JUSI Vol. 1, No. 2
September 2011

ISSN 2087-8737

Implementasi *Quality of Service* Menggunakan Metode *Hierarchical Token Bucket*

Imam Riadi*, Wahyu Prio Wicaksono

Program Studi Sistem Informasi, Universitas Ahmad Dahlan
Jalan Prof. Dr. Soepomo, S.H., Janturan, Yogyakarta

E-mail: imam_riadi@uad.ac.id

ABSTRAK *Perkembangan jaringan Internet memunculkan permasalahan khususnya pada pengelolaan bandwidth. Linux sebagai sistem operasi yang bersifat terbuka, menawarkan berbagai metode untuk membantu proses pengelolaan bandwidth, salah satunya dengan menggunakan metode* Hierarchical Token Bucket *(HTB) yang menjamin para pengguna jaringan mendapatkan bandwidth sesuai yang telah ditentukan. HTB memungkinkan client memperoleh bandwidth minimum yang disediakan. Penelitian ini dilakukan dengan mengumpulkan data dari berbagai sumber yang terkait, kemudian melakukan eksperimen dengan mengimplementasikan HTB dengan menambahkan program aplikasi untuk membantu administrator mengelola dan mengatur alokasi bandwidth tiap client. Berdasarkan hasil penelitian ini pengelolaan bandwith dapat dibagi secara merata serta kualitas koneksi jaringan lebih stabil.*

Keywords: *Bandwidth, Hierarchical Token Bucket, Manajemen, Jaringan.*

1 Pendahuluan

Perkembangan layanan komunikasi data saat ini sangatlah cepat, layanan yang ada tidak digunakan secara individual tetapi layanan ini digunakan secara massal dan hampir serentak dalam tiap waktu. Banyak insitusi maupun lembaga organisasi atau pendidikan yang menggunakan layanan internet secara serentak, penggunaan layanan Internet yang beragam sifatnya secara bebas dapat mengakses semua aplikasi yang ada dalam internet seperti *email, web, chatting, browsing*, dan *multimedia*. Penyebabnya *bandwidth* yang ada telah terambil banyak untuk memenuhi user pertama dan kedua karena untuk melihat video secara *online* atau *download* yang membutuhkan *bandwidth* yang cukup besar, sehingga untuk user ketiga mengalami *delay*.

Quality of Service (QoS) memegang peranan yang sangat penting dimana Linux sebagai salah satu sistem operasi Linux telah menawarkan beberapa teknik QoS untuk memfasilitasi proses manajemen *bandwidth* pada suatu jaringan Internet. Salah satu metode yang dapat diterapkan yaitu *Hierarchical Token Bucket* (HTB), metode ini akan menjamin pengguna jaringan mendapatkan bandwith yang sesuai sehingga kinerja jaringan Internet tetap berjalan dengan baik dan lancar.

2 Landasan Teori

2.1 *Quality of Service*

Quality of Service (QoS) merupakan metode pengukuran tentang seberapa baik jaringan dan merupakan suatu usaha untuk mendefinisikan karakteristik dan sifat dari satu servis (Ferguson & Huston, 1998). QoS digunakan untuk mengukur sekumpulan atribut kinerja yang telah dispesifikasikan dan diasosiasikan dengan suatu servis.

QoS didesain untuk membantu *end user* menjadi lebih produktif dengan memastikan bahwa user mendapatkan kinerja yang handal dari aplikasi-aplikasi berbasis jaringan. QoS mengacu pada kemampuan jaringan untuk menyediakan layanan yang lebih baik pada trafik jaringan tertentu melalui teknologi yang berbeda-beda. QoS menawarkan kemampuan untuk mendefinisikan atribut-atribut layanan jaringan yang disediakan, baik secara kualitatif maupun kuantitatif.

Komponen-komponen dari QoS adalah:

a) **Delay**, merupakan total waktu yang dilalui suatu paket dari pengirim ke penerima melalui sebuah jaringan. Delay pengiriman ke penerima pada dasarnya tersusun atas *hardware latency*, delay akses, dan delay transmisi.

b) **Jitter**, merupakan variasi delay antar paket yang terjadi pada jaringan berbasis IP. Besarnya nilai jitter akan sangat dipengaruhi oleh variasi beban trafik dan besarnya tumbukan antar-paket (*congestion*) yang ada dalam jaringan tersebut. Semakin besar beban trafik di dalam jaringan akan menyebabkan semakin besar pula peluang terjadinya *congestion*, dengan demikian nilai jitter-nya akan semakin besar. Semakin besar nilai jitter akan mengakibatkan nilai QoS akan semakin turun. Kategori kinerja jaringan berbasis IP dalam jitter versi *Telecommunications and Internet Protocol Harmonization Over Networks* (TIPHON) mengelompokkan menjadi empat kategori penurunan kinerja jaringan berdasarkan nilai jitter seperti terlihat pada Tabel 1.

Tabel 1. Kategori Jitter

Kategori Degradasi	Peak Jitter
Sangat bagus	0 ms
Bagus	75 ms
Sedang	125 ms
Jelek	225 ms

c) **Bandwidth**, merupakan kapasitas atau daya tampung kabel *Ethernet* agar dapat dilewati trafik paket data dalam jumlah tertentu. Bandwidth juga biasa berarti jumlah konsumsi paket data per satuan waktu dinyatakan dengan satuan *bit per second* (bps) (Santosa, 2004).

d) ***Latency***, apabila mengirimkan data sebesar 3Mbyte pada saat jaringan sepi waktunya 5 menit tetapi pada saat ramai sampai 15 menit, hal ini disebut

latency. Latency pada saat jaringan sibuk berkisar 50-70 msec (Santosa, 2004).

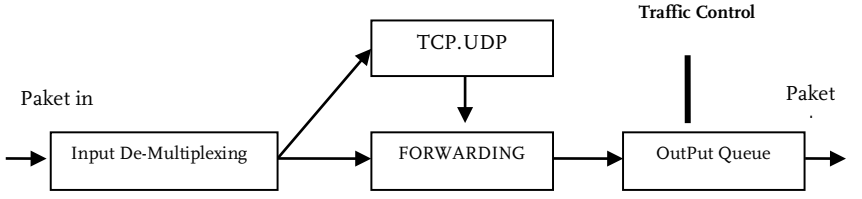
e) ***Losses***, jumlah paket yang hilang saat pengiriman paket data ke tujuan, kualitas terbaik pada jaringan LAN/WAN jika jumlah losses paling kecil (Santosa, 2004).

f) **Ping**, (*Packet Internet Gropher*) merupakan salah satu program yang digunakan untuk menguji komunikasi antar komputer dalam sebuah jaringan melalui protokol TCP/IP. Ping akan mengirimkan *Internet Control Message Protocol* (ICMP) *Echo Request* messages pada IP Address komputer yang dituju dan meminta respons dari komputer tersebut (Muuss, 1977). Ping memiliki kategori ukuran kinerja jaringan berdasarkan ping seperti terlihat pada Tabel 2.

Tabel 2. Kategori Ping

Ping	Packet loss	Keterangan
< 50 ms	0 %	Hasil memuaskan
± 90 ms	0 %	Baik
± 150 ms	1 %	Cukup
± 300 ms	3 %	Kategori kurang baik, kesulitan untuk menjalankan aplikasi streaming/game
> 500 ms	20 %	Jelek

Implementasi manajemen *bandwidth* pada jaringan berbasis TCP/IP dapat menggunakan beberapa *tools* yang dapat dipakai, baik dalam bentuk *software*, maupun *hardware*. *Tools* tersebut ada yang berharga sangat mahal, seperti Cisco, dan ada juga yang bersifat gratis seperti aplikasi-aplikasi manajemen *bandwidth* di Linux. Prinsip dasar implementasi manajemen *bandwidth* pada Linux dapat dijelaskan pada Gambar 1. (Ferguson & Huston, 1998)



Gambar 1. Linux Traffic Control (Ferguson & Huston, 1998)

Input demultiplexer akan memeriksa apakah paket yang datang ditunjukan untuk *node local*. Jika ya, maka paket akan dikirimkan ke layer yang lebih tinggi untuk pemrosesan lebih lanjut. Jika tidak, maka paket akan diteruskan ke *blok forwarding*. *Blok forwarding*, yang mungkin juga dapat menerima paket lokal dari layer yang lebih tinggi, akan melihat pada tabel *routing* dan menentukan *hop* selanjutnya bagi paket tersebut. Paket tersebut akan diantrikan untuk ditransmisikan pada *interface output*. Di titik inilah fungsi dari pengontrolan trafik pada Linux akan diterapkan. Pengontrolan trafik pada Linux dapat digunakan untuk membangun kombinasi yang komples dari disiplin antrian,

kelas-kelas, dan *filter-filter* yang akan mengontrol paket-paket yang dikirimkan pada interface output.

2.2 *Hierarchical Token Bucket*

Hierarchical Token Bucket (HTB) merupakan teknik penjadwalan paket yang digunakan kebanyakan router berbasis Linux, dikembangkan pertama kali oleh Martin Devara (2002). HTB diklaim menawarkan kemudahan pemakaian dengan teknik peminjaman dan implementasi pembagian trafik yang lebih akurat. Dasar kerja HTB hampir sama dengan disiplin antrian CBQ bahkan diagram blok sistem CBQ dengan HTB tidak ada bedanya, hanya saja pada *General Scheduler* HTB menggunakan mekanisme *Deficit Round Robin* (DRR) dan pada blok umpan balik, *Estimator*, HTB tidak menggunakan *Eksponential Weighted Moving Average* (EWMA) melainkan *Token Bucket Filter* (TBF).

Pada HTB terdapat parameter *ceil* sehingga kelas akan selalu mendapatkan *bandwidth* diantara *base link* dan nilai *ceil* linknya. Parameter ini dapat dianggap sebagai estimator kedua, sehingga setiap kelas dapat meminjam *bandwidth* selama *bandwidth* total yang diperoleh memiliki nilai dibawah nilai *ceil*. Hal ini mudah diimplementasikan dengan cara tidak mengijinkan proses peminjaman *bandwidth* pada saat kelas telah melampaui link ini (keduanya *leaves* dan interior dapat memiliki *ceil*). Apabila nilai ceil sama dengan nilai base link, maka akan memiliki fungsi yang sama seperti parameter bounded pada CBQ, dimana kelas-kelas tidak dijinkan untuk meminjam bandwidth. Sedangkan jika nilai ceil diset tak terbatas atau dengan nilai yang lebih tinggi seperti kecepatan link yang dimiliki, maka akan didapat fungsi yang sama seperti kelas *non bounded* (Yudha, 2007)

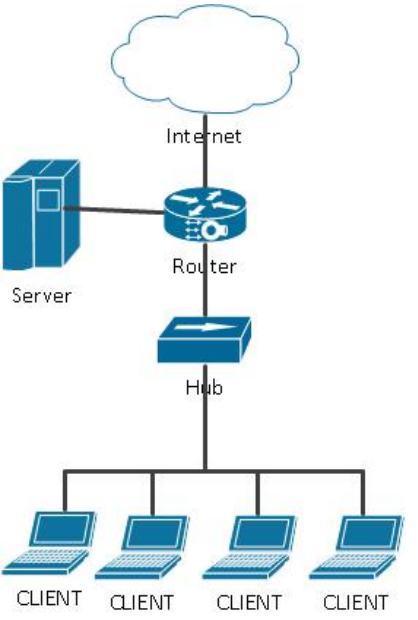
Penjadwalan pengiriman paket antrian, maka HTB menggunakan suatu proses penjadwalan yang dapat dijelaskan sebagai berikut (Devara, 2002)

(a) Class, merupakan parameter yang diasosiasikan dengan rate yang dijamin (assured rate) AR, ceil rate CR, prioritas P, level dan quantum. Class dapat memiliki parent. Selain AR dan CR, didefinisikan juga actual rate atau R, yaitu rate dari aliran paket yang meninggalkan class dan diukur pada suatu perioda waktu tertentu.
(b) Leaf, merupakan class yang tidak memiliki anak. Hanya leaf yang dapat memegang antrian paket.
(c) Level, dari kelas menentukan posisi dalam suatu hirarki. Leaf-leaf memiliki level 0, root class memiliki level=jumlah level-1 dan setiap inner class memiliki level kurang dari satu dari parentnya.
(d) Mode, dari class merupakan nilai-nilai buatan yang diperhitungkan dari R, AR dan CR. Mode-mode yang mungkin adalah: Merah: R > CR; Kuning: R <= CR and R > AR; Hijau selain di atas

3 Pembahasan

3.1 Topologi Jaringan

Topologi yang digunakan dalam mengembangkan penelitian ini menggunakan topologi star seperti terlihat pada Gambar 3.

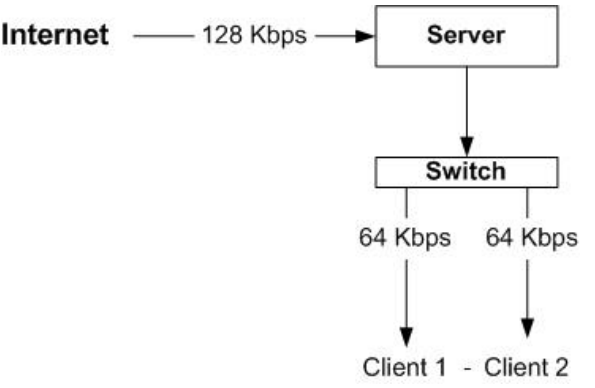


Gambar 3. Topologi Star

Masing-masing *workstation* dihubungkan langsung ke *server* dengan perantara *switch/Hub*. Topologi ini memiliki kabel sendiri untuk setiap *workstation* ke *server*, maka jalur komunikasi dalam kabel akan semakin lebar sehingga akan meningkatkan kinerja secara keseluruhan. Jika terdapat gangguan di suatu jalur maka gangguan hanya akan terjadi dalam komunikasi antara *workstation* yang bersangkutan dengan *server*.

3.2 Manajemen Bandwidth

Manajemen *bandwidth* ini akan membatasi penggunaan *bandwidth* jaringan Internet, manajemen dilakukan untuk membagi rata *bandwidth* per-*client* agar tidak terjadi *congestion*, jika sebuah jaringan Internet belum menerapkan manajemen *bandwidth* maka salah satu *client* menggunakan *bandwidth* secara penuh, *client-client* setelahnya akan mengalami antrian permintaan paket data dan mendapatkan *bandwidth* ketika permintaan paket data dari client 1 terpenuhi. Hal ini dapat mengganggu client-client lain dan mengganggu kinerja dari jaringan internet itu sendiri seperti terlihat pada Gambar 4.

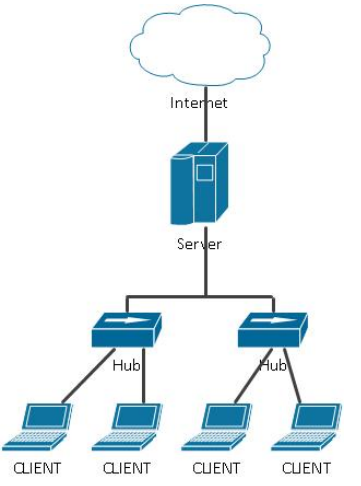


Gambar 4. Proses manajemen bandwidth

Berdasarkan Gambar 4 maka *server* akan mendapatkan bandwidth sebesar 128 Kbps, setiap *client* akan melakukan *browsing* dan *download/upload*, client memiliki aturan yang berbeda sehingga tidak terjadi gangguan antara *client* 1 dan *client* 2 tetapi dapat meminjam *bandwidth* jika salah satu client tidak aktif.

3.3 Perancangan Sistem Jaringan

Perancangan sistem jaringan merupakan tahap pembuatan rancangan jaringan, dalam penelitian ini menggunakan rancangan seperti Gambar 5, terdapat sebuah server menggunakan sistem operasi Linux Fedora Core 9 kernel 2.6.25-14 dan memiliki 30 client.



Gambar 5. Perancangan sistem jaringan

3.4 Perancangan Pembatasan Bandwidth

Perancangan pembatasan dilakukan dengan menggunakan bantuan aplikasi yang khusus dikembangkan untuk melakukan hal ini. Program ini merupakan alat bantu bagi *administrator* untuk mempermudah menjalankan, mengelola atau mengatur *bandwidth* tiap *client* yang semula atau awalnya dengan mengetikkan perintah-perintah di konsol untuk mengelola, menjalankan dan

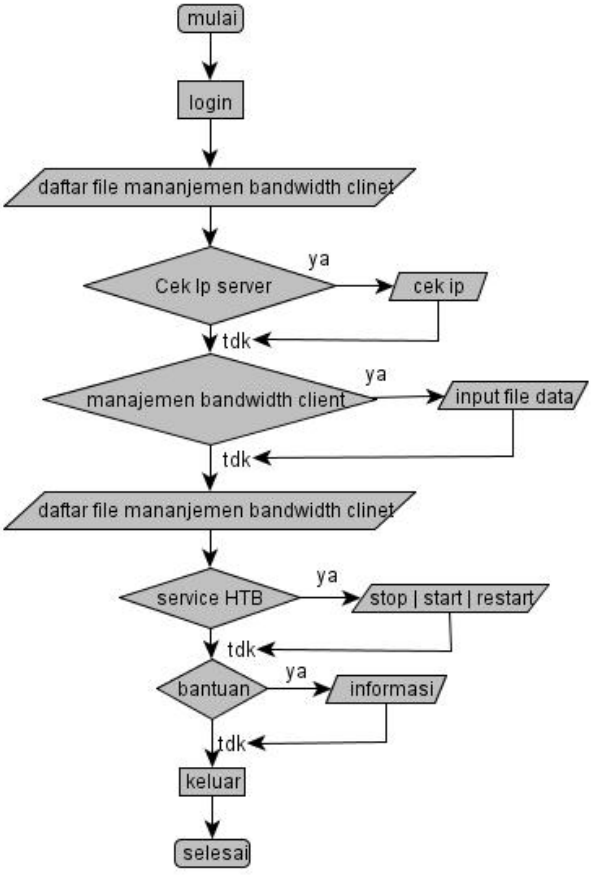
mengatur bandwidth. Program ini dibuat dengan menggunakan bahasa pemrograman PHP dan MySQL sebagai *database* untuk menyimpan file-file konfigurasi *bandwidth*, untuk selanjutnya program ini akan disebut dengan nama *imanbw (Interface Manajemen Bandwidth*). Skenario rancangan untuk menentukan pembatasan *bandwidth* setiap *client* dapat dilihat seperti pada Tabel 3.

Tabel 3. Perancangan pembatasan bandwidth

Client	Rate	Ceil
Komp 1	64	128
Komp 2	64	128
Komp 3	64	128
Komp 4	64	128
Komp 5	64	128

Client	Rate	Ceil
Komp 6	64	128
Komp 7	64	128
Komp 8	64	128
Komp 9	64	128
Komp 10	64	128

Program *imanbw* yang diterapkan memiliki cara kerja seperti terlihat pada Gambar 6.



Gambar 6. Cara kerja *imanbw*

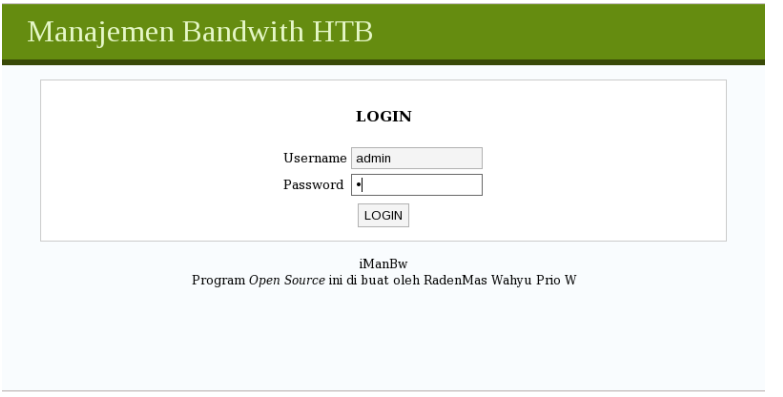
Cara kerja program *imanbw* sebagai berikut :

(a) User admin login ke *imanbw*
(b) Halaman utama akan langsung menuju ke daftar file
(c) Halaman utama memiliki beberapa menu diantaranya menu cek IP, jika ya maka akan melihat IP dari server, jika tidak maka dapat dilanjutkan memilih menu selanjutnya.
(d) Menu manajemen bandwidth, jika ya maka akan melakukan input file data konfigurasi, jika tidak maka akan langsung menuju halaman daftar file.
(e) Menu service, jika ya maka perintah untuk menjalankan HTB, jika tidak maka dapat menuju menu help untuk mengetahui tata cara pemakaian imanbw.
(f) Jika sudah melakukan konfigurasi user admin dapat keluar dari imanbw atau kembali login lagi untuk melakukan aktifitas yang dibutuhkan.

4 Implementasi HTB

4.1 Implementasi *imanbw*

Implementasi *imanbw* dapat dilakukan dengan menjalankan *servis httpd* dan *mysql*, setelah semuanya aktif maka selanjutnya dapat diakses menggunakan browser dengan alamat *http://localhost/manbw* dari server, bila dari client diketikkan *http://ipaddress_server/manbw* seperti terlihat pada Gambar 7.



Gambar 7. Implemenyasi *imanbw*

4.2 Tabel User dan Network

Tabel user diperlukan untuk *login imanbw* dan tabel *network* untuk file-file konfigurasi HTB agar mudah di edit oleh *administrator*. Pembuatan tabel *user* dan *network* sebagai berikut:

Kode program 1. Create table users

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL auto_increment,
  `username` varchar(55) NOT NULL,
  `password` varchar(55) NOT NULL,
  PRIMARY KEY  (`id`)
```

```
) TYPE=MyISAM  AUTO_INCREMENT=2 ;
```

Kode program 2. Create table networks

```
CREATE TABLE `networks` (
  `id` int(11) NOT NULL auto_increment,
  `nama` varchar(255) NOT NULL,
  `r2q` varchar(255) NOT NULL,
  `rate` varchar(255) NOT NULL,
  `quantum` varchar(255) NOT NULL,
  `leaf` varchar(255) NOT NULL,
  `ceil` varchar(255) NOT NULL,
  `rule` varchar(255) NOT NULL,
  `mark` varchar(255) NOT NULL,
  PRIMARY KEY  (`id`)
) TYPE=MyISAM  AUTO_INCREMENT=32 ;
```

4.3 Konfigurasi imanbw

Imanbw memerlukan konfigurasi penyesuaian dengan *path* konfigurasi HTB di atas agar dapat dijalankan sesuai fungsinya. Konfigurasi yang pertama adalah tentang koneksi *database* dengan *imanbw* karena fungsi database di sini sebagai penyimpan file-file konfigurasi HTB agar dapat di edit dan dihapus dengan mudah, seperti dibawah ini:

Kode program 3. Koneksi database dengan imanbw

```
<?
mysql_connect('localhost','root','123456');
mysql_select_db('bwmeter');
//menghapus session pesan
unset($_SESSION['mesg']);
?>
```

Koneksi di atas disesuaikan dengan database yang telah dibuat dan tersimpan dengan nama file *database.php*. Konfigurasi kedua yaitu menyamakan path untuk meletakkan *file-file* konfigurasi HTB dan eksekusi *file htb.init*.

Kode program 4. Path direktori HTB

```
<?
include 'function.php';
define("PATH_NETWORKS","/etc/sysconfig/htb/");
?>
```

Konfigurasi di atas menunjukkan tempat *file-file* konfigurasi HTB disimpan ketika menggunakan *imanbw* dan tersimpan dengan nama *setting.php*. Melakukan eksekusi htb.init dengan imanbw juga harus disesuaikan dengan path di lokasi htb.init yaitu di */usr/local/sbin* tersimpan dengan nama *service.php*.

Kode program 5. Path eksekusi htb.init

```
<?
session_start();
if($_POST['action'] == 'Start'){
        if($output = shell_exec('/usr/local/sbin/htb.init start'))
                $_SESSION['mesg'] = 'Service has been started.<br>';
        else
                $_SESSION['mesg'] = 'Service started failed.<br>';
}
elseif($_POST['action'] == 'Stop'){
        if($output = shell_exec('/usr/local/sbin/htb.init stop'))
                $_SESSION['mesg'] = 'Service has been stopped.<br>';
```

```
        else
                $_SESSION['mesg'] = 'Service stopped failed.<br>';
}
elseif($_POST['action'] == 'Restart'){
        if($output = shell_exec('/usr/local/sbin/htb.init restart'))
                $_SESSION['mesg'] = 'Service has been restarted.<br>';
        else
                $_SESSION['mesg'] = 'Service restarted failed.<br>';
}
$_SESSION['mesg'] .= "<pre>$output</pre>";
header("location:index.php?menu=service");
?>
```

4.4 Pengujian

Pengujian dilakukan terhadap *bandwidth* ketika sebelum menerapkan HTB dan sesudah menerapkan HTB. Pengujian ini dilakukan untuk mengetahui apakah metode HTB dalam manajemen bandwidth dapat berjalan dengan baik dan lancar. Dalam pengujian ini dilakukan dengan metode *Comparation Test*. *Comparation test* merupakan pengujian yang membandingkan hasil dari aturan yang telah ditentukan untuk mendapatkan data yang identik dengan aturan-aturan yang telah diberikan, sehingga dapat dilihat perbedaannya.

Pengujian dilakukan menggunakan *iperf* dan *ping*, dengan sintaks perintah:

Seluruh client melakukan aktifitasnya, server memberikan perintah *iperf –c ip_client –t 10 –i 1 ke setiap clinet-nya* maka akan mendapatkan nilai intervals, transfer, rate.

Hasil pengujian perbandingan seperti terlihat pada Tabel 4.

Tabel 4. Hasil perbandingan tanpa HTB dan dengan HTB

Client	Rate (KB/s)	Loss	Response Time (ms)
Komp 3	72.1	0 %	214
Komp 9	1.2	33.33%	1486

Ket = tanpa HTB

Client	Rate (KB/s)	Loss	Response Time (ms)
Komp 3	62.4	0%	63
Komp 9	21.1	0%	290

Ket = dengan HTB

Hasil diatas secara keseluruhan menunjukkan bahwa HTB dapat melakukan pembagian bandwidth dengan baik dengan membandingkan *hasil loss paket* data dan *response time* selain *rate* sebagai hasil ukuran perbandingan bandwidth.

5 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan maka dapat diambil kesimpulan bahwa, HTB dapat melakukan pembatasan *bandwidth* dengan baik dari seluruh *client* yang ada dan program *imanbw* yang dikembangkan dapat membantu memudahkan *administrator* dalam melakukan manajemen *bandwidth* di lingkungan jaringan yang dikelolanya.

Referensi

Devara, M., 2002, *Hierarchical Token Bucket Theory*, http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm

Ferguson, P. & Huston, G., 1998, *Quality of Service*, John Wiley & Sons Inc.

Muuss, M., 1977, *The Story of PING program*, http://ftp.arl.mil/~mike/ping.html

Santosa, B., 2004, *Manajemen bandwidth internet dan intranet*, http://stream.plasa.com/onno/gfe/view.php?file=referensi_bahasa_indonesia_2/network/bwmanagement.pdf

Yudha., 2007, *Disiplin Antrian (Queueing Discipline/Qdisc)*, http://omyudha.multiply.com/journal/item/9

***