

BAB IV

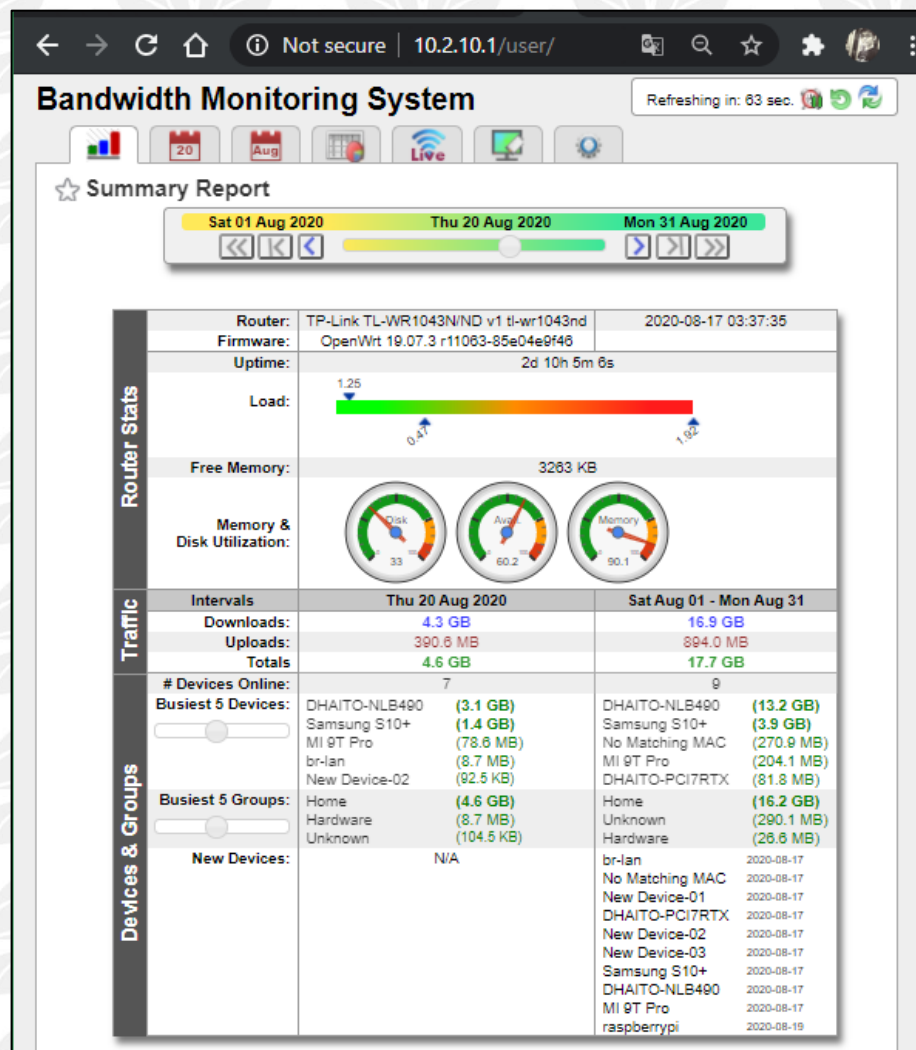
HASIL DAN PEMBAHASAN

4.1 Hasil Perancangan *Bandwidth Monitoring System*

Pada bab ini penulis akan membahas mengenai hasil perancangan *Bandwidth Monitoring System*. Hasil perancangan akan dibahas pada setiap tab fitur dari *Bandwidth Monitoring System*. Berikut ini hasil perancangan yang telah dibuat:

4.1.1 *Summary Report*

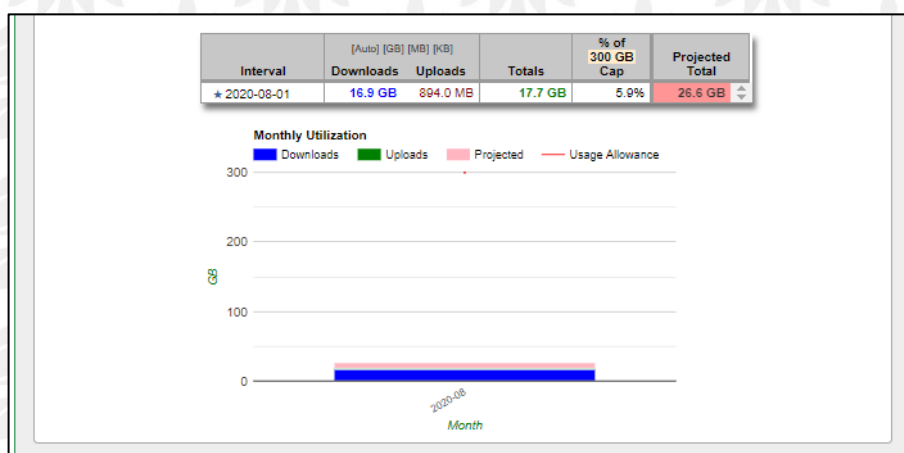
Halaman *Summary Report* digunakan untuk memberikan informasi ringkas keseluruhan informasi *bandwidth*. Dapat dilihat pada Gambar 4.1 terdapat sebuah *slider* dan *button* untuk mengganti informasi mengenai *bandwidth* berdasarkan hari yang kita inginkan, terlihat bahwa informasi yang ditampilkan adalah tanggal 20 Agustus 2020. Terdapat tabel yang berisi tiga informasi pokok, antara lain *Router Stats*, *Traffic*, dan *Groups & Devices*. *Router Stats* memberikan informasi mengenai perangkat dan *firmware* yang digunakan yaitu Raspberry Pi 3 B+ dan OpenWRT v19.07.3, kemudian uptime selama 2 hari 10 jam dengan server *load* sedang, sisa RAM sebesar 3263KB dan *disk utility*. *Traffic* memberikan informasi *download* dan *upload* pada tanggal 20 agustus 2020 yaitu hari berjalan sekarang dengan detail *download* sebesar 4.3GB, *upload* 390Mb, dan totalnya 4.6GB. Sedangkan pada bulan berjalan sekarang dengan detail *download* sebesar 16.9GB, *upload* 894MB, dan totalnya 17.7GB. *Devices & Groups* memberikan informasi jumlah *device* yang aktif terhubung atau pernah terhubung dan berapa besar data yang telah digunakan, dapat dilihat bahwa pada tanggal 20 Agustus terdapat 7 *device* sedang aktif terhubung ke jaringan, beberapa contoh *device* yang terhubung adalah DHAITO-NLB490 pemakaian 3.1GB, Samsung S10+ pemakaian 1.4GB. Sedangkan pada bulan berjalan Agustus pernah ada 9 *device* terhubung ke jaringan, salah satunya DHAITO-NLB490 pemakaian 13.2GBGB, Samsung S10+ pemakaian 3.9GB. Pada *list device* terdapat *slider* yang dapat kita atur untuk banyaknya *list device* yang ingin kita tampilkan. *New Device* berguna untuk menampilkan *device* yang baru terhubung di jaringan.



Gambar 4.1 Halaman *Summary - Bandwidth Monitoring System* (1)

Pada Gambar 4.2 masih dalam halaman *Summary Report*, dapat dilihat ada sebuah tabel dan grafik. Informasi tabel dalam setiap baris digunakan untuk memberikan informasi interval pada pemakaian *bandwidth* setiap bulan. Dilihat pada pemakaian *bandwidth* sejak tanggal 1 Agustus yaitu *download* sebesar 16.9GB dan *upload* sebesar 894GB dan totalnya 17.7GB. Kemudian ada informasi bahwa 5.9% of 300GB FUP, maksudnya adalah total pemakaian 5.9% dari total 300GB FUP. 300GB adalah batas FUP dari Telkom Indihome sebagai ISP, karena di sini penulis menggunakan Telkom sebagai jasa Internet, dan nilai FUP ini bisa kita sesuaikan pada halaman konfigurasi. Pada *projected total* 26,6GB adalah prediksi dalam pemakaian sebulan, nilai ini didapatkan dari rata-rata penggunaan *bandwidth* oleh setiap *device* dalam bulan berjalan,

ini hanya sebagai info saja dan tidak bisa digunakan sebagai acuan pasti, karena pemakaian *bandwidth* mungkin bisa berubah drastis pada hari berikutnya. Sedangkan grafik memberikan informasi seperti halnya pada tabel, dapat dilihat pemakaian *bandwidth* pada bulan Agustus masih sangat sedikit, *download* (berwarna biru), *upload* (berwarna hijau), *projected total* (berwarna merah muda) tidak sampai menyentuh 50GB. Sedangkan garis berwarna merah berada pada 300GB yang merupakan FUP dari penggunaan internet.



Gambar 4.2 Halaman *Summary* - *Bandwidth Monitoring System* (2)

4.1.1.1 Beberapa Script Pada Halaman *Summary Report*

Sebagian *Script* berikut digunakan untuk tabel *Summary Report* bagian *Router Stats* sedangkan untuk device dan traffic akan dijelaskan pada halaman *Hourly*, *Monthly Usage Report* dan *Groups And Devices*.

```

if [ -f "/etc/openwrt_release" ] ; then
    installedfirmware=$(cat /etc/openwrt_release | grep -i
'DISTRIB_DESCRIPTION' | cut -d'"' -f2)
elif [ "$_firmware" -eq "2" ] ; then
    routermodel=$(nvram get model)
    installedversion=$(nvram get buildno)_$(nvram get extendno)
    installedtype='merlin'
elif [ "$_has_nvram" -eq "1" ] ; then
    installedfirmware=$(uname -o)
    routermodel=$(nvram get DD_BOARD)
    installedversion=$(nvram get os_version)
    installedtype=$(nvram get dist_type)
fi
if [ -d /tmp/sysinfo/ ] ; then
    local model=$(cat /tmp/sysinfo/model)
    local board=$(cat /tmp/sysinfo/board_name)
    routermodel="$model $board"
fi

```

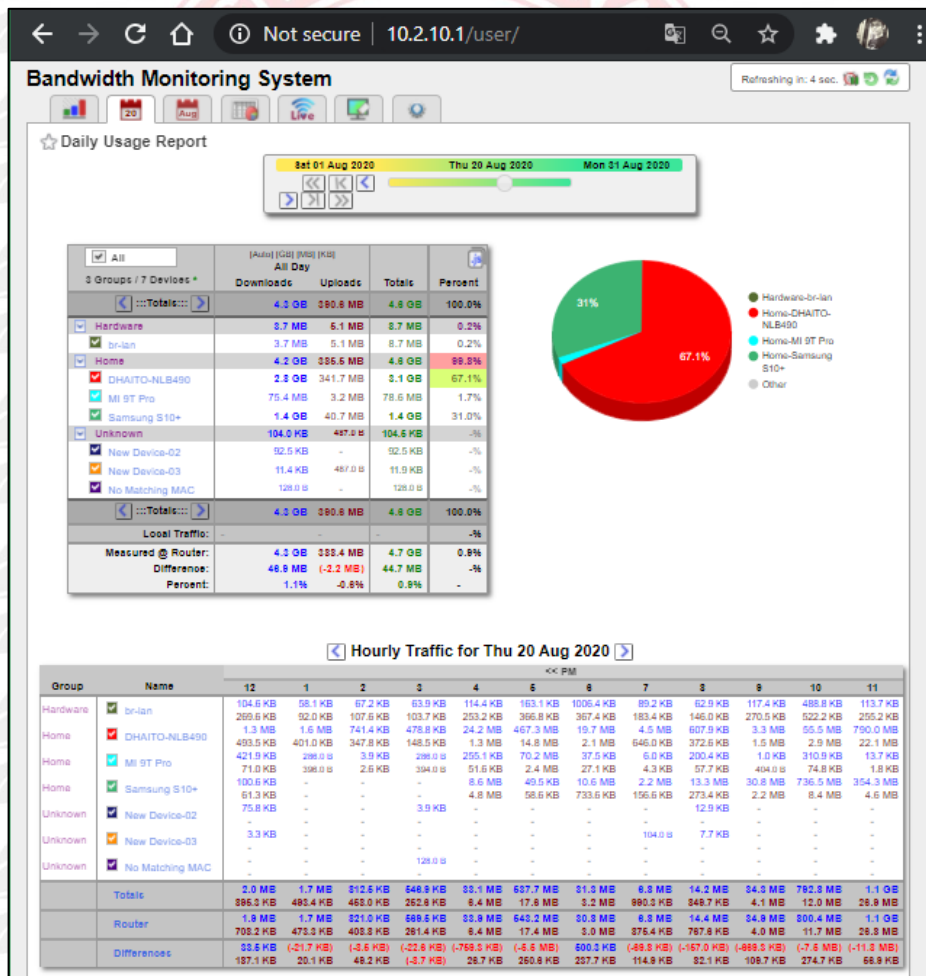
Script di atas digunakan untuk menampilkan *router* model dengan memasukkan perintah `nvrnm` kemudian mengambil `buildno` dan menyimpannya pada `routermodel`. Begitu juga mendapatkan info versi OpenWRT yaitu dengan memasukkan perintah `cat /etc/openwrt_release` kemudian mengambil `DISTRIB_DESCRIPTION` dan menyimpannya pada `installedfirmware`.

```
updateServerStats ()
{
    local cTime=$(date +%T)
    if [ -z "$sl_max" ] || [ "$sl_max" \< "$load5" ] ; then
        sl_max=$load5
        sl_max_ts="$cTime"
    fi
    if [ -z "$sl_min" ] || [ "$load5" \< "$sl_min" ] ; then
        sl_min=$load5
        sl_min_ts="$cTime"
    fi
}
getMI ()
{
    local result=$(echo "$meminfo" | grep -i "^$1:" | cut -d' ' -f2)
    [ -z "$result" ] && result=0
    echo "$result"
    $send2log "getMI: $1=$result" 0
}
etHourlyHeader() {
    $send2log "getHourlyHeader: $1 $2" 0
    meminfo=$(cat /proc/meminfo | tr -s ' ')
    local freeMem=$(getMI "MemFree")
    local bufferMem=$(getMI "Buffers")
    local cacheMem=$(getMI "Cached")
    local availMem=$(( $freeMem+$bufferMem+$cacheMem ))
    local disk_utilization=$(df "${d_baseDir}" | tail -n 1 | tr -s ' ' | cut -d' ' -f5)
}
```

Script diatas digunakan untuk mendapatkan info server load dan disk utilization. Function `updateServerStats` digunakan untuk mendapatkan server load pada `sl_min` dan `sl_max`. Function `getMI` untuk memperoleh kebutuhan data dengan menjalankan perintah `$meminfo` kemudian hasil yang diperoleh proses pada function `etHourlyHeader()` untuk mendapatkan data server load dan disk utilization.

4.1.2 Daily Usage Report

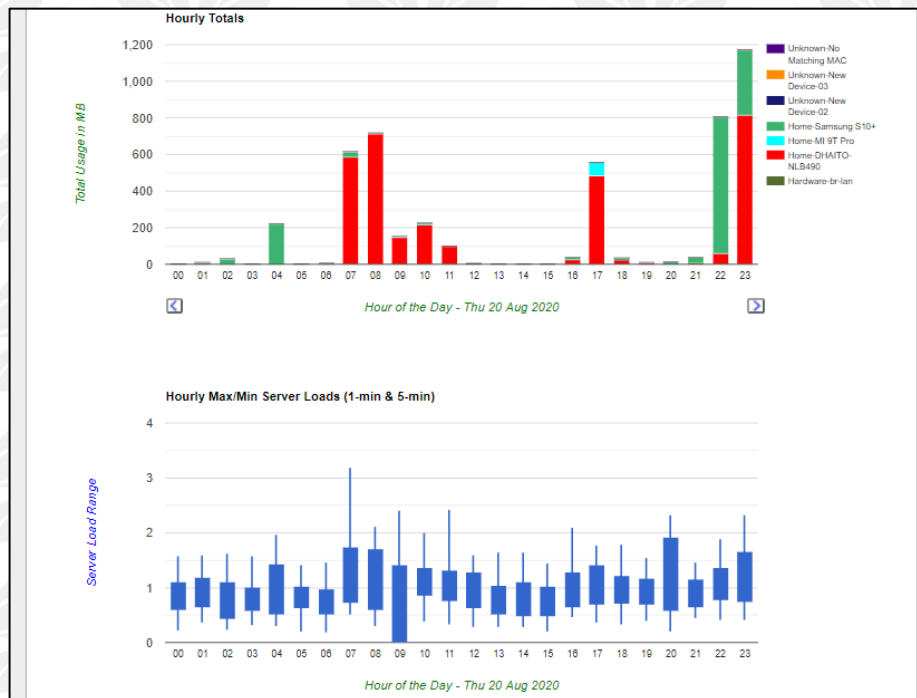
Halaman *Daily Usage Report* berguna untuk mendapatkan informasi ringkas keseluruhan informasi *bandwidth*. Pada Gambar 4.3 terdapat sebuah *slider* dan *button* untuk mengganti informasi mengenai *bandwidth* berdasarkan hari yang kita inginkan seperti halnya dengan *Summary Report*, Terlihat bahwa informasi yang ditampilkan adalah tanggal 20 Agustus 2020. Dapat dilihat ada sebuah tabel mengenai rincian penggunaan *bandwidth* harian pada 20 Agustus. Pada tanggal tersebut ada 3 *groups* dan 7 *device* yang pernah aktif menggunakan internet. Misalnya, *group Home* pada hari tersebut menggunakan *bandwidth* dengan rincian *download* 4.2GB, *upload* 385MB, total 4.6GB, dan menggunakan 99,3% dari total penggunaan harian.



Gambar 4.3 Halaman *Daily Usage* - *Bandwidth Monitoring System* (1)

Pada Gambar 4.3 Lebih spesifiknya pada tabel, pada *group home* terdapat 3 *device* dengan rincian penggunaan *device* DHAITO-NLB490 yaitu *download* 2.8GB, *upload* 341,7MB, total 3.1GB, dan menggunakan 67,1% dari total penggunaan harian, *device* MI9T Pro yaitu *download* 75.4MB, *upload* 3.2MB, total 78.6GMB, dan menggunakan 1,7% dari total penggunaan harian, dan *device* Samsung S10+ yaitu *download* 1.4GB, *upload* 40.7MB, total 1.4GB, dan menggunakan 31% dari total penggunaan harian. Dapat dilihat total penggunaan *bandwidth* pada *Bandwidth Monitoring System* pada tanggal 20 Agustus adalah *download* 4.3GB, *upload* 390.8MB, Total 4.8GB, dengan 100% penggunaan harian. Sedangkan total pengukuran *bandwidth* oleh *router* sebesar *download* 4.3GB, *upload* 388.4MB, Total 4.8GB. Terlihat bahwa selisih pengukuran pada *Bandwidth Monitoring System* dan *router* adalah *download* 4.8MB, *upload* 2.2MB, total 44.7MB, selisih 0.9%, dan selisih tersebut masih tergolong wajar dan *Bandwidth Monitoring System* masih bisa dikatakan berjalan dengan baik.

Pada Gambar 4.3 terdapat diagram *pie* yang menunjukkan persentase dari penggunaan *bandwidth* harian dari setiap *device*. Sebagai contoh, Terlihat bahwa *device* DHAITO-HOME (warna merah) adalah pemakai *bandwidth* terbesar yaitu 67.1%, Samsung S10+ (warna hijau) sebesar 31%, dan MI 9T Pro sebesar 1.7%. Sedangkan pada tabel *Hourly Traffic* menunjukkan pemakaian *bandwidth* oleh setiap *device* pada jam 12 pagi sampai jam 12 malam, dan detail penggunaan setiap jam dapat diatur berdasarkan tanggal yang dipilih. Sebagai contoh, pada tanggal 20 Agustus 2020 pada jam 11 malam (PM) *device* DHAITO-NLB490 menggunakan *bandwidth* dengan rincian *download* 790MB dan *upload* 22.1MB. *Device* Samsung S10+ menggunakan *bandwidth* dengan rincian *download* 354.3MB dan *upload* 4.6MB. Lalu total pada perhitungan *bandwidth* oleh *Bandwidth Monitoring System* pada jam 11 malam adalah sebesar *download* 1.1GB dan *upload* 28.9MB. Perhitungan *bandwidth* oleh *router* pada jam 11 malam adalah sebesar *download* 1.1GB dan *upload* 28.9MB. Selisih perhitungannya yaitu *download* 11.8MB dan *upload* 68.9KB, selisih tersebut masih bisa dikatakan wajar dan jaringan tidak ada kendala.



Gambar 4.4 Halaman *Daily Usage - Bandwidth Monitoring System* (2)

Pada gambar 4.4 masih pada halaman *Daily Usage* terdapat dua grafik, yaitu *Hourly Totals* dan *Hourly Min Max Server*. Grafik *Hourly Total* menunjukkan grafik pemakaian *bandwidth* oleh setiap *device* pada setiap jam dari mulai jam 00 pagi hingga jam 12 malam, terlihat pada jam 11 malam adalah waktu paling besar menggunakan *bandwidth* dan *device* yang dominan adalah DHAITO-NLB490 (warna merah) dan Samsung S10+ (warna hijau). *Hourly Min Max Server Load* menunjukkan grafik server *load* oleh *router* pada setiap jam dari mulai jam 00 pagi hingga jam 12 malam dengan rentang 1 menit dan 5 menit, pada grafik terlihat server *load* tertinggi ada pada jam 7 pagi dengan detail pada 1 menit mempunyai beban sekitar 0.5 sampai 3.3, sedangkan pada 5 menit mempunyai beban sekitar 0.70 sampai 1.70.

4.1.2.1 Beberapa Script Pada Halaman *Daily Usage*

Berikut adalah Sebagian script untuk memperoleh penggunaan *bandwidth* secara harian. Function `tallyHourlyData_1` digunakan mendapatkan data *Hourly Usage* yang ditampung pada `hrlyData`.

```
tallyHourlyData_1() {
  macTotals_0() {
    for line in $(echo "$macEntries")
```

```

do
    $send2log "line: $line" -1
    do_tot=$(digitAdd $do_tot $(getCV "$line" 'down'))
    up_tot=$(digitAdd $up_tot $(getCV "$line" 'up') )
done
newline=$(setNewLine_0)
hr_results="$hr_results
$newline"
}
}

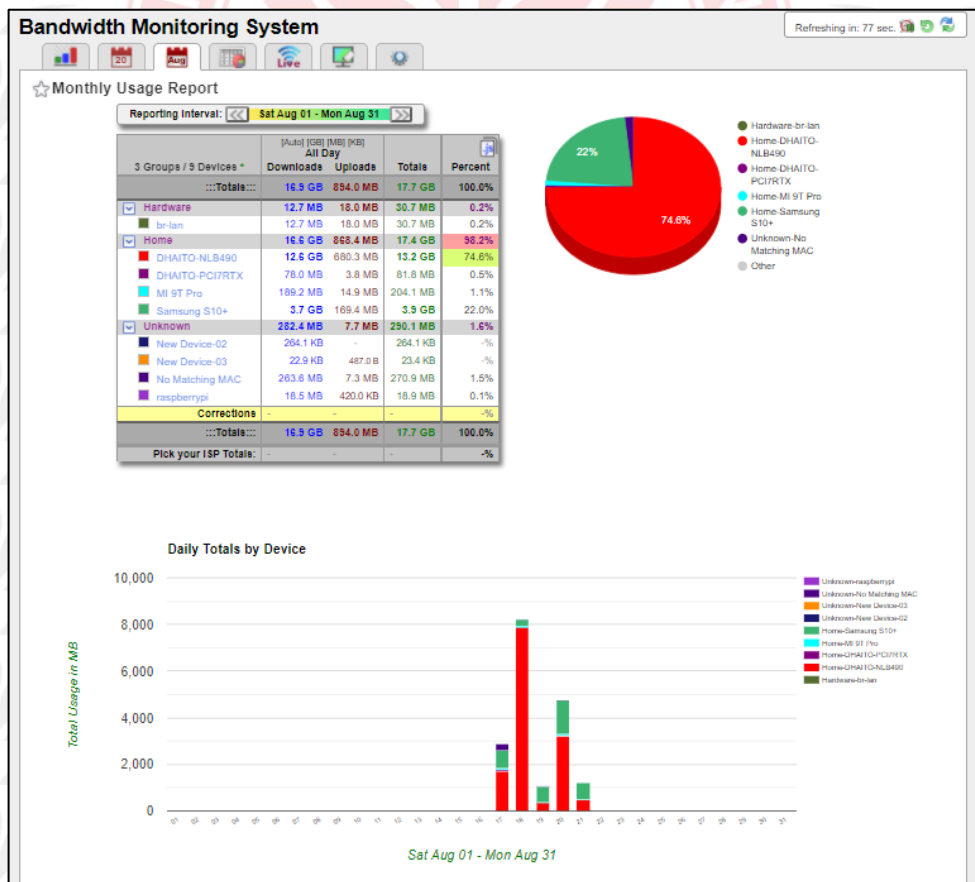
local hrlyData=$(echo "$hrlyData" | grep "^hu")
$send2log "hrlyData: $hrlyData" -1
local macList=$(echo "$hrlyData" | grep -o
"\\"mac\":"\"[^\\\"]\\{1,\\}\"" | tr 'A-Z' 'a-z' | sort -k1 | uniq -c
| cut -d'"' -f4)
$send2log "macList: $macList" -1
IFS=$'\n'
local gt_down=0
local gt_up=0
local gt_ul_down=0
local gt_ul_up=0
for mac in $(echo "$macList")
dgt=$(eval $dgt_fn)
hr_results="$hr_results
$dgt"}

```



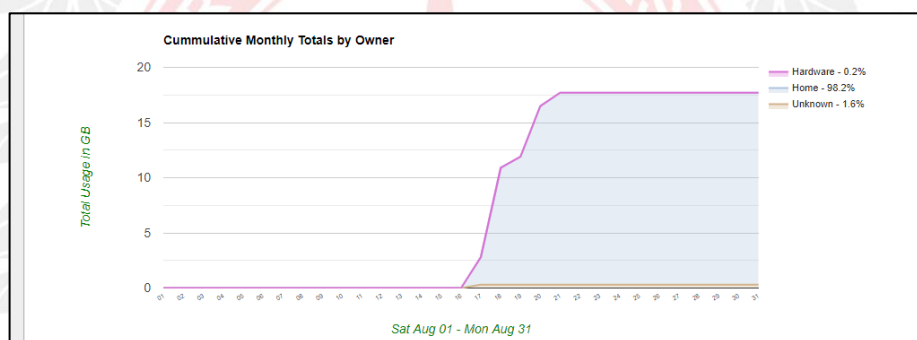
4.1.3 Monthly Usage Report

Halaman *Monthly Usage Report* berguna untuk mendapatkan informasi ringkas keseluruhan informasi *bandwidth* dalam bulan. Dapat dilihat pada Gambar 4.5 terdapat sebuah *button* untuk mengganti informasi mengenai *bandwidth* berdasarkan bulan yang kita inginkan, Terlihat bahwa informasi yang ditampilkan adalah bulan Agustus dari tanggal 1 sampai 31. Dapat dilihat ada sebuah tabel mengenai rincian penggunaan *bandwidth* bulanan pada bulan Agustus. Pada tanggal tersebut ada 3 *groups* dan 9 *device* yang pernah aktif menggunakan internet. Misalnya, *group hardware* pada bulan tersebut menggunakan *bandwidth* dengan rincian *download* 12.7MB, *upload* 18MB, total 30.7MB, dan menggunakan 0.2% dari total penggunaan bulanan, dan *group Home* pada bulan tersebut menggunakan *bandwidth* dengan rincian *download* 12.6GB, *upload* 680.3MB, total 13.2GB, dan menggunakan 74.6% dari total penggunaan bulanan.



Gambar 4.5 Halaman *Monthly Usage* - *Bandwidth Monitoring System* (1)

Pada Gambar 4.5 Lebih spesifiknya pada tabel, pada *group home* terdapat 4 *device* dengan rincian penggunaan *device* DHAITO-NLB490 yaitu *download* 12.6GB, *upload* 680,3MB, total 13.2GB, dan menggunakan 74.6% dari total penggunaan harian, *device* MI9T Pro yaitu *download* 189.2MB, *upload* 14.9MB, total 204.1MB, dan menggunakan 1,1% dari total penggunaan harian, *device* Samsung S10+ yaitu *download* 3.7GB, *upload* 169.4MB, total 3.94GB, dan menggunakan 22% dari total penggunaan harian, dan *device* DHAITO-PCI7RTX yaitu *download* 78MB, *upload* 3.8MB, total 81.8MB, dan menggunakan 0.5% dari total penggunaan harian. Dapat dilihat total penggunaan *bandwidth* pada *system Bandwidth Monitoring System* pada tanggal 20 Agustus adalah *download* 16.9GB, *upload* 834MB, Total 17.7GB, dengan 100% penggunaan harian.



Gambar 4.6 Halaman *Monthly Usage - Bandwidth Monitoring System* (2)

Pada gambar 4.6 masih pada halaman *Monthly Usage* terdapat grafik yang menunjukkan penggunaan *bandwidth* dari setiap *group* dalam sebulan yaitu bulan Agustus tanggal 1 sampai 31. Terlihat pada grafik penggunaan terbanyak ada pada *group Home* yang ditunjukkan dengan warna biru muda, *group* ini mendominasi penggunaan *bandwidth* dalam sebulan yaitu sekitar 17GB. Sedangkan pada *group* lain yaitu *hardware* dan *unknown* masing-masing tidak melebihi sampai 300MB.

4.1.3.1 Beberapa Script Pada Halaman *Monthly Usage*

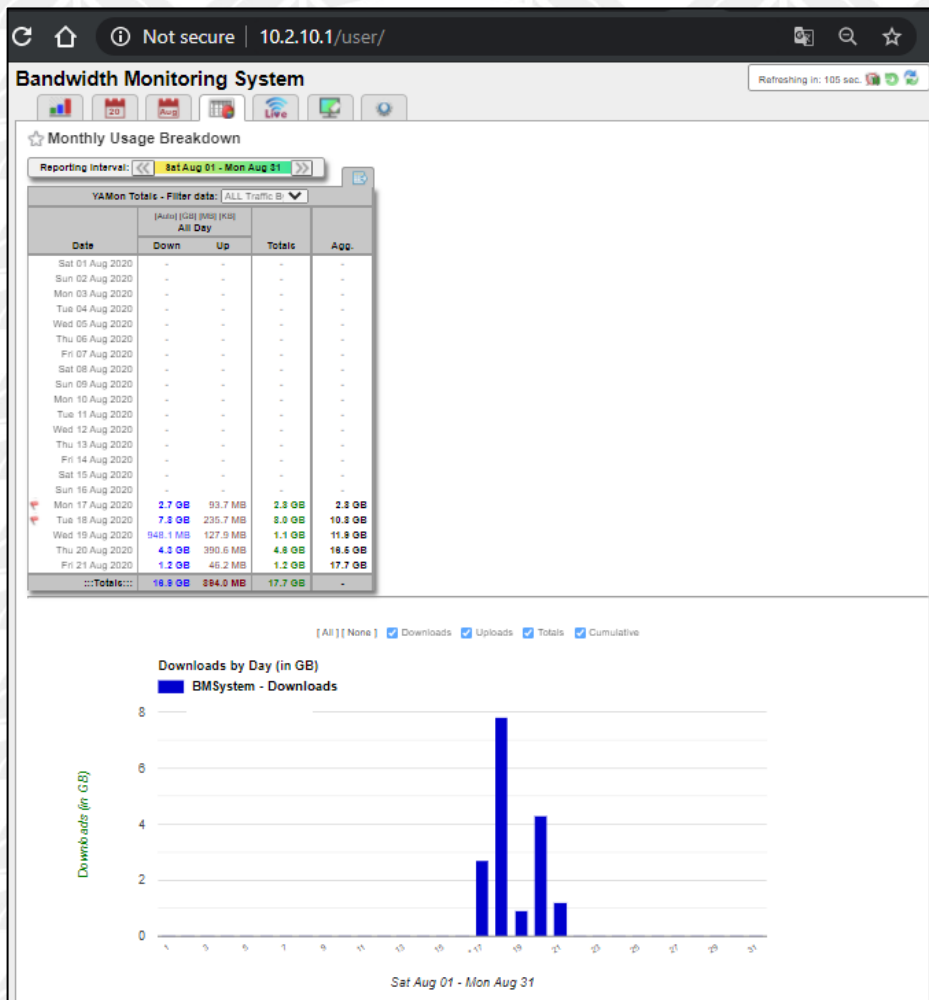
Berikut adalah potongan *script* untuk memperoleh data penggunaan *bandwidth* secara bulanan. *Fuction* yang digunakan untuk

adalah `calcMonthlyTotal()` dan `getTotals_1()` berperan dalam menghitung data dari harian menjadi bulanan.

```
calcMonthlyTotal()
{
  getTotals_1()
  {
    IFS=$'\n'
    for line in $(echo "$dgt1")
    do
      mt_down=$(digitAdd $mt_down $(getCV "$line" 'down'))
      mt_up=$(digitAdd $mt_up $(getCV "$line" 'up'))
      mt_ul_down=$(digitAdd $mt_ul_down $(getCV "$line"
'ul_do'))
      mt_ul_up=$(digitAdd $mt_ul_up $(getCV "$line" 'ul_up'))
    done
    unset IFS
  }
  if [ -z "$(echo $vars | grep 'monthly_total')" ]; then
    local rotf=$(echo "$mud" | grep -v '^var')
    local mt="var monthly_total_down=\"\""
    var monthly_total_up=""
    if [ "$_unlimited_usage" -eq "1" ]; then
      mt="$mt
fi
echo "$vars
$mt
$rotf" > $_macUsageDB
fi
updateInDB "monthly_total_down" "$mt_down"
updateInDB "monthly_total_up" "$mt_up"
updateInDB "monthly_updated" "$(date +%Y-%m-%d %H:%M:%S)"
local mt_tot=$(digitAdd $mt_up $mt_down)
local mt_tot_gb=$(toGB $mt_tot 0)
}
```

4.1.4 Monthly Usage Breakdown

Halaman *Monthly Usage Breakdown* berguna untuk mendapatkan rincian informasi keseluruhan total penggunaan *bandwidth* dalam bulan dan di list dalam tiap harian. Dapat dilihat pada Gambar 4.5 terdapat sebuah *button* untuk mengganti informasi mengenai *bandwidth* berdasarkan bulan yang kita inginkan dan sebuah *combobox* untuk memilih *device* atau *group* sebagai filter untuk dilihat rincian penggunaan *bandwidth* dalam sebulan. Terlihat bahwa informasi yang ditampilkan adalah bulan Agustus dari tanggal 1 sampai tanggal berjalan yaitu tanggal 21.



Gambar 4.7 Halaman *Monthly Breakdown - Bandwidth Monitoring System* (1)

Terlihat pada Gambar 4.7 di bagian tabel bahwa dari tanggal 1 Agustus sampai 16 Agustus tidak ada data atau informasi penggunaan *bandwidth*. Hal ini dikarenakan *Bandwidth Monitoring System* belum dijalankan, dengan kata

lain penulis belum mengaktifkannya dan baru tanggal 17 Agustus *Bandwidth Monitoring Sistem* digunakan. Karena pada filter *combobox* penulis memilih *all traffic by day* maka informasi yang ditampilkan adalah total keseluruhan penggunaan *bandwidth* oleh *device* dalam sehari dari tanggal 16 Agustus sampai hari berjalan yaitu tanggal 22. Total dari penggunaan *bandwidth* dalam bulan berjalan Agustus yaitu *download* 16.9GB, *upload* 884MB, dan total 17.7GB.

Pada grafik yang ditampilkan pada Gambar 4.7 merupakan grafik *Download by Day* untuk memberikan informasi mengenai penggunaan *bandwidth* pada sisi *download*. Informasi menampilkan total *download* dalam sebulan dari tanggal 1 sampai 31 Agustus. Karena pada filter *combobox* dipilih *all traffic by day*, maka yang ditampilkan adalah total penggunaan semua *device*, tidak dirincikan pada *device* tertentu. Dapat dilihat penggunaan tertinggi ada pada tanggal 18 Agustus, yaitu 7.8GB.



Gambar 4.8 Halaman *Monthly Breakdown - Bandwidth Monitoring System (2)*

Pada grafik yang ditampilkan pada Gambar 4.8 merupakan grafik *Upload by Day* untuk memberikan informasi mengenai penggunaan *bandwidth* pada sisi *upload*. Informasi menampilkan total *upload* dalam sebulan dari tanggal

1 sampai 31 Agustus. Karena pada filter *combobox* dipilih *all traffic by day*, maka yang ditampilkan adalah total penggunaan *upload* semua *device*, tidak dirincikan pada *device* tertentu. Dapat dilihat penggunaan tertinggi ada pada tanggal 18 Agustus, yaitu 390.6MB. Sedangkan pada grafik *Total Traffic by Day* adalah kombinasi antara graffik *Download by Day* dan *Upload by Day*.

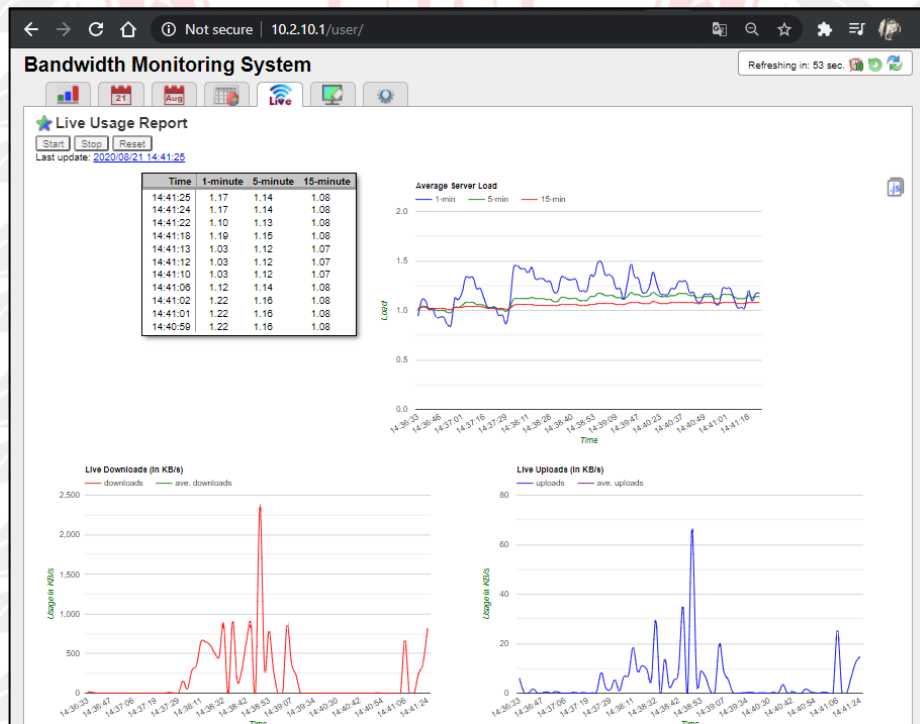
4.1.4.1 Beberapa Script Pada Halaman *Monthly Breakdown*

Berikut adalah potongan script untuk memperoleh data penggunaan bandwidth secara bulanan per hari. Fuction yang digunakan untuk adalah `updateUsage()`, `updateUsage_0()`, dan `updateUsage_1()` untuk melist penggunaan harian dalam sebulan.

```
updateUsage()
{
    $send2log "updateUsage: $1 $2" 0
    local cmd=$1
    local chain=$2
    local hr=$(date +%H)
    _ud_list=''
    local iptablesData=$(eval $cmd $_tMangleOption -nL
"$chain" -vxZ | tr -s '-' ' ' | grep -vi RETURN | grep "^[1-
9]" | cut -d' ' -f3,8,9)
    if [ -z "$iptablesData" ] ; then
        $send2log ">>> $cmd returned no data... returning " -1
        return
    fi
    createUDList "$iptablesData"
    $send2log "iptablesData-->
$iptablesData" -1
    _ud_list=$(echo "$_ud_list" | tail -n+2)
    $send2log "_ud_list-->
$_ud_list" 0
    IFS=$'\n'
    for line in $_ud_list
    }
updateUsage_0()
{ #update just IPv4 traffic
    $send2log "updateUsage_0" 0
    updateUsage 'iptables' "$BMS_IP4"
}
updateUsage_1()
{ #update both IPv4 & IPv6 traffic
    $send2log "updateUsage_1" 0
    updateUsage 'iptables' "$BMS_IP4"
    updateUsage 'ip6tables' "$BMS_IP6"}
```

4.1.5 Live Usage Report

Halaman *Live Usage* yang bisa dilihat pada Gambar 4.9 pada *Bandwidth Monitoring System* digunakan untuk mendapatkan informasi mengenai *bandwidth load*, *server load*, *active current device*, dan *active connection* secara *real-time* setiap 2 detik. *Server load* ditampilkan dalam bentuk tabel dan grafik dengan parameter 1-5-15 menit dan berapa besar beban yang digunakan pada waktu tertentu. *Bandwidth load* menampilkan besar keseluruhan *bandwidth* yang diakses baik *download* maupun *upload*. Sedangkan *current device* dan *active connection* digunakan untuk memantau aktivitas koneksi pengguna. Bisa dilihat pada gambar, ada tiga buah *button* yaitu *stop*, *pause*, dan *reset*. Tombol *stop* untuk menghentikan proses *monitoring*, *start* untuk memulai proses *monitoring*, dan *reset* untuk menghapus dan memulai dari awal proses *monitoring*. Sedangkan *last update* adalah *monitoring* terakhir yang dilakukan oleh sistem. Pada tabel *server load* akan mengambil 10 catatan terakhir dari *server load* dan terus melakukan proses *update* terlihat pada gambar waktu pencatatan dari pukul 14:41:25 hingga pukul 14:40:59.



Gambar 4.9 Halaman *Live Usage* - *Bandwidth Monitoring System* (1)

Pada Gambar 4.9 Grafik *Server Load* menunjukkan beban tertinggi ada pada rentang 1 menit (warna biru) pada pukul 14:39:09 yaitu sekitar 1.5. Sedangkan pada *live monitor download* dan *upload bandwidth usage* tertinggi ada pada pukul 14:38:53 yaitu *download* sekitar 2400KBps dan *upload* sekitar 66KBps.

1 Current Device						27 Active Connections					
Time	Group	Device	Total	KB/s	Total	KB/s	Source	Port	External Destinations Only	Port	Bytes
14:45:23	# devices: 1	# connections: 32	119.0 B	0.0	119.0 B	0.0	br-lan	55058	208.67.220.220	443	350900
-	Home	DHAITO-NLB490	119.0 B	0.0	119.0 B	0.0	DHAITO-NLB490	49198	36.92.231.77	443	40725
14:45:20	# devices: 1	# connections: 32	1.4 MB	368.0	55.1 KB	13.8	DHAITO-NLB490	56119	74.125.24.93	443	6375
-	Home	DHAITO-NLB490	1.4 MB	368.0	55.1 KB	13.8	DHAITO-NLB490	85333	74.125.10.58	443	1922458
14:45:16	# devices: 1	# connections: 31	119.0 B	0.1	119.0 B	0.1	DHAITO-NLB490	83528	34.223.130.205	443	6929
-	Home	DHAITO-NLB490	119.0 B	0.1	119.0 B	0.1	DHAITO-NLB490	53123	36.92.231.77	443	82984
14:45:14	# devices: 1	# connections: 31	124.6 B	0.1	159.0 B	0.2	DHAITO-NLB490	60616	36.92.231.77	443	20500
-	Home	DHAITO-NLB490	124.6 B	0.1	159.0 B	0.2	DHAITO-NLB490	82179	34.192.131.251	8002	239105
14:45:10	# devices: 1	# connections: 31	164.0 KB	164.0	8.7 KB	8.7	DHAITO-NLB490	82521	40.119.211.203	443	6962
-	Home	DHAITO-NLB490	164.0 KB	164.0	8.7 KB	8.7	DHAITO-NLB490	85992	74.125.200.188	5228	1514
14:45:09	# devices: 1	# connections: 27	1.6 MB	803.0	47.9 KB	24.0	DHAITO-NLB490	85526	36.92.231.77	443	12715
-	Home	DHAITO-NLB490	1.6 MB	803.0	47.9 KB	24.0	DHAITO-NLB490	84741	1.1.1.1	443	110291
14:45:02	# devices: 1	# connections: 39	194.8 B	0.1	80.0 B	0.1	DHAITO-NLB490	84735	54.241.223.196	443	10084
-	Hardware	br-lan	104.0 B	0.1	80.0 B	0.1	DHAITO-NLB490	85380	172.217.194.94	443	2215
14:44:57	# devices: 3	# connections: 38	325.0 B	0.2	955.0 B	0.5	DHAITO-NLB490	85093	149.154.171.237	443	22906
-	Hardware	br-lan	44.0 B	0.0	40.0 B	0.0	DHAITO-NLB490	82043	35.174.127.31	443	474887
-	Home	Samsung S10+	128.0 B	0.1	232.0 B	0.1	DHAITO-NLB490	56134	74.125.130.113	443	3459
-	Home	DHAITO-NLB490	153.0 B	0.1	683.0 B	0.3	DHAITO-NLB490	80685	172.217.27.46	443	2892
14:44:55	# devices: 1	# connections: 43	2.1 MB	1064.5	37.0 KB	18.5	DHAITO-NLB490	85105	103.253.147.57	443	2081
-	Home	DHAITO-NLB490	2.1 MB	1064.5	37.0 KB	18.5	DHAITO-NLB490	85533	74.125.130.113	443	4460
14:44:49	# devices: 1	# connections: 44	72.6 B	0.0	119.0 B	0.0	DHAITO-NLB490	82020	34.192.13.251	8001	256831
-	Home	DHAITO-NLB490	72.6 B	0.0	119.0 B	0.0	DHAITO-NLB490	80591	172.217.194.113	443	3457
-	Samsung S10+		44206	74.125.130.188	5228	4804					
-	Samsung S10+		42972	47.74.170.155	5222	17075					
-	Samsung S10+		44346	191.117.98.24	80	894					
-	Samsung S10+		49198	47.74.193.5	5222	47833					
-	Unknown: 10.2.10.221		40370	74.125.68.188	5228	6544					

Gambar 4.10 Halaman *Live Usage - Bandwidth Monitoring System* (2)

Pada Gambar 4.10 terdapat dua tabel yaitu *Current Device* dan *Active Connections*. *Current Device* menunjukkan jumlah *device* yang aktif sekarang dan dalam waktu rentang tertentu. Sebagai contoh pada pukul 14:45:20 ada 1 *device* yaitu DHAITO-NLB490 pada *group home* terdeteksi sedang melakukan aktifitas dalam jaringan dengan jumlah data 1.4MB dan *speed* 368KBps pada *download*, jumlah data 55.1KB dan *speed* 13.8KBps pada *upload*. *Active Connections* menunjukkan jumlah *device* yang berkomunikasi dengan *host* luar melalui *port* tertentu. Sebagai contoh Samsung S10+ dengan *port* 44206 sedang melakukan koneksi ke *host* 74.125.130.188 di *port* 5228 dengan jumlah data 4804B.

4.1.5.1 Beberapa Script Pada Halaman *Live Usage*

Berikut adalah potongan script untuk memonitor server load, dan bandwidth usage secara real-time. Fuction yang digunakan untuk adalah `doliveUpdates_0`, `doliveUpdates_1` mengupdate data setiap 2 detik.

```
doliveUpdates_0()
{ #doliveUpdates=0
  $send2log "doliveUpdates" -1
  return
}
```



```

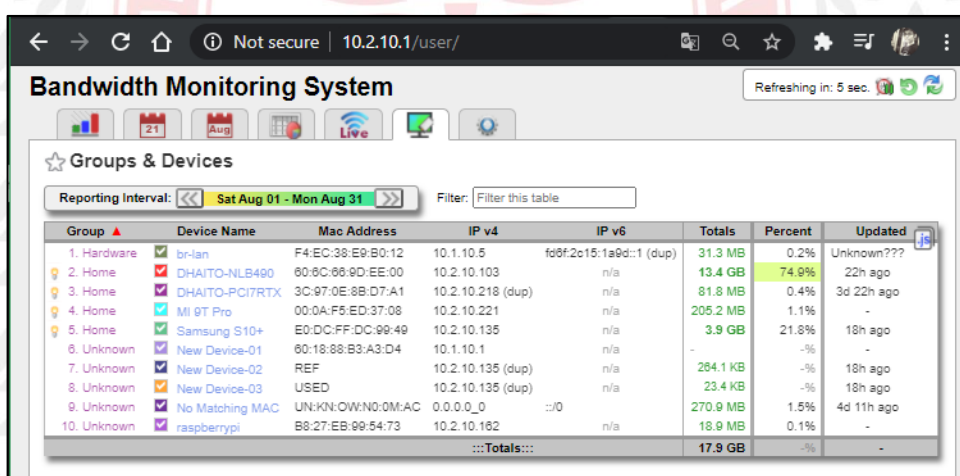
}
doliveUpdates_1()
{ #doliveUpdates=1
  $send2log "doliveUpdates_1" -1
  $send2log "_liveFilePath: $_liveFilePath" -1
  local loadavg=$(cat /proc/loadavg)
  $send2log ">>> loadavg: $loadavg" -1
  load1=$(echo "$loadavg" | cut -f1 -d" ")
  load5=$(echo "$loadavg" | cut -f2 -d" ")
  local load15=$(echo "$loadavg" | cut -f3 -d" ")
  local cTime=$(date +%T)
  echo "var last_update='$_cYear/$_cMonth/$_cDay $cTime'"
  serverload($load1,$load5,$load15)" > $_liveFilePath
  if [ "$_doCurrConnections" -eq "1" ] ; then
    $send2log ">>> curr_connections" -1
    local ddd=$(awk "$_contrack_awk" "$_contrack")
    err=$(echo "$ddd" 2>&1 1>> $_liveFilePath)
    $send2log "curr_connections >>>\n$ddd" -1
    [ -n "$err" ] && $send2log "ERR >>> doliveUpdates
(ddd): $err" 0
    #echo "$ddd" >> $_liveFilePath
  fi
  $send2log ">>> _liveusage: $_liveusage" -1
  echo "$_liveusage" >> $_liveFilePath
  _liveusage=' '
  [ "$_doArchiveLiveUpdates" -eq "1" ] && cat "$_liveFilePath"
>> $_liveArchiveFilePath
}

```



4.1.6 Group & Device

Pada Gambar 4.11 merupakan halaman *Groups And Devices* dari *Bandwidth Monitoring System*. Bagian ini digunakan sebagai daftar *groups* dan *devices* atau daftar pengguna yang pernah terhubung ke jaringan. Terdapat sebuah *button* untuk mengganti informasi mengenai *device* yang pernah terhubung ke jaringan berdasarkan bulan yang kita inginkan dan sebuah *textbox* sebagai filter untuk mencari *device* atau *group* yang diinginkan. Di halaman ini kita dapat melihat *Group*, *Devices*, *MAC Address*, *IPv4*, *IPv6*, total pemakaian *bandwidth*, Persentase pemakaian *bandwidth*, *update* aktivitas terakhir. Dapat dilihat di gambar. Dapat dilihat pada gambar ada sekitar 10 *device* dan 3 *group* yang pernah terhubung ke jaringan dalam bulan Agustus. Device DHAITO-NLB490 terdeteksi dengan *MAC Address* 60:6C:66:9D:EE:00, *IP Address* 10.2.10.103. total penggunaan *bandwidth* 13.4GB dari penggunaan total semua *device* 17.9GB, sekitar 74% total penggunaan *bandwidth* pada bulan Agustus, dan *update* terakhir sekitar 22 jam yang lalu.



Group	Device Name	Mac Address	IP v4	IP v6	Totals	Percent	Updated
1. Hardware	br-lan	F4:EC:38:E9:B0:12	10.1.10.5	fe8f:2c15:1a9d::1 (dup)	31.3 MB	0.2%	Unknown???
2. Home	DHAITO-NLB490	60:6C:66:9D:EE:00	10.2.10.103	n/a	13.4 GB	74.9%	22h ago
3. Home	DHAITO-PCI7RTX	3C:97:0E:8B:D7:A1	10.2.10.218 (dup)	n/a	81.8 MB	0.4%	3d 22h ago
4. Home	Mi 9T Pro	00:0A:F5:ED:37:08	10.2.10.221	n/a	205.2 MB	1.1%	-
5. Home	Samsung S10+	E0:DC:FF:DC:99:49	10.2.10.135	n/a	3.9 GB	21.8%	18h ago
6. Unknown	New Device-01	60:18:88:B3:A3:D4	10.1.10.1	n/a	-	-%	-
7. Unknown	New Device-02	REF	10.2.10.135 (dup)	n/a	284.1 KB	-%	18h ago
8. Unknown	New Device-03	USED	10.2.10.135 (dup)	n/a	23.4 KB	-%	18h ago
9. Unknown	No Matching MAC	UN:KN:OW:NO:0M:AC	0.0.0.0_0	:::0	270.9 MB	1.5%	4d 11h ago
10. Unknown	raspberrypi	B8:27:EB:99:54:73	10.2.10.162	n/a	18.9 MB	0.1%	-
:::Totals:::					17.9 GB	-%	-

Gambar 4.11 Halaman *Groups & Devices* - *Bandwidth Monitoring System*

4.1.6.1 Beberapa Script Pada Halaman *Groups & Devices*

Berikut adalah potongan script untuk menambah daftar *device* yang terhubung ke jaringan. Fuction yang digunakan untuk adalah `add2UsersJS()` untuk mendapatk IP dan *MAC Address* dan memasukan ke dalam daftar *Groups & Devices*.

```

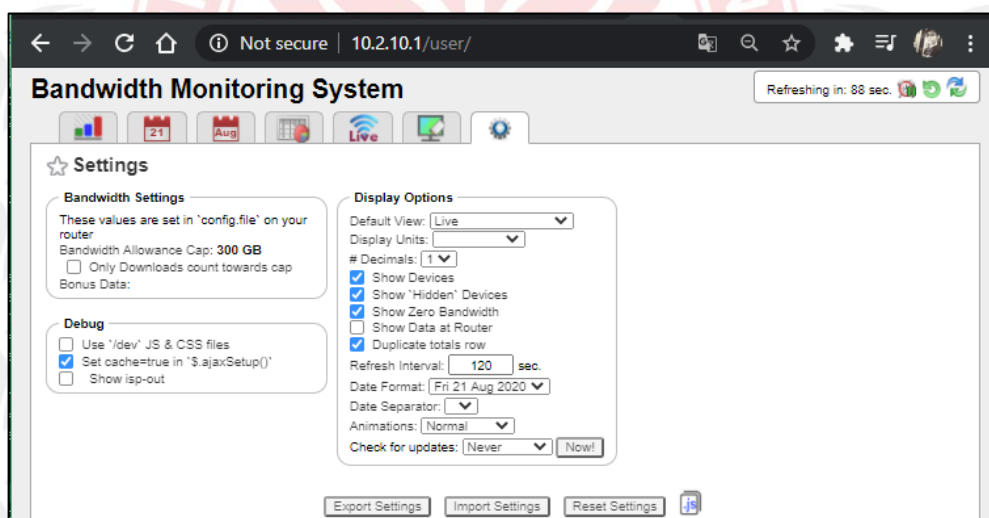
addUsersJS ()
{
$send2log "add2UsersJS: $1 $2 $3 $4 $5" 0
  local mac=$1
  local ip=$2
  local is_ipv6=$3
  local oname=''
  local dname=''
  [ -n "$4" ] && oname="$4"
  [ -n "$5" ] && dname="$5"
  $send2log "add2UsersJS: $mac $ip$is_ipv6 / $oname /
$dname" 1
  [ -z "$ip" ] || clearDupIPs
  local ds=$(date +"%Y-%m-%d %H:%M:%S")
  if [ -z "$dname" ] ; then
    local deviceName=$(getDeviceName $mac)
    $send2log "deviceName-->$deviceName" -1
    if [ -z "$_do_separator" ] ; then
      dname="$deviceName"
    else
      [ -z "$oname" ] &&
oname=${deviceName%"$_do_separator"*}
      dname=${deviceName#"$_do_separator"}
    fi
  fi
}

```



4.1.7 Settings

Halaman Settings bisa dilihat pada Gambar 4.12 digunakan untuk mengatur beberapa konfigurasi dari aplikasi web yang digunakan. Beberapa konfigurasi yang biasa dilakukan adalah mengatur *Fair Usage Policy* (FUP) pada *bandwidth* yang kita punya dari berlangganan internet pada *Internet Service Provider* (ISP). Kita juga dapat mengatur apakah hanya *download* yang masuk dalam perhitungan FUP pada pemakaian *bandwidth*. Pada konfigurasi debug kita dapat memilih versi debug yang kita inginkan, apakah menggunakan `"/dev"` JS dan CSS, dan mengaktifkan *cache* pada `$.ajax.Setup()`. Sedangkan konfigurasi *display option* untuk mengatur *refresh setting* dan tampilan UI dari *Bandwidth Monitoring System*. Tombol *Export Settings* digunakan untuk memindahkan setingan, *Import Setting* untuk memasang konfigurasi, dan *Reset Settings* untuk mengembalikan setingan ke awal sistem.



Gambar 4.12 Halaman Settings - Bandwidth Monitoring System.

4.1.7.1 Beberapa Script Pada Halaman Settings

Berikut adalah script yang digunakan dalam halaman Settings untuk melakukan pengecekan jika perubahan terjadi. Function yang berperan dalam kasus ini adalah `checkConfig()` kemudian konfigurasi baru akan disimpan dalam `config.file` dan akan dijalankan oleh sistem dengan konfigurasi yang baru.

```
checkConfig()
{
  local dcf=$(date -r "$d_baseDir/default_config.file" +%s)
  local cf=$(date -r "$_configFile" +%s)
  $send2log "checkConfig: dcf: $dcf cf: $cf" 0
  [ "$cf" \< "$dcf" ] && return
  touch "$d_baseDir/default_config.file"
  $send2log "checkConfig >>> config.file has changed! Resetting
setInitValues ---" 2
  setConfigJS
  [ "$enable_ftp" -eq 1 ] && send2FTP "$_configFile"
  updateHourly
  setInitValues
}
```



4.2 Hasil Pengujian AQM

Bagian ini berisi evaluasi terhadap kinerja dari algoritma antrian yakni Drop Tail, Fq_Codel, dan Cake dalam yang diuji. Beberapa matriks kinerja akan digunakan untuk mengevaluasi efektivitas dari algoritma yang ada. Matriks ini meliputi nilai *throughput*, *packet loss*, dan nilai *latency* pada simulasi sebagai berikut:

1. *Video on Demand* (VOD) dari Youtube berjudul Céline Dion - Hits Medley (Live in Boston) pada <https://www.youtube.com/watch?v=cvu28pKql8g>.
Detail: 10 menit 4 detik pada resolusi 1920x1080 piksel (FHD 1080p).
2. File *Download* dari Filearena.net dengan nama *Blank-Test-File* pada *link download* http://mirror.filearena.net/pub/speed/SpeedTest_512MB.dat
Detail: Ukuran File 512MB dengan Tipe File: DAT.

Kebutuhan *bandwidth* minimal pada *video on demand* Youtube dapat dilihat pada Tabel 4.1.

Tabel 4.1 Perhitungan Kebutuhan *Bandwidth* untuk VOD.

Ukuran Berkas Video (Megabit)	177,13 MB (1417 Mbit)
Durasi (detik)	10 Menit 4 Detik (604 detik)
Minimal Bandwidth	1417Mb / 604s = 2,346026 Mbps =2.346 Kbps
Minimal Bandwidth untuk 3 komputer	3pc * 2346 Kbps 7.038 Kbps

Sedangkan *bandwidth* minimal pada file download untuk 3 komputer dapat dilihat pada Tabel 4.2

Tabel 4.2 Perhitungan Kebutuhan *Bandwidth* untuk File Download

Ukuran Berkas File	512 MB (4096 Mbit)
Durasi	4096Mb / 15Mbps @ ~300 detik 4096Mb / 10Mbps @ ~420 detik 4096Mb / 05Mbps @ ~840 detik

4.2.1 Pengaturan *Traffic Shaping* Pada OpenWRT

Konfigurasi *traffic shaping* dilakukan pada perangkat Raspberry Pi dengan OpenWRT pada paket *management bandwidth Simple Queue Management* SQM yang sudah dipasang. Pada Gambar 4.1 *meunjukkan* bagian SQM untuk menentukan batas limit *bandwidth* yang diterapkan dalam pengujian yang telah dilakukan.

Smart Queue Management
With **SQM** you can enable traffic shaping, better mixing (Fair Queueing), active queue length management (AQM) and prioritisation on one network interface.

Queues Delete

Basic Settings Queue Discipline Link Layer Adaptation

Enable this SQM instance.

Interface name **wan0 (wan)**

Download speed (kbit/s) (ingress) set to 0 to selectively disable ingress shaping:

Upload speed (kbit/s) (egress) set to 0 to selectively disable egress shaping:

Create log file for this SQM instance under /var/run/sqm/\${interface_name}.[start|stop]-sqm.log.

Verbosity of SQM's output into the system log. **info (default)**

Add Save & Apply Save Reset

Gambar 4.13 SQM Pada OpenWRT untuk menentukan *Limit Bandwidth*

Smart Queue Management
With **SQM** you can enable traffic shaping, better mixing (Fair Queueing), active queue length management (AQM) and prioritisation on one network interface.

Queues Delete

Basic Settings Queue Discipline Link Layer Adaptation

Queueing disciplines useable on this system. After installing a new qdisc, you need to restart the router to see updates! **cake**

Queue setup script **simple.qos**

layer_cake.qos:
This uses the cake qdisc as a replacement for both htb as shaper and fq_codel as leaf qdisc. This exercises cake's diffserv profile(s) as different "layers" of priority. This script requires that cake is selected as qdisc, and forces its usage. See: <http://www.bufferbloat.net/projects/codel/wiki/Cake> for more information

piece_of_cake.qos:
This just uses the cake qdisc as a replacement for both htb as shaper and fq_codel as leaf qdisc. It just does not come any simpler than this, in other words it truly is a "piece of cake". This script requires that cake is selected as qdisc, and forces its usage. See: <http://www.bufferbloat.net/projects/codel/wiki/Cake> for more information

simple.qos:
BW-limited three-tier prioritisation scheme with your qdisc on each queue. (default)

simplest.qos:
Simplest possible configuration: HTB rate limiter with your qdisc attached.

simplest_tbf.qos:
Simplest possible configuration (TBF): TBF rate limiter with your qdisc attached. TBF may give better performance than HTB

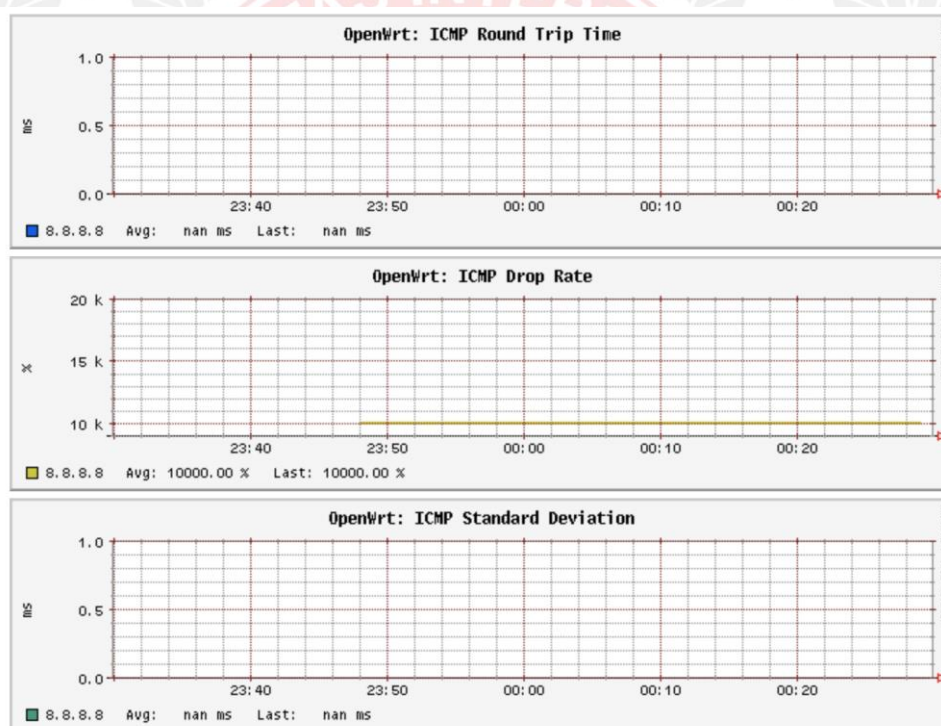
Save & Apply Save Reset

Gambar 4.14 SQM OpenWRT Untuk Menentukan Disiplin Antrian

Sedangkan pada Gambar 4.2 adalah bagian SQM untuk menentukan metode antrian yang diterapkan yang terdiri dari tiga pilihan yakni Drop Tail (mematikan fungsi AQM), Fq_Codel, dan Cake. Semua konfigurasi akan diatur sesuai skenario yang diuji..

4.2.2 Hasil Pengujian

Data yang diperoleh berupa *throughput* (kbps), *packet drop*, dan nilai *latency* (ms) yang akan diambil dari Luci-Statistic yang terdapat pada OpenWRT. Simulasi pengecekan akan dilakukan satu per satu untuk semua kondisi yang sudah ditentukan. Gambar di bawah ini contoh informasi yang diberikan oleh Luci-Statistic pada OpenWRT.



Gambar 4.15 Contoh tampilan Luci-Statistic pada OpenWRT.

Antar muka pada Luci Statistic pada area *latency* dapat dilihat pada Gambar 4.15. Terdapat tiga grafik utama pada bagian ping seperti pada gambar yang ditampilkan. Grafik pertama adalah *Round Trip Time* yang digunakan untuk melakukan *monitoring* terhadap *latency* pada waktu tertentu, grafik menampilkan perjalanan *ping* selama 60 menit dan berapa besar *latency* yang terjadi. Grafik kedua menampilkan *Drop Rate* yang digunakan untuk

monitoring paket yang di drop pada waktu tertentu, grafik menampilkan perjalanan paket drop selama 60 menit dan berapa besar jumlah paket yang di drop. Sedangkan pada grafik *Standart deviation* digunakan untuk memperoleh hasil standar deviasi dari *latency* yang terjadi selama 60 menit. Semua data tersebut digunakan untuk mengetahui kualitas jaringan yang dimiliki dan sebagai pengambilan keputusan untuk langkah selanjutnya. .

Data informasi yang akan di ambil dari adalah *throughput* (kbps), *packet drops*, dan nilai *latency* (ms). Data statistik akan diperbarui tiap 5 detik untuk kemudian diolah dalam aplikasi Microsoft Excel untuk mendapatkan nilai rata - rata yang bisa dilihat pada lampiran laporan tugas akhir penulis. Setelah dilakukan perhitungan nilai rata - rata dari tiap parameter yang di uji pada semua simulasi, langkah selanjutnya adalah melakukan analisa perbandingan kinerja algoritma antrian dengan ketentuan sebagai berikut :

1. Perbandingan kinerja algoritma Drop Tail dengan tiga kapasitas *bandwidth* yakni 15Mbps, 10Mbps, dan 5Mbps pada simulasi VOD dan *file downloads*.
2. Perbandingan kinerja algoritma Fq_Codel dengan tiga kapasitas *bandwidth* yakni 15Mbps, 10Mbps, dan 5Mbps pada simulasi VOD dan *file downloads*.
3. Perbandingan kinerja algoritma Cake dengan tiga kapasitas *bandwidth* yakni 15Mbps, 10Mbps, dan 5Mbps pada simulasi VOD dan *file downloads*.
4. Perbandingan nilai *packet loss* untuk tiga algoritma antrian dengan tiga kapasitas *bandwidth* yakni 15Mbps, 10Mbps, dan 5Mbps pada simulasi VOD dan *file downloads*.
5. Perbandingan nilai *latency* untuk tiga algoritma antrian dengan tiga kapasitas *bandwidth* yakni 15Mbps, 10Mbps, dan 5Mbps pada simulasi VOD dan *file downloads*.
6. Perbandingan nilai *throughput* untuk tiga algoritma antrian dengan tiga kapasitas *bandwidth* yakni 15Mbps, 10Mbps, dan 5Mbps pada simulasi VOD dan *file downloads*.

4.2.3 Kinerja Algoritma Drop Tail

Algoritma Drop Tail atau Tail Drop akan membuang paket (*drop*) jika kondisi *buffer* penuh atau terjadi antrian paket. Secara opsional, AQM dapat dinonaktifkan dengan memilih None (Drop Tail) dalam konfigurasi *ShapingObject*. Ini berarti bahwa tidak ada *drop preemptive* yang dilakukan, dan paket yang akan masuk dijatuhkan ketika antrian penuh (*tail dropping*).

Tabel 4.3 Kinerja Algoritma Drop Tail

Bandwidth (Mbps)	Throughput (Kbps)		Drop		Latency (ms)	
	VOD	DL	VOD	DL	VOD	DL
15	9018	8007,8	0	3322,8	50,4	64
10	8354,9	7954	146,2	10212,9	151,1	177,2
5	4602,7	3511,1	1101,1	12945,2	230,7	234,1

Pada Tabel 4.3 terlihat pada simulasi VOD, kapasitas bandwidth 15Mbps belum ada paket yang dijatuhkan atau nol (0), nilai *latency*-nya 50,4ms dan *throughput* yang dihasilkan sebesar 9018Kbps. Tidak ada paket yang dijatuhkan merupakan hal yang wajar karena *bandwidth* yang tersedia melebihi dari kebutuhan minimalnya. Pada kapasitas 10Mbps paket yang dijatuhkan adalah 146,2 paket dengan nilai *latency*-nya 151,1ms dan *throughput* yang dihasilkan sebesar 8354,9Kbps. Sedangkan pada kapasitas *bandwidth* 5 Mbps paket yang dijatuhkan sebesar 1101,1 paket, nilai *latency*-nya 230,7ms dan *throughput* yang dihasilkan sebesar 4602,7Kbps.

Sedangkan pada simulasi *file download* kapasitas bandwidth 15 Mbps paket yang dijatuhkan 3322,8 paket, nilai *latency*-nya 64 ms dan *throughput* yang dihasilkan sebesar 8007,8 Kbps. Paket data sudah ada yang dijatuhkan karena *traffic shaping* menjaga agar *bandwidth* yang tersedia bisa dipakai secara merata. Pada kapasitas 10 Mbps paket yang dijatuhkan adalah 10212,9 paket dengan nilai *latency*-nya 177,2 ms dan *throughput* yang dihasilkan sebesar 7954 Kbps. Sedangkan pada kapasitas bandwidth 5 Mbps paket yang dijatuhkan sebesar 12945,2 paket, nilai *latency*-nya 234,1 ms dan *throughput* yang dihasilkan sebesar 3511,1 Kbps.

Semakin kecil kapasitas *bandwidth*-nya maka semakin besar nilai paket yang akan di jatuhkan (*drop packet*) dan semakin tinggi nilai *latency*-nya sedangkan *throughput*-nya akan semakin kecil. Kondisi ini wajar karena semakin kecil *bandwidth* yang tersedia maka semakin besar nilai *latency* karena jaringan semakin padat. Semakin kecil *bandwidth* yang tersedia maka semakin kecil pula *throughput*-nya, hal ini dikarenakan paket data yang dikirimkan oleh server menyesuaikan kapasitas *bandwidth* yang ada di *client*.

Kategori *latency* dari ITU-T.G.1010, nilai *latency* Drop Tail memiliki nilai antara 50,4ms sampai dengan 230,7ms, ini termasuk golongan bagus karena berada di antara 150ms – 300ms.

4.2.4 Kinerja Algoritma Fq_Codel

Fq_Codel melakukan *early drops* berdasarkan waktu yang dihabiskan oleh paket dalam antrian. Jika waktu untuk mencapai kepala antrian (*head of the queue*) lebih dari target latensi untuk interval (secara *default* 50 ms), *early drops* akan dilakukan. Tingkat *early drop* dimulai dari yang rendah kemudian meningkat dengan jumlah interval berikutnya di mana waktu dalam antrian melebihi tujuan latensi. Algoritma Fq_Codel berusaha menjaga *latency*-nya tetap rendah dengan mengorbankan nilai drop yang tinggi.

Tabel 4.4 Kinerja Algoritma Fq_Codel

Bandwidth (Mbps)	Throughput (Kbps)		Drop (Packet)		Latency (ms)	
	VOD	DL	VOD	DL	VOD	DL
15	8845,5	8525,9	2393,9	3303,3	15	18,8
10	8007,7	8149,7	10387,2	9714,9	18,9	33
5	4428,2	3740	10679,1	13266,5	20,5	37,8

Tabel 4.4 memperlihatkan pada simulasi VOD kapasitas *bandwidth* 15Mbps paket yang di jatuhkan 2393 paket, nilai *latency*-nya 15ms dan *throughput* yang dihasilkan sebesar 8845,5 Kbps. Pada kapasitas 10 Mbps paket yang di jatuhkan adalah 10387,2 paket dengan nilai *latency*-nya 18,9ms dan *throughput* yang dihasilkan sebesar 8007,7Kbps. Sedangkan pada kapasitas

bandwidth 5 Mbps rerata paket yang dijatuhkan sebesar 10679,1 paket, nilai *latency*-nya 20,5ms dan *throughput* yang dihasilkan sebesar 4428,2Kbps. Pada Fq_Codel, walaupun pada kondisi kapasitas *bandwidth* 15Mbps yang lebih besar dari kebutuhan minimalnya, paket tetap ada yang dijatuhkan. Hal ini dikarenakan Fq_Codel berusaha menjaga nilai *latency* tetap rendah dengan cara melakukan drop paket.

Sedangkan pada simulasi *file download* kapasitas *bandwidth* 15 Mbps paket yang dijatuhkan 3303,3 paket, nilai *latency*-nya 18,8 ms dan *throughput* yang dihasilkan sebesar 8525,9 Kbps. Pada kapasitas 10 Mbps paket yang dijatuhkan adalah 9714,9 paket dengan nilai *latency*-nya 33 ms dan *throughput* yang dihasilkan sebesar 8149,7 Kbps. Sedangkan pada kapasitas *bandwidth* 5 Mbps paket yang dijatuhkan sebesar 13266,5 paket, nilai *latency*-nya 37,8 ms dan *throughput* yang dihasilkan sebesar 3740 Kbps.

Semakin kecil kapasitas *bandwidth*-nya maka semakin besar nilai paket yang akan dijatuhkan (*drop packet*) dan semakin tinggi nilai *latency*-nya sedangkan *throughput*-nya akan semakin kecil. Kondisi ini wajar karena semakin kecil *bandwidth* yang tersedia maka semakin besar nilai *latency* karena jaringan semakin padat. Nilai *latency* dari tiga skenario kapasitas *bandwidth* tidak berbeda jauh relatif hampir sama. Hal tersebut wajar karena Fq_Codel berusaha menjaga nilai *latency* tetap rendah.

Semakin kecil *bandwidth* yang tersedia maka semakin kecil pula *throughput*-nya, hal ini dikarenakan paket data yang dikirimkan oleh server menyesuaikan kapasitas *bandwidth* yang ada di *client*. Berdasar kategori *latency* dari ITU-T.G.1010, nilai *latency* Fq_Codel mempunyai nilai berkisar antara 15ms sampai 37,8ms, ini termasuk kategori Sangat Bagus karena di bawah 150ms.

4.2.5 Kinerja Algoritma Cake

Cake melakukan penjatuhan paket awal (*early drop*) dengan probabilitas yang telah ditetapkan untuk *ShapingObject* untuk prioritas tersebut. Cake juga menyesuaikan *drop probability* untuk *ShapingObject* berdasarkan antrian yang digunakan (*queue usage*). Ketika *ShapingObject* kosong, *drop probability*

berkurang. Ketika antrian penuh (paket *enqueued* sama dengan ukuran antrian), antrian akan meningkatkan *drop probability*. Hal ini digunakan untuk membuat tren ukuran antrian (*the queue size trend*) untuk menjaga latensi .

Tabel 4.5 Kinerja Algoritma Cake

Bandwidth (Mbps)	Throughput (Kbps)		Drop (Packet)		Latency (ms)	
	VOD	DL	VOD	DL	VOD	DL
15	8996,9	8290,5	0	3260	40,2	43,6
10	8293,9	8311,7	617,8	11776,2	77,2	87,9
5	4598,7	3891,4	1481	13036,1	89,3	91,5

Pada tabel 4.5 terlihat pada simulasi VOD kapasitas *bandwidth* 15Mbps belum ada paket yang dijatuhkan atau paket yang dijatuhkan nilainya nol (0), nilai *latency*-nya 40,2ms dan *throughput* yang dihasilkan sebesar 8996,9Kbps. Paket belum ada yang dijatuhkan karena kapasitas *bandwidth* yang tersedia lebih daripada kebutuhan minimalnya. Pada kapasitas 10Mbps paket yang dijatuhkan adalah 617,8 paket dengan nilai *latency*-nya 77,2ms dan *throughput* yang dihasilkan sebesar 8293,9Kbps. Sedangkan pada kapasitas *bandwidth* 5 Mbps paket yang dijatuhkan sebesar 1481 paket, nilai *latency*-nya 89,3ms dan *throughput* yang dihasilkan sebesar 4598,7Kbps.

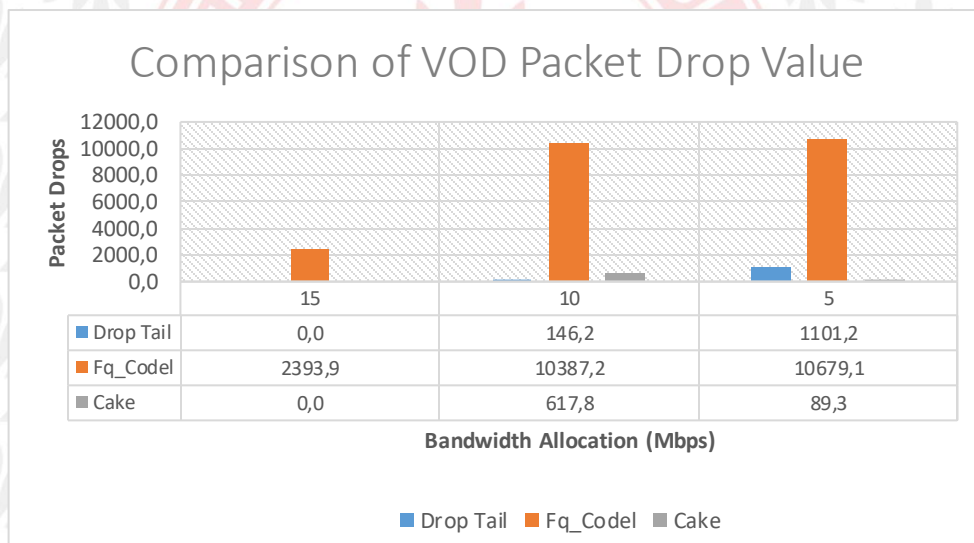
Sedangkan pada simulasi *file download* kapasitas *bandwidth* 15 Mbps paket yang dijatuhkan 3260 paket, nilai *latency*-nya 43,6 ms dan *throughput* yang dihasilkan sebesar 8290,5 Kbps. Pada kapasitas 10 Mbps rerata paket yang dijatuhkan adalah 11776,2 paket dengan nilai *latency*-nya 87,9 ms dan *throughput* yang dihasilkan sebesar 8311,7 Kbps. Sedangkan pada kapasitas *bandwidth* 5 Mbps rerata paket yang dijatuhkan sebesar 13036,1 paket, nilai *latency*-nya 91,5 ms dan *throughput* yang dihasilkan sebesar 3891,4 Kbps.

Semakin kecil kapasitas *bandwidth*-nya maka semakin besar nilai paket yang akan dijatuhkan (*drop packet*) dan semakin tinggi nilai *latency*-nya sedangkan *throughput*-nya akan semakin kecil. Kondisi ini wajar karena semakin kecil *bandwidth* yang tersedia maka semakin besar nilai *latency* karena jaringan semakin padat. Semakin kecil *bandwidth* yang tersedia maka

semakin kecil pula throughput-nya, hal ini dikarenakan paket data yang dikirimkan oleh server menyesuaikan kapasitas *bandwidth* yang ada di *client*. Berdasar kategori *latency* dari ITU-T.G.1010, nilai *latency* Cake mempunyai nilai berkisar antara 40 ms sampai 91,5 ms, ini termasuk kategori Sangat Bagus karena di bawah 150ms.

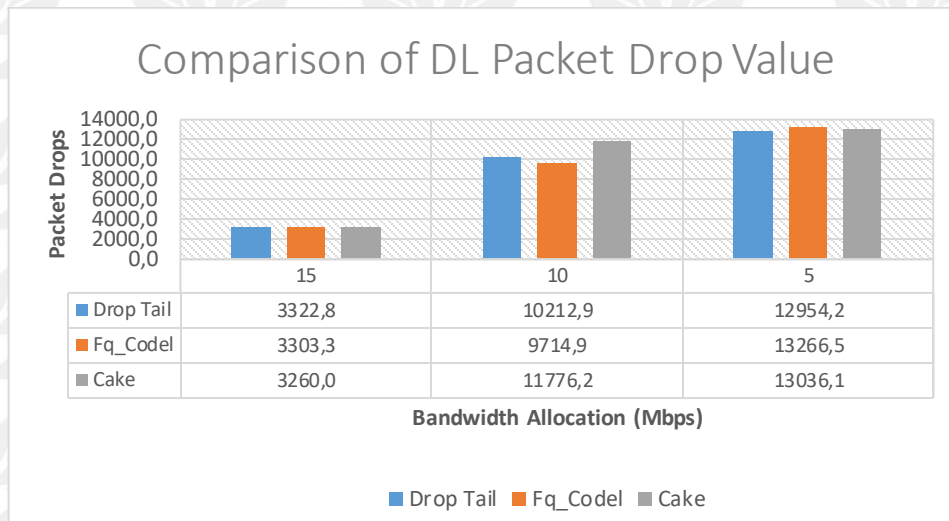
4.2.6 Perbandingan Jumlah *Packet-Drop*

Packet-drops adalah jumlah paket yang dijatuhkan dikarenakan beberapa kondisi seperti kondisi penampungan (*buffer*) yang sudah penuh, kerusakan pada perangkat jaringan, dan lain – lain. Pemantauan *packet-drops* dilakukan pada perangkat Luci-Statistic pada OpenWRT. Berikut ini adalah grafik perbandingan rata-rata jumlah paket yang dijatuhkan (drop) untuk tiga algoritma antrian dengan tiga kapasitas *bandwidth* yang berbeda



Gambar 4.16 Perbandingan Jumlah Paket Drop pada VOD

Pada gambar 4.17 memperlihatkan paket yang dijatuhkan pada *Fq_Codel* selalu lebih tinggi daripada dua algoritma lainnya. Pada kapasitas *bandwidth* 15Mbps, Drop Tail dan Cake tidak menjatuhkan paket sedangkan pada *Fq_Codel* paket yang dijatuhkan adalah 2393,9 paket. Hal tersebut wajar karena pada Drop Tail dan Cake paket akan dijatuhkan jika terjadi antrian atau adanya *packet loss*, sedangkan pada *Fq_Codel* paket akan tetap akan dijatuhkan untuk menjaga nilai *latency* agar tetap rendah sesuai target.

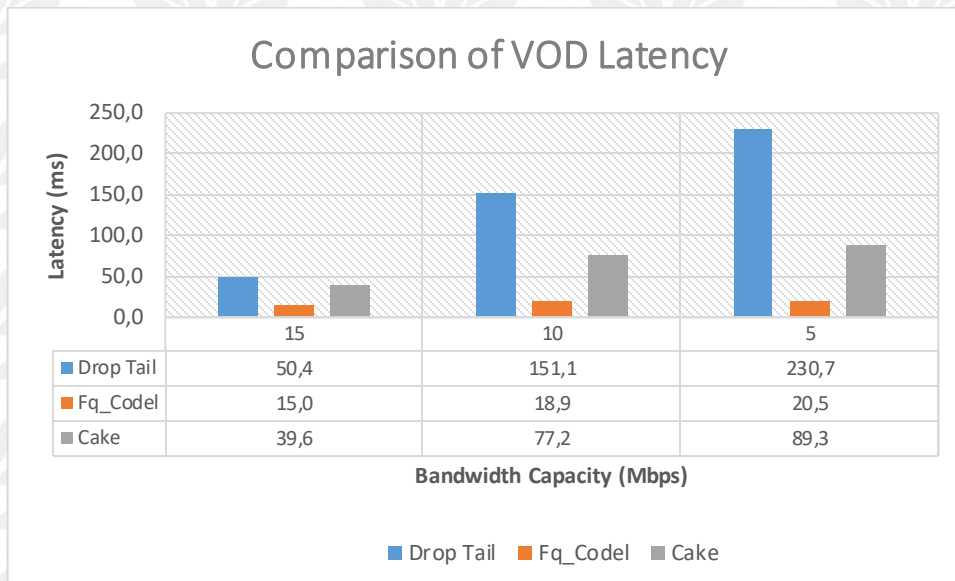


Gambar 4.17 Perbandingan Jumlah *Drop Packet* pada *File Download*.

Pada gambar 4.18 menginformasikan bahwa pada simulasi *file download*, semua algoritma antrian menjatuhkan paket. Fq_Codel memiliki jumlah paket yang dijatuhkan paling tinggi kedua pada kapasitas *bandwidth* 15 Mbps setelah Drop Tail. Sedangkan pada kapasitas 10 Mbps jumlah paket yang dijatuhkan paling kecil dibandingkan Drop Tail dan Cake. Drop Tail dan Cake pada *bandwidth* 15Mbps menjatuhkan paket karena algoritma antrian berusaha membagi jumlah paket yang dikirim lebih merata sehingga ada paket yang dijatuhkan agar tidak terjadi paket yang dikirim ke satu tujuan lebih tinggi dari tujuan lainnya.

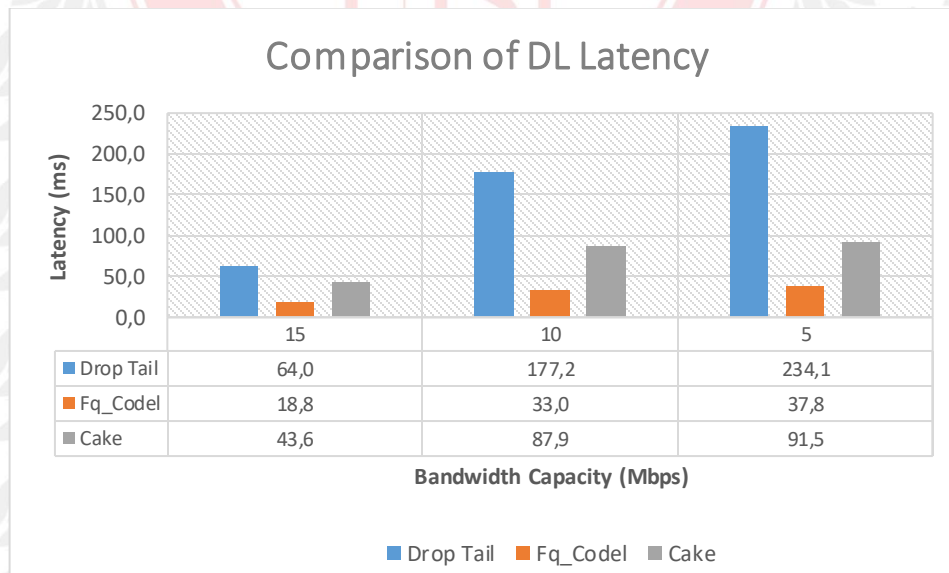
4.2.7 Perbandingan Nilai *Latency*

Latency atau *Delay* adalah selisih waktu kirim antara satu paket dengan paket lainnya. Terjadi penundaan waktu kirim karena adanya proses transmisi dari satu titik ke titik tujuan. Nilai rata - rata *latency* diambil dari perangkat Luci-Statistic pada OpenWRT. Berikut ini adalah grafik perbandingan nilai *latency* pada tiga algoritma antrian dengan tiga kapasitas *bandwidth* yang berbeda yang bisa dilihat pada Gambar 4.19.



Gambar 4.18 Perbandingan Nilai *Latency* pada VOD.

Pada gambar 4.19, terlihat pada simulasi VOD nilai *latency* Fq_Codel paling kecil dibandingkan Drop Tail dan Cake. Hal tersebut wajar karena Fq_Codel bertujuan untuk menjaga nilai *latency* yang rendah sesuai target dengan kompensasi jumlah paket yang dijatuhkan menjadi besar. Nilai *latency* Fq_Codel pada ketiga skenario kapasitas *bandwidth* hampir sama dengan perbedaan selisih tidak banyak.



Gambar 4.19 Perbandingan Nilai *Latency* pada *File Download*.

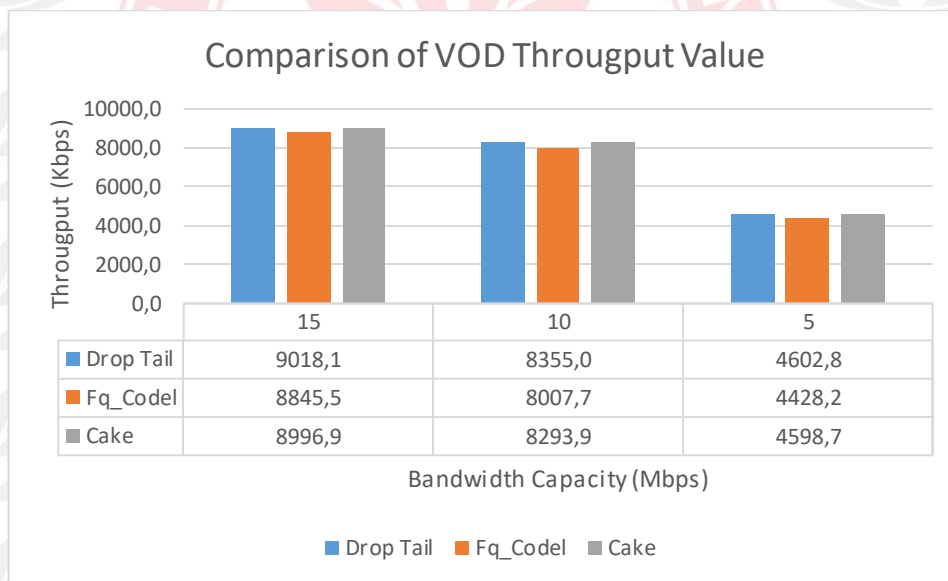
Pada Gambar 4.20 memperlihatkan perbandingan nilai *latency* pada file *download*, algoritma Fq_Codel memiliki nilai *latency* terkecil dan cenderung

stabil di bawah 50 ms, dibandingkan algoritma Drop Tail dan Cake. Kecilnya nilai *latency* pada algoritma Fq_Codel harus dibayar dengan kompensasi kehilangan paket drop yang tinggi.

Berdasarkan tabel kategori *latency* atau *delay* yang dikeluarkan oleh ITU-T.G.1010, nilai *latency* pada perbandingan ketiga algoritma termasuk kategori Sangat Bagus karena di bawah 150ms kecuali Drop Tail pada kapasitas *bandwidth* 15Mbps dan 10Mbps termasuk kategori Bagus karena nilai *latency*-nya antara 150ms - 300ms.

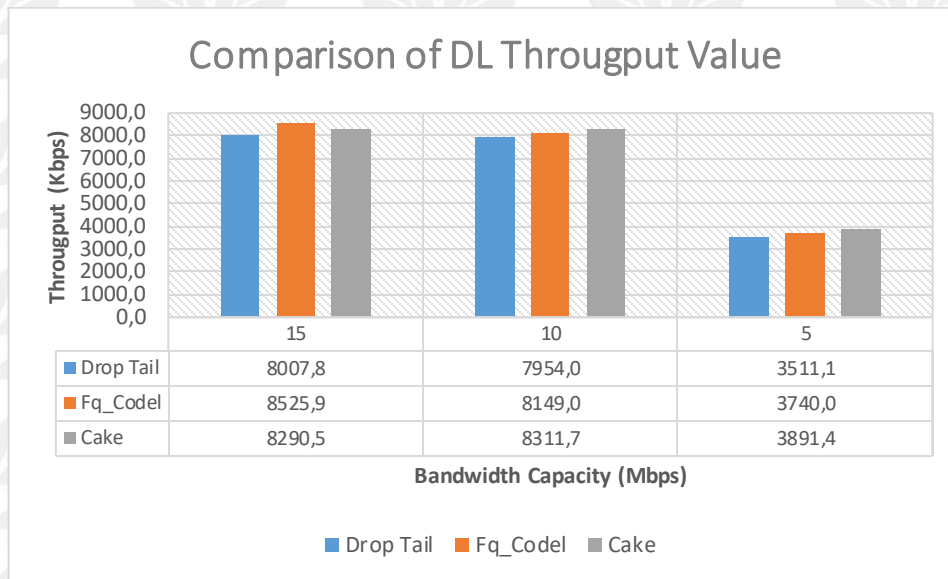
4.2.8 Perbandingan Nilai *Throughput*

Throughput merupakan jumlah total *byte* yang diterima oleh *receiver* pada selang waktu saat paket pertama dan terakhir dikirimkan. Berikut ini adalah grafik perbandingan nilai *throughput* untuk tiga algoritma antrian dengan tiga kapasitas *bandwidth* yang berbeda yang bisa dilihat pada Gambar 4.21.



Gambar 4.20 Perbandingan Nilai *Throughput* pada VOD.

Pada gambar 4.21 perbandingan *throughput* simulasi VOD, diperoleh informasi bahwa Drop Tail mencapai nilai *throughput* paling tinggi pada kapasitas *bandwidth* 15 Mbps sebesar 9018,1 Kbps. Cake dapat mencapai nilai *throughput* paling tinggi pada kapasitas *bandwidth* 10 Mbps sebesar 8293,9 Kbps dan kapasitas 5 Mbps sebesar 4598,7 Kbps.



Gambar 4.21 Perbandingan Nilai *Througput* pada *File Dwonload*.

Dari gambar 4.22 pada simulasi file download di atas terlihat Fq_Codel memiliki nilai *througput* paling tinggi pada kapasitas *bandwidth* 15Mbps sebesar 8525,9Kbps. Cake dapat mencapai nilai *througput* paling tinggi pada kapasitas *bandwidth* 10Mbps sebesar 8311,7Kbps dan kapasitas 5Mbps sebesar 3891,4Kbps.

4.2.9 Penilaian Kinerja *Quality-of-Services* (QoS)

Berdasarkan uraian kinerja jumlah paket yang dijatuhkan (*Packet-Drops*), nilai *latency*, dan *througput* akan di buat pembobotan nilai dengan ketentuan :

1. Jumlah *packet drop* tertinggi mendapat nilai 1 dan terendah mendapat nilai 3. Sedangkan jumlah *packet drop* di antara tertinggi dan terendah mendapat nilai 2.
2. Nilai *latency* terendah mendapat nilai 3 dan tertinggi mendapat nilai 1. Sedangkan nilai *latency* di antara tertinggi dan terendah mendapat nilai 2.
3. Nilai *througput* tertinggi mendapat nilai 3 dan terendah mendapat nilai 1. Sedangkan nilai *througput* di antara tertinggi dan terendah mendapat nilai 2.

Berikut ini adalah tabel hasil pembobotan nilai dari hasil pengujian yang telah dilakukan terhadap tiga algoritma.

Tabel 4.6 Pembobotan Nilai Kinerja QoS untuk VOD dan File *Dwownload*.

Bandwidth (Mbps)	VOD			DL		
	Drop Tail	Fq_Codel	Cake	Drop Tail	Fq_Codel	Cake
15	6	6	6	6	6	6
10	6	7	7	7	7	7
5	6	6	7	5	7	6
Total	18	19	20	18	20	19

Berdasarkan Tabel 4.6 terlihat bahwa algoritma Cake mendapat nilai paling tinggi dibanding lainnya untuk simulasi VOD. Kemudian diikuti dengan algoritma Drop Tail lalu Fq_Codel. Sedangkan algoritma Fq_Codel unggul untuk simulasi *file download*, diikuti dengan algoritma Cake lalu Drop Tail.

(Halaman ini sengaja dikosongkan)

