Evaluasi Penggunaan Linux Router untuk Pembatasan Bandwidth Aplikasi Internet Tertentu

Willy Santoso, Henry Novianus Palit, Justinus Andjarwirawan
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jln. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
willy.santoso94@gmail.com, hnpalit@petra.ac.id, justin@petra.ac.id

ABSTRAK

Menurut laporan Cisco pada tahun 2015, penggunaan internet telah meningkat lebih dari lima kali dalam 5 tahun terakhir, dan hal tersebut diproyeksikan akan meningkat tiga kali selama 5 tahun ke depan. Peningkatan penggunaan internet tersebut banyak disebabkan oleh aplikasi-aplikasi internet tertentu seperti Flash, MPEG, *video streaming*, dan P2P *file sharing*. Oleh karena itu, pengaturan *bandwidth* dengan cara *traffic shaping* pada Linux dilakukan agar dapat memberikan kebutuhan internet yang sesuai dengan kelas pengguna. nDPI digunakan untuk mengeklasifikasikan *traffic data* dalam jaringan internet.

Pengerjaan penelitian berfokus pada evaluasi penggunan nDPI pada aplikasi P2P bittorrent. Evaluasi dilakukan menggunakan iptables untuk melakukan *marking traffic data* dan traffic control (tc) untuk melakukan *shaping*. *Bandwidth monitoring* menggunakan aplikasi iptraf pada Linux dan aplikasi networx pada klien.

Berdasarkan pengujian yang dilakukan, traffic shaping dengan nDPI ditambah dengan *cache* berhasil dilakukan pada aplikasi P2P bittorrent. Percobaan nDPI dengan *cache* dilakukan pada beberapa ukuran *cache* dan *shaping* pada beberapa *bandwidth*. nDPI dengan ukuran cache 1024 *entries* merupakan besar *cache* yang direkomendasikan. nDPI tidak dapat sepenuhnya mengklasifikasikan *traffic data* bittorrent dengan benar.

Kata Kunci: nDPI, cache, LRU, bittorrent, Linux Centos, traffic shaping, traffic control

ABSTRACT

According to Cisco report in 2015, Internet usage has increased more than five times in the last 5 years, and it was projected to increase three times over the next 5 years. Increasing use of the internet is mostly caused by certain internet applications such as Flash, MPEG, video streaming, and P2P file sharing. Therefore, bandwidth configuration with traffic shaping method in Linux is needed in order to provide the needs of the Internet in accordance with user class. nDPI is used for classifying the data traffic in the Internet network.

Work on the thesis focuses on the evaluation of the use of nDPI on bittorrent P2P applications. The evaluation was done using iptables to perform the data traffic marking and traffic control (tc) to do the bandwidth shaping. Bandwidth monitoring used iptraf applications on Linux and Networx application on the client.

Based on testing performed, traffic shaping with nDPI coupled with cache successfully limited the bandwidth usage of bittorrent P2P applications. The nDPI with cache was evaluated in different cache sizes and bandwidth limits. nDPI with size cache of 1024 entries is recommended. nDPI can not fully classifying bittorrent traffic correctly.

Keywords: *nDPI, cache, LRU, bittorrent, Linux Centos, traffic shaping, traffic control*

1. PENDAHULUAN

Semakin berkembangnya teknologi informasi sekarang ini, maka kebutuhan akses internet semakin meningkat pula. Penggunaan internet sudah menjadi kebutuhan penting bagi pengguna untuk memperoleh informasi. Menurut laporan Cisco, penggunaan internet telah meningkat lebih dari lima kali dalam 5 tahun terakhir, dan hal tersebut diproyeksikan akan meningkat tiga kali selama 5 tahun ke depan [2].

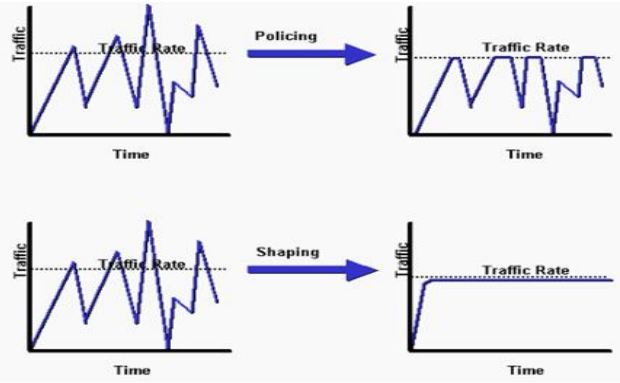
Peningkatan penggunaan internet tersebut banyak disebabkan oleh aplikasi-aplikasi internet tertentu seperti Flash, Windows Media, MPEG, *video streaming*, P2P *file sharing*, dan lain-lain. Trend menunjukkan pertumbuhan terus-menerus dan pada tahun 2018, lalu lintas video yang akan mengambil 79% dari semua lalu lintas internet yang digunakan pengguna [4]. Sebagai contoh dalam suasana kantor (jaringan LAN), penggunaan akses internet untuk aplikasi video streaming dan aplikasi P2P yang berlebihan dapat mengakibatkan terhambatnya akses internet untuk aplikasi lainnya seperti email atau *website browsing*. Maka dari itu aplikasi internet yang digunakan untuk menangani hal-hal pekerjaan penting harus diberikan prioritas akses internet yang lebih tinggi. *Traffic policy* dan *traffic shaping* merupakan dua teknik yang dapat digunakan untuk mengatasi masalah akses internet pada penggunaan aplikasi internet tersebut.

Sebagai solusi untuk permasalahan tersebut dilakukan penelitian penggunaan *traffic shaping* pada *router* dalam sistem operasi Linux. Penelitian ini melakukan evaluasi pembatasan *bandwidth* dengan menggunakan pedoman umum untuk mengkonfigurasi *traffic shaping* pada pola-pola akses internet yang berbeda. Dengan adanya penelitian ini diharapkan dapat memberikan pengalaman penggunaan internet yang lebih baik kepada pengguna. Untuk lebih spesifik, konfigurasi *traffic shaping* harus digunakan dengan benar untuk dapat memberikan pelayanan internet yang sesuai dengan kelas pengguna.

2. LANDASAN TEORI

2.1. Traffic Shaping dan Traffic Policing

Traffic Shaping dan Traffic Policing merupakan 2 teknik metode yang dapat digunakan untuk mengatur lalu lintas paket data yang berada dalam jaringan internet komputer [2]. *Traffic Shaping* adalah metode yang digunakan untuk mengatur *transfer* data pada sebuah jaringan untuk menjamin kinerja dan kualitas jaringan. Cara kerja *Traffic Shaping* adalah dengan menunda paket yang mempunyai prioritas lebih rendah dan mendahulukan paket yang mempunyai prioritas lebih tinggi. Sedangkan *Traffic Policing* adalah metode yang digunakan untuk memberi batasan maksimal *transfer* data pada sebuah jaringan. *Traffic Policing* mengatur batasan maksimal *transfer* data. *Traffic Policing* memberikan dua pilihan tindakan untuk sebuah paket yaitu jika paket termasuk dalam batasan maksimal maka paket akan di transmisikan, jika paket melanggar batasan maksimal maka paket akan dibuang. Contoh perbandingan *Traffic Shaping* dan *Traffic Policing* dapat dilihat pada Gambar 1.



Gambar 1 Perbandingan *Traffic Shaping* dan *Policing* [3]

2.2. Linux CentOS

CentOS adalah sebuah distro *platform* Linux yang stabil, mudah dikelola, dan bersumber dari Red Hat Enterprise Linux (RHEL). Sejak 2004 Maret, Linux CentOS dapat digunakan oleh publik secara gratis dan menjadi distro yang didukung oleh komunitas Red Hat. Oleh karena itu CentOS mempunyai fungsi yang mirip dengan Red Hat Enterprise Linux (RHEL). CentOS Linux adalah sistem operasi Linux yang gratis dan bebas untuk didistribusikan. CentOS dikembangkan oleh sebuah tim kecil dari *developer* inti. *Developer* inti tersebut didukung oleh komunitas pengguna yang aktif Linux seperti administrator sistem, administrator jaringan, manajer, kontributor inti Linux, dan pecinta Linux dari seluruh dunia [1].

2.3. Queueing Discipline

Queueing Discipline [5] adalah bagaimana cara paket disimpan untuk sementara sambil menunggu untuk dikirimkan. Banyak macam *Queueing Discipline* bisa digunakan untuk menentukan paket mana yang akan dikirimkan dan yang akan dibuang. *Queueing Discipline* juga dapat menentukan berapa lama paket akan menunggu untuk dikirim, sehingga dapat berguna untuk pengaturan QoS pada suatu jaringan internet. *Queueing* discipline dibagi menjadi 2 bagian yaitu *classless* dan *classful*. Berikut penjelasan dari masing-masing bagian *Queueing discipline* tersebut :

Stochastic Fairness Queueing (SFQ) adalah sebuah implementasi dari sebuah algoritma antrian yang adil. Algoritma *SFQ* memiliki keakuratan yang paling rendah jika dibandingakan dengan algoritma pengaturan *queue* yang lain. Akan tetapi algoritma SFQ ini lebih adil dalam pembagian aliran data. Cara kerja dari algorima SFQ ini adalah aliran data dibagi menjadi beberapa antrian *FIFO* yang sangat besar dimana setiap paket dari FIFO *queue* dikirimkan secara bergantian dengan sistem *round robin*. Hal tersebut menyebabkan sebuah perilaku yang adil dalam pembagian aliran data. *SQF* disebut '*stochastic*' karena tidak hanya memakai sebuah antrian dalam satu sesi *transfer* data. *SFQ* mempunyai algoritma yang bisa membagi aliran data dengan sejumlah antrian FIFO menggunakan algoritma *hashing*..

Token Bucket Filter (TBF) merupakan sebuah *qdisc* sederhana yang hanya mengirim paket yang diterima selama tidak melewati batas yang ditentukan. Akan tetapi algoritma *Token Bucket Filter* memiliki kemampuan *short burst* dimana paket jaringan yang dikirimkan melalui algoritma ini mampu melebihi batas pengiriman paket untuk sementara waktu. *TBF* sangat presisi dalam jaringan internet dan penggunaan prosessor yang efisien. *TBF* terdiri dari *buffer* (*bucket*) yang dalam tingkatan tertentu (*token rate*) terus diisi oleh potongan informasi virtual yang disebut *token*. Parameter yang paling penting dari *bucket* adalah ukurannya dan seberapa banyak *bucket* tersebut dapat menyimpan *token*.

PRIO qdisc adalah algoritma pengaturan *queue* dimana melakukan pembagian aliran data berdasarkan *filter* yang telah dikonfigurasi dalam kelas-kelas *queue*. Saat sebuah paket data di *enqueued* ke *PRIO qdisc*, maka sesuai dengan konfigurasi akan masuk kedalam salah satu kelas *queue*. Standarnya dalam PRIO qdisc akan dibuat tiga kelas *queue*. Kelas – kelas ini akan memakai *FIFO qdisc* akan tetapi dapat juga diganti dengan algoritma qdisc lainnya. Sedangkan saat sebuah paket data harus di *dequeued* dari kelas *queue*, pengaturan akan berjalan berdasarkan prioritas dimana kelas *queue* yang lebih rendah hanya dapat mengirim paket data ke kelas *queue* yang lebih tinggi apabila kelas yang lebih tinggi tersebut tidak ada antrian paket data.

HTB merupakan algoritma improvisasi dari *Token Bucket Filter* yang menggunakan banyak kelas *queue* dalam konfigurasinya. Setiap kelas pada HTB merupakan sebuah TBF *qdisc* dimana kelas-kelas TBF *qdisc* tersebut berbentuk seperti *tree* dengan kompisisi *root* dan *leaves*. Cara kerja *HTB* seperti *CBQ* hanya saja *HTB* tidak menghitung waktu *idle bandwitdth* dari jaringan. Karena *HTB* adalah *Classful Token Bucket Filter* maka *HTB* mempunyai beberapa parameter yang dapat digunakan untuk *shaping* pertukaran paket data antar kelas *queue* dan menentukan prioritas dari kelas-kelas *queue* yang digunakan.

2.4. nDPI (Open Deep Packet Inspection)

nDPI adalah DPI *library* yang dikembangkan oleh Ntop [7]. Dirilis di bawah lisensi LGPL, tujuannya adalah untuk memperluas library protokol baru yang dinyatakan hanya tersedia pada versi berbayar dari Open DPI. Pada nDPI digunakan algoritma *string matching* Aho-Corasick, untuk proses pencocokan paket data dengan *signature* yang dimiliki nDPI. Selain platform Unix, nDPI juga mendukung penggunaanya pada sistem operasi Windows. Selain itu, nDPI juga dapat digunakan untuk aplikasi pemantauan lalu lintas jaringan internet. nDPI digunakan oleh ntop dan nProbe untuk menambahkan deteksi aplikasi-lapisan protokol, terlepas dari port aplikasi yang digunakan. Hal ini menandakan bahwa suatu aplikasi adalah mungkin berjalan menggunakan protokol yang

dikenal pada port yang tidak standar (misalnya mendeteksi protokol http pada non port selain 80), dan juga sebaliknya (misalnya mendeteksi aplikasi Skype yang berjalan pada port 80). Hal tersebut menyebabkan konsep penggunaan port tidak sama dengan aplikasi yang digunakan. Berikut beberapa contoh kategori protokol yang dimiliki oleh nDPI :

- Aplikasi Multimedia (Youtube, Netflix, Instagram)
- Aplikasi P2P (Bitorrent, Gnuttela, eDonkey)
- Aplikasi Game (Steam,World of Warcraft, Halflife 2)
- Dan lain-lain

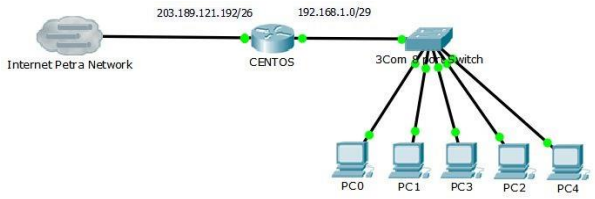
3. DESAIN SISTEM

3.1. Desain Jaringan

Dalam penelitian ini membutuhkan 2 jenis jaringan yaitu jaringan publik dan jaringan *private*. Jaringan publik adalah jaringan yang menyediakan akses internet yang dimiliki oleh UK Petra. Jaringan *private* merupakan jaringan lokal yang terhubung dengan komputer-komputer *host*. Jaringan *private* tersebut terhubung dengan jaringan publik melalui komputer Linux *server* yang berfungsi sebagai *router*. Jaringan *private* ini terdiri dari *switch* dan komputer-komputer *host* yang terhubung. Topologi jaringan *private* ini dapat dilihat pada Gambar 2.

Komputer *server* memiliki 2 *interface* ethernet yaitu ethernet 0 dan ethernet 1. Ethernet 0 merupakan *interface* penghubung untuk jaringan *private*, sedangkan ethernet 1 merupakan *interface* untuk jaringan publik UK Petra. Berikut pengaturan IP *address* masing-masing *interface* tersebut: untuk ethernet 1 memiliki IP *address* 203.189.121.203, *subnet mask* 255.255.255.192, *default gateway* 203.189.121.193, serta pengaturan DNS *server* 203.189.121.4 dan 203.189.121.7. Sedangkan pengaturan jaringan ethernet 0 memiliki IP address 192.168.1.1, *subnet mask* 255.255.255.248.

Pemberian IP address untuk komputer-komputer di lab tersebut menggunakan DHCP dari komputer *server* dengan pengaturan berikut: komputer-komputer *host* akan memiliki IP *address* dengan batas antara 192.168.1.2 – 192.168.1.6, *subnet mask* 255.255.255.248, dan *default gateway* 192.168.1.1. Serta memiliki pengaturan DNS server 203.189.120.4 dan 203.189.120.7.



Gambar 2. Desain Jaringan

3.1.1. Spesifikasi Komputer Server

Komputer *server* berfungsi sebagai *server* sekaligus berfungsi sebagai *router* antara jaringan publik UK Petra dan jaringan *private*. Komputer *server* akan menggunakan sistem operasi CentOS versi 7.1503 yang merupakan versi terbaru dari CentOS. Untuk menjadikan *server* sebagai *router* maka dibutuhkan konfigurasi dan instalasi paket-paket yang dibutuhkan termasuk juga nDPI. Komputer sebagai *server* yang ada di laboratorium-SC memiliki spesifikasi sebagai berikut:

Processor : Intel(R) Core(TM)2 Duo CPU E4400 @ 2.00GHz
RAM : 3GB
Ethernet Card 0 : 100 Mbps
Ethernet Card 1 : 100 Mbps
Storage : 80gb

3.1.2. Spesifikasi Jaringan Private

Berikut adalah spesifikasi jaringan *private* yang digunakan untuk penelitian ini di lab Sistem Cerdas (SC):

Topologi Lab SC : menggunakan topologi star
Switch : 3Com 8 port 10/100 Mbps
Kabel LAN : kabel utp category 5E

3.2. Desain Cache

Cache akan digunakan pada beberapa protokol *library* nDPI seperti bittorrent dan aplikasi *game online* steam. *Cache* akan berfungsi sebagai tempat penyimpanan sementara IP, dimana IP yang tersimpan adalah IP dari paket data yang sudah terklasifikasikan oleh *library* nDPI. *Cache* diharapkan dapat mempercepat dan meningkatkan presentase pendeteksian suatu klasifikasi protokol dalam library nDPI. Pembuatan *cache* akan menggunakan tempat penyimpanan *array static* dalam struct. Algoritma *cache replacement* yang akan digunakan adalah *Least Recently Used* (LRU). LRU merupakan algoritma *cache replacement* dasar yang penggunaanya dapat diimplementasi pada suatu sistem nyata. Berdasarkan jurnal [6] penggunaan LRU menunjukan bahwa algoritmanya lebih baik dari pada beberapa alogritma *cache* replacement lainya, akan tetapi pada penggunanya dapat memakan *resource* CPU yang tinggi. Segmen Program 1 merupakan gambaran desain dari struktur struct yang akan digunakan sebagai *cache*. Struct dari *cache* tersebut akan menyimpan beberapa *variabel* seperti :

- btaddr : bertipe data unassigned integer 32 bit yang digunakan untuk menyimpan *source* IP *address* dari paket data.
- Age : bertipe data long long integer yang digunakan untuk menghitung seberapa lama suatu *record* paket data telah tersimpan dalam *cache*.
- Hit : bertipe data long long integer yang digunakan untuk menghitung frekuensi seberapa sering suatu record IP dalam *cache* telah digunakan.
- Count_lru : bertipe data long long integer yang digunakan untuk menghitung *least recently used* dari suatu *record* dalam *cache*.

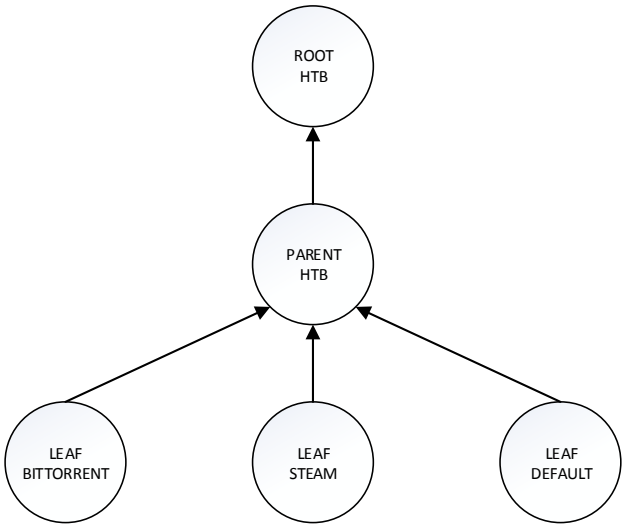
Segmen Program 1. Struktur struct dari *cache* LRU

```
#define CACHE_SIZE  1024
struct btcache
{
        u_int32_t btaddr;
        long long int hit, age;
        long long int count_lru;
};
```

3.3. Desain Queue HTB

Desain kelas *queue* dalam HTB ini akan melakukan *traffic shaping* pada kelas-kelas *queue leaf*. HTB terdiri dari strurktur hiearki *queue* seperti yang terlihat dalam Gambar 3. Struktur *queue* pada gambar tersebut terdiri dari root sebagai kelas utama, kemudian *parent* yang merupakan kelas yang dikonfigurasi untuk memiliki total *bandwidth*. Kemudian *inner class* dari *parent queue* terdiri dari 3 *leaf* yaitu *leaf* bittorrent, *leaf* Steam, *leaf default*. *Traffic* paket data yang masuk kedalam *queue* HTB ini diklasifikasikan sesuai *marking* dari iptables.

Leaf bittorrent dikonfigurasi untuk membatasi *bandwidth traffic* data bittorrent. *Leaf* Steam digunakan untuk membatasi *traffic* data aplikasi *game online* steam. Sedangakan *leaf default* dikonfigurasi untuk membatasi *traffic* data yang tidak terklasifikasikan dari iptables.



Gambar 3. Desain kelas HTB

4. IMPLEMENTASI SISTEM

Bab ini menjelaskan implementasi iptables dengan *libary* nDPI, penggunaan *cache* pada *library* nDPI, konfigurasi tc *queueing discipline*, dan konfigurasi *network monitoring*. Konfigurasi iptables dan tc akan menggunakan *script* dengan bahasa bash. Sedangkan implementasi *cache* menggunakan bahasa pemrograman C.

4.1. Konfigurasi iptables bittorrent

Seperti yang terlihat pada Pada Segmen Program 2 merupakan *script* bash yang berisi perintah untuk mengklasifikasi *traffic* data bittorrent sesuai dengan nDPI. Perintah tersebut dibuat pada tabel *mangle* pada iptables. Pada iptables dibuat *chain* baru "BITTORRENT" dimana akan berisi *rule* untuk melakukan *marking* paket data yang terklasifikasikan bittorrent oleh nDPI dengan perintah "MARK". Pada *traffic* data bittorrent akan menggunakan *mark* 1. Pengecekan *traffic* data bittorrent akan dilakukan pada saat data dari internet masuk ke *interface* ethernet 1 (*Prerouting*) dan saat data keluar dari ethernet 0 menuju ethernet 1 (*Postrouting*). Pada *chain* "BITTORRENT" juga dilakukan pencatatan pada log sistem.

Segmen Program 2. Bash *script* iptables bittorrent

```
#!/bin/sh
iptables -t mangle -N BITTORRENT
iptables -t mangle -A PREROUTING -i eth1 -m
ndpi --bittorrent -j BITTORRENT
iptables -t mangle -A POSTROUTING -o eth1 -
m ndpi --bittorrent -j BITTORRENT
iptables -t mangle -A BITTORRENT -j LOG --
log-prefix "IPT-BitTorrent:" --log-level 7
iptables -t mangle -A BITTORRENT -j MARK --
set-mark 1
iptables -t mangle -A BITTORRENT -j RETURN
```

4.2. Implementasi cache LRU

Pada Segmen Program 3 merupakan implementasi algoritma *cache replacement* LRU yang digunakan ketika isi dari *cache* sudah penuh dan harus ada *record* IP yang harus dihapus.

Segmen Program 3. Fungsi algoritma *cache replacement* LRU

```
int indexdel=-1;

long long int temp_age=-1,temp_lru=-1;

for(i=0;i<CACHE_SIZE;i++)
{
       if(arr[i].hit != 0){
              if(temp_lru<arr[i].count_lru)
              {
               temp_lru= arr[i].count_lru;
              }
       }
}
for(i=0;i<CACHE_SIZE;i++){
       if(arr[i].hit != 0){
       if(temp_lru==arr[i].count_lru){
              if(temp_age < arr[i].age)
                     {
                     temp_age = arr[i].age;
                     indexdel=i;
                     }
              }
       }
}
```

4.3. Konfigurasi HTB Queue

Seperti yang terlihat pada Segmen Program 4 merupakan *script* yang berisi perintah-perintah konfigurasi tc kelas HTB. Konfigurasi HTB digunakan pada *interface* ethernet 0 router Linux. *Bandwidth* pada kelas *parent* akan dikonfigurasi dengan kecepatan 4000 kbps atau setara dengan 500 KB/s. Kemudian pada kelas *leaf* dengan *identifier* 1:10 merupakan kelas *queue* untuk *traffic* data bittorrent / steam. Batas *bandwidth* yang dikonfigurasi pada bittorrent adalah 1600 kbps (200 KB/s). Pada kelas *leaf* dengan *identifier* 1:30 merupakan kelas *queue* untuk *traffic* data yang tidak terklasifikasikan. Limit *bandwidth* yang dikonfigurasi pada kelas *default* adalah 2400 kbps (300 KB/s).

Segmen Program 4. Bash *Script* konfigurasi HTB queue

```
#!/bin/sh


tc qdisc add dev eth0 root handle 1: htb
default 30


tc class add dev eth0 parent 1: classid 1:1
htb rate 4000kbit
tc class add dev eth0 parent 1:1 classid
1:10 htb rate 1600kbit ceil 1600kbit
tc class add dev eth0 parent 1:1 classid
1:30 htb rate 2400kbit ceil 2400kbit


tc filter add dev eth0 parent 1:0 protocol
ip handle 1 fw flowid 1:10
```

5. PENGUJIAN SISTEM

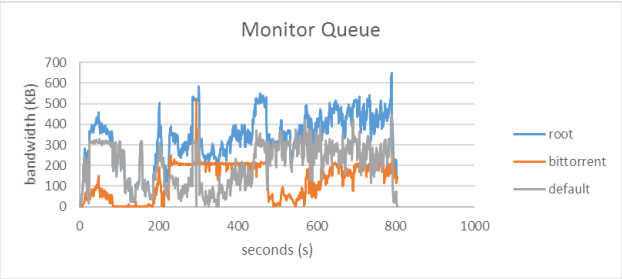
Menjelaskan pengujian-pengujian yang dilakukan yaitu mengenai kinerja nDPI pada aplikasi P2P (bittorrent) dan kinerja nDPI dengan *cache* pada beberapa ukuran *cache* dan kecepatan *bandwidth*.

5.1. Pengujian Aplikasi Bittorrent

5.1.1. Pengujian Bittorrent dengan nDPI

Pengujian Bittorrent ini dilakukan dengan menggunakan *library* dari nDPI yang berupa *file* bittorrent.c. Penggunan nDPI bittorrent akan ditambahkan pada *router* Linux dengan menggunakan perintah iptables. *Traffic shaping* dilakukan dengan menggunakan konfigurasi tc. Konfigurasi iptables dan tc menggunakan *script* yang terdapat dalam bab sebelumnya. Konfigurasi tc *queue* menggunakan kelas HTB dengan *leaf* 2 kelas. Kelas yang pertama merupakan bittorrent dengan batas *bandwidth* 200 KB/s dan kelas yang kedua merupakan *default* dengan batas *bandwidth* 300 KB/s. Konfigurasi *queue* tersebut digunakan untuk mengevaluasi keberhasilan *library* nDPI dalam melakukan pengklasifikasian paket data bittorrent.

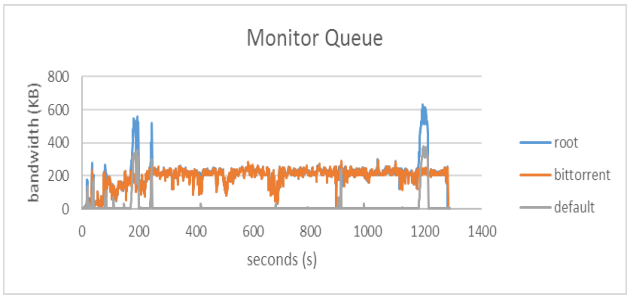
Pengujian aplikasi bittorrent ini juga dievaluasi dengan memonitor *bandwidth traffic* internet dari *router* dengan mengunakan 2 *script* monitor yang terdapat dalam bab sebelumnya. Pada Gambar 4 merupakan hasil grafik monitor bandwidth internet setiap detik pada *interface* ethernet 0 *router* Linux. Pada grafik hasil monitor tersebut terlihat bahwa *bandwidth* aplikasi bittorrent tidak stabil pada batas seharusnya (200KB/s). *Bandwidth* rata-rata yang digunakan oleh torrent selama pengujian berlangsung yaitu 298KB/s. Terlihat juga pada grafik hasil monitor tiap kelas pada konfigurasi *queue* yang digunakan. Sepeti yang terlihat pada grafik monitor *queue* tersebut terdapat banyak *traffic* pada kelas *queue default*. Hal ini disebabkan nDPI yang digunakan pada iptables, banyak tidak mendeteksi paket-paket data bittorrent. Oleh sebab itu iptables tidak dapat melakukan *marking* dan pada konfigurasi *queue* paket data bittorrent akan masuk ke dalam kelas *default*. Hal tersebut menyebabkan *bandwidth traffic* internet total aplikasi torrent akan melebihi batas.



Gambar 4. Grafik Monitor *queue* bittorrent nDPI

5.1.2. Pengujian BitTorrent dengan nDPI cache 1024 pada bandwidth 200KB

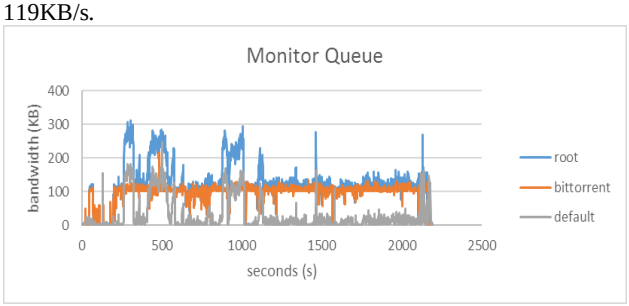
Pengujian aplikasi bittorrent ini juga dievaluasi dengan memonitor *bandwidth traffic* internet dari *router* dengan menggunakan 2 *script* monitor yang terdapat dalam bab 4. Pada Gambar 5 merupakan hasil grafik monitor *bandwidth* internet setiap detik pada *interface* Ethernet 0 router Linux. Pada grafik hasil monitor *interface* Ethernet 0 tersebut terlihat juga bahwa *bandwidth* torrent stabil pada limit seharusnya (200KB/s). *Bandwitdh* rata-rata yang digunakan oleh torrent selama pengujian berlangsung yaitu 196KB/s. Pada grafik hasil monitor tiap kelas pada konfigurasi *queue* yang digunakan. Seperti yang terlihat pada grafik monitor *queue* tersebut *bandwidth* berjalan pada batas dan terkadang sedikit *traffic* berada dalam kelas *default*. Adanya *traffic* pada kelas *default* menyebakan kecepatan *bandwidth* bitttorrent pada klien melebihi batasnya. Hal tersebut disebabkan adanya *traffic* data bittorrent yang tidak dapat diklasifikasikan oleh nDPI bittorrent *cache*.



Gambar 5. Grafik Monitor *queue* bittorrent nDPI *cache* 1024 *bandwidth* 200KB

5.1.3. Pengujian Bittorrent dengan nDPI cache 1024 pada bandwidth 100KB

Pengujian nDPI *cache* 1024 ini juga dievaluasi dengan memonitor *bandwidth traffic* internet dari *router* dengan menggunakan 2 *script* monitor yang terdapat dalam bab 4. Pada Gambar 6 merupakan hasil grafik monitor *bandwidth* internet setiap detik pada *interface* ethernet 0 *router* Linux. Pada grafik hasil monitor *interface* ethernet 0 tersebut terlihat bahwa *bandwidth* bittorrent rata-rata berada pada batas seharusnya yaitu 100KB/s. Pada grafik hasil monitor tiap kelas pada konfigurasi *queue* yang digunakan. Dikarenakan nDPI bittorrent yang tidak dapat mengklasifikasikan *traffic* torrent secara maksimal maka menyebabkan adanya *traffic* pada kelas *default*. Adanya *traffic* pada kelas *default* menyebakan *bandwidth* aplikasi bittorrent pada klien melebihi batas yang seharusnya. Terlihat dari grafik monitor *queue* pada waktu awal pengujian kecepatan *bandwidth* tidak stabil. Rata-rata kecepatan *bandwidth* yang digunakan oleh bittorrent selama pengujian berlangsung yaitu 119KB/s.

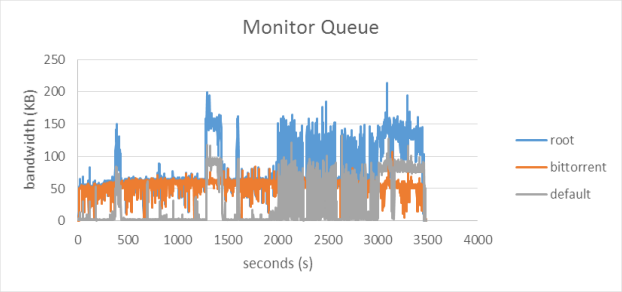


Gambar 6. Grafik Monitor *queue* bittorrent nDPI cache 1024 *bandwidth* 100KB

5.1.4. Pengujian Bittorrent dengan nDPI cache 1024 pada bandwidth 50KB

Pengujian nDPI *cache* 1024 ini juga dievaluasi dengan memonitor *bandwidth traffic* internet dari *router* dengan menggunakan 2 *script* monitor yang terdapat dalam bab 4. Pada Gambar 7 merupakan hasil grafik monitor *bandwidth* internet

setiap detik pada *interface* ethernet 0 *router* Linux. Pada grafik hasil monitor *interface* ethernet 0 tersebut terlihat bahwa *bandwidth* bittorrent rata-rata berada pada batas seharusnya yaitu 50KB/s. Pada grafik hasil monitor tiap kelas pada konfigurasi *queue* yang digunakan. Dikarenakan nDPI bittorrent yang tidak dapat mengklasifikasikan *traffic* torrent secara maksimal, sehingga yang menyebabkan kecepatan *bandwidth* aplikasi bittorrent melebihi batas *bandwidth* yang seharusnya. Terlihat dari grafik monitor *queue* pada waktu akhir pengujian kecepatan *bandwidth* berjalan menjadi tidak stabil. Rata-rata kecepatan *bandwidth* yang digunakan oleh bittorrent selama pengujian berlangsung yaitu 75 KB/s.



Gambar 7. Grafik Monitor *queue* bittorrent nDPI *cache* 1024 *bandwidth* 50KB

5.1.5. Ringkasan Bab

Tabel 1. Hasil pengujian *cache* pada berbagai *size*

No	Type	Hit	Miss	Total Access	Ratio Hitted (%)
1	nDPI + cache 512	3070	3427	6497	47.25%
2	nDPI + cache 1024	3490	2600	6090	57.31%
3	nDPI + cache 2048	3009	2155	5164	58.27%
4	nDPI + cache 4096	2719	3567	6286	43.26%

Tabel 1. menunjukkan percobaan aplikasi bittorrent yang menggunakan nDPI *cache* pada berbagai *size*. Dari tabel tersebut dapat dilihat bahwa *cache* dengan *size* 512 dan 4096 merupakan besar *cache* yang tidak baik dimana rasio *cache* lebih tidak digunakan (*miss* yang lebih besar dari *hit*). Sedangkan *cache* dengan besar 2048 memiliki rasio *hit* yang baik akan tetapi dibandingkan dengan *cache* dengan 1024 yang 2x lebih kecil rasio *hit*nya hanya berbeda kurang dari 1%. Oleh karena itu dapat disimpulkan bahwa *cache* dengan *size* 1024 merupakan besar *cache* yang paling direkomendasikan dalam penggunannya pada nDPI.

Tabel 2. Hasil pengujian nDPI *cache* pada berbagai kecepatan *bandwidth*

Type	Bittorrent bandwidth (KB/s)	Default bandwidth (KB/s)	Total bandwidth (KB/s)
nDPI + cache 200 KB/s	192 KB/s	16 KB/s	208 KB/s
nDPI + cache 100 KB/s	96 KB/s	29 KB/s	125 KB/s
nDPI + cache 50 KB/s	55 KB/s	26 KB/s	81 KB/s

Tabel 2. menunjukkan percobaan nDPI dengan *cache* pada berbagai kecepatan *bandwidth*. *Bandwidth* untuk bittorrent diuji pada kecepatan 200KB/s, 100KB/s dan 50 KB/s. Dari tabel tersebut dapat disimpulkan bahwa dengan melakukan implementasi nDPI *cache* berhasil dilakukan *traffic shaping* bittorrent pada kecepatan *bandwidth* yang telah ditentukan. Akan tetapi dikarenakan nDPI tidak dapat mengklasifikasikan *traffic* data bittorrent dengan sempurna maka akan menimbulkan *traffic* pada kelas *default*.

6. KESIMPULAN DAN SARAN

Dari proses perancangan, pembuatan, dan hasil evaluasi pengujian Linux *router* untuk melakukan *traffic shaping* dengan iptables, nDPI, dan *traffic control* dapat diambil kesimpulan sebagai berikut:

- Linux dapat dikonfigurasi menjadi sebagai *router* untuk melakukan *forward* paket-paket data dalam jaringan.
- Kemampuan nDPI untuk mengklasifikasikan *traffic* data bittorrent belum sepenuhnya sempurna sehingga beberapa paket data yang seharusnya bittorrent menjadi tidak terklasifikasikan.
- Kemampuan penggunaan iptables untuk melakukan *marking* pada paket-paket data dalam jaringan berfungsi dengan baik. Akan tetapi proses *marking* paket data sangat bergantung pada kemampuan nDPI untuk mengklasifikasikan jenis paket data yang ada.
- Implementasi *cache* pada *library* nDPI bittorrent dapat meningkatkan keberhasilan *traffic shaping* pada Linux *router*.
- Konfigurasi *queue* HTB dapat melakukan *traffic shaping* yang sesuai dengan limit *bandwidth* yang dikonfigurasi.

Saran untuk penggunaan nDPI *library* dan *traffic control* pada Linux *router* adalah sebagai berikut:

- Penggunaan nDPI *library* yang digunakan pada penelitian ini dapat dilakukan *upgrade* ke dalam *library* nDPI versi yang lebih baru sehingga pengelompokan paket-paket data dapat lebih sempurna.
- Pengaturan konfigurasi *traffic control* dalam Linux dapat lebih dioptimasi lagi kemampuan untuk melakukan *traffic shaping*nya dengan kombinasi *queueing discipline* yang berbeda-beda.

7. REFERENSI

[1] CentOS. *CentOS Linux*. URI= http://www.centos.org/about/

[2] Cisco. 2015. Cisco Visual Networking Index: Forecast and Methodology, 2013-2018. *Cisco White Paper C11-481360.URI=* http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn--next-generation-network/white_paper_c11-481360.html

[3] Cisco. 2005. QoS Policing: Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting. *Cisco Document ID* 19645. URI= http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html

[4] Gebert, S., Pries, R., Schlosser, D., and Heck, K. 2012. Internet Access Traffic Measurement and Analysis. *Proc. of 4th International Conference on Traffic Monitoring and Analysis (TMA)*. Vienna (Austria). pp. 29–42.

[5] Hubert, B. 2012. *Traffic Control*. URI= http://www.lartc.org/howto/

[6] Swain, D., Dash, N.B., Shamkuwar, D.O., Swain, D. 2011. Analysis and Predictability of Page Replacement Techniques towards Optimized Performance. *IJCA Proceedings on International Conference on Recent Trends in Information Technology and Computer Science (ICRTITCS-2011)*, icrtitcs(3):12-16.

[7] The ntop Team. *Open and Extensible LGPLv3 Deep Packet Inspection Library*. URI= http://www.ntop.org/products/deep-packet-inspection/ndpi