PEMBATAS KECEPATAN DATA PADA PC-ROUTER BERBASIS LINUX MENGGUNAKAN DISIPLIN ANTRIAN *HIERARCHICAL TOKEN BUCKET* (HTB)

Traffic Shaping on Linux Based PC-Router
Using Hierarchical Token Bucket (HTB) Queuing Discipline

Hari Siswantoro
Program Studi Teknik Elektro Unsoed

ABSTRACT

A common problem which occurred on a network router is bottle-neck phenomenon. This problem appear if the router is connecting high speed network (e.g. a 100 Mbps fast ethernet LAN) to low speed network (e.g. 512 kbps DSL internet line). To avoid traffic congestion, there must be a mechanism to control the link sharing. This paper discusses the use of Hierarchical Token Bucket (HTB) algorithm, which is embedded in Linux kernel, to share the bandwidth on a link in a controlled fashion. Hierarchical link sharing allows multiple hosts, agencies, or protocol families to share the bandwidth. By implementing the HTB algorithm, we intend to achieve a fair bandwidth sharing according to the policy.

Keywords*: traffic shaping, linux router, hierarchical token bucket (HTB)*

PENDAHULUAN

Dalam jaringan komputer dikenal empat parameter yang digunakan untuk menentukan tingkat kualitas layanan (*Quality of Service*), yaitu reliabilitas, tunda waktu, *jitter* dan lebar jalur (*bandwidth*) (Tanenbaum, 2003). Penelitian ini dititikberatkan untuk membahas parameter terakhir yaitu *bandwidth*. Seiring dengan perkembangan jaringan komputer dan aplikasinya, kebutuhan *bandwidth* menjadi semakin besar. *Bandwidth* menjadi sumber daya yang terbatas dan harus diatur penggunaannya untuk mengantisipasi kemacetan jalur.

Salah satu teknik yang digunakan untuk mengantisipasi kemacetan jalur adalah dengan membatasi kecepatan transmisi data yang keluar dari sebuah *router* (Floyd, 2001). Jika kecepatan transmisi data dibatasi maka diharapkan *bandwidth* yang jumlahnya terbatas dapat dimanfaatkan secara efektif dan kemacetan jalur dapat dihindari. Untuk membatasi kecepatan transmisi data, berbagai algoritma telah diusulkan. Di antara algoritma tersebut, antara lain adalah algoritma *fair queueing* (Nagle, 1987 disalin oleh Tanenbaum), yang kemudian disempurnakan oleh Demers et. al. pada tahun 1990 (disalin oleh Tanenbaum) menjadi *byte-by-byte round robin fair queueing*. Kedua algoritma tersebut memberikan prioritas yang sama terhadap semua paket data, padahal sering terjadi bahwa terdapat paket data tertentu yang membutuhkan *bandwidth* lebih besar dibanding dengan paket data lain. Kelemahan ini kemudian diperbaiki oleh Shreedhar dan Varghese pada tahun 1995 (disalin oleh Tanenbaum) dengan mengajukan algoritma *weighted fair queueing*.

Pada tahun 1995, Floyd dan Jacobson mengusulkan sebuah mekanisme pembagian *bandwidth* menggunakan metode hirarkis. *Bandwidth* sebuah jalur akan dibagi untuk beberapa sub-jaringan, kemudian di sub-jaringan masing-masing akan dibagi lagi untuk berbagai *client* atau protokol yang membutuhkannya. Metode ini juga memperbolehkan *bandwidth* yang dialokasikan untuk sebuah kelas dapat dipinjam oleh kelas yang lain jika sedang tidak dipergunakan. Metode ini kemudian lebih dikenal dengan *Class Based Queueing* (*CBQ*).

Sarolahti et. al, pada tahun 2007 mengemukakan perlunya penentuan laju pengiriman data melalui jalur yang sedang digunakan. Penentuan laju tersebut diberi nama algoritma *Quick-Start* dan digunakan untuk menggantikan Slow-Start yang selama ini digunakan. Dalam artikel ini, besar kapasitas jalur yang akan dibagi sudah ditentukan terlebih dahulu berdasarkan kapasitas *uplink* router.

Melalui penelitian yang dilakukannya, Devera pada tahun 2001, mengatakan bahwa

CBQ terlalu kompleks dan tidak optimal untuk berbagai situasi. Kemudian dia mengusulkan metode baru yang diberi nama *Hierarchical Token Bucket (HTB)*. Pendekatan hirarkis yang digunakan dalam HTB memudahkan pembagian *bandwidth* untuk berbagai kelas, dan memberikan jaminan *bandwidth*, serta memungkinkan untuk menyebutkan jumlah *bandwidth* yang boleh dipinjam oleh kelas lain. Disiplin antrian HTB sebenarnya mirip dengan CBQ tetapi HTB tidak berhenti saat penghitungan waktu untuk melakukan pemangkasan laju data. Selain itu parameter yang digunakan dalam HTB lebih sedikit dan sederhana dibanding CBQ.

Devera mengimplementasikan HTB pada *PC-router* yang menggunakan sistem operasi *Linux*. Kernel *Linux* sendiri pertama kali dikembangkan oleh Linus Torvalds pada tahun 1991 dan saat ini merupakan salah satu sistem operasi yang populer digunakan sebagai *router* maupun *server* di internet. Sifatnya yang *open source* menyebabkan Linux banyak digunakan dan diadopsi oleh para peneliti jaringan.

Dalam penelitiannya Devera menggunakan HTB untuk membagi *bandwidth* pada beberapa kelas protokol. Padahal beberapa kasus pembagian *bandwidth* justru tidak menggunakan kelas protokol melainkan berupa kelas yang terdiri atas beberapa sub-jaringan maupun *host*. Seperti yang terjadi pada sebuah jaringan warnet maupun jaringan penyedia jasa layanan internet. Penelitian yang dilaporkan pada artikel ini merupakan implementasi HTB untuk membatasi kecepatan data pada PC-router Linux dengan anggota kelas yang berupa *host*. *Script* yang digunakan dalam penelitian ini didasarkan atas manual yang ditulis oleh Hubert (2002).

METODE PENELITIAN

Penelitian ini dilakukan dengan metode simulasi jaringan yang terdiri atas sebuah PC-router berbasis Linux dan sepuluh buah komputer *client*. PC-router dihubungkan ke internet untuk menyimulasikan *bottle-neck*. PC-router dilengkapi dengan HTB dan aplikasi pemantau trafik data IPtraf. IPtraf digunakan untuk mengamati trafik secara *real time*, sehingga kecepatan transmisi data yang keluar masuk router akan terpantau. Sedangkan untuk membangkitkan trafik yang besar dan konstan, *client* akan melakukan transfer berkas berukuran besar.

Policy atau kebijakan yang diterapkan adalah, jika semua *client* atau *host* sedang aktif dan membutuhkan *bandwidth* yang besar, maka *bandwidth* akan dibagi secara adil dan merata. Jika ada *client* yang tidak aktif, maka jatah *bandwidth*-nya boleh digunakan oleh *client* lain yang membutuhkannya.

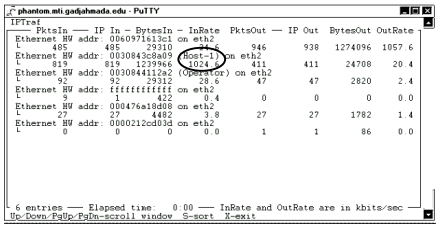
HASIL DAN PEMBAHASAN

Dalam jaringan simulasi, terdapat N buah *host* atau sub-jaringan yang saling berbagi *bandwidth* satu sama lain. Diinginkan supaya *bandwidth* sisa dari sebuah kelas dapat dibagi secara merata untuk kelas-kelas yang lain. Untuk mempermudah penggambaran, dimisalkan jumlah host N=10 dan *bandwidth* yang dibagi sebesar 1 Mbps. Untuk menangani kasus ini, dibuat sebuah *shell script* yang menjalankan disiplin antrian HTB di bawah ini.

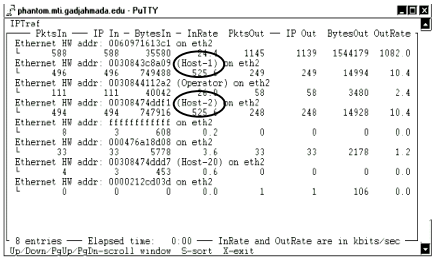
```
#! /bin/bash
# HTB Traffic Shaping script

# Menghapus sisa-sisa konfigurasi
tc qdisc del dev eth2 root 2>
/dev/null > /dev/null

# Singkatan untuk beberapa
perintah yang sering digunakan
TCC="tc class add dev eth2 parent"
TCQ="tc qdisc add dev eth2 parent"
TCFU32="tc filter add dev eth2
parent 1:0 protocol ip prio 25 u32
match ip dst"
# Root Class
tc qdisc add dev eth2 root handle
1: htb default 100
$TCC 1: classid 1:1 htb rate 1Mbit
ceil 1Mbit burst 2k
# Default Class -> Host 10
$TCC 1:1 classid 1:100 htb rate
100kbit ceil 1Mbit burst 2k prio 1
$TCQ 1:100 handle 100: sfq perturb
10
# Host-1
```

```
$TCC 1:1 classid 1:11 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:11 handle 11: sfq perturb
10

$TCFU32 172.24.12.11 flowid 1:11

# Host-2

$TCC 1:1 classid 1:12 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:12 handle 12: sfq perturb
10

$TCFU32 172.24.12.12 flowid 1:12

# Host-3

$TCC 1:1 classid 1:13 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:13 handle 13: sfq perturb
10

$TCFU32 172.24.12.13 flowid 1:13

# Host-4

$TCC 1:1 classid 1:14 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:14 handle 14: sfq perturb
10

$TCFU32 172.24.12.14 flowid 1:14

# Host-5

$TCC 1:1 classid 1:15 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:15 handle 15: sfq perturb
10

$TCFU32 172.24.12.15 flowid 1:15

# Host-6

$TCC 1:1 classid 1:16 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:16 handle 16: sfq perturb
10

$TCFU32 172.24.12.16 flowid 1:16

# Host-7

$TCC 1:1 classid 1:17 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:17 handle 17: sfq perturb
10

$TCFU32 172.24.12.17 flowid 1:17

# Host-8

$TCC 1:1 classid 1:18 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:18 handle 18: sfq perturb
10

$TCFU32 172.24.12.18 flowid 1:18

# Host-9

$TCC 1:1 classid 1:19 htb rate
100kbit ceil 1Mbit burst 2k prio 1

$TCQ 1:19 handle 19: sfq perturb
10

$TCFU32 172.24.12.19 flowid 1:19
```

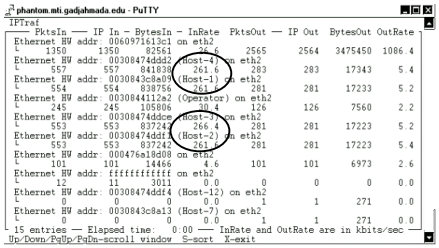
Setelah *script* di atas dijalankan, untuk melihat sejauh mana disiplin antrian HTB bekerja perlu dilakukan sebuah uji coba. Uji coba ini dilakukan dengan mengalirkan trafik ke masing-masing *class* dengan beberapa kondisi. Ada sepuluh kondisi, kondisi pertama : hanya ada satu *host* aktif, kondisi kedua : hanya ada dua *host* aktif, demikian seterusnya. Maksud dari perlakuan ini adalah untuk mengetahui sejauh mana HTB adil membagikan sisa *bandwidth* yang tidak terpakai. Gambar-gambar di bawah ini merupakan cuplikan sesaat pemantau laju data IPTraf dari beberapa kondisi.



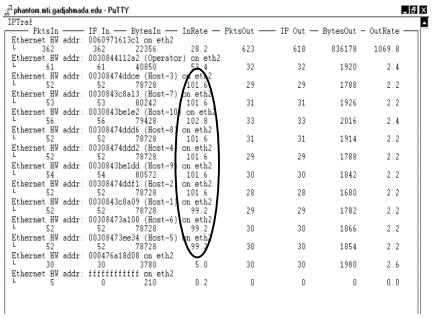
Gambar 1. Satu *host* aktif menggunakan *bandwidth* sendiri.



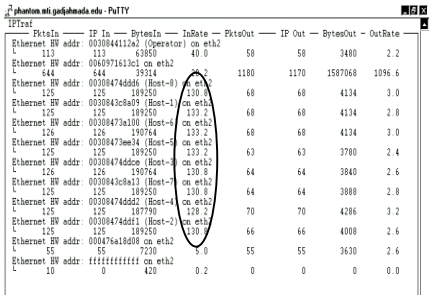
Gambar 2. Dua *host* aktif.



Gambar 3. Empat *host* aktif



Gambar 5. Sepuluh *host* aktif semua.



Gambar 4. Delapan *host* aktif.

Jika dibuat sebuah rekapitulasi, maka akan diperoleh hasil pengukuran kecepatan data yang menuju ke tiap-tiap host seperti terlihat pada tabel 1 dan tabel 2 berikut ini (kecepatan data dalam satuan kbps atau *kilo bits per second*).

Tabel 1. Hasil pengukuran laju data (1-5 *Host*)

No Host	1 Host	2 Host	3 Host	4 Host	5 Host
1	1025	526	354	262	208
2	0	526	349	262	211
3	0	0	349	266	211
4	0	0	0	262	208
5	0	0	0	0	211
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0

Tabel 2. Hasil pengukuran laju data (6-10 *Host*)

No Host	6 Host	7 Host	8 Host	9 Host	10 Host
1	157	153	133	116	99
2	155	150	131	116	102
3	157	150	131	116	102
4	155	148	128	116	102
5	155	148	133	119	99
6	155	153	133	119	99
7	0	150	131	116	102
8	0	0	131	116	102
9	0	0	0	115	102
10	0	0	0	0	103

Dari data tersebut dapat disimpulkan bahwa disiplin antrian HTB yang dipasang berhasil mengatur laju data ke tiap-tiap *host* dan membagi sisa *bandwidth* yang ada dengan adil ke semua *host*. Meskipun dari angka-angka tersebut tampak adanya ketidakseragaman dan ketidakbulatan kecepatan, hal ini masih dalam batas kewajaran karena selisih angkanya hanya sekitar 2-5 kbps. Penyebabnya adalah parameter *burst* pada HTB yang memungkinkan kecepatan sebuah kelas dapat sedikit menembus parameter *ceil rate*.

KESIMPULAN DAN SARAN

Kesimpulan

1. Disiplin antrian HTB pada PC-router Linux dapat digunakan untuk mengatur kecepatan data pada kelas yang berupa *host* atau sub-jaringan dengan model hirarkis dan peminjaman *bandwidth* yang tidak digunakan.
2. Penggolongan kelas *host* atau sub-jaringan dapat dilakukan menggunakan *classifier* U32.

Saran

1. Untuk mendapatkan data yang dapat direkam untuk keperluan analisa di lain waktu, sebaiknya digunakan pemantau data yang sekaligus menyimpannya. IPtraf tidak mempunyai fasilitas untuk menyimpan data, tetapi hanya menampilkan sesaat sehingga harus segera dicatat secara manual.
2. Penelitian lebih lanjut dapat dilakukan untuk mengamati kemampuan HTB mengatur kecepatan data pada kelas dengan prioritas yang tidak sama.

DAFTAR PUSTAKA

Devera, M., 2001, *HTB Manual Page,* http://www.cdi.cz/~devik/qos/htb

Floyd, S., 2001, *A Report on Some Recent Developments in TCP Congestion Control*, IEEE Communications Magazine, April

Floyd, S., Jacobson, V., 1995, *Link Sharing and Resource Management Models for Packet Networks*,IEEE/ACM Transactions on Networking, Vol. 3 No. 4, Agustus

Hubert, B., 2002, *Linux Advanced Routing & Traffic Control HOWTO*,http://lartc.org/HOWTO/cvs/2.4 routing/output/2.4routing. html

Sarolahti, P., Allman, M., Floyd, S., 2007, *Determinig an Appropriate Sending Rate Over an Underutilized Network Path*, Elsevier Computer Networks, Vol. 51 Issue 7, May

Tanenbaum, A.S., 2003, *Computer Networks, Fourth Edition,* Prentice Hall, New Jersey

Torvalds, L., 1991-2008, *Linux Kernel Development*,http://www.kernel. org