IMPLEMENTASI ALGORITMA GREEDY SEBAGAI PENENTUAN ALOKASI BANDWIDTH SECARA DINAMIS

Mikail Ilham Bramantya*), Imam Santoso[1] dan Aghus Sofwan[2]

[1]Departemen Teknik Elektro, Universitas Diponegoro, Semarang,
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

*) *E-mail: mikaililham@students.undip.ac.id*

Abstrak

Penelitian ini bertujuan untuk membuat dan mengimplementasi program alokasi bandwidth dinamis yang dapat mengalokasikan besar bandwidth pada setiap node jaringan berdasarkan keadaan trafik di node tersebut. Pada penelitian ini, Algoritma Greedy digunakan dalam penentuan besar bandwidth karena dianggap mampu dalam memecahkan masalah kombinatorial dalam mengoptimisasi sebuah sistem. Pengujian dilakukan pada jaringan uji coba dan UNDIPConnect Fakultas Teknik Universitas Diponegoro. Berdasarkan hasil pengujian, program berhasil mengambil informasi besar trafik pada jaringan serta mengalokasikan bandwidth ke setiap node sesuai dengan perhitungan. Didapatkan juga rata-rata nilai Jain's Fairness Index program sebesar 0.585 saat pensimulasian program pada jaringan UNDIPConnect Fakultas Teknik Universitas Diponegoro yang diberi trafik berdasarkan data rekap trafik dari Agustus 2019 hingga Juli 2020.

Kata kunci: Alokasi Bandwidth, Algoritma Greedy, Trafik, Jain's Fairness Index

Abstract

This final project research aims to create and implement a dynamic bandwidth allocation program that can allocate bandwidth to each network node based on the traffic conditions at that node. In this study, the Greedy Algorithm is used in determining the amount of bandwidth because it is considered capable of solving combinatorial problems in optimizing a system. Testing was conducted on both prototype network and UNDIPConnect, of Engineering Faculty of Universitas Diponegoro. Based on the test results, the program is successful in retrieving traffic information on the network and allocating bandwidth to each node according to the equation. There was also an average score for the Jain's Fairness Index program of 0.585 when simulating the program on the UNDIPConnect network of the Faculty of Engineering, Diponegoro University, which was given traffic based on traffic recap data from August 2019 to July 2020.

Keywords: Bandwidth Allocation, Greedy Algorithm, Traffic, Jain's Fairness Index

1. Pendahuluan

Pada era digital ini, koneksi internet telah menjadi sebuah kebutuhan bagi sebagian besar orang. Berdasarkan survei yang dilakukan oleh Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) pada tahun 2018 [1], sebanyak 171,17 juta atau sekitar 64,8% masyarakat Indonesia telah menggunakan internet dalam kehidupan sehari-harinya. Bagi mahasiswa, koneksi internet merupakan salah satu penunjang kegiatan akademik. Tidak hanya mahasiswa, dosen dan pegawai lainnya di sebuah perguruan tinggi pun juga membutuhkan koneksi internet untuk dapat melakukan kegiatan operasional sehari-hari. Untuk memenuhi kebutuhan tersebut, Universitas Diponegoro memberikan fasilitas Wi-Fi, UNDIPConnect, yang dapat memberikan akses internet berkecepatan tinggi kepada penggunanya. Setiap pemilik email Universitas Diponegoro dapat menggunakan fasilitas ini selama masih berada di dalam jangkauan jaringan. Universitas Diponegoro pun telah memasang access point yang tersebar di wilayah kampus sehingga fasilitas Wi-Fi dapat diakses dengan mudah.

Algoritma Greedy dapat digunakan untuk memecahkan masalah kombinatorial dalam mengoptimisasi sebuah sistem [2][3][4]. Algoritma ini melakukan optimasi dengan cara bertahap sesuai dengan kriteria optimal yang diinginkan[5][6][7]. Pada penelitian [8] dan [9] dilakukan manajemen *bandwidth* menggunakan algoritma Greedy dengan menjadikan tingkat prioritas layanan sebagai salah satu masukan dalam pengalokasian bandwidth.

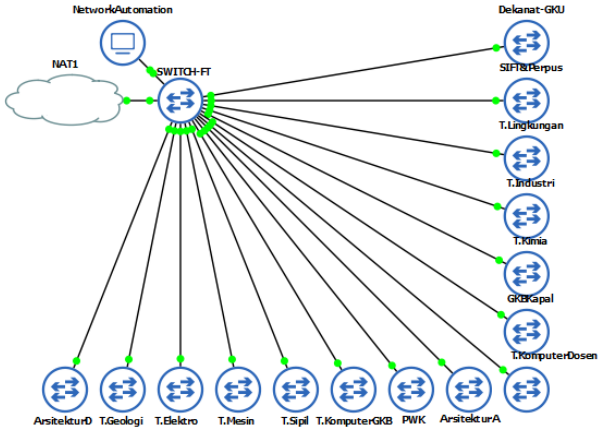
Penelitian [10] dan [11] dilakukan untuk mengevaluasi efisiensi manajemen *bandwidth* yang digunakan pada suatu perusahaan. Metode *traffic shaping* digunakan peneliti adalah dengan teknik *Packet First In First Out* (PFIFO) dan *Per Connection Queue* (PCQ). Utilitas jaringan menggunakan metode PCQ juga menunjukkan hasil yang lebih baik dimana nilai standar deviasinya paling kecil jika dibandingkan dengan penggunaan metode

PFIFO dan tidak menggunakan *traffic shaper*. Hal ini menandakan bahwa distribusi *bandwidth* pada jaringan lebih seimbang.

Penelitian ini dilakukan untuk menciptakan sistem manajemen bandwidth yang dapat mengutilisasi sumber daya bandwidth jaringan secara optimal dengan memperhatikan performansi jaringan yang memberikan kenyamanan kepada pengguna dalam mengakses jaringan. Sistem yang diusulkan dalam penelitian ini mampu mengalokasikan bandwidth sesuai kebutuhan yang besarnya ditentukan dari trafik data yang terbangkitkan dari pengguna. Sistem juga dapat mengalokasikan bandwidth secara dinamis mengalokasikan bandwidth jika terjadi peningkatan atau penurunan trafik data di lokasi-lokasi tertentu.

2. Metode

Pembuatan program dilakukan dengan menggunakan 2 (dua) perangkat lunak, yaitu GNS3 2.2.11 sebagai *simulator* untuk menjalankan program serta Sublime Text 3 sebagai *text editor* untuk menuliskan senarai program. Program dirancang untuk bisa mengalokasikan bandwidth pada jaringan UNDIPConnect Fakultas Teknik Universitas Diponegoro. Topologi UNDIPConnect Fakultas Teknik Universitas Diponegoro.



Gambar 1. Topologi Perancangan Program

Pada Gambar 1 terlihat terdapat 3 (dua) komponen utama, yaitu *NetworkAutomation*, NAT, dan *Switch*. Komponen *NetworkAutomation* berfungsi untuk menjalankan program alokasi *bandwidth* dimana hasil perhitungan program akan diimplementasikan pada antarmuka *Switch*. Dalam proses pembuatan program, komponen *Switch-FT* dihubungkan dengan 15 *Switch* yang merepresentasikan 15 *node* departemen dan bagian administrasi yang ada di Fakultas Teknik Universitas Diponegoro. Komponen NAT berfungsi untuk memberikan IP address kepada jaringan.

2.1. Memasukkan Nama Antarmuka

Program yang dibuat hanya akan melakukan pengalokasian *bandwidth* pada antarmuka yang diinginkan. Maka dari itu, nama antarmuka harus dimasukkan secara manual sebelum program berjalan di sebuah *file* yang terpisah. Pada pembuatan program ini digunakan 15 node yang merepresentasikan departemen dan bagian administrasi yang ada di Fakultas Teknik Universitas Diponegoro.

Tabel 1. Hubungan Node dengan Switch-FT

No.	Nama Node	Antarmuka pada *Switch-FT*
1	*NetworkAutomation*	*Ethernet0/0*
2	NAT	*Ethernet0/1*
3	Dekanata-GKU	*Ethernet1/0*
4	SIFT&Perpus	*Ethernet1/1*
5	T. Lingkungan	*Ethernet1/2*
6	T. Industri	*Ethernet1/3*
7	T. Kimia	*Ethernet2/0*
8	GKB Kapal	*Ethernet2/1*
9	T. Komputer Dosen	*Ethernet2/2*
10	Arsitektur A	*Ethernet2/3*
11	PWK	*Ethernet3/0*
12	T. Komputer GKB	*Ethernet3/1*
13	T. Sipil	*Ethernet3/2*
14	T. Mesin	*Ethernet3/3*
15	T. Elektro	*Ethernet4/0*
16	T. Geologi	*Ethernet4/1*
17	Arsitektur D	*Ethernet4/2*

Nama antarmuka yang dipilih harus dimasukkan ke dalam satu file pada komponen NetworkAutomation. Pembuatan file ini dapat dilakukan dengan cara mengakses console pada NetworkAutomation. antarmuka *Ethernet0/0* dan *Ethernet0/1* tidak dituliskan dalam program. *Ethernet1/0* hingga *Ethernet4/2* dimasukkan karena tersambung dengan *node* yang ingin diikutsertakan dalam program pengalokasian *bandwidth*. Pada perancangan ini, dibuat *file* baru yang memiliki nama *"listinterface"*.

2.2. Koneksi Telnet

Pada perancangan program ini, jenis komunikasi antara kedua komponen menggunakan protokol *telnet*. Untuk melakukan komunikasi, protokol *telnet* membutuhkan *Internet Protocol* (IP) *address* komponen tujuan.

```
import getpass
HOST = "X.X.X.X" #IP Address Destinasi
user = input("Enter your username: ")
password = getpass.getpass()
```

2.3. Parameter Alokasi *Bandwidth*

Paramater yang digunakan dalam program ini terdiri dari 3 parameter, yaitu besar total *bandwidth*, batas trafik berlebih, dan *bandwidth* minimal. Untuk mendeklarasikan ketiga parameter tersebut, program ditambahkan senarai sebagai berikut:

```
total_BW = xxxx #dalam satuan bit
BW_min = xxxx #dalam satuan bit
Trafik_max = xxxx #dalam satuan bit
```

Nilai yang dimasukkan ke dalam senarai tersebut adalah dalam satuan bit.

2.4. Login Telnet

Protokol *telnet* mengharuskan pengguna yang ingin mengakses sebuah komponen secara *remote* untuk memasukkan *username* dan *password* yang telah terdaftar. Pengguna *telnet* harus melakukan *login* di awal pengaksesan. Oleh karena itu, program dirancang untuk melakukan *login* dengan memasukkan informasi koneksi telnet yang telah dilakukan di tahap pertama.

```
tn = telnetlib.Telnet(HOST)
tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")
```

Komponen yang dituju adalah sesuai dengan informasi *IP address* yang dimasukkan di awal program. *Username* dan *password* yang digunakan juga sesuai dengan informasi yang dimasukkan pada awal program. Informasi tersebut dienkripsikan dengan skema *American Standard Code for Information Interchange* (ASCII) agar dapat dibaca oleh komponen *Switch.*

2.5. Alokasi Awal *Bandwidth*

Saat pertama kali program dijalankan, program akan mengalokasikan *bandwidth* pada antarmuka sebesar parameter *bandwidth* minimal. Hal ini dilakukan untuk menyetarakan besar *bandwidth* awal pada semua antarmuka serta memunculkan informasi tambahan yang akan digunakan pada langkah selanjutnya.

```
tn.write(b"enable\n")
tn.write(b"cisco\n")
tn.write(b"terminal length 0\n")
tn.write(b"config t\n")
f = open ('listinterface')
for interface in f:
    interface = interface.strip() tn.write(b"int
    " + str(interface).encode('ascii') + b"\n")
    tn.write(b"bandwidth " + str(BW_min).encode
    ('ascii') + b"\n")
    tn.write(b"exit\n")
tn.write(b"exit\n")
tn.write(b"exit\n")
print (tn.read_all().decode('ascii'))
```

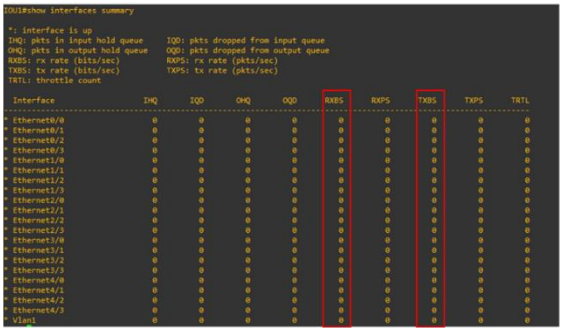
Senarai di atas berfungsi untuk mengalokasikan *bandwidth* ke semua antarmuka yang terdapat pada *file* nama antarmuka. Senarai ini berfungsi untuk menuliskan perintah pada *Switch* dengan menggunakan protokol *telnet*.

2.5. Pengambilan Data Trafik dan *Bandwidth*

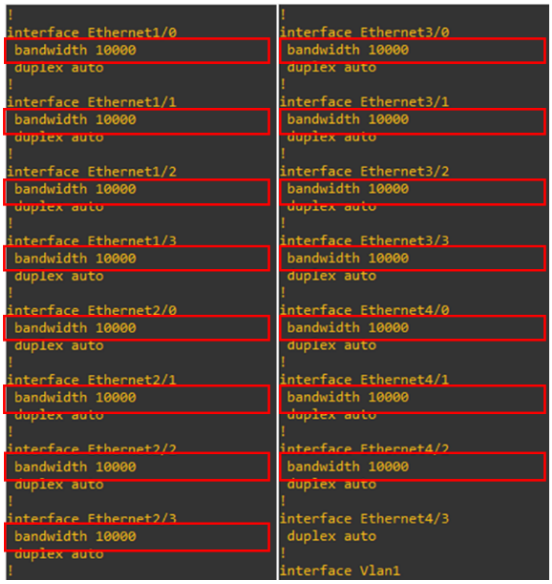
Hal pertama yang dilakukan pada *loop* program ini adalah mengambil informasi trafik dan besar *bandwidth* pada antarmuka. Hal tersebut dilakukan dengan senarai sebagai berikut:

```
tn.write(b"enable\n")
tn.write(b"cisco\n")
tn.write(b"terminal length 0\n")
tn.write(b"show interfaces summary\n")
tn.write(b"show run\n")
tn.write(b"exit\n")
str1 = tn.read_all().decode('ascii')
```

Senarai di atas berfungsi untuk menuliskan perintah menunjukkan informasi komponen *Switch*. Senarai ini berfungsi untuk menampilkan informasi keadaan trafik dan menampilkan besar *bandwidth* yang telah dialokasikan pada setiap antarmuka seperti pada Gambar 2 dan Gambar 4 Senarai pada baris terakhir berfungsi untuk menyimpan informasi-informasi sebelumnya.



Gambar 2. Informasi Keadaan Trafik



Gambar 3. Informasi Besar Bandwidth Antarmuka.

Setelah informasi trafik dan *bandwidth* didapatkan, informasi-informasi tersebut disusun dalam sebuah *list* agar lebih mudah untuk diolah. Informasi-informasi tersebut disusun ke dalam sebuah *list* seperti terlihat pada Gambar 4.

```
xyz = []
f = open ('listinterface')
for interface in f:
    interface = interface.strip()
    int1  = (str1.index(interface))
    intinfo1 = ((str1[int1: int1 +
    113]).split())
    int2 = (str1.index('interface ' +
    str(interface)))
    intinfo2 = ((str1[int2: int2 + 50]).split())
    array =[(intinfo1[0]), (int(intinfo1[5])) +
    (int(intinfo1[7])),(int(intinfo2[3]))]
    xyz.append(array)
```

```
Ini list XYZ (Antarmuka | Trafik | Bandwidth):
['Ethernet1/0', 0, 10000]
['Ethernet1/1', 0, 10000]
['Ethernet1/2', 0, 10000]
['Ethernet1/3', 0, 10000]
['Ethernet2/0', 0, 10000]
['Ethernet2/1', 0, 10000]
['Ethernet2/2', 0, 10000]
['Ethernet2/3', 0, 10000]
['Ethernet3/0', 0, 10000]
['Ethernet3/1', 0, 10000]
['Ethernet3/2', 0, 10000]
['Ethernet3/3', 0, 10000]
['Ethernet4/0', 0, 10000]
['Ethernet4/1', 0, 10000]
['Ethernet4/2', 0, 10000]
```

Gambar 4. Tampilan List XYZ.

2.5. Perhitungan Alokasi *Bandwidth* dengan Algoritma *Greedy*

Merujuk pada penelitian [8], kategori tersebut adalah antarmuka *idle*, *non-greedy*, dan *greedy*. Kategori *idle* merupakan antarmuka yang memiliki trafik di bawah parameter *bandwidth* minimal, *non-greedy* merupakan antarmuka yang memiliki trafik di atas *bandwidth* minimal dan di bawah batas trafik berlebih, dan *greedy* merupakan antarmuka yang memiliki trafik di atas batas trafik berlebih.

Pengalokasian *bandwidth* langsung pada kategori *idle* dan *non-greedy*. Antarmuka *idle* akan dialokasikan *bandwidth* sebesar *bandwidth* minimal dan antarmuka *non-greedy* akan dialokasikan sebesar nilai terkecil antara 2 kali besar trafik atau nilai batas trafik berlebih. Antarmuka *greedy* akan dialokasikan sebesar batas trafik berlebih dan diikutsertakan dalam perhitungan tambahan *bandwidth*.

Selanjutnya, program akan memberikan nilai prioritas berdasarkan besar nilai trafik berlebih. Nilai prioritas diinisialisasikan menjadi 4 tingkatan seperti terlihat pada Tabel 3.3. Setelah itu program akan menambahkan nilai value pada tiap antarmuka dan mengurutkan antarmuka berdasarkan *value*nya.

Tabel 2. Penentuan Tingkat Prioritas

Tingkat Prioritas	*Value*	Keterangan
A	1	Trafik Berlebih ≤ 0,25 x Trafik_max
B	2	Trafik Berlebih ≤ 0,5 x Trafik_max
C	3	Trafik Berlebih ≤ 0,75 x Trafik_max
D	4	Trafik Berlebih > 0,75 x Trafik_max

```
for l in range (len(list_a)):
    item_list_a = list_a[l]
    trafik_lebih = item_list_a[1] - Trafik_max
    if trafik_lebih <= 0.25 * Trafik_max
        list_a[l].append(1)
    elif trafik_lebih <= 0.5 * Trafik_max:
        list_a[l].append(2)
    elif trafik_lebih <= 0.75 * Trafik_max:
        list_a[l].append(3)
    elif trafik_lebih > 0.75 * Trafik_max:
        list_a[l].append(4)
list_a.sort(key=lambda x: x[3], reverse=True)
```

Setelah didapatkan klasifikasi tingkat prioritas, selanjutnya dilakukan penyesuaian nilai *bandwidth*. Penentuan nilai alokasi *bandwidth* berdasarkan tingkat prioritas (tinggi, sedang, dan rendah) menggunakan metode *Fractional Knapsack Problem* dengan menggunakan persamaan (1) yang diusulkan pada [8].

$$Wi = Xi \times Spare\_BW \quad (1)$$

Keterangan:
Wi : Alokasi *bandwidth* value ke-i (bps)
Xi : Proporsi *Bandwidth* ($0 < Xi < 1$)
$Spare\_BW$: Sisa *Bandwidth* (bps)

Kemudian program akan menyusun informasi tersebut dalam sebuah matriks. Data yang disusun adalah nilai *value* dan nilai alokasi *bandwidth* pada *value* tersebut (Wi). Lalu, program akan mengelompokkan antarmuka berdasarkan nilai *value*nya.

Setelah dilakukan pengelompokkan dengan senarai di atas, program akan membuat tabel yang berisikan informasi kelebihan trafik tiap antarmuka pada nilai *value i*. Informasi trafik berlebih dibutuhkan untuk melakukan perhitungan pengalokasian *bandwidth* setiap nilai *value*. Perhitungan tersebut dilakukan dengan persamaan (2).

$$Rij = \frac{Cij}{Ci} \quad (2)$$

Keterangan:
Rij : Proporsi *bandwidth* antarmuka ke-*j* pada value ke-*i*
Cij : Konsumsi *bandwidth* antarmuka ke-*j* pada value ke-*i*
Ci : Konsumsi *bandwidth* pada value ke-*i*

Selanjutnya, penghitungan alokasi *bandwidth* tiap layanan dilakukan menggunakan persamaan (3).

$$Wij = Rij \times Wi \quad (3)$$

Keterangan:
Wij : Alokasi *bandwidth* antarmuka ke-j pada *value* ke-i
Rij : Proporsi *bandwidth* antarmuka ke-j pada *value* ke-i
Wi : Alokasi *bandwidth* value ke-i

2.6. Alokasi *Bandwidth* Hasil Perhitungan

Dari informasi yang didapatkan, program akan kembali melakukan koneksi *telnet* ke komponen *Switch* untuk melakukan alokasi *bandwidth* pada tiap antarmuka. Hal ini dilakukan dengan senarai sebagai berikut:

```
tn = telnetlib.Telnet(HOST)
tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")

#Rangkaian command untuk mengalokasikan
bandwidth
tn.write(b"enable\n")
tn.write(b"cisco\n")
tn.write(b"terminal length 0\n")
tn.write(b"config t\n")

#Mengalokasikan bandwidth sesuai hasil
perhitungan
for l in range (len(xyz)):
    item_xyz = xyz[l]
    tn.write(b"int " +
    (str(item_xyz[0])).encode('ascii') +
    b"\n")
    tn.write(b"bandwidth " +
    (str(item_xyz[2])).encode('ascii') + b"\n")
    tn.write(b"exit\n")
tn.write(b"exit\n")
```

Setelah alokasi *bandwidth* pada komponen *Switch* selesai, program akan kembali ke awal *loop* untuk melakukan pengambilan data trafik. Interval pengulangan *loop* dapat diatur sesuai dengan keinginan pengguna. Hal ini dilakukan dengan senarai sebagai berikut:

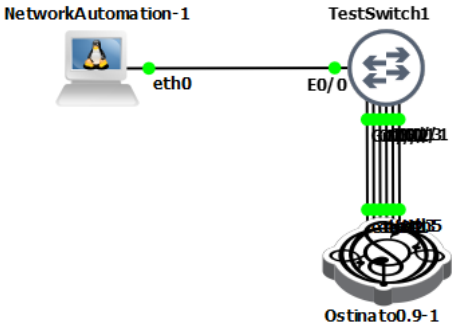
```
import schedule
import time

schedule.every(10).seconds.do(dynamicBW)

while True:
    schedule.run_pending()
    time.sleep(1)
```

Fungsi pustaka *schedule* dan *time* agar program dapat mengenali satuan waktu dan membaca waktu. Program akan terus berjalan sesuai interval yang ditentukan pada senarai di atas.

3. Pengujian dan Analisis

3.1 Implementasi Program Alokasi Bandwidth Dinamis

Sebuah jaringan diberikan trafik percobaan untuk menguji program dalam mengalokasikan *bandwidth* pada tiap antarmuka sesuai perhitungan algoritma *Greedy* yang digunakan. Jaringan percobaan ini terdiri dari 6 *node* yang membangkitkan trafik dan kelima *node* tersebut dikoneksikan ke antarmuka sebuah *switch* di mana program alokasi *bandwidth* dinamis akan dijalankan pada *switch* tersebut seperti pada Gambar 5.



Gambar 5. Topologi Percobaan

No.	Komponen	Antarmuka pada *Switch*
1.	*NetworkAutomation*	*Ethernet0/0*
2.	*eth1*	*Ethernet0/1*
3.	*eth2*	*Ethernet0/2*
4.	*eth3*	*Ethernet0/3*
5.	*eth4*	*Ethernet1/0*
6.	*eth5*	*Ethernet1/1*
7.	*eth6*	*Ethernet1/2*

Nama antarmuka harus dimasukkan ke dalam file '*listinterface*'. Keenam *node* terhubung dengan antarmuka *switch*. Trafik kemudian dibangkitkan menggunakan komponen '*Ostinato*' yang dihubungkan langsung dengan *node*. Trafik dari semua *node* memiliki destinasi di '*NetworkAutomation*' yang telah terhubung dengan *switch* pada antarmuka *Ethernet0/0*. Dengan skenario seperti ini, trafik akan melewati *switch* terlebih dahulu sehingga *switch* dapat membaca besar trafik yang melewati antarmuka *switch* sehingga program dapat mengambil informasi tersebut untuk proses pengalokasian *bandwidth*. Besar trafik yang dikirimkan pada tiap *node* dirincikan pada Tabel 3.

Tabel 3. Trafik Tiap Antarmuka.

No.	Antarmuka	Besar Trafik (bits)
1.	*Ethernet0/1*	5000
2.	*Ethernet0/2*	16000
3.	*Ethernet0/3*	2000
4.	*Ethernet1/0*	10000
5.	*Ethernet1/1*	18000
6.	*Ethernet1/2*	20000

Parameter program pengalokasian *bandwidth* kemudian ditentukan dengan mengubah senarai dalam program. Parameter yang diatur adalah sebagai berikut:

- *Bandwidth* Total : 100000 bps
- *Bandwidth* Minimal : 4000 bps
- Trafik Maksimal : 12000 bps

Selanjutnya dilakukan perhitungan untuk mengetahui besar alokasi *bandwidth* yang sesuai dengan perhitungan alokasi *bandwidth* dan dilakukan juga pengimplementasian program alokasi *bandwidth* dinamis pada jaringan uji coba. Hal ini bertujuan untuk mengetahui kesesuian program dalam mengalokasi *bandwidth* dengan perhitungan yang telah ditentukan.

Tiap *node* dapat diklasifikasikan berdasarkan besar trafiknya. Tiap *node* memiliki prioritasnya masing-masing yang akan mempengaruhi besar *bandwidth* saat pengalokasian. Maka dari itu, trafik setiap *node* dapat diklasifikasikan berdasarkan tingkat prioritasnya seperti pada Tabel 4.4

Tabel 4. Klasifikasi Antarmuka Percobaan Perhitungan.

No.	Nama *Node*	Besar Trafik	Tingkat Prioritas	*Value*	Keterangan
1.	*Ethernet*0/1	5000	-	-	*Non-Greedy*
2.	*Ethernet*0/2	16000	B	2	*Greedy*
3.	*Ethernet*0/3	2000	-	-	*Idle*
4.	*Ethernet*1/0	10000	-	-	*Non-Greedy*
5.	*Ethernet*1/1	18000	B	3	*Greedy*
6.	*Ethernet*1/2	20000	C	3	*Greedy*

Untuk antarmuka *idle*, alokasi *bandwidth* yang diberikan adalah sebesar parameter *bandwidth* minimal. Antarmuka *non-greedy* akan diberikan alokasi *bandwidth* sebesar nilai terkecil antara 2 kali besar trafik atau nilai batas trafik berlebih. Antarmuka *greedy* akan diberikan alokasi *bandwidth* sementara sebesar nilai parameter trafik maksimal sebelum diikutsertakan dalam penambahan alokasi selanjutnya. Dengan proses ini, alokasi *bandwidth* sementara tiap antarmuka dapat terlihat pada Tabel 4.

Tabel 5. Alokasi *Bandwidth* Sementara Percobaan Perhitungan.

Antarmuka	Trafik	Alokasi BW Sementara
*Ethernet*0/1	5000	10000
*Ethernet*0/2	16000	12000
*Ethernet*0/3	2000	4000
*Ethernet*1/0	10000	12000
*Ethernet*1/1	18000	12000
*Ethernet*1/2	20000	12000
	Jumlah:	62000

Antarmuka yang berkategori *greedy* dikelompokkan kembali berdasarkan tingkat prioritasnya. Dari 3 antarmuka yang berkategori *greedy*, terdapat 2 tingkat prioritas yang berbeda, yaitu tingkat prioritas B dan C.

Tabel 6. Antarmuka Tiap Tingkat Prioritas.

Antarmuka	Tingkat Prioritas	Value
*Ethernet*0/2	B	2
*Ethernet*1/1	B	2
*Ethernet*1/2	C	3

Terdapat 2 tingkat prioritas yang berbeda yaitu B dan C sehingga alokasi *bandwidth* tiap tingkat prioritas dapat dilakukan.

Tabel 7. Proporsi dan Alokasi *Bandwidth* Tambahan Tingkat Prioritas.

Tingkat Prioritas	Proporsi Tingkat Prioritas (x_m)	Alokasi Tingkat Prioritas (w_m)
B	0,4	15200
C	0,6	22800
Total	1	38000

Nilai proporsi tiap antarmuka pada tiap tingkat prioritas (r_{mn}) dapat dihitung. Nilai r_{mn} dihitung berdasarkan besar trafik berlebih tiap antarmuka C_{mn} dan trafik berlebih total tiap tingkat prioritas C_m.

Tabel 8. Proporsi Antarmuka Tiap Tingkat Prioritas.

Tingkat Prioritas	Antarmuka	Trafik Berlebih Antarmuka (C_{mn})	Proporsi Antarmuka (r_{mn})
B	*Ethernet*0/2	4000	0,4
	*Ethernet*1/1	6000	0,6
C	*Ethernet*1/2	8000	1

Setelah didapatkan nilai (r_{mn}), alokasi *bandwidth* tambahan tiap antarmuka (w_{mn}) dapat dilakukan.

Tabel 9. Alokasi Bandwidth Tambahan Tiap Antarmuka

Tingkat Prioritas	Alokasi Tingkat Prioritas (w_m)	Antarmuka	Proporsi Antarmuka (r_{mn})	Alokasi *Bandwidth* Tambahan (w_{mn})
B	15200	*Ethernet*0/2	0,4	6080
		*Ethernet*1/1	0,6	9120
C	22800	*Ethernet*1/2	1	22800

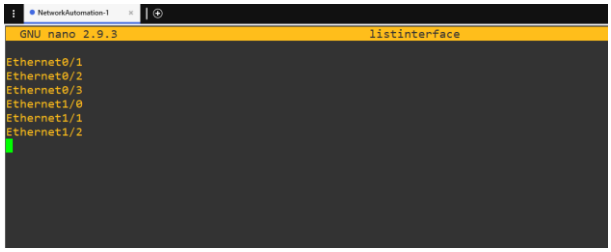
Nilai alokasi *bandwidth* tambahan (w_{mn}) telah didapatkan sehingga nilai akhir alokasi *bandwidth* tiap antarmuka bisa ditetapkan dengan menjumlahkan nilai w_{mn} kepada antarmuka *greedy*. Hasil akhir alokasi *bandwidth* bisa dilihat pada Tabel 10.

Tabel 10. Hasil Alokasi Bandwidth Percobaan Perhitungan.

No.	Antarmuka	Besar Trafik	*Alokasi Bandwidth*
1.	*Ethernet*0/1	5000	10000
2.	*Ethernet*0/2	16000	18080
3.	*Ethernet*0/3	2000	4000
4.	*Ethernet*1/0	10000	12000
5.	*Ethernet*1/1	18000	21120
6.	*Ethernet*1/2	20000	34800

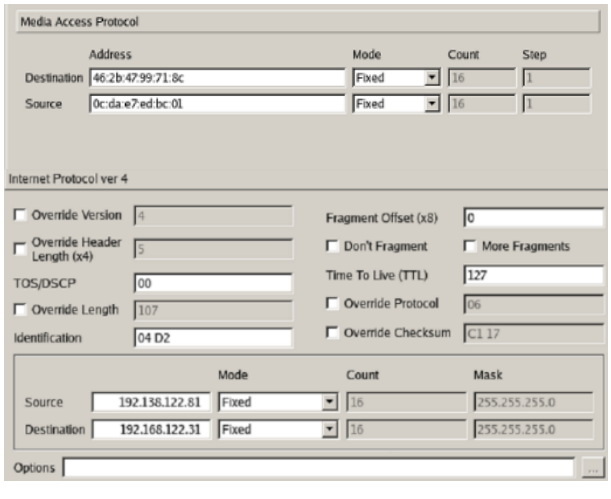
Program alokasi *bandwidth* dinamis dijalankan pada jaringan untuk mengatur besar *bandwidth* tiap antarmuka yang diinginkan setelah tersambung ke sebuah *switch* pada jaringan percobaan yang telah dibuat. Dengan menggunakan skenario topologi dan besar trafik yang sama seperti sebelumnya, program dijalankan pada *switch* untuk membuktikan kebenaran pengalokasian *bandwidth* telah sesuai dengan perhitungannya.

Hal pertama yang perlu dilakukan untuk menjalankan program adalah menyebutkan antarmuka-antarmuka yang ingin diikutsertakan dalam pengalokasian *bandwidth*. Nama antarmuka tersebut disebutkan dalam file '*listinterface*' seperti pada Gambar 6 dan disimpan di komponen *NetworkAutomation.*



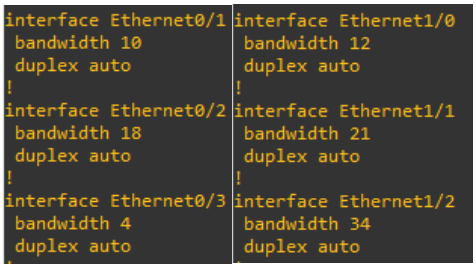
Gambar 6. Tampilan File '*listinterface*'.

Setelah antarmuka dicantumkan pada file 'listinterface', trafik data dibangkitkan dengan menyambungkan tiap *node* dengan komponen '*Ostinato*' sebagai pembangkit trafik seperti terlihat pada Gambar 7.



Gambar 7. Tampilan Pengalamatan Trafik pada Ostinato.

Program akan membaca informasi RXBS dan TXBS lalu akan mengolahnya sehingga didapatkan hasil akhir *bandwidth* yang dialokasikan. Hasil pengalokasian dapat terlihat seperti Gambar 8.

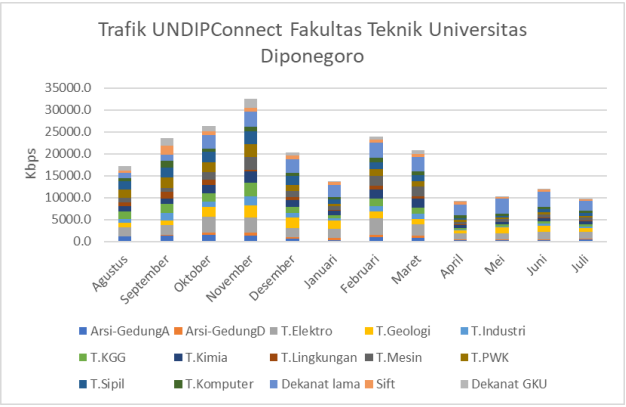


Gambar 8. Hasil Akhir Alokasi Bandwidth.

Berdasarkan Gambar 8, program telah bekerja sesuai dengan perhitungan. *Bandwidth* pada Gambar 8 memiliki satuan *kilobits* sehingga terdapat pembulatan ke bawah saat mengalokasikan *bandwidth* dari hasil perhitungan program.

3.2 *Bandwidth Fairness* Program pada Jaringan UNDIPConnect Fakultas Teknik Universitas Diponegoro

Pada bagian ini, program alokasi *bandwidth* dinamis disimulasikan dengan menggunakan topologi jaringan UNDIPConnect Fakultas Teknik Universitas Diponegoro untuk mengetahui nilai *Jain's Fairness Index.* Trafik dibangkitkan dalam jaringan untuk penentuan *bandwidth* tiap antarmuka yang dilakukan oleh program. Trafik yang digunakan merupakan data rekap trafik UNDIPConnect pada Fakultas Teknik dari bulan Agustus 2019 sampai bulan Juli 2020 yang didapatkan dari hasil pengambilan data di BAPSI Universitas Diponegoro. Besar trafik dapat dilihat pada Gambar 4.9 juga menggambarkan besar trafik pada Agustus 2019 hingga Juli 2020. Trafik tersebut merupakan rerata *sampling* tiap bulan selama setahun.



Gambar 9. Trafik UNDIPConnect Agustus 2019 – Juli 2020.

keadaan trafik pada jaringan UNDIPConnect Fakultas Teknik Universitas Diponegoro dari Agustus 2019 hingga Juli 2020. Trafik tertinggi terjadi pada bulan November

2019 di mana tercatat besar rata-rata trafik bulan itu adalah sekitar 32.581,5 Kbps atau sekitar 32 Mbps. Besar trafik terendah tercatat pada April 2020 di mana rata-rata besar trafik pada bulan itu adalah 9.160,5 Kbps atau sekitar 9,1 Mbps.

Sesuai hasil wawancara pengambilan data di BAPSI Universitas Diponegoro, *bandwidth* yang dimiliki oleh Universitas Diponegoro kampus Tembalang adalah sebesar 3 *Gigabit per second* (Gbps). Sementara, pengguna UNDIPConnect Fakultas Teknik diasumsikan sebanyak jumlah mahasiswa dan pegawai Fakultas Teknik Universitas Diponegoro. Alokasi *bandwidth* pada Fakultas Teknik Universitas Diponegoro persentase pengguna di Fakultas Teknik Universitas Diponegoro dari seluruh pengguna di Universitas Diponegoro. Berdasarkan [12] dan [13] didapatkan data seperti terlihat pada Tabel 11.

Tabel 11. Jumlah Mahasiswa dan Pegawai[12][13]**.**

	Jumlah di Fakultas Teknik	Total di Universitas Diponegoro
Mahasiswa	6211	55834
Pegawai	536	3682
Total	6747	59516

Dari data tersebut, alokasi *bandwidth* UNDIPConnect Fakultas Teknik Universitas Diponegoro berdasarkan presentase pengguna dapat dihitung sebagai berikut:

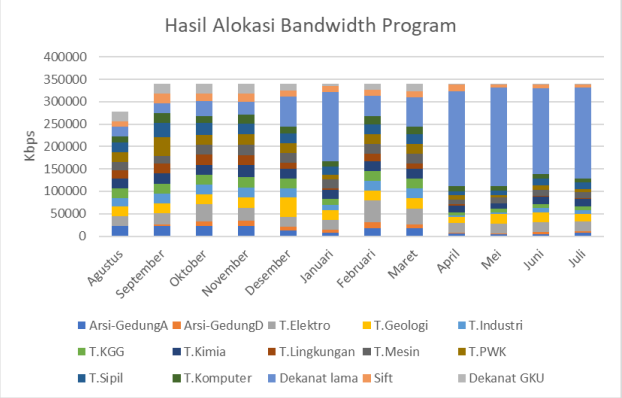
$$\begin{aligned} Alokasi\ BW\ FT &= Persentase\ Pengguna\ FT \times Bandwidth\ Total \\ &= \left(\frac{6747}{59516} \times 100\%\right) \times 3\ Gbps \\ &= 11{,}33\% \times 3\ Gbps \\ &\approx 0{,}34\ Gbps \end{aligned}$$

Dengan didapatkannya besar alokasi *bandwidth* Fakultas Teknik Universitas Diponegoro, parameter alokasi banwdith pada program dapat ditetapkan sebagai berikut:

- *Bandwidth* Total : 340000000 bps
- *Bandwidth* Minimal : 1100000 bps
- Trafik Maksimal : 22000000 bps

Besar parameter trafik maksimal pada percobaan ini adalah hasil dari pembagian besar *bandwidth* total dengan banyaknya jumlah node. Sementara, besar parameter bandwidth minimal adalah 5% dari besar trafik maksimal. Parameter-parameter tersebut diatur dalam program sebelum program dijalankan.

Trafik selama setahun kemudian dibangkitkan dalam jaringan. Program pun dijalankan sehingga pengalokasian *bandwidth* dapat dilakukan sesuai dengan keadaan trafik pada jaringan. Hasil alokasi *bandwidth* program dapat terlihat pada Gambar 10 grafik besar *bandwidth* tiap node pada tiap bulan dari Agustus 2019 hingga Juli 2020.



Gambar 10. *Alokasi Bandwidth* Tiap Node pada Agustus 2019 – Juli 2020.

Besar *bandwidth* tiap node berubah-ubah. Total *bandwidth* yang dialokasikan sebesar 340.000 Kbps atau 340 Mbps pada September 2019 hingga Juli 2020. Perbedaan terjadi pada Agustus 2019 di mana bandwidth yang dialokasikan hanya sebesar 277.760 Kbps atau sekitar 277,7 Mbps. Hal ini terjadi karena tidak ada *node* yang memiliki trafik melebihi parameter batas trafik berlebih sehingga sisa *bandwidth* tidak dialokasikan ke *node* manapun.

Berdasarkan besar alokasi *bandwidth* tiap node yang telah didapatkan, nilai dari *Jain's Fairness Index* [14][15] dapat dihitung dengan menggunakan persamaan (4).

$$I_{JF} = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x^2{}_i} \qquad (4)$$

Keterangan:
I_{JF} : *Jain's Fairness Index* ($0 \leq I_{JF} \leq 1$)
x_i : *Bandwidth* dari komponen *i*
n : Jumlah komponen

Tabel 12. Hasil Perhitungan *Jain's Fairness Index*.

Bulan	Jain's Fairness Index (I_{JF})
Agustus'19	0.913789226
September'19	0.907119257
Oktober'19	0.930220456
November'19	0.976316814
Desember'19	0.719485659
Januari'20	0.288025566
Februari'20	0.837302742
Maret'20	0.750904903
April'20	0.165679432
Mei'20	0.153599299
Juni'20	0.200595125
Juli'20	0.178967429
Rata-Rata	**0.585167159**

Berdasarkan Tabel 12, terlihat nilai *Jain's Fairness Index* alokasi *bandwidth* oleh program dari bulan Agustus 2019 sampai Juli 2020. Nilai I_{JF} tertinggi terdapat pada bulan

November 2019 yaitu sebesar 0,976316814. Sementara, nilai I_{JF} terendah terdapat pada bulan Mei 2020 yaitu sebesar 0,153599299. Nilai rata-rata I_{JF} hasil alokasi program yaitu sebesar 0,585167159. Perbedaan nilai I_{JF} disebabkan terdapatnya variasi antara pengalokasian *bandwidth* tiap node. Pada bulan November 2020, hasil alokasi dari *bandwidth* tiap node memiliki variasi yang kecil dikarenakan keadaan trafik pada bulan tersebut relatif tinggi dibandingkan bulan-bulan lainnya. Hal ini menyebabkan perbedaan besar alokasi *bandwidth* pada tiap node lebih kecil. Sedangkan pada bulan Mei 2020, terdapat variasi nilai alokasi *bandwidth* yang besar pada *node* Dekanat Lama dikarenakan seluruh sisa *bandwidth* dialokasikan ke node ini. Hal ini menyebabkan alokasi *bandwidth* pada node Dekanat Lama sangat besar dibandingkan *node-node* lainnya sehingga nilai I_{JF} menjadi kecil.

4. Kesimpulan

Berdasarkan penelitian berjudul "Implementasi Algoritma *Greedy* Sebagai Penentuan Alokasi *Bandwidth* Secara Dinamis", dapat disimpulkan bahwa, Pembuatan program pengalokasian *bandwidth* secara dinamis dengan menggunakan algoritma *Greedy* dapat dilakukan dengan menggunakan bahasa pemrogram *Python* di mana program dijalankan pada simulasi dengan membaca informasi trafik yang ada pada jaringan, program berhasil memberikan alokasi *bandwidth* ke setiap *link* dengan nilai yang berbeda tergantung dengan pengklasifikasian setiap *link* berdasarkan besar trafik pada *link* tersebut, penambahan *bandwidth* pada link berkategori *greedy* telah sesuai dengan perhitungan algoritma *Greedy* yang ada dengan membulatkan ke bawah hasil perhitungan, serta dengan menggunakan topologi jaringan UNDIPConnect Fakultas Teknik Universitas Diponegoro dengan membangkitkan besar trafik berdasarkan rekap data trafik bulan Agustus 2019 sampai Juli 2020, program memiliki rata-rata *Jain's Fairness Index* sebesar 0.585167159 dengan nilai tertinggi sebesar 0.976316814 pada November 2019 serta nilai terkecil sebesar 0.153599299 pada Mei 2020.

Referensi

[1] APJII, "Penetrasi & Profil Perilaku Pengguna Internet Indonesia Tahun 2018," *Apjii*, p. 51, 2019, [Online]. Available: www.apjii.or.id.

[2] S. A. Curtis, "The classification of greedy algorithms," *Sci. Comput. Program.*, vol. 49, no. 1–3, pp. 125–157, 2003, doi: 10.1016/j.scico.2003.09.001.

[3] R. N. Howell, "Algorithms: A Top-Down Approach," *Dept. Comput. Inf. Sci. Kansas State Univ.*, vol. 9, pp. 374–391, 2009.

[4] V. Chandru and M. R. Rao, *Combinatorial optimization*. 2004.

[5] S. Radhakrishnan, D. Kolippakkam, and V. S. Mathura, *Introduction to algorithms*. 2007.

[6] H. Li, Q. Xia, and Y. Wang, "Research and Improvement of Kruskal Algorithm," *J. Comput. Commun.*, vol. 05, no. 12, pp. 63–69, 2017, doi: 10.4236/jcc.2017.512007.

[7] B. A. Forouzan, *DATA COMMUNICATIONS AND NETWORKING, FOURTH EDITION*, 4th ed., vol. 2018. McGraw-Hill, 2018.

[8] A. R. Muttaqi, S. Wahjuni, and S. N. Neyman, "Manajemen Alokasi Bandwidth Layanan Internet Menggunakan Fractional Knapsack Problem," *J. Teknol. dan Sist. Komput.*, vol. 7, no. 2, pp. 71–76, 2019, doi: 10.14710/jtsiskom.7.2.2019.71-76.

[9] C. Cerrone, R. Cerulli, and B. Golden, "Carousel greedy: A generalized greedy algorithm with applications in optimization," *Comput. Oper. Res.*, vol. 85, pp. 97–112, 2017, doi: 10.1016/j.cor.2017.03.016.

[10] M. Wairisal and N. Surantha, "Design and Evaluation of Efficient Bandwidth Management for a Corporate Network," *Proc. 2018 Int. Conf. Inf. Manag. Technol. ICIMTech 2018*, no. September, pp. 98–102, 2018, doi: 10.1109/ICIMTech.2018.8528162.

[11] I. A. Bustoni, A. E. Permanasari, and I. Hidayah, "Otomasi Manajemen Bandwidth Jaringan UGM-Hotspot menggunakan Metode SARIMA dan Fuzzy Logic Tsukamoto," *Jur. Tek. Elektro Dan Teknol. Inf. FT UGM*, 2015.

[12] "E-Duk - Elektronik Daftar Urut Kepangkatan Undip." https://e-duk.apps.undip.ac.id/index.php/statistik/stkepegawaian.html.

[13] PDDikti, "PDDikti - Pangkalan Data Pendidikan Tinggi," *PDDikti*. 2020, [Online]. Available: https://pddikti.kemdikbud.go.id/pt.

[14] C. Touati, E. Altman, and J. Galtier, "On fairness in Bandwidth Allocation," *Sophia*, no. September, 2001.

[15] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems." 1998, [Online]. Available: http://arxiv.org/abs/cs/9809099.