersebut memungkinkan setiap pengguna memperoleh bandwidth sesuai kebijakan yang telah ditentukan tanpa mengganggu pengguna lain, sehingga performa jaringan tetap terjaga meskipun terjadi peningkatan lalu lintas data.[11]

HTB merupakan algoritma manajemen bandwidth yang mampu memberikan jaminan laju bit (*bitrate*) pada setiap aliran trafik, mendukung pembagian bandwidth berlebih (*excess bandwidth sharing*), serta menyediakan mekanisme prioritas trafik secara hierarkis. Dengan mekanisme tersebut, HTB mampu menjaga kualitas layanan jaringan (*Quality of Service/QoS*) secara lebih optimal.

Metode HTB bekerja dengan mengatur distribusi bandwidth secara hierarkis sehingga setiap kelas trafik memperoleh alokasi bandwidth sesuai kebutuhan dan prioritasnya. Mekanisme tersebut mampu meningkatkan efisiensi penggunaan bandwidth sekaligus menjaga kualitas layanan jaringan.[12]

Implementasi HTB pada jaringan internet mampu menjaga kestabilan throughput serta menekan terjadinya kemacetan trafik. Setelah HTB diterapkan, bandwidth dibagi menjadi 10 Mbps untuk Client 1 dan 5 Mbps untuk Client 2.

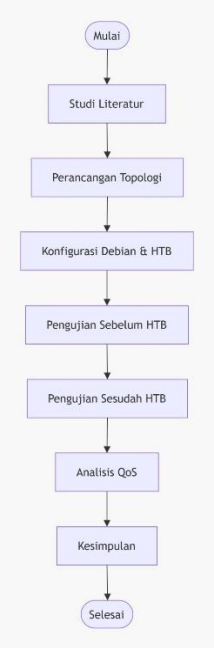
DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

Konfigurasi:

```
tc qdisc del dev enp0s3 root
tc qdisc add dev enp0s3 root handle 1: htb default 20
tc class add dev enp0s3 parent 1: classid 1:1 htb rate 15mbit ceil 15mbit
tc class add dev enp0s3 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit
tc class add dev enp0s3 parent 1:1 classid 1:20 htb rate 5mbit ceil 5mbit
tc filter add dev enp0s3 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.10.10 flowid 1:10
tc filter add dev enp0s3 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.10.20 flowid 1:20
```

2.4 Tahapan Penelitian

Alur penelitian dapat ditulis sebagai berikut.



Gambar 2. Flowchart

2.5 Metode Pengujian QoS

Metode pengujian yang terlihat pada dokumen adalah sebagai berikut.

1. Pengujian throughput dilakukan menggunakan *iperf3* dengan protokol TCP pada mode *reverse* (-R), sehingga proses pengiriman data berlangsung dari server ke client.
2. Pengujian delay dan packet loss dilakukan menggunakan perintah *ping* sebanyak 100 paket ke alamat IP tujuan.
3. Pengukuran jitter dilakukan menggunakan *iperf3* dengan protokol UDP, namun dokumentasi hasil pengujian belum ditampilkan secara konsisten pada kondisi sebelum dan sesudah penerapan HTB.
4. Analisis dilakukan secara komparatif dengan membandingkan parameter QoS pada dua client sebelum dan sesudah implementasi HTB.

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

2.6 Parameter Quality of Service (QoS)

Pengukuran Quality of Service (QoS) dilakukan berdasarkan parameter throughput, delay, jitter, dan packet loss karena keempat parameter tersebut mampu menggambarkan performa jaringan secara objektif dan menjadi dasar dalam mengevaluasi kualitas layanan internet.[13]

Pernyataan tersebut diperkuat oleh Rahmadhan, Khairil, dan Al Akbar (2024) yang menjelaskan bahwa keberhasilan implementasi manajemen bandwidth dapat dianalisis melalui parameter throughput, delay, jitter, dan packet loss karena parameter tersebut mampu menggambarkan kestabilan distribusi bandwidth serta kualitas komunikasi data pada jaringan komputer.[14]

Rumus Throughtput:

Throughput merupakan indikator keberhasilan pengiriman data pada jaringan yang menunjukkan jumlah data yang berhasil diterima dalam satuan waktu tertentu. Semakin tinggi nilai throughput, maka semakin baik performa jaringan dalam mentransmisikan data.[15]

Nilai throughput digunakan untuk mengetahui kemampuan efektif jaringan dalam mentransmisikan data. Semakin besar nilai throughput, maka semakin baik performa jaringan tersebut.

$$\text{Throughput} = \frac{\text{Total data diterima}}{\text{Waktu pengujian}}$$

Pada penelitian ini, nilai throughput operasional diambil dari output iperf3 bagian receiver karena pengujian memakai opsi -R.

Rumus Delay:

Delay merupakan parameter penting dalam pengukuran QoS yang menunjukkan waktu tempuh paket dari pengirim menuju penerima. Nilai delay yang rendah menunjukkan bahwa jaringan mampu merespons komunikasi data secara cepat dan stabil meskipun diterapkan mekanisme pembatasan bandwidth menggunakan HTB.[16]

$$\text{Delay rata-rata} = \frac{\sum \text{delay}}{n}$$

Pada bukti praktikum yang tersedia, nilai delay diambil dari Average RTT pada output ping.

Delay merupakan waktu yang dibutuhkan paket data untuk mencapai tujuan.

Rumus Packet Loss:

$$\text{Packet Loss}(\%) = \frac{\text{Paket dikirim} - \text{Paket diterima}}{\text{Paket dikirim}} \times 100\%$$

Rumus Jitter:

$$\text{Jitter rata-rata} = \frac{\sum |d_i - d_{i-1}|}{n-1}$$

Pada penelitian ini, bila tersedia, nilai jitter praktis dibaca langsung dari kolom Jitter pada output iperf3 UDP.

Keterangan:

a. D_n = delay paket ke-n

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

b. D_{n-1} = delay paket sebelumnya

c. n = jumlah paket

Standar Evalusi Quality of Service (QoS)

Pada penelitian ini, standar TIPHON digunakan sebagai acuan untuk menganalisis kualitas layanan jaringan sebelum dan sesudah penerapan algoritma Hierarchical Token Bucket (HTB).

Berdasarkan ETSI TR 101 329 (TIPHON), kualitas layanan jaringan (*Quality of Service/QoS*) dievaluasi menggunakan beberapa parameter utama, seperti delay, jitter, dan packet loss. Parameter-parameter tersebut digunakan untuk menggambarkan kemampuan jaringan dalam memberikan layanan komunikasi yang stabil dan dapat diterima oleh pengguna.[17]

Tabel 3. Standar Kategori QoS Berdasarkan TIPHON

Parameter	Sangat Baik	Baik	Sedang	Buruk
Delay	< 150 ms	150–300 ms	300–450 ms	> 450 ms
Jitter	0–75 ms	75–125 ms	125–225 ms	> 225 ms
Packet Loss	0%	1–3%	4–15%	> 15%

3. Hasil dan Diskusi

Hasil penelitian menunjukkan bahwa penerapan algoritma Hierarchical Token Bucket (HTB) pada server Debian 12 mampu mengendalikan alokasi bandwidth sesuai kebijakan yang ditetapkan. Sebelum penerapan HTB, kedua client memiliki throughput yang hampir sama, yaitu sekitar 94 Mbits/sec. Setelah HTB diterapkan, throughput menurun menjadi 8,81 Mbps/sec pada Client 1 dan 4,71 Mbps/sec pada Client 2, yang menunjukkan bahwa mekanisme *traffic shaping* berjalan dengan baik.

Hasil penelitian ini sejalan dengan penelitian Abdullah, Wibowo, dan Fiqar (2024) yang menyatakan bahwa implementasi algoritma Hierarchical Token Bucket mampu mengendalikan alokasi bandwidth sesuai kebijakan yang telah ditetapkan tanpa menyebabkan penurunan kualitas layanan jaringan secara signifikan. Dengan demikian, penerapan HTB tidak hanya berfungsi sebagai pembatas bandwidth, tetapi juga sebagai mekanisme untuk menjaga kestabilan parameter Quality of Service.[18]

Dari sisi Quality of Service (QoS), implementasi HTB tidak menurunkan kualitas jaringan secara signifikan. Nilai delay pada kedua client tetap rendah dan *packet loss* tetap sebesar 0%, sehingga stabilitas koneksi jaringan masih terjaga.

Parameter jitter menunjukkan nilai yang rendah, namun analisisnya dilakukan secara terbatas karena dokumentasi pengujian belum konsisten. Oleh karena itu, evaluasi keberhasilan implementasi HTB pada penelitian ini lebih difokuskan pada parameter throughput, delay, dan *packet loss*.

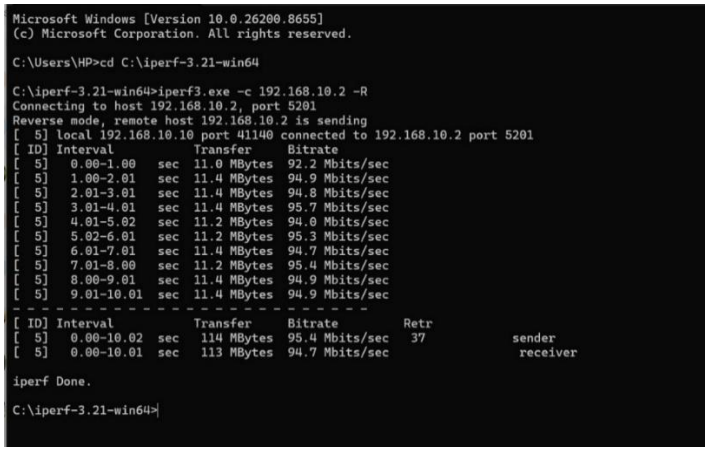
Secara keseluruhan, hasil penelitian menunjukkan bahwa HTB efektif digunakan untuk manajemen bandwidth pada server Debian 12 karena mampu mengatur pembagian bandwidth tanpa mengurangi kualitas layanan jaringan secara signifikan.

Hasil penelitian ini juga sejalan dengan penelitian Hamudi Alamsyah dan Somantri (2022) yang menyatakan bahwa penerapan metode HTB mampu meningkatkan pemerataan distribusi bandwidth sehingga setiap pengguna memperoleh alokasi bandwidth sesuai prioritas tanpa menurunkan kualitas layanan jaringan secara signifikan.[19]

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

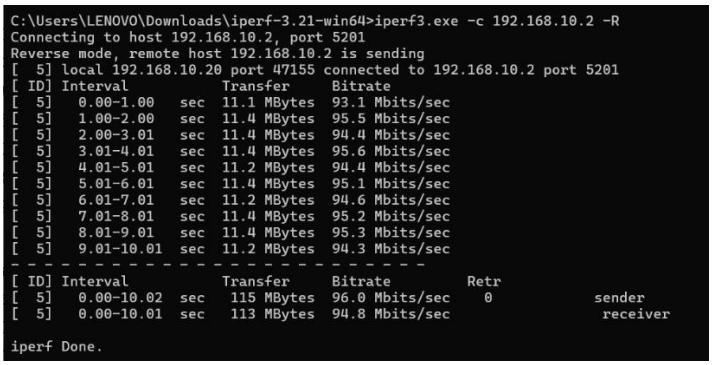
3.1. Hasil Pengujian Sebelum HTB

3.1.1 Throughtpuut Sebelum HTB



Gambar 3. Throughput Sebelum HTB pada Client 1

Berdasarkan screenshot pengujian, perintah yang digunakan adalah *iperf3 -c 192.168.10.2 -R*. Opsi *-R* menunjukkan bahwa aliran data dikirim dari server ke client. Hasil pengujian menunjukkan nilai *sender* sebesar 95,4 Mbps/sec dan *receiver* sebesar 94,7 Mbps/sec. Nilai *receiver* tersebut digunakan sebagai throughput efektif pada Client 1 sebelum penerapan HTB.



Gambar 4. Throughput Sebelum HTB pada Client 2

Pada Client 2 sebelum penerapan HTB, hasil pengujian *iperf3* menunjukkan nilai *sender* sebesar 96,0 Mbps/sec dan *receiver* sebesar 94,8 Mbps/sec. Nilai tersebut hampir sama dengan hasil yang diperoleh Client 1, sehingga dapat disimpulkan bahwa sebelum penerapan HTB kedua client memperoleh kapasitas bandwidth yang relatif serupa saat diuji secara individual.

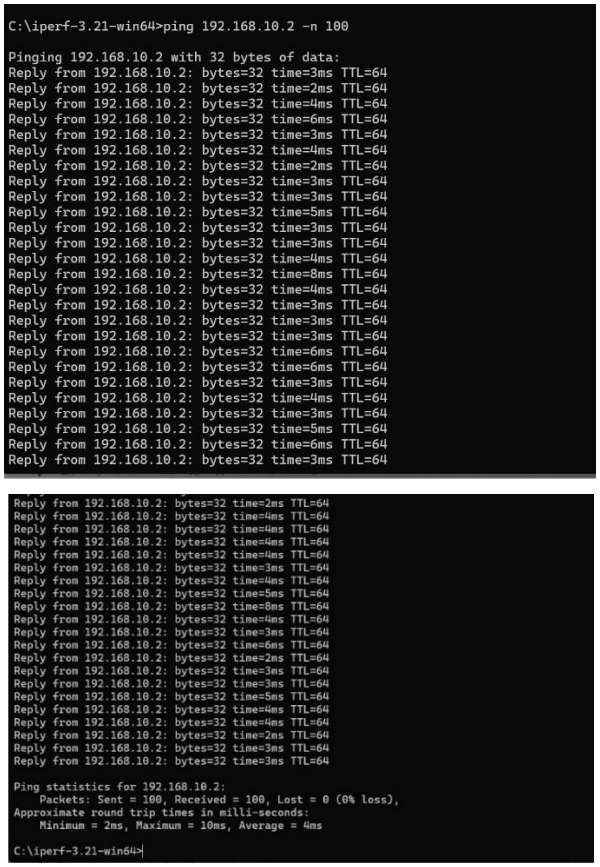
Tabel 4. hasil throughput sebelum HTB

Client	Sender	Receiver
Client 1	95.4 Mbps/sec	94.7 Mbps/sec
Client 2	96.0 Mbps/sec	94.8 Mbps/sec

Sebelum penerapan HTB, kedua client menunjukkan nilai throughput yang relatif sama. Hal ini mengindikasikan bahwa belum terdapat pembatasan bandwidth yang signifikan, sehingga setiap client dapat memanfaatkan kapasitas jaringan secara maksimal. Namun, kondisi tersebut juga menunjukkan bahwa belum terdapat kebijakan alokasi bandwidth yang berbeda antarclient.

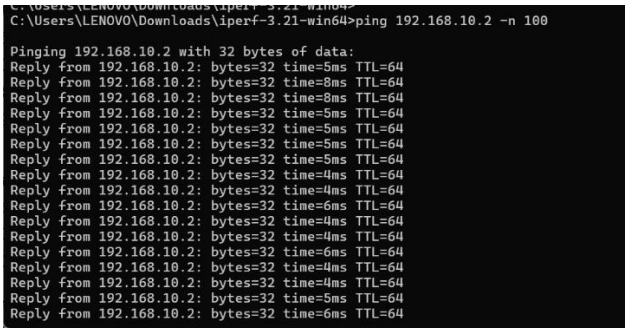
DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

3.1.2. Delay Sebelum HTB

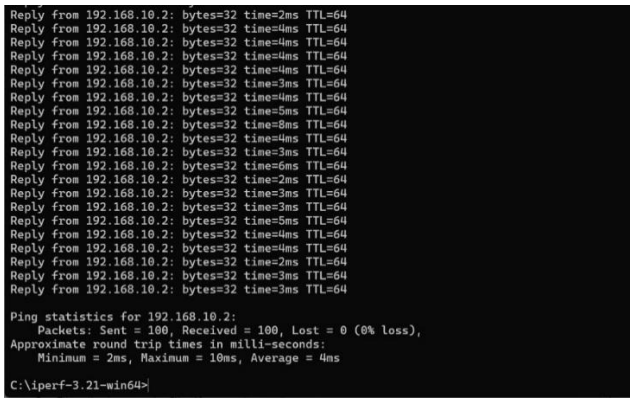


Gambar 5. Delay Sebelum HTB pada Client 1

Hasil pengujian *ping* pada Client 1 sebelum penerapan HTB menunjukkan nilai *round trip time* (RTT) minimum sebesar 2 ms, maksimum 10 ms, dan rata-rata 4 ms. Nilai tersebut menunjukkan bahwa kondisi jaringan sebelum penerapan HTB memiliki waktu tunda yang relatif rendah dan stabil.



DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)



Gambar 6. Delay Sebelum HTB pada Client 2

Hasil pengujian *ping* pada Client 2 sebelum penerapan HTB menunjukkan nilai *round trip time* (RTT) minimum sebesar 2 ms, maksimum 10 ms, dan rata-rata 4 ms. Nilai tersebut menunjukkan bahwa performa jaringan pada kedua client sebelum penerapan HTB masih relatif sama dan stabil.

Tabel 5. Delay sebelum HTB

Client	Minimum	Maximum	Average
Client 1	2 ms	10 ms	4 ms
Client 2	2 ms	10 ms	4 ms

Nilai rata-rata *round trip time* (RTT) sebesar 4 ms pada kedua client menunjukkan bahwa kondisi jaringan sebelum penerapan HTB masih responsif dan stabil. Hasil pengujian juga tidak menunjukkan adanya peningkatan delay yang signifikan.

3.1.3 Packet Loss Sebelum HTB

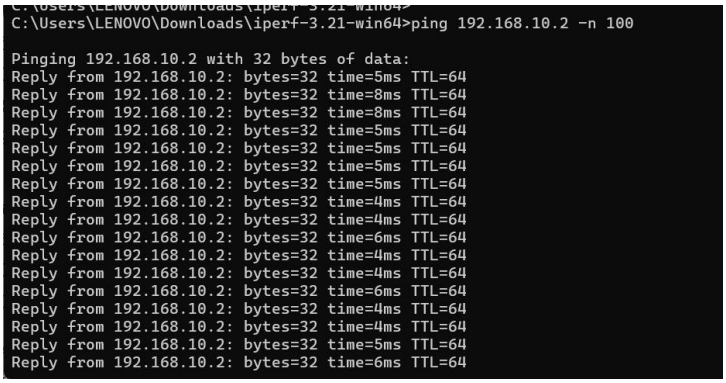
Berdasarkan Gambar 5 dan Gambar 6, hasil pengujian *ping* menunjukkan nilai *Sent* = 100, *Received* = 100, dan *Lost* = 0. Dengan demikian, packet loss pada kedua client sebelum penerapan HTB adalah sebesar 0%.

Tabel 6. packet loss sebelum HTB

Client	Sent	Received	Lost	Packet Loss
Client 1	100	100	0	0%
Client 2	100	100	0	0%

Tidak ditemukannya *packet loss* menunjukkan bahwa koneksi jaringan sebelum penerapan HTB berada dalam kondisi stabil. Hal ini mengindikasikan bahwa permasalahan utama pada kondisi awal bukan terletak pada kehilangan paket, melainkan belum adanya pengaturan bandwidth yang terstruktur.

3.1.4 Jitter Sebelum HTB

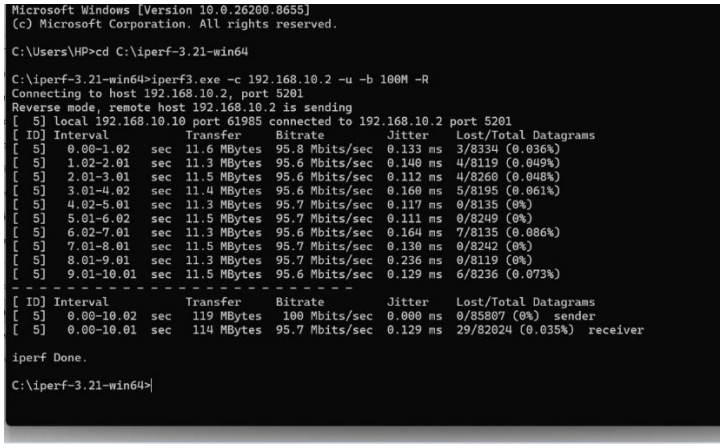


DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

Gambar 7. Jitter Sebelum HTB pada Client 1

Meskipun pada gambar diberi label sebagai pengujian jitter sebelum HTB pada Client 1, output yang ditampilkan masih berupa hasil *ping* dan bukan hasil *iperf3* UDP yang memuat nilai jitter. Oleh karena itu, nilai jitter sebelum penerapan HTB pada Client 1 tidak dapat ditentukan secara langsung dari dokumentasi yang tersedia.

Pengukuran jitter sebelum penerapan HTB hanya dapat dilakukan pada Client 2 karena dokumentasi pengujian UDP pada Client 1 tidak tersedia. Oleh karena itu, analisis jitter sebelum HTB dilakukan secara terbatas.



Gambar 8. Jitter Sebelum HTB pada Client 2

Pada gambar yang diberi label pengujian jitter sebelum HTB pada Client 2, perintah yang digunakan adalah *iperf3 -u -b 100M -R*. Hasil pengujian menunjukkan nilai *receiver* sebesar 95,7 Mbits/sec, *jitter* sebesar 0,129 ms, serta *lost/total datagrams* sebanyak 29/82024 (0,035%). Nilai tersebut menunjukkan bahwa variasi delay pada jaringan masih tergolong rendah. Namun, kesesuaian label client pada dokumentasi ini masih memerlukan verifikasi lebih lanjut.

Tabel 7. Jitter sebelum HTB

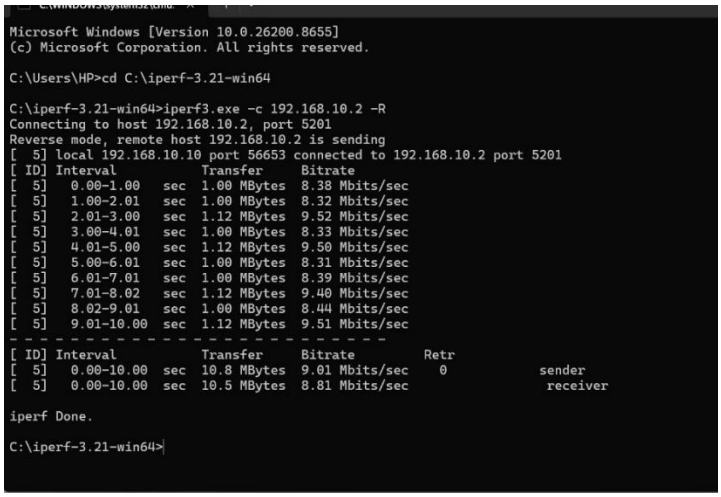
Client	Metode yang tampil	Nilai jitter eksplisit
Client 1	Screenshot berlabel jitter, tetapi isi masih ping	Tidak dapat ditetapkan dari dokumen
Client 2	iperf3 -u -b 100M -R	0.129 ms

Analisis jitter sebelum penerapan HTB tidak dapat dilakukan secara menyeluruh karena dokumentasi yang tersedia tidak konsisten. Dokumentasi pada Client 1 tidak menampilkan nilai jitter, sedangkan pada Client 2 nilai jitter ditampilkan melalui pengujian UDP. Oleh karena itu, data jitter sebelum HTB tidak tersedia secara lengkap untuk kedua client.

3.2 Hasil Pengujian Sesudah HTB

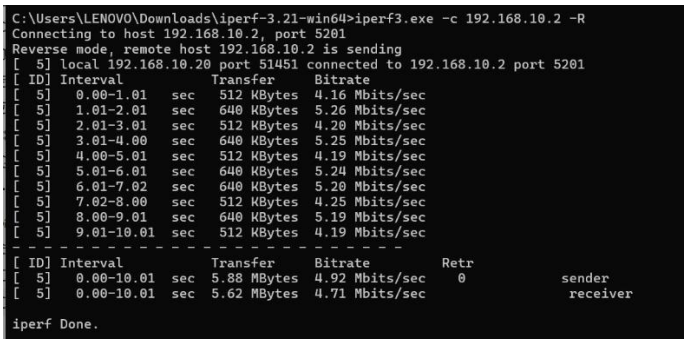
3.2.1 Throughput Sesudah HTB

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)



Gambar 9. Throughput Sesudah HTB pada Client 1

Setelah penerapan HTB, hasil pengujian menunjukkan throughput pada Client 1 sebesar 9,01 Mbps/sec pada *sender* dan 8,81 Mbps/sec pada *receiver*. Penurunan throughput ini menunjukkan bahwa mekanisme *traffic shaping* telah berjalan dengan baik, sehingga bandwidth dapat dikendalikan sesuai kebijakan yang telah ditetapkan.



Gambar 10. Throughput Sesudah HTB pada Client 2

Setelah penerapan HTB, hasil pengujian pada Client 2 menunjukkan throughput sebesar 4,92 Mbps/sec pada *sender* dan 4,71 Mbps/sec pada *receiver*. Nilai ini lebih rendah dibandingkan Client 1, yang menunjukkan bahwa alokasi bandwidth pada Client 2 dibatasi lebih ketat.

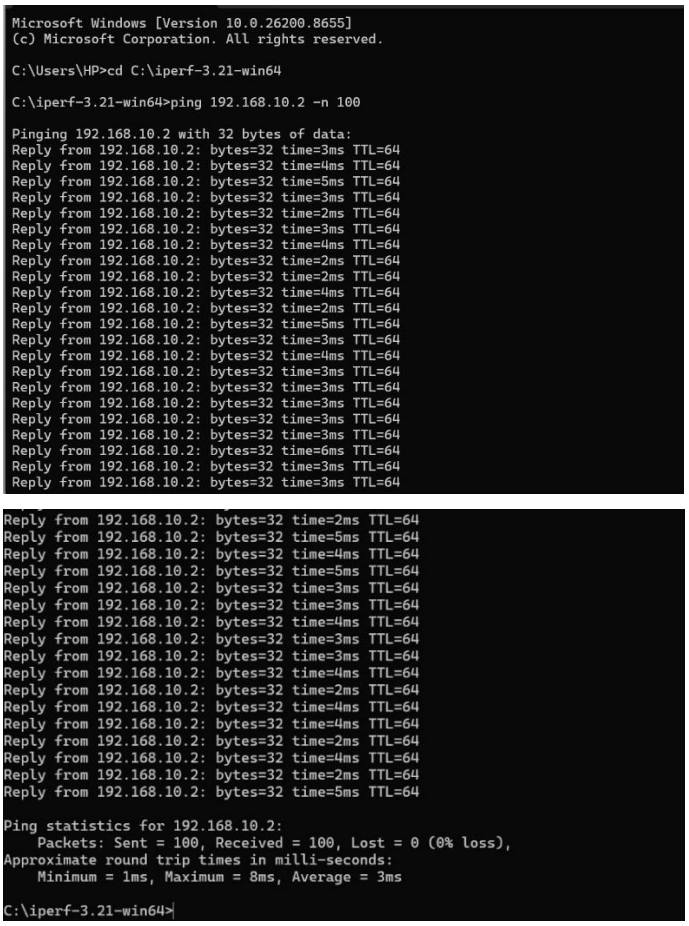
Tabel 8. Throughput sesudah HTB

Client	Sender	Receiver
Client 1	9.01 Mbps/sec	8.81 Mbps/sec
Client 2	4.92 Mbps/sec	4.71 Mbps/sec

Dibandingkan dengan kondisi sebelum HTB, throughput efektif setelah penerapan HTB mengalami penurunan yang signifikan pada kedua client. Client 1 mempertahankan sekitar 9,3% dari throughput awal, sedangkan Client 2 sekitar 5,0%. Hasil ini menunjukkan bahwa kebijakan alokasi bandwidth yang berbeda pada masing-masing client telah diterapkan dan berjalan dengan baik.

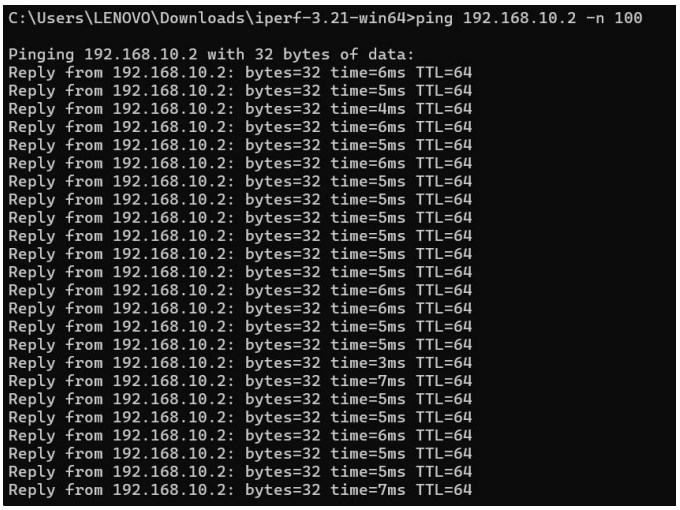
3.2.2 Delay Sesudah HTB

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

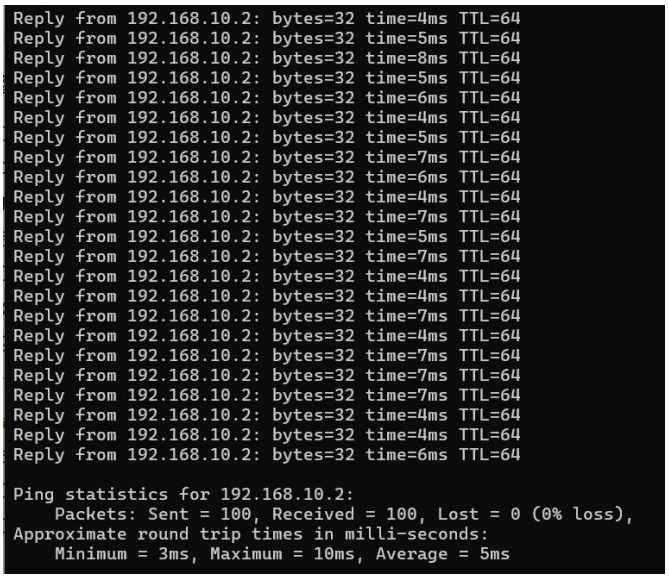


Gambar 11. Delay Sesudah HTB pada Client 1

Pada Client 1 setelah penerapan HTB, hasil pengujian *ping* menunjukkan nilai *round trip time* (RTT) minimum sebesar 1 ms, maksimum 8 ms, dan rata-rata 3 ms. Hasil ini menunjukkan bahwa rata-rata RTT mengalami penurunan dari 4 ms menjadi 3 ms setelah HTB diterapkan.



DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)



Gambar 12. Delay Sesudah HTB pada Client 2

Pada Client 2 setelah penerapan HTB, hasil pengujian *ping* menunjukkan nilai *round trip time* (RTT) minimum sebesar 3 ms, maksimum 10 ms, dan rata-rata 5 ms. Dibandingkan dengan kondisi sebelum HTB, rata-rata RTT pada Client 2 mengalami peningkatan dari 4 ms menjadi 5 ms.

Tabel 9. Delay sesudah HTB

Client	Minimum	Maximum	Average
Client 1	1 ms	8 ms	3 ms
Client 2	3 ms	10 ms	5 ms

Hasil pengujian menunjukkan bahwa berdasarkan standar TIPHON, nilai delay sebesar 3 ms dan 5 ms masih termasuk kategori sangat baik.". Hal ini menunjukkan bahwa HTB mampu membatasi throughput tanpa menyebabkan peningkatan RTT yang signifikan. Pada Client 1, nilai delay mengalami sedikit penurunan, sedangkan pada Client 2 terjadi peningkatan kecil, namun masih dalam batas yang rendah dan stabil.

3.2.3 Packet Loss Sesudah HTB

Berdasarkan Gambar 11 dan Gambar 12, hasil pengujian *ping* menunjukkan nilai *Sent* = 100, *Received* = 100, dan *Lost* = 0. Dengan demikian, packet loss setelah penerapan HTB tetap sebesar 0% pada kedua client.

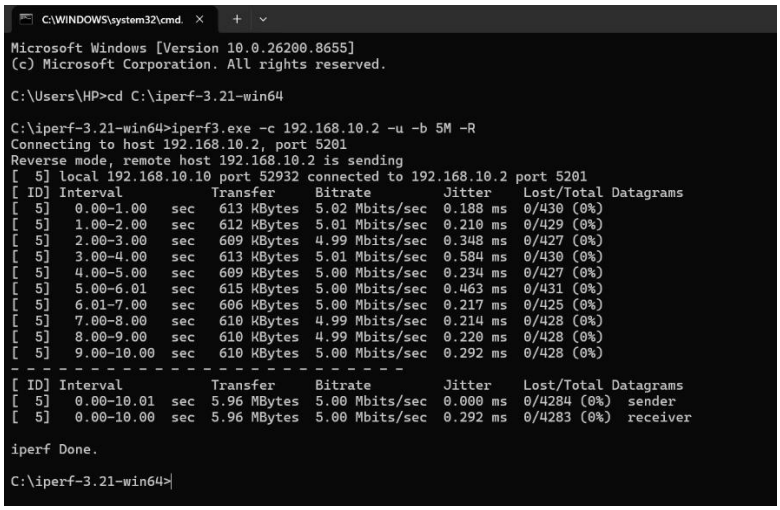
Tabel 10. Packet loss sesudah HTB

Client	Sent	Received	Lost	Packet Loss
Client 1	100	100	0	0%
Client 2	100	100	0	0%

Berdasarkan hasil pengujian, penerapan HTB tidak menyebabkan terjadinya *packet loss*. Hal ini menunjukkan bahwa mekanisme pembatasan bandwidth yang diterapkan tidak memengaruhi stabilitas konektivitas jaringan pada kedua client.

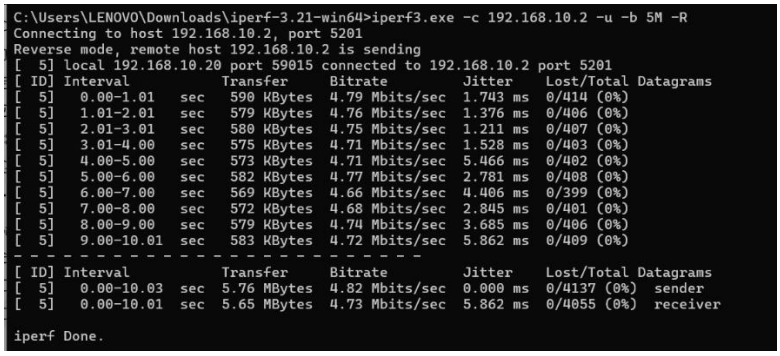
DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

3.2.4 Jitter Sesudah HTB



Gambar 13. Jitter Sesudah HTB pada Client 1

Pada pengujian menggunakan perintah *iperf3 -u -b 5M -R*, hasil yang diperoleh menunjukkan nilai *receiver* sebesar 5,00 Mbps/sec, *jitter* sebesar 0,292 ms, serta *lost/total datagrams* sebesar 0/4283 (0%). Hasil tersebut menunjukkan bahwa nilai jitter setelah penerapan *traffic shaping* masih tergolong rendah.



Gambar 14. Jitter Sesudah HTB pada Client 2

Berdasarkan hasil pengujian menggunakan *iperf3 -u -b 5M -R*, diperoleh nilai throughput *receiver* sebesar 4,73 Mbps/sec dengan nilai *jitter* sebesar 5,862 ms. Selain itu, tidak ditemukan kehilangan paket (*packet loss*) dengan nilai 0%. Hasil tersebut menunjukkan bahwa jaringan tetap stabil setelah penerapan HTB.

Tabel 11. jitter sesudah HTB

Client	Perintah Pengujian	Throughput Receiver	Jitter	Packet Loss
Client 1	iperf3 -u -b 5M -R	5,00 Mbps/sec	0,292 ms	0%
Client 2	iperf3 -u -b 5M -R	4,73 Mbps/sec	5,862 ms	0%

Hasil pengujian menunjukkan bahwa nilai jitter sesudah HTB sebesar 0,292 ms pada Client 1 dan 5,862 ms pada Client 2. Meskipun terdapat perbedaan nilai jitter, kedua client tidak mengalami packet loss (0%), sehingga kualitas layanan jaringan setelah penerapan HTB tetap stabil. Berdasarkan standar TIPHON, nilai jitter sebesar 0,292 ms dan 5,862 ms masih termasuk kategori sangat baik karena berada pada rentang 0–75 ms.

"Perbedaan nilai jitter antara Client 1 dan Client 2 dipengaruhi oleh perbedaan alokasi bandwidth. Client 2 memperoleh bandwidth lebih kecil sehingga mengalami antrean paket yang lebih panjang."

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

2.3 Perbandingan Sebelum dan Sesudah HTB

Tabel 12. Perbandingan utama yang dapat diverifikasi

Parameter	Client 1 Sebelum HTB	Client 1 Sesudah HTB	Client 2 Sebelum HTB	Client 2 Sesudah HTB
Throughput (*receiver*)	94,7 Mbps/sec	8,81 Mbps/sec	94,8 Mbps/sec	4,71 Mbps/sec
Delay rata-rata (RTT)	4 ms	3 ms	4 ms	5 ms
Packet Loss	0%	0%	0%	0%
Jitter	Tidak tersedia	0,292 ms	0,129 ms	5,862 ms*

Berdasarkan Tabel 11, throughput kedua client mengalami penurunan setelah penerapan HTB, yang menunjukkan bahwa mekanisme pembatasan bandwidth berjalan dengan baik. Meskipun throughput menurun, nilai delay tetap rendah dan packet loss tetap 0% pada kedua client. Hal ini menunjukkan bahwa penerapan HTB mampu mengendalikan bandwidth tanpa menurunkan stabilitas jaringan secara signifikan.

4. Kesimpulan

Berdasarkan hasil penelitian dan pengujian Quality of Service (QoS) yang telah dilakukan pada server Debian 12 menggunakan algoritma Hierarchical Token Bucket (HTB), dapat disimpulkan bahwa penerapan algoritma HTB berhasil mengelola dan mengendalikan penggunaan bandwidth sesuai dengan kebijakan yang telah ditetapkan. Sebelum penerapan HTB, kedua client memperoleh throughput yang relatif sama, yaitu sebesar 94,7 Mbps pada Client 1 dan 94,8 Mbps pada Client 2 karena belum terdapat mekanisme pembagian bandwidth. Setelah penerapan HTB, throughput mengalami perubahan menjadi 8,81 Mbps pada Client 1 dan 4,71 Mbps pada Client 2. Hasil tersebut menunjukkan bahwa mekanisme pembagian bandwidth menggunakan HTB berjalan dengan baik sesuai konfigurasi yang diterapkan, yaitu alokasi bandwidth sebesar 10 Mbps untuk Client 1 dan 5 Mbps untuk Client 2. Selain itu, penerapan HTB tidak memberikan dampak negatif yang signifikan terhadap kualitas layanan jaringan. Hasil pengujian menunjukkan bahwa nilai delay pada kedua client tetap berada pada kategori sangat baik berdasarkan standar TIPHON, yaitu berkisar antara 3 ms hingga 5 ms. Nilai packet loss juga tetap sebesar 0% baik sebelum maupun sesudah penerapan HTB, sehingga dapat disimpulkan bahwa tidak terdapat paket data yang hilang selama proses transmisi berlangsung.Berdasarkan hasil pengujian jitter, nilai jitter sesudah penerapan HTB diperoleh sebesar 0,292 ms pada Client 1 dan 5,862 ms pada Client 2. Meskipun terjadi peningkatan nilai jitter pada Client 2 akibat alokasi bandwidth yang lebih kecil, nilai tersebut masih berada dalam kategori sangat baik berdasarkan standar TIPHON. Hal ini menunjukkan bahwa penerapan HTB mampu mengatur distribusi bandwidth tanpa menurunkan stabilitas jaringan secara signifikan. Secara keseluruhan, algoritma Hierarchical Token Bucket (HTB) terbukti efektif digunakan sebagai metode manajemen bandwidth pada server Debian 12 karena mampu mengatur distribusi bandwidth secara lebih adil, terkontrol, dan efisien, sekaligus mempertahankan kualitas layanan jaringan agar tetap optimal.

Referensi

[1] Aprianto Budiman, M. Ficky Duskarnaen, and Hamidillah Ajie, "Analisis Quality of Service (Qos) Pada Jaringan Internet Smk Negeri 7 Jakarta," *PINTER J. Pendidik. Tek. Inform. dan Komput.*, vol. 4, no. 2, pp. 32–36, 2020, doi: 10.21009/pinter.4.2.6.

[2] N. K. Daulay, "ANALISIS QUALITY OF SERVICE (QOS) PADA JARINGAN INTERNET DI UNIVERSITAS BINA INSAN LUBUKLINGGAU MENGGUNAKAN METODE HIERARCHICAL TOKEN BUCKET (HTB) QUALITY ANALYSIS OF SERVICES IN INTERNET NETWORK IN INSAN LUBUKLINGGAU UNIVERSITY USING HIERARCHICAL TOK," vol. 3, 202. https://doi.org/10.32502/digital.v3i1.2471

[3] D. Fiantis, "Manajemen Trafik Menggunakan HTB Untuk Meningkatkan Kualitas Layanan IP Network," *J. Sains Terap.*, vol. 11, no. 1, pp. 88–103, 2021. https://doi.org/10.29244/jstsv.11.1.88-103

[4] M. Bosk, M. Gaji, S. Schwarzmann, S. Lange, and T. Zinner, "HTBQueue : A Hierarchical Token Bucket Implementation for the OMNeT ++ / INET Framework," pp. 1–9. https://doi.org/10.48550/arXiv.2109.12879

[5] S. Budin and I. Riadi, "Traffic Shaping Menggunakan Metode HTB (Hierarchical Token Bucket) pada Jaringan Nirkabel," vol. 1, no. 3, pp. 144–152, 2019, doi: 10.12928/biste.v1i3.1100. DOI: 10.12928/biste.v1i3.1100

[6] Y. Akbar, R. A. Pratama, S. Sugiyono, and F. Jawad, "and Civil Engineering Implementation of Mikrotik Network Management Using Quality of Service Features with the Hierarchical Token Bucket Algorithm," 2025. https://doi.org/10.61132/ijmecie.v1i4.271

[7] H. Dermawan and A. Hasibuan, "Metode Penelitian Eksperimen : Prinsip , Prosedur , dan Aplikasi dalam Penelitian Ilmiah," pp. 4–7, 2025. https://doi.org/10.56211/factory.v3i2.729

[8] A. Ardi, "Pemodelan dan Simulasi Algoritma Hierarchical Token Bucket (HTB) dalam Optimalisasi Penggunaan Bandwidth pada Mikrotik," vol. 4, no. 1, pp. 1–7, 2025. https://doi.org/10.30606/rjti.v4i1.3244

[9] Subhiyanto, "Implementasi Manajemen Bandwidth dengan Metode Hierarchical Token Bucket (HTB) dan Per Connection Queue

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

(PCQ) pada STMIK Antar Bangsa," *J. Tek. Inform.*, vol. 7, no. 2, pp. 69–73, 2021, doi: 10.51998/jti.v7i2.436.

[10] K. Oren *et al*., "PENERAPAN METODE HIERARCHICAL TOKEN BUCKET PADA JARINGAN INTERNET (STUDI KASUS :," vol. 13, no. 3, 1950. https://doi.org/10.23960/jitet.v13i3.6859

[11] F. Wadly, W. Fitriani, M. Muttaqin, and P. Veranda, "Implementation of Hierarchical Token Bucket (HTB) for Fair Bandwidth Allocation in Small Business Network Environments," pp. 1142–1153.

[12] E. Arfan Sugandi, D. Juardi, zhari Ali Ridha, U. Singaperbangsa Karawang, and K. Jawa Barat, "Implementasi Metode Hierarchical Token Bucket (Htb) Dalam Manajemen Bandwith Jaringan Internet (Studi Kasus Kantor Desa Sumber Sari)," *J. Mhs. Tek. Inform.*, vol. 7, no. 4, pp. 2749–2755, 2023. https://doi.org/10.36040/jati.v7i4.7194

[13] F. T. I. Usn, "Pengukuran Kualitas layanan Internet di Universitas Sembilanbelas November Kolaka Berdasarkan QoS dan QoE," vol. 12, no. 2, pp. 132–142. https://doi.org/10.36774/jusiti.v12i2.1425

[14] I. Ramadhan and A. Al Akbar, "Qos Analysis In The Application Of HTB At Computer Laboratory Of Engineering Faculty Of Universitas Muhammadiyah Bengkulu Analisis Qos Dalam Penerapan Htb Pada Laboraturium Komputer Fakultas Teknik Universitas Muhammadiyah Bengkulu," vol. 3, no. 2, pp. 151–158, 2024. https://doi.org/10.37676/jmcs.v3i2.6595

[15] I. Awaliyani and R. Z. A. Aziz, "Analisis Quality Of Service (QoS) Jaringan Internet Pelanggan Pada ISP Jalurdata . Net Menggunakan Metode Hierarchical Token Bucket (HTB) dan Per Connection Queue (PCQ)," pp. 425–436, 2024, doi: 10.33364/algoritma/v.21-2.1792.

[16] H. Kusbandono, T. Lestariningsih, T. Septianto, and P. N. Madiun, "Comparative Analysis of Quality of Service (QoS) on WLAN Network Bandwidth Management using HTB Method with PCQ," vol. 3, no. 10, pp. 4797–4810, 2024. https://doi.org/10.55927/eajmr.v3i10.11675

[17] ETSI, "Etsi tr 101 329-1," *Telecommun. Internet Protoc. Harmon. Over Networks Release 3; End-to-end Qual. Serv. TIPHON Syst. Part 1 Gen. Asp. Qual. Serv.*, vol. 2, pp. 1–201, 2000.

[18] R. K. Abdullah, S. P. Wibowo, and T. P. Fiqar, "Analisis QOS Jaringan Internet ITK dengan Metode Hierarchical Token Bucket," pp. 280–293. doi:10.35889/progresif.v20i1.1533

[19] H. Alamsyah and S. Somantri, "Perancangan dan Impelementasi QoS Di Mikrotik Menggunakan Metode HTB (Studi Kasus SMP MBS Al Karimah Cibadak)," *J. Tek. Inform. UNIKA St. Thomas*, vol. 07, pp. 14–22, 2022, doi: 10.54367/jtiust.v7i1.1648.

DOI: https://doi.org/10.31004/riggs.v5i2.11612
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)