IMPLEMENTASI DAN ANALISIS OPTIMASI *BANDWIDTH* DENGAN *QUEUE TREE* MENGGUNAKAN ALGORITMA HTB STUDI KASUS: PT. SUMBER KREASI FUMIKO

Taufik Rahman
Program Studi Manajemen Informatika
Akademi Manajemen Informatika dan Komputer Bina Sarana Informatika Jakarta
Jl. RS. Fatmawati No.24 Jakarta Selatan
Email: taufik@bsi.ac.id

ABSTRACT

Queue Tree is one of the Bandwidth management becomes essential for multi-service networks, more and varied applications that can be served by a network effect on the use of links in the network. The links that exist should be able to handle the needs of the application client will tesebut even though in a state of congestion, there must be a guarantee that the link still works properly despite an explosion in demand for application. Bandwidth management has an imPortant role in the distribution of quotas for these types of applications that can access the link, in addition to Bandwidth management to provide better Bandwidth guarantees to applications that have been allocated Bandwidth to continuously transmit data in accordance with the allocation though there is congestion in the network or the client even in Under certain circumstances when the Bandwidth allocation that is owned by an application or client is not used, then the Bandwidth Manager idle Bandwidth can be temporarily transferred to the classroom who are experiencing Queues, this gives the advantage of a class Queue accelerating loss while optimizing the use of existing links. Hierarchical Token Bucket (HTB) as the implementer of the available bandwidth management can be run on linux operating system platform is also available on Mikrotik Router which are Queue HTB tree is proper bandwidth management analysis, is expected to use a precise and accurate will make the network work optimally.

Keywords**:** *Bandwidth*, HTB, *Queue Tree*.

1. PENDAHULUAN

Jaringan komputer telah berkembang secara eksplosif. Dua dekade yang lalu, beberapa orang memiliki akses ke jaringan. Sekarang, komunikasi komputer telah menjadi bagian penting dari infrastruktur setiap organisasi. Jaringan digunakan dalam setiap aspek bisnis, termasuk periklanan, produksi, pengiriman, perencanaan, penagihan, dan akuntansi. Akibatnya, sebagian besar perusahaan memiliki jaringan ganda. Pada bidang pendidikan di semua tingkatan menggunakan jaringan komputer memudahkan siswa dan guru akses seketika ke informasi di perpustakaan *online* di seluruh dunia. Setiap negara dan kantor pemerintah daerah telah menggunakan jaringan, seperti halnya organisasi militer. Singkatnya, jaringan komputer telah berada di mana-mana.

Dengan berbagai koneksi *internet* yang ada seperti dial-up, ISDN, jaringan *lease line* adalah berbagai cara untuk membagi koneksi antara pengguna *internet*. Tetapi muncul pertanyaan seberapa efisien teknik berbagi yang digunakan sehubungan dengan kecepatan dan keamanan. Dengan munculnya teknologi nirkabel, berbagi koneksi *internet* dapat dibuat efisien, jika akan merancang sebuah jaringan LAN atau Wi-Fi pada sebuah instansi atau organisasi. Dalam studi ini, bertujuan membahas bagaimana mengatasi masalah pembagian bandwidth internet antar pengguna, kemudian di jelaskan pembagian *Bandwidth internet* antar pengguna, rincian perangkat keras, alokasi alamat IP dan topologi jaringan yang sebelum dan sesudah.

2. LANDASAN TEORI

2.1. *Bandwidth*

Bandwidth adalah kapasitas atau daya tampung kabel *ethernet* agar dapat dilewati trafik paket data dalam jumlah tertentu. *Bandwidth* juga bisa berarti jumlah konsumsi paket data per satuan waktu di*n*yatakan dengan satuan *bit per second* [bps]. *Bandwidth internet* di sediakan oleh provider *internet* dengan jumlah tertentu tergantung sewa pelanggan. Dengan *QoS* dapat mengatur agar *client* tidak

menghabiskan *Bandwidth* yang di sediakan oleh provider.

Bandwidth kontrol adalah seperangkat mekanisme yang mengontrol kecepatan data alokasi, variabilitas keterlambatan, pengiriman tepat waktu, dan kehandalan dalam pengiriman**.**

2.2. *Queue Tree*

Queue Tree hanya menciptakan satu arah antrian pada salah satu HTB. Hal ini juga satu-satunya cara bagaimana cara menambahkan antrian pada antarmuka yang terpisah. Dengan cara ini sangat mungkin untuk memudahkan konfigurasi *mangle*, tidak perlu menandai yang terpisah untuk *download* dan *upload*, *upload* hanya akan mendapatkan *interface public* dan *download* hanya akan mendapatkan *interface private*. Itu juga dimungkinkan untuk memiliki antrian ganda (contoh: prioritas lalu lintas pada *global-in* atau *global-out*, pembatasan per klien pada *interface* yang keluar).

Jika terdapat *simple Queue* dan *Queue Tree* di HTB yang sama, maka *simple Queue* yang akan mendapatkan traffic pertama. *Queue Tree* tidak diperintahkan, semua lalu lintas lewat bersama-sama. *Queue Tree* harus digunakan bila akan menggunakan data *rate* yang mutakhir pengalokasian ditentukan berdasarkan protokol, *Port* dan kelompok alamat IP. Awalnya harus menandai aliran paket dengan tanda pada /ip *firewall mangle* dan kemudian menggunakan tanda tersebut sebagai pengidentifikasi untuk aliran paket pada *Queue Tree*.

2.3. *Hierarichal Token Bucket (HTB)*

Hierarchical Token Bucket adalah suatu disiplin antrian *classful* yang berguna untuk menerapkan penanganan yang berbeda untuk berbagai jenis lalu lintas. Secara umum, hanya bisa mengatur satu antrian untuk *interface*. Pada *Router* antrian yang melekat pada HTB. Antrian dapat ditambahkan pada *simple Queue/Queue Tree* yang terdapat pada HTB. Dengan demikian memiliki beberapa sifat yang berasal dari *parent Queue*. Sebagai contoh, untuk mengatur kecepatan data maksimum pada *workgroup* dan kemudian mendistribusikan jumlah lalu lintas data antara anggota kelompok kerja.

Hierarchical Token Bucket (HTB) memungkinkan untuk membuat struktur antrian hirarki dan menentukan hubungan antar antrian, seperti "*parent-child*" atau "*child-child"*.

Antrian setidaknya memiliki satu anak (*child*) yang menjadi antrian dalam (*inner Queue*), semua antrian tanpa *child*-antrian daun (*leaf Queue*). *Leaf Queue* membuat konsumsi trafik yang sebenarnya, sedang *inner Queue* bertanggung jawab hanya pada distribusi trafik. Semua *leaf Queue* diperlakukan secara sama.

Setiap antrian pada HTB memiliki dua batas *rate*:

1. CIR (*Committed Information Rate*) - (*limit-at* di *router*) skenario terburuk, aliran akan mendapatkan jumlah lalu lintas tidak peduli berapapun (diasumsi kan benar-benar dapat mengirim data begitu banyak).
2. MIR (*Maximal Information Rate*) - (*Max-limit* di *router*) skenario kasus yang terbaik, laju yang mengalir bisa mendapatkan lebih, jika ada *Queue-parent* yang meluangkan *Bandwidth*.

Dengan kata lain, pada awalnya *limit-at* (CIR) dari semua antrian akan dipenuhi, jika antrian *child* akan mencoba untuk meminjam data *rate* yang diperlukan dari *parent* mereka dalam rangka untuk mencapai *Max-limit* (MIR) mereka.

2.4. *MikroTik Router*

MikroTik Router adalah sistem operasi *in*dependen berbasis l*in*ux untuk IA-32 *router* dan *router* yang kecil. Hal ini tidak memerlukan komponen tambahan dan tidak memiliki *pre-requirements* perangkat lunak. Hal ini dirancang dengan menggunakan antarmuka yang mudah, namun sangat *powerfull* yang memungkinkan seorang administrator jaringan untuk menggunakan struktur jaringan dan fungsi, yang akan membutuhkan pendidikan panjang di tempat lain hanya dengan mengikuti manual referensi dan bahkan tanpa itu.

MikroTik RouterOS dapat diinstal pada komputer standar menjadi *router* jaringan yang kuat. Dengan menambahkan *Interface* jaringan atau kartu jaringan pada komputer standar untuk memperluas kemampuan *router*. Dapat di *remote control* dengan aplikasi w*in*dows yang mudah dan *real-time* (*WinBox*).

MikroTik Router terdapat beberapa fitur seperti otentikasi, otorisasi, *client account*, *routing, mpls, Queues, firewall*, *IP Telephony* yang memungkinkan *Voice over IP* (VoIP) dan lain-lain.

3. METODE PENELITIAN

3.1. Studi Kepustakaan

Mempelajari literatur tentang teori dasar yang mendukung studi ini yaitu konfigurasi *router/gateway* dan manajemen *Bandwidth* dengan *Queue Tree* menggunakan algoritma HTB pada *mikrotik router***.**

3.2. Konsep dan Perancangan Sistem

Pada tahap ini dilakukan konsep kebutuhan sistem yang akan dibuat dan menjadi dasar untuk perancangan sistem, seperti besarnya kapasitas *bandwidth* yang tersedia, jumlah *client* serta proporsi alokasi *bandwidth* untuk masing-masing *client*, *topologi* jaringan.

1. Implementasi

Pada tahap ini dilakukan pembuatan *bandwidth control* dengan *Queue Tree* sesuai dengan analisis dan perancangan sistem. Pada tahap implementasi ini langkah-langkah yang dilakukan adalah:

1. Instalasi dan konfigurasi *router/gateway* (*MikroTik Router*).
2. Konfigurasi *mangle* untuk menandai paket.
3. Konfigurasi *bandwidth control Queue Tree*.
4. Konfigurasi *tool iperf*.

3.4. Pengujian

Pada tahap ini dilakukan pengujian sistem apakah berjalan sesuai dengan tujuan studi, yaitu:

1. Pengaturan algoritma *Hieraricahl Token Bucket* menggunakan *mangle* dan *address list*, dan *bandwidth control* dengan menggunakan *Bandwidth control Queue Tree*.
2. Menganalisa hasilnya yaitu menggunakan *tool monitoring bandwidth MikroTik Router* dengan *Queue Tree statistics*, *graphing* dan *tool iperf*.
3. Memantau aktifitas penggunaan *bandwidth* tiap-tiap *client* sesuai dengan rancangan percobaan.

3.5. Penerapan *Bandwidth*

Penerapan dilakukan dengan cara membangkitkan trafik dari *server* ke *client* dengan *tool Iperf* untuk mendapatkan nilai *Throughput* dilakukan masing-masing setiap lima detik untuk setiap percobaan, lalu diambil nilai rata-rata dari sampel yang diambil. Sedangkan untuk mendapatkan nilai *response time* dilakukan dengan cara melakukan *ping* dari *Mikrotik Router* ke *client*. Data yang akan diambil adalah *Throughput* dan *response time.*

1. Penerapan *Bandwidth* Pra-HTB

Sebelum *Hierarchical Token Bucket* diterapkan pada jaringan, dilakukan suatu pengukuran terhadap k*in*erja jaringan. Hal ini dimaksudkan agar terlihat bagaimana efek yang akan terjadi setelah diterapkannya *Hierarchical Token Bucket* pada jaringan, terutama pada masalah *Throughput* dan *respon time* dari jaringan.

1.Pengukuran *Throughput* (dengan menggunakan *Iperf*)

a.Trafik besar akan dikirim ke masing-masing *client*, yang diharapkan akan mengakibatkan pemakaian *bandwidth* secara penuh, misal dengan mengirimkan sejumlah paket. *Output* yang akan diambil:

a). *Throughput client* 1
b). *Throughput client* 2

b. Trafik dikirim ke kedua *client* secara bersamaan yang mengakibatkan pemakaian *bandwidth* secara penuh, misal dengan mengirimkan sejumlah paket secara bersamaan ke kedua *client*. *Output* yang akan diambil:

a). *Throughput client* 1
b). *Throughput client* 2

2. Pengukuran *response time* (dengan menggunakan p*in*g) paket size 200 *byte*

a. Mengukur *response time* server-*client*1 pada saat tidak ada pemakaian
b. Mengukur *response time* server-*client*1 pada saat pemakaian BW penuh oleh *client*1
c. Mengukur *response time* server-*client*2 pada saat pemakaian BW penuh oleh *client*2
d. Mengukur *response time* server-*client*1 pada saat pemakaian BW penuh oleh *client*1 dan 2
e. Mengukur *response time* server-*client*2 pada saat pemakaian BW penuh oleh *client*1 dan 2

2. Penerapan *Bandwidth Queue Tree* Menggunakan HTB

a. Penerapan Pembagian *Bandwidth* Berdasarkan *Port* dan Aplikasi Tertentu

Sebelum dilakukan pengukuran, dibuat pembagian *bandwidth* kepada dua kelas yang berbeda *Port* dengan cara mengimplementasikan *Hierarchical Token Bucket* pada *output interface* dari server. Kelas pertama (semua trafik yang menuju *Port* 80) diberikan jatah *Bandwidth* 384 kbit/s, dan kelas kedua (semua trafik yang menuju *Port* 4899) diberikan jatah *Bandwidth* 128 kbit/s. Sehingga *bandwidth* maksimal l*in*k adalah 512 kbit/s. Masing-masing kelas, satu sama lain dapat sal*in*g memakai *bandwidth* yang sedang tidak digunakan. Skenario di atas diterjemahkan dengan per*in*tah sebagai berikut:

1. [adm*in*@MT_YONGKI] ip *firewall mangle*> *add chain=prerouting protocol=tcp* dst-*Port*=80 *action=mark-*

connection new-connection-mark=HTTP_*conn passthrough*=yes
2. [adm*in*@MT_YONGKI] ip *firewall mangle> add chain=prerouting protocol=tcp connection mark*=HTTP_*conn action=mark-packet new-packet-mark*=HTTP *passthrough=no*
3. [adm*in*@MT_YONGKI] *Queue Tree> add name*=aplikasi *parent*=global-total *packet-mark= Queue-type=default priority*=3 *max-limit*=524288
4. [adm*in*@MT_YONGKI] *Queue Tree> add name*=HTTP *parent*=aplikasi *packet-mark*=HTTP_*conn Queue-type=default priority*=4 *max-limit*=393216
5. [adm*in*@MT_YONGKI] *Queue Tree> add name*=RADM*IN parent*=aplikasi *packet-mark*=RADM*IN_conn Queue-type=default priority*=4 *max-limit*=131072

b. Penerapan Pengaruh Perioritas dalam Manajemen *Bandwidth*

Sebelum dilakukan pengukuran, dibuat pembagian *bandwidth* kepada tiga kelas yang berbeda *Port* dengan cara mengimplementasikan *Hierarchical Token Bucket* pada *output interface* dari *router*.

Perioritas pertama, yaitu trafik datacenter 192.168.1.250, diberikan jatah *Bandwidth* 100Mb/s dari global-total. Perioritas kedua, yaitu trafik DVR, diberikan jatah *bandwidth* 100Mb/s dari global-total. Perioritas ketiga, yaitu *parent-download*, diberikan jatah *bandwidth* 4Mb/s dari global-*out*, dan *parent-upload* diberikan jatah *bandwidth* 2Mb/s dari global-*in*. Perioritas keempat, yaitu trafik aplikasi, diberikan jatah *bandwidth* 512 kbit/s. Perioritas kedelapan, yaitu semua trafik yang menggunakan P2P, diberikan jatah *bandwidth* 128 kbit/s. Skenario diatas dapat diterjemahkan sebagai berikut:

1. [adm*in*@MT_YONGKI] *Queue Tree> add name*=datacenter *parent*=global-total *packet-mark*=datacenter *Queue-type=default priority*=1 *max-limit*=104857600
2. [adm*in*@MT_YONGKI] *Queue Tree> add name*=DVR *parent*=global-*out Queue-type=default priority*=2 *max-limit*=104857600
3. [adm*in*@MT_YONGKI] *Queue Tree> add name=Port* dan aplikasi *parent*=global-total *Queue-type=default priority*=3 *max-limit*=524288
4. [adm*in*@MT_YONGKI] *Queue Tree> add name=Parent-Download parent*=global-*out Queue-type=default priority*=4 *max-limit*=419430
5. [adm*in*@MT_YONGKI] *Queue Tree> add name=Parent-Upload parent*=global-*in Queue-type=default priority*=4 *max-limit*=2097152
6. [adm*in*@MT_YONGKI] *Queue Tree> add name*=all-P2P *parent*=global-total *packet-mark*=koneksi-P2P *Queue-type=default priority*=8 *max-limit*=131072

4. HASIL DAN PEMBAHASAN

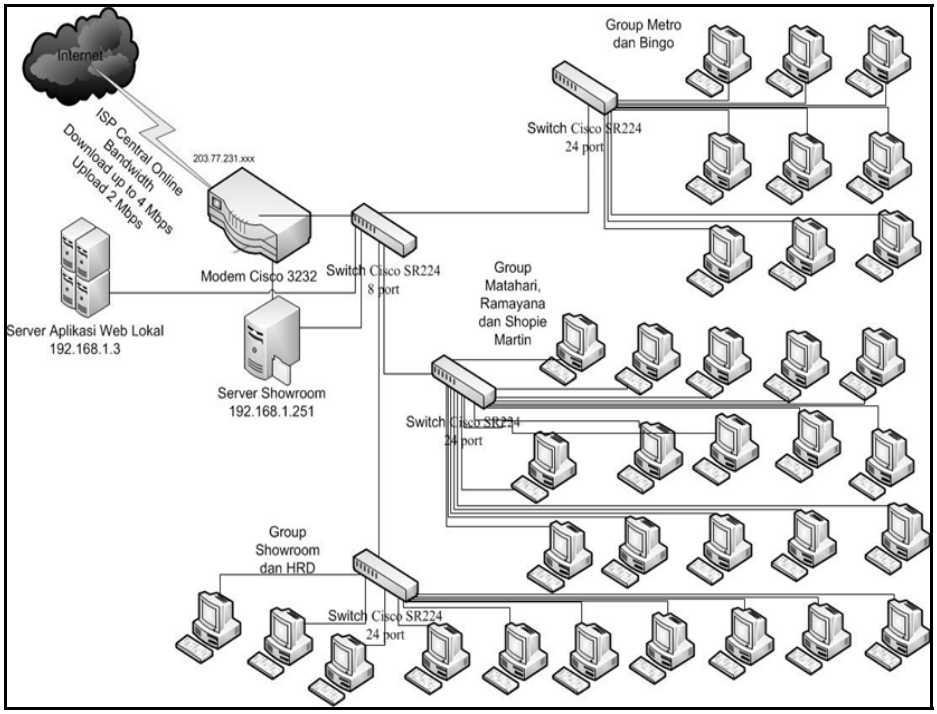
4.1. Skema *Topologi Network* PT. SUMBER KREASI FUMIKO

Untuk mempermudah analisa terhadap penggunaan *bandwidth* pada PT. SUMBER KREASI FUMIKO perlu didukung oleh skema *topologi network*. Skema *topologi network* yang dimaksud adalah skema *topologi* sebelum penerapan *Queue Tree* dengan algoritma HTB. Pada Gambar 4.1 terlihat jelas bahwa koneksi *internet* dari *modem ADSL* langsung dihubungkan dengan *switch* kemudian ke komputer *client*. Belum adanya pembagian alokasi *bandwidth* pada setip *client*, sehingga *client* yang lebih dulu mengakses *internet* maka *client* tersebut yang akan mendapatkan *bandwidth* yang lebih besar.

Pada skema topologi yang terdapat pada gambar 1, menunjukan komponen *network* yang digunakan terlihat pada tabel 1 dibawah ini.

Tabel 1. Komponen *Network*

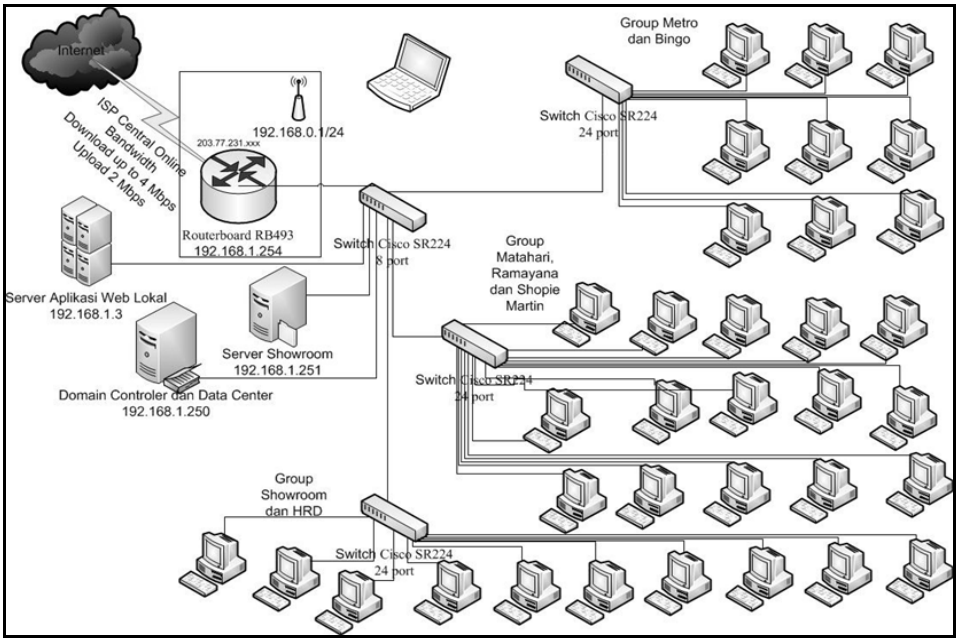
No	JENIS KOMPONEN	FUNGSI
1	L*in*e Telephone	Koneksi PSTN Sentral Telepon
2	Modem Cisco 3232	Koneksi *Internet* ADSL 4 Mbps
3	Switch	Koneksi Modem ADSL, dan Koneksi *Client*



Gambar 1. Topologi Network PT. SUMBER KREASI FUMIKO
Sumber : PT. SUMBER KREASI FUMIKO

4.2. Rancangan Percobaan

Untuk melakukan penerapan *Queue Tree* dengan algoritma HTB dan *bandwidth control*, maka dibuatlah *topologi* dengan penambahan *MikroTik Router* dan perangkat keras lainnya, dapat dilihat pada gambar 2.



Gambar 2. Topologi jaringan dengan penerapan Queue Tree dengan algoritma HTB dan bandwidth control

4.3. Hasil Pembahasan penerapan *Bandwidth* Pra-HTB

Dilakukan dengan cara membangkitkan trafik dari server ke *client* dengan *tool iperf* untuk mendapatkan nilai *Throughput* penerapan dilakukan masing-masing setiap lima detik untuk setiap percobaan, lalu diambil nilai rata-rata dari sampel yang diambil. Sedangkan untuk mendapatkan nilai *response time* dilakukan dengan cara melakukan p*i*ng dari *Mikrotik Router* ke *client*.

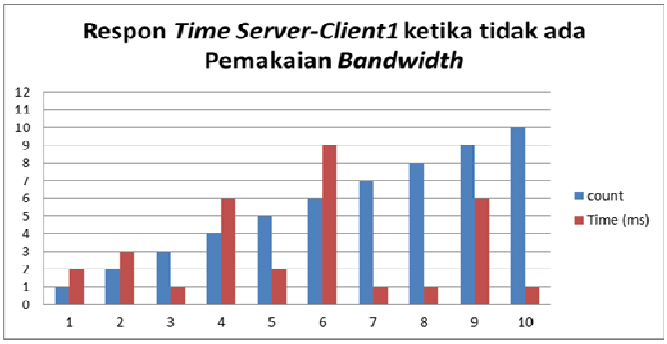
1. Trafik dikirim hanya ke *client* 1

Tabel 2 menunjukkan bagaimana performansi jaringan *client1* dengan mengirimkan sejumlah paket ketika *client1* memakai *bandwidth* penuh sendirian sebelum diterapkannya HTB.

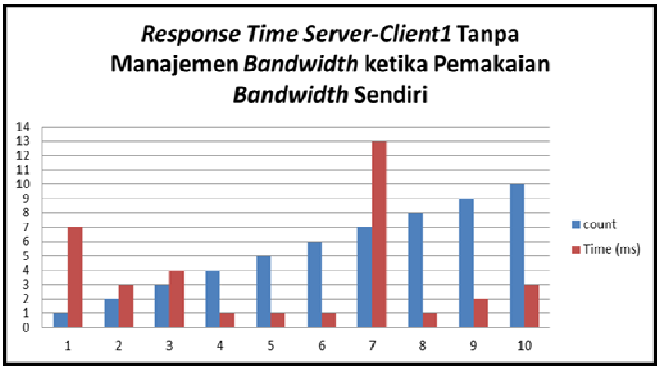
Tabel 2. Monitor*ing Client* 1 (192.168.1.28) Tanpa Manajemen *Bandwidth* ketika Pemakaian *bandwidth* Sendiri

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.1	6.25	5.18
6	10.2	5.50	4.52
11	10.2	6.38	5.25
16	10.2	5.75	4.71
21	10.3	6.12	4.98
26	10.2	5.62	4.64
31	10.2	6.38	5.25
36	10.2	6.25	5.16
41	10.2	6.00	4.94
46	10.4	6.25	5.06
Average	10.22	6.05	4.969

Gambar 3 menunjukkan response time server ke *client1* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client1* tidak memakai *Bandwidth* sendirian sebelum diterapkannya HTB.



Gambar 3. *Response Time* Server-*Client*1 Tanpa Manajemen *Bandwidth* ketika tidak ada Pemakaian *Bandwidth*



Gambar 4. *Response Time* Server-*Client*1 Tanpa Manajemen *Bandwidth* ketika Pemakaian *Bandwidth* Sendiri

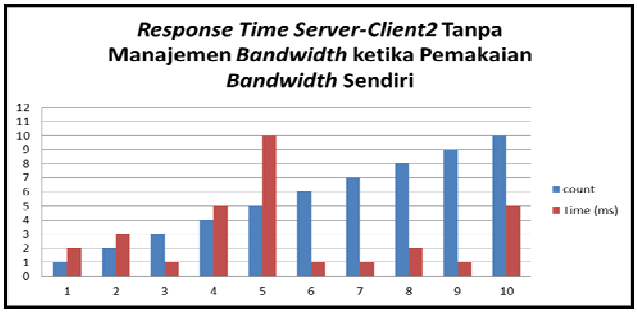
2. Trafik dikirim hanya ke *client* 2

Tabel 3 menunjukkan bagaimana performansi jaringan *client 2* dengan mengirimkan sejumlah paket ketika *client2* memakai *bandwidth* penuh sendirian sebelum diterapkannya HTB.

Tabel 3. Monitor*ing Client* 2 (192.168.1.49) Tanpa Manajemen *Bandwidth* ketikapemakaian *bandwidth* sendiri

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.1	6.00	4.99
6	10.1	6.25	5.20
11	10.0	6.00	5.03
16	10.2	6.25	5.16
21	10.1	5.88	4.87
26	10.0	5.88	4.91
31	10.0	6.00	5.02
36	10.1	5.88	4.87
41	10.1	6.25	5.21
46	10.1	5.62	4.65
Average	10.08	6.001	4.991

Gambar 5 menunjukkan *response time* server ke *client* 2 dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client* 2 memakai *bandwidth* penuh sendirian sebelum diterapkannya HTB.



Gambar 5. *Response Time* Server-*Client*2 Tanpa Manajemen *Bandwidth* ketika Pemakaian *Bandwidth* Sendiri

3. Trafik dikirim ke *client* 1 dan 2

a. Performansi jaringan *client 1*

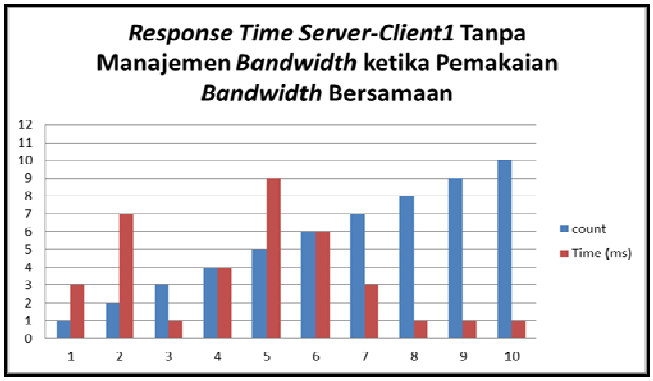
Tabel 4 menunjukkan bagaimana performansi jaringan *client 1* dengan mengirimkan sejumlah paket ketika *client 1* dan *client 2* memakai *bandwidth* penuh secara bersamaan sebelum diterapkannya HTB.

Tabel 4. Performansi Jaringan *Client* 1 (192.168.1.28) Tanpa Manajemen *bandwidth* ketika Pemakaian *bandwidth* Bersamaan

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.2	6.00	4.94
6	10.1	5.75	4.79
11	10.1	6.12	5.07
16	10.0	5.88	4.93
21	10.1	6.38	5.30
26	10.3	5.75	4.70
31	10.1	6.25	5.18
36	10.2	6.25	5.13

41	10.3	6.00	4.89
46	10.2	5.50	4.54
Average	10.16	5.988	4.947

Gambar 6 menunjukkan *response time client1* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client1 & client2* memakai *bandwidth* penuh secara bersamaan sebelum diterapkannya HTB.



Gambar 6. *Response Time* Server-*Client*1 Tanpa Manajemen *Bandwidth* ketika Pemakaian *Bandwidth* Bersamaan

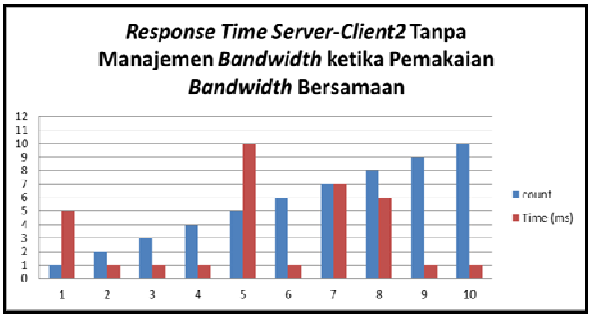
b. Performansi jaringan *client 2*

Tabel 5 menunjukkan bagaimana performansi jaringan *client 2* dengan mengirimkan sejumlah paket ketika *client 1* dan *client 2* memakai *Bandwidth* penuh secara bersamaan sebelum diterapkannya HTB.

Tabel 5. Performansi Jaringan *Client* 2 (192.168.1.49) Tanpa Manajemen *Bandwidth* ketika Pemakaian *Bandwidth* Bersamaan

time	*in*terval (Sec)	transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.1	6.12	5.10
6	10.0	5.25	4.41
11	10.0	5.88	4.90
16	10.1	5.25	4.35
21	10.0	6.25	5.23
26	10.1	6.12	5.10
31	10.1	5.62	4.65
36	10.1	6.12	5.11
41	10.1	5.88	4.90
46	10.0	6.12	5.14
Average	10.06	5.861	4.889

Gambar 7 menunjukkan *response time client 2* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client 1 & client 2* memakai *bandwidth* penuh secara bersamaan sebelum diterapkannya HTB.



Gambar 7. *Response Time* Server-*Client*2 Tanpa Manajemen *Bandwidth* ketika Pemakaian *Bandwidth* Bersamaan

4.4. Hasil Pembahasan Penerapan *Bandwidth* HTB

a. Hasil Penerapan Pembagian *Bandwidth* Berdasarkan *Port* dan Aplikasi Tertentu

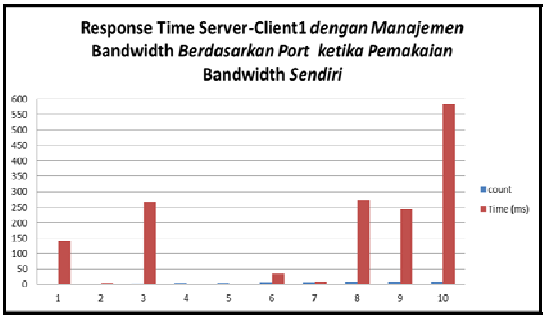
1. Trafik dikirim hanya ke *client* 1

Tabel 6 menunjukkan bagaimana performansi jaringan *client1* dengan mengirimkan sejumlah paket ketika *client1* memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan *Port*.

Tabel 6. Performansi Jaringan *Client1* (192.168.1.28) dengan Manajemen *Bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Sendiri.

Time	*Interval* (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.1	6.25	5.21
6	10.0	6.12	5.14
11	10.1	6.25	5.17
16	10.2	5.50	4.53
21	10.0	6.25	5.23
26	10.7	6.25	4.91
31	10.1	6.12	5.07
36	10.7	6.50	5.09
41	10.4	5.62	4.55
46	10.0	6.25	5.23
Average	10.23	6.111	5.013

Gambar 8 menunjukkan bagaimana *response time server* ke *client 1* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client 1* memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan *Port*.



Gambar 8. *Response Time Server-Client1* dengan Manajemen *Bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Sendiri

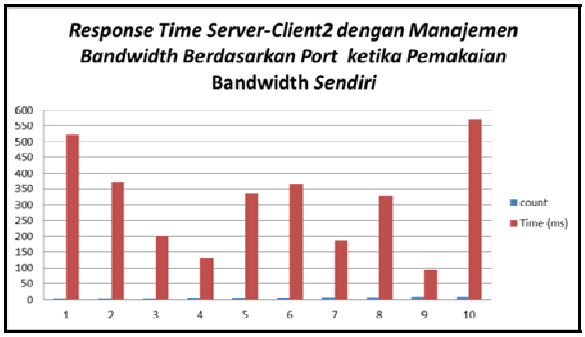
2. Trafik dikirim hanya ke *client* 2

Tabel 7 menunjukkan bagaimana performansi jaringan *client 2* dengan mengirimkan sejumlah paket ketika *client 2* memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan *Port*.

Tabel 7. Performansi Jaringan *Client 2* (192.168.1.49) dengan Manajemen *Bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Sendiri

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.2	6.00	4.92
6	10.1	6.25	5.21
11	10.0	6.00	5.02
16	10.2	6.00	4.96
21	10.0	6.25	5.22
26	10.0	5.62	4.73
31	10.0	5.88	4.92
36	10.0	5.50	4.59
41	10.1	6.25	5.20
46	10.0	6.12	5.12
Average	10.06	5.987	4.989

Gambar 9 menunjukkan respon *time* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client 2* memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan *Port*.



Gambar 9. *Response Time* Server-*Client*2 dengan Manajemen *Bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Sendiri

3. Trafik dikirim ke *client* 1 dan *client* 2

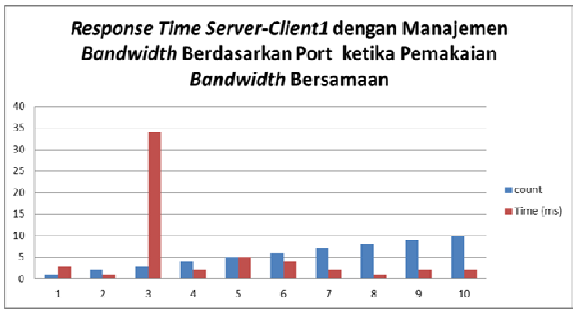
a. Performansi jaringan pada *client* 1

Tabel 8 menunjukkan bagaimana performansi jaringan *client 1* dengan mengirimkan sejumlah paket ketika *client 1* & *client2* memakai *bandwidth* penuh secara bersamaan setelah diterapkannya HTB berdasarkan *Port*.

Tabel 8. Performansi Jaringan *Client1* (192.168.1.28) dengan Manajemen *Bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Bersamaan

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.2	6.00	4.94
6	10.1	6.25	5.20
11	10.2	5.88	4.85
16	10.1	6.38	5.29
21	10.1	6.25	5.17
26	10.1	5.25	4.36
31	10.2	6.12	5.03
36	10.2	6.12	5.06
41	10.0	5.75	4.81
46	10.2	6.25	5.16
Average	10.14	6.025	4.987

Gambar 10 menunjukkan *response time server* ke *client1* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client1 & client2* memakai *bandwidth* penuh secara bersamaan setelah diterapkannya HTB berdasarkan *Port*.



Gambar 10. *Response Time Server-Client1* dengan Manajemen *Bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Bersamaan

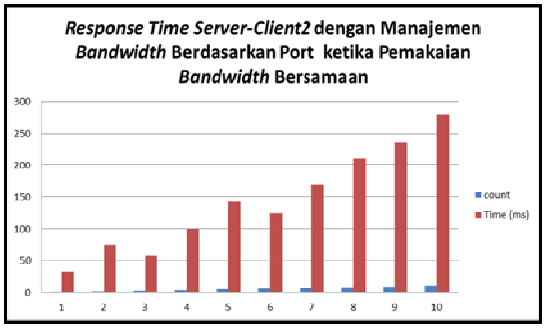
b. Performansi jaringan pada *client* 2

Tabel 9 menunjukkan bagaimana performansi jaringan *client2* dengan mengirimkan sejumlah paket ketika *client1 & client2* memakai *bandwidth* penuh secara bersamaan setelah diterapkannya HTB berdasarkan *Port*.

Tabel 9. Performansi Jaringan *Client2* (192.168.1.49) dengan Manajemen *bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Bersamaan

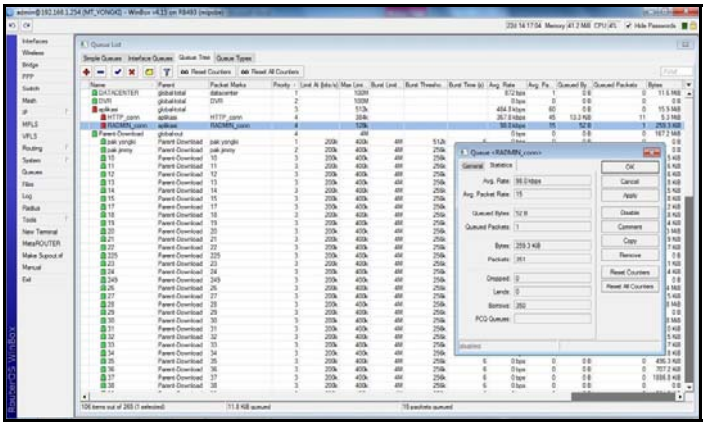
Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.1	5.62	4.68
6	10.0	6.25	5.24
11	10.1	5.75	4.76
16	10.1	5.88	4.88
21	10.0	6.25	5.24
26	10.2	6.38	5.26
31	10.0	6.00	5.03
36	10.1	5.75	4.79
41	10.1	6.12	5.07
46	10.1	6.00	4.98
Average	10.08	6	4.993

Gambar 11 menunjukkan *response time server* ke *client2* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client1 & client2* memakai *bandwidth* penuh secara bersamaan setelah diterapkannya HTB berdasarkan *Port*.



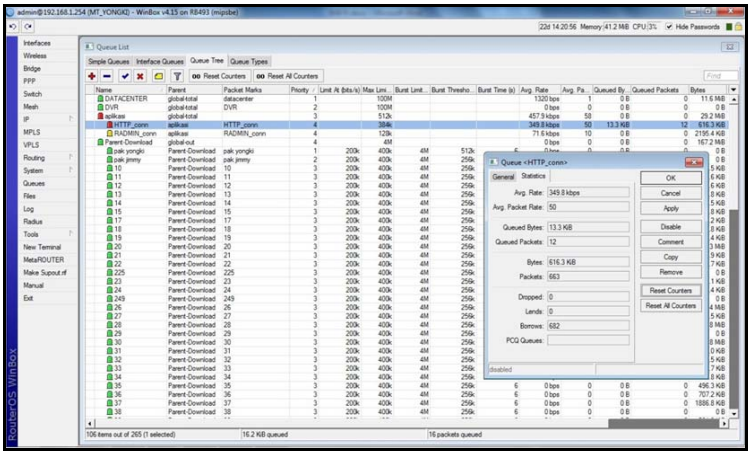
Gambar 11. *Response Time Server-Client2* dengan Manajemen *Bandwidth* Berdasarkan *Port* ketika Pemakaian *Bandwidth* Bersamaan

Setelah dilakukan menguji dan dan melihat hasil dari sebelum di terapkan *Queue Tree* dengan metode HTB, Selanjutnya hasil dari pada pembagian *bandwidth* berdasarkan *Port* dan penerapannya pada aplikasi tertentu. Pada aplikasi Radmin, yang mana untuk aplikasi ini diberi *bandwidth* maksimal 128kbps dan setelah mencapai dan melebihi *bandwidth* yang diberikan, maka *Queue* akan berwarna merah pada gambar 12.



Gambar 12. Statistik *Queue Tree* Pada Aplikasi Radmin

Hal ini berarti membuktikan bekerjanya *Queue Tree* yang diberlakukan dan setelah melewati batas yang diberikan maka *Queue Tree* akan menekan koneksi terhadap *Queue* tersebut. Setelah pengujian pada aplikasi Radmin, dilakukan juga pengujian pada *Port* 80 atau aplikasi web yang bentukan *Queue Tree* nya pada gambar di atas dengan *parent* nya dari *global-total* dengan perioritas empat dan *bandwidth* maksimal 384 kbps.



Gambar 13. Statistik *Queue Tree* Pada Aplikasi Web atau *Port* 80

Pada gambar 13 adalah hasil statistik *Queue Tree* pada aplikasi web, yang mana untuk aplikasi ini diberi *bandwidth* maksimal 384kbps dan setelah mencapai dan melebihi *bandwidth* yang diberikan, maka *Queue* akan berwarna merah. Hal ini berarti membuktikan bekerjanya *Queue Tree* yang diberlakukan dan setelah melewati batas yang diberikan maka *Queue Tree* akan menekan koneksi terhadap *Queue* tersebut.

b. Hasil Penerapan Penggunaan Perioritas dalam Manajemen *Bandwidth*

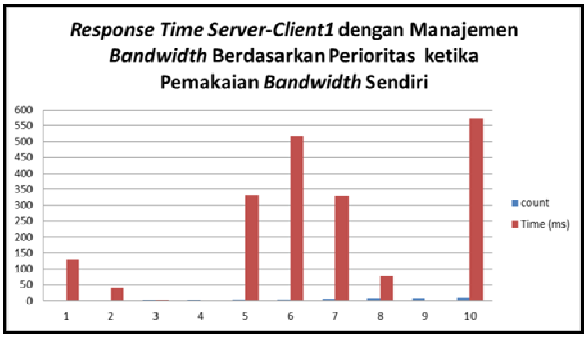
1. Trafik dikirim hanya ke *client* 1

Tabel 10 menunjukkan bagaimana performansi jaringan *client1* dengan mengirimkan sejumlah paket ketika *client1* dengan perioritas 1 memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan perioritas.

Tabel 10. Performansi Jaringan Client1 (192.168.1.28) Dengan Manajemen Bandwidth Berdasarkan Perioritas Ketika Pemakaian Bandwidth

Time	*I*nterval (Sec)	Transfer (M*Bytes*)	*Bandwidth* (Mbps)
1	10.2	6.25	5.13
6	10.1	6.00	4.99
11	10.1	6.25	5.21
16	10.0	6.12	5.12
21	10.3	6.38	5.21
26	10.1	6.38	5.29
31	10.2	6.38	5.27
36	10.1	6.38	5.27
41	10.2	6.00	4.91
46	10.1	5.88	4.89
Average	10.14	6.202	5.129

Gambar 15 *response time server* ke *client1* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client1* dengan perioritas 1 memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan perioritas.



Gambar 15. Response Time Server-Client1 dengan Manajemen Bandwidth Berdasarkan Perioritas ketika Pemakaian Bandwidth Sendiri

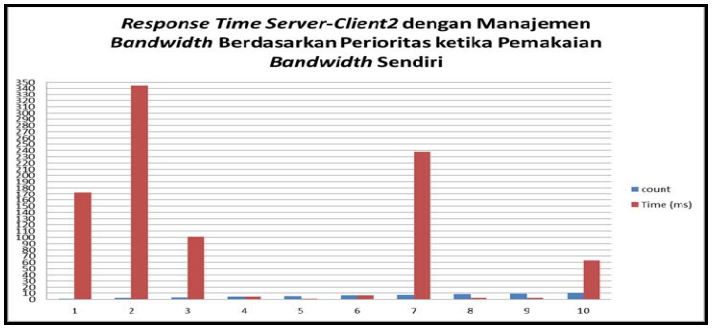
2. Trafik dikirim hanya ke *client* 2

Tabel 11 menunjukkan bagaimana performansi jaringan *client2* dengan mengirimkan sejumlah paket ketika *client2* dengan perioritas 2 memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan perioritas.

Tabel 11. Performansi Jaringan Client2 (192.168.1.11) dengan Manajemen Bandwidth Berdasarkan perioritas ketika Pemakaian Bandwidth Sendiri.

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.2	6.38	5.27
6	10.0	6.12	5.13
11	10.1	6.38	5.28
16	10.1	6.25	5.21
21	10.0	6.12	5.12
26	10.0	6.38	5.33
31	10.2	5.62	4.64
36	10.1	6.00	5.00
41	10.0	6.12	5.12
46	10.0	6.38	5.35
Average	10.07	6.175	5.145

Gambar 14 menunjukkan *response time server* ke *client2* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client2* dengan perioritas 2 memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan perioritas.



Gambar 14. *Response Time Server-Client2* dengan Manajemen *Bandwidth* Berdasarkan Perioritas ketika Pemakaian *Bandwidth* Sendiri.

3. **Trafik dikirim hanya ke *client* 3**

Tabel 12 menunjukkan bagaimana performansi jaringan *client*3 dengan mengirimkan sejumlah paket ketika *client*3 dengan perioritas 3 memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan perioritas.

Tabel 12. Performansi Jaringan *Client3* (192.168.1.49) dengan Manajemen *Bandwidth* Berdasarkan perioritas ketika Pemakaian *Bandwidth*.

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	0.1	6.00	4.99
6	10.0	6.12	5.13
11	0.1	6.12	5.08
16	10.0	6.12	5.12
21	10.2	6.00	4.93
26	10.0	6.12	5.12
31	10.0	5.62	4.73
36	10.0	6.12	5.13
41	10.0	6.12	5.12
46	10.1	5.88	4.90
Average	10.05	6.022	5.025

Gambar 15 menunjukkan *response time server* ke *client*3 dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika *client*3 dengan perioritas 3 memakai *bandwidth* penuh sendirian setelah diterapkannya HTB berdasarkan perioritas.

Gambar 15. Response Time Server-Client3 dengan Manajemen Bandwidth Berdasarkan Perioritas ketika Pemakaian Bandwidth Sendiri.

4. **Trafik dikirim ke *client* 1, *client* 2 dan *client* 3**

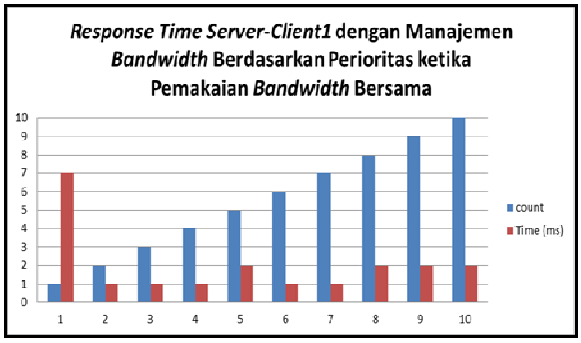
   a. Performansi jaringan *client1*

   Tabel 13 menunjukkan bagaimana performansi jaringan *client1* dengan mengirimkan sejumlah paket ketika trafik dikirimkan ke *client1*, *client2* dan *client*3 secara bersamaan setelah diterapkannya HTB berdasarkan perioritas.

Tabel 13. Performansi Jaringan Client1 (192.168.1.28) dengan Manajemen Bandwidth Berdasarkan perioritas ketika Pemakaian Bandwidth Bersamaan.

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.5	6.25	5.01
6	10.0	6.25	5.24
11	10.1	5.50	4.58
16	10.0	6.25	5.22
21	10.1	6.25	5.19
26	10.3	.00	4.89
31	0.1	6.25	5.20
36	0.7	6.38	5.00
41	10.2	5.88	4.82
46	10.2	6.25	5.13
Average	10.22	6.126	5.028

Gambar 16 menunjukkan *response time server* ke *client1* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika trafik dikirimkan ke *client1*, *client2* dan *client*3 secara bersamaan setelah diterapkannya HTB berdasarkan perioritas.



Gambar 16 Response Time Server-Client1 dengan Manajemen Bandwidth Berdasarkan Perioritas ketika Pemakaian Bandwidth Bersama

b. Performansi jaringan *client2*

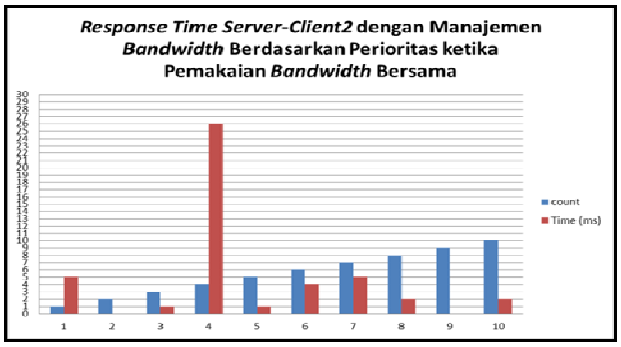
Tabel 14 menunjukkan bagaimana performansi jaringan *client2* dengan mengirimkan sejumlah paket ketika trafik dikirimkan ke *client1*, *client2* dan *client*3 secara bersamaan setelah diterapkannya HTB berdasarkan perioritas.

Tabel 14. Performansi Jaringan *Client2* (192.168.1.11) dengan Manajemen *Bandwidth* Berdasarkan perioritas ketika Pemakaian *Bandwidth* Bersamaan

Time	*I*nterval (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.2	6.38	5.26
6	10.1	6.38	5.28
11	10.1	5.88	4.88
16	10.2	6.38	5.27
21	10.0	6.25	5.25
26	10.0	6.25	5.22
31	10.1	6.38	5.30
36	10.0	6.25	5.25

41	10.0	5.38	4.50
46	10.1	6.00	4.98
Average	10.08	6.153	5.119

Gambar 17 menunjukkan *response time server* ke *client2* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika trafik dikirimkan ke *client1*, *client2* dan *client3* secara bersamaan setelah diterapkannya HTB berdasarkan perioritas.



Gambar 17. Response Time Server-Client2 dengan Manajemen Bandwidth Berdasarkan Perioritas ketika Pemakaian Bandwidth Bersama.

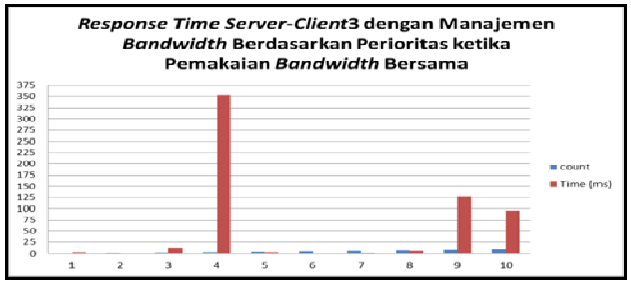
c. Performansi jaringan *client3*

Tabel 15 menunjukkan bagaimana performansi jaringan *client3* dengan mengirimkan sejumlah paket ketika trafik dikirimkan ke *client1*, *client2* dan *client3* secara bersamaan setelah diterapkannya HTB berdasarkan perioritas.

Tabel 15 Performansi Jaringan Client3 (192.168.1.49) dengan Manajemen Bandwidth Berdasarkan Port ketika Pemakaian Bandwidth Bersamaan.

Time	*Interval* (Sec)	Transfer (M*Byte*)	*Bandwidth* (Mbps)
1	10.1	6.12	5.10
6	10.0	6.25	5.24
11	10.0	6.12	5.13
16	10.0	6.12	5.14
21	10.0	6.25	5.24
26	10.0	6.25	5.23
31	10.1	6.25	5.20
36	10.0	6.12	5.12
41	10.2	5.88	4.84
46	10.0	6.12	5.15
Average	10.04	6.148	5.139

Gambar 18 menunjukkan *response time server* ke *client3* dengan mengirimkan paket 200, TTL 128, sebanyak 10 kali dengan *timeout* 1000ms ketika trafik dikirimkan ke *client1*, *client2* dan *client3* secara bersamaan setelah diterapkannya HTB berdasarkan perioritas.



Gambar 18. Response Time Server-Client3 dengan Manajemen Bandwidth Berdasarkan Perioritas ketika Pemakaian Bandwidth Bersama.

Setelah melakukan implementasi dan analisa, akhirnya sampai pada kesimpulan teknis, yaitu mengambil nilai *average* dari hasil *Throughput* pra-HTB, hasil *Throughput* HTB berdasarkan *Port* dan aplikasi dan hasil *Throughput* pengaruh dari pada perioritas dalam manajemen *bandwidth* baik secara penggunaan *bandwidth* penuh sendiri oleh *client1* atau *client2*, maupun penggunaan *bandwidth* secara bersamaan. Tabel 16 menunjukkan kesimpulan teknis *average Throughput* atau *bandwidth* aktual yang digunakan untuk mentransfer data dalam waktu tertentu.

Tabel 16. Kesimpulan Teknis *Average Throughput*

1.pra HTB:			
*client*1	10.22	6.05	4.969
*client*2	10.08	6.001	4.991
average			**4.98**
bersamaan:			
*client*1	10.16	5.988	4.947
*client*2	10.06	5.861	4.889
average			**4.918**
2.HTB *Port* dan aplikasi:			
*client*1	10.23	6.111	5.013
*client*2	10.06	5.987	4.989
average			**5.001**
bersamaan;			
*client*1	10.14	6.025	4.987
*client*2	10.08		4.993
average			**4.99**
3.Perioritas:			
*client*1	10.14	6.202	5.129
*client*2	10.07	6.175	5.145
*client*3	10.05	6.022	5.025
average			**5.099667**
bersamaan;			
*client*1	10.22	6.126	5.028
*client*2	10.08	6.153	5.119
*client*3	10.04	6.148	5.139
average			**5.095333**

5. KESIMPULAN

Setelah melakukan implementasi dan analisa *bandwidth* dengan menggunakan *Queue Tree* HTB, maka dapat diambil kesimpulan bahwa suatu jaringan tanpa manajemen *bandwidth* yang baik akan berakibat pada *Throughput* yang tidak terkontrol dan akan menyebabkan pemakaian yang tidak seimbang. Hal ini dapat dibuktikan dengan *Throughput* yang dihasilkan pada simulasi pra-HTB.

Setelah penerapan HTB maka pemakaian akan terjadi pemakaian yang seimbang dan sesuai dengan perioritas yang ditentukan oleh *Mikrotik Router*. Hal ini dapat terlihat pada hasil simulasi HTB berdasarkan perioritas. *Response time* pada saat *client* telah melebihi batas kuota atau *bandwidth* yang diberikan, maka akan mencapai *average* 309 ms. Hal ini dapat terlihat pada hasil eksperimen HTB berdasarkan *Port* dan aplikasi yang telah diberikan *bandwidth* tersendiri.

Pemfilteran berdasarkan protokol *icmp* mungkin dapat diterapkan untuk memperbaiki nilai waktu respon pada manajemen *bandwidth*. Untuk hasil yang lebih baik, bisa menerapkan solusi manajemen yang sesuai dengan kebijakan perusahaan atau *in*stansi, seperti penetapan perioritas *bandwidth* berdasarkan pada kebijakan, memblokir atau membolehkan pengaturan situs atau web sesuai dengan kebijakan

DAFTAR PUSTAKA

Ali Pangera, Abas, Analisa Perband*in*gan HTB dan CBQ (CLASS BASED QUEUING) Untuk Mengatur *Bandwidth* Menggunakan L*IN*UX, Jurnal DASI Desember 2004, STMIK AMIKOM Yogyakarta.<http://p3m.amikom.ac.id/p3m/dasi/des04/01%20-%20STMIK%20AMIKOM%20Yogyakarta%2

0Makalah%20ABBAS%20analisis%20perbandingan_%2014.pdf>

Benita. Yaron, (2005), *Kernel Korner-Analysis of the HTB Queuing Discipline,* Linux Journal, Volume 2005 Issue 131, March 2005, Specialized Systems Consultants, Inc. Seattle, WA, USA. <http://delivery.acm.org/10.1145/1060000/1053503/7562.html?key1=1053503&key2=3138586921&coll=DL&dl=ACM&CFID=4629351&CFTOKEN=48740083 >

Cai. Kan, Blackstock. Michael, Lotun. Reza, J. Feeley. Michael, Krasic. Charles, Wang. Junfang (2007), *Wireless unfairness: alleviate MAC congestion first!*, WinTECH '07 Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization,43-50, ACM *New* York, NY, USA. <http://delivery.acm.org/10.1145/1290000/1287777/p43-cai.pdf?key1=1287777&key2=0833044921&coll=DL&dl=ACM&CFID=4629351&CFTOKEN=48740083 >

Devera Aka Devic, Martin. May 5 2002, *Hierarchical Token Bucket theory*.<luxik.cdi.cz/~devik/qos/htb/manual/theory.htm>

Ferguson, Paul. Huston, Geoff (1998), *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, John Wiley & Sons.< http://www194.megaupload.com/files/d7003bfdda4e5b8b755f55716c0dbc1e/quality-of-service-delivering-qos-on-the-internet-and-in-corporate-networks.9780471243588.25444.pdf >

Glenn, Berg`s. (1998). *Training Guides MSCE Second Edition Networking Essentials* (2rd ed.). United States of America: *New* Riders Publishing<http://phoenixalley.com/ebooks/Networking/MSCE%20Training%20Guides%20-%20Networking%20Essentials.pdf>

Kiruthika. M, Smita. D, and Dhanashree. H (Januari 2009). *Sharing internet connection through Wi-Fi network*. ICAC3 '09 Proceedings of the International Conference on Advances in Computing, Communication and Control, 659-663. ACM, *New* York, NY, USA. <http://delivery.acm.org/10.1145/1530000/1523237/p659-kiruthika.pdf?key1=1523237&key2=5063586921&coll=DL&dl=ACM&CFID=4629351&CFTOKEN=48740083>

Leiner. Barry M, Cerf. Vinton G, Clark. David D, Kahn. Robert E, Kleinrock . Leonard, Lynch. Daniel C, Postel. Jon, Roberts. Lawrence G, Wolff. Stephen S (Februari 1997) *The past and future history of the Internet*. Communications of the ACM, Volume 40 Issue 2, 102 – 108. ACM *New* York, NY, USA. <http://delivery.acm.org/10.1145/260000/253741/p102-leiner.pdf?key1=253741&key2=8256586921&coll=DL&dl=ACM&CFID=4629351&CFTOKEN=48740083>

MikroTik *Router*OS™ v2.9 Reference Manual. *Document Revision* 3.40, *September 26,* 2007. Mikrotik, *Router*OS and *Router*BOARD are trade*mark*s of Mikrotikls SIA. <http://www.mikrotik.com/testdocs/ros/2.9/refman2.9.pdf>

MikroTik *Router*OS™ v3.0 Reference Manual . *Document Revision* 3.92, *February 11, 2008.* Mikrotik, *Router*OS and *Router*BOARD are trade*mark*s of Mikrotikls SIA. <http://www.mikrotik.com/testdocs/ros/3.0/refman3.0.pdf>

Mikrotik, 2010. Manual:*Queue*. <http://wiki.mikrotik.com/wiki/Manual:*Queue*>

Mikrotik, 2010. Manual:*Queue-Burst*. <http://wiki.mikrotik.com/wiki/Manual:*Queues_-_Burst*>

Proboyekti, Umi. 2009. Jaringan Komputer. Diambil dari: <http://lecturer.ukdw.ac.id/othie/Jaringan_Komputer.pdf>