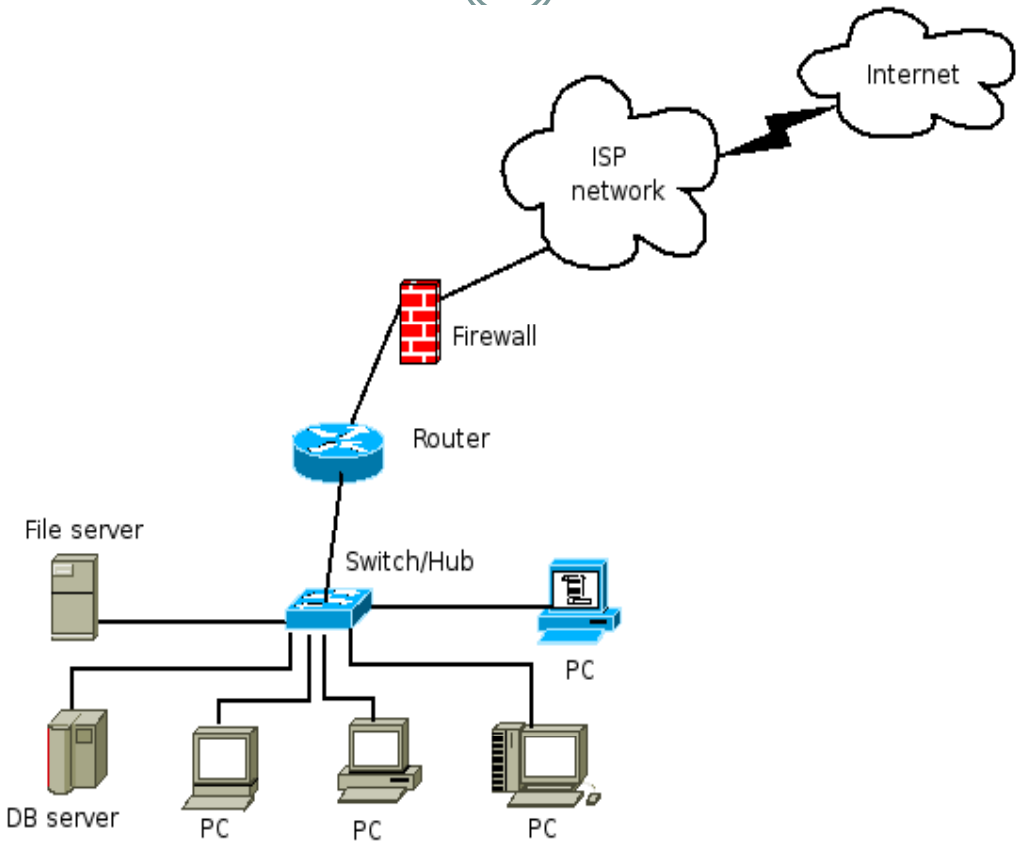
Bandwidth Management

Muhammad Zen Samsono Hadi, ST. MSc.

Lab. Komunikasi Digital
Gedung D4 Lt. 1
EEPIS-ITS

Network Diagram



Introduction

- Bandwidth manajemen merupakan cara pengaturan bandwidth supaya terjadi pemerataan pemakaian bandwidth
- Cara melakukan :
  - Dari Proxy
  - Dengan Qos / Traffick Shapping : HTB, CBQ

Bandwidth Management

Bandwidth Management (Traffic Control/Shaping) adalah suatu istilah yang ditujukan pada suatu subsistem antrian *packet* dalam/pada suatu jaringan atau *network devices*. Secara singkat *traffic control/shaping* adalah suatu usaha mengontrol *traffic* jaringan sehingga *bandwidth* lebih optimal dan performa network lebih terjamin. Fungsi dan operasi traffic control pada kernel linux terdiri dari komponen-komponen berikut ini:

- queueing disciplines (qdisc)
- classes
- filters
- policer

Bandwidth Management

Queueing discipline (Qdisc) ***bertanggungjawab untuk mentransmisikan data.***

Classes dipasang pada qdisc dan mengandung/berisi traffic. Setiap class yang tidak memiliki child class selalu memiliki 1 qdisc yang berasosiasi dengannya untuk mentransmisikan paket-paket data, dan qdisc tersebut menampung seluruh traffic yang masuk/mengalir ke dalam class tersebut.

Filters dipasang pada qdisc dan class dan men-split traffic menjadi beberapa child-class yang berbeda.

Policers digunakan untuk meyakinkan filters sesuai/match hanya dengan satu rate paket tertentu. **Policers** dapat dishare oleh beberapa filter berbeda dan pada interface-interface berbeda.

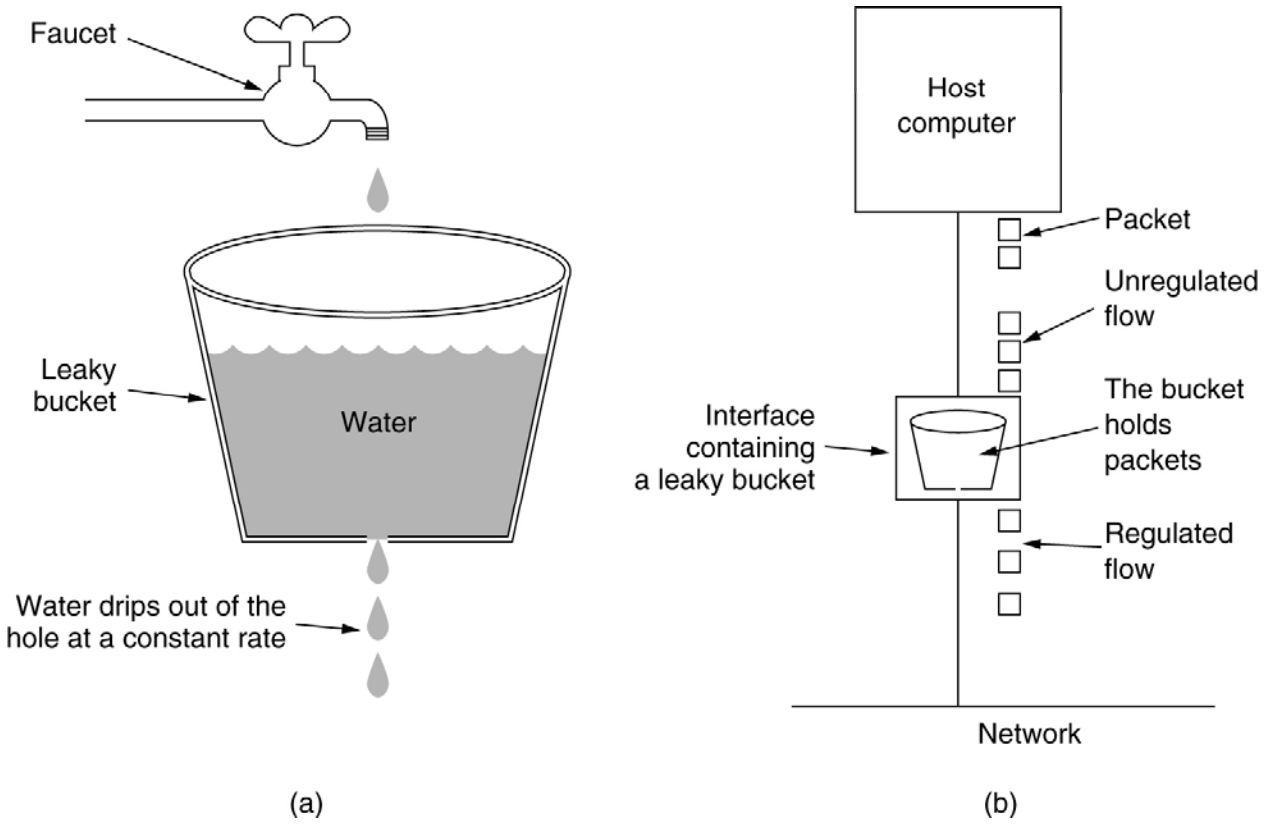
Traffic Shaping

- Traffic shaping controls the *rate* at which packets are sent (not just how many)
- At connection set-up time, the sender and carrier negotiate a traffic pattern (shape)
- Two traffic shaping algorithms are:
  - Leaky Bucket
  - Token Bucket

The Leaky Bucket Algorithm

- The **Leaky Bucket Algorithm** used to control rate in a network. It is implemented as a single-server queue with constant service time. If the bucket (buffer) overflows then packets are discarded.

The Leaky Bucket Algorithm

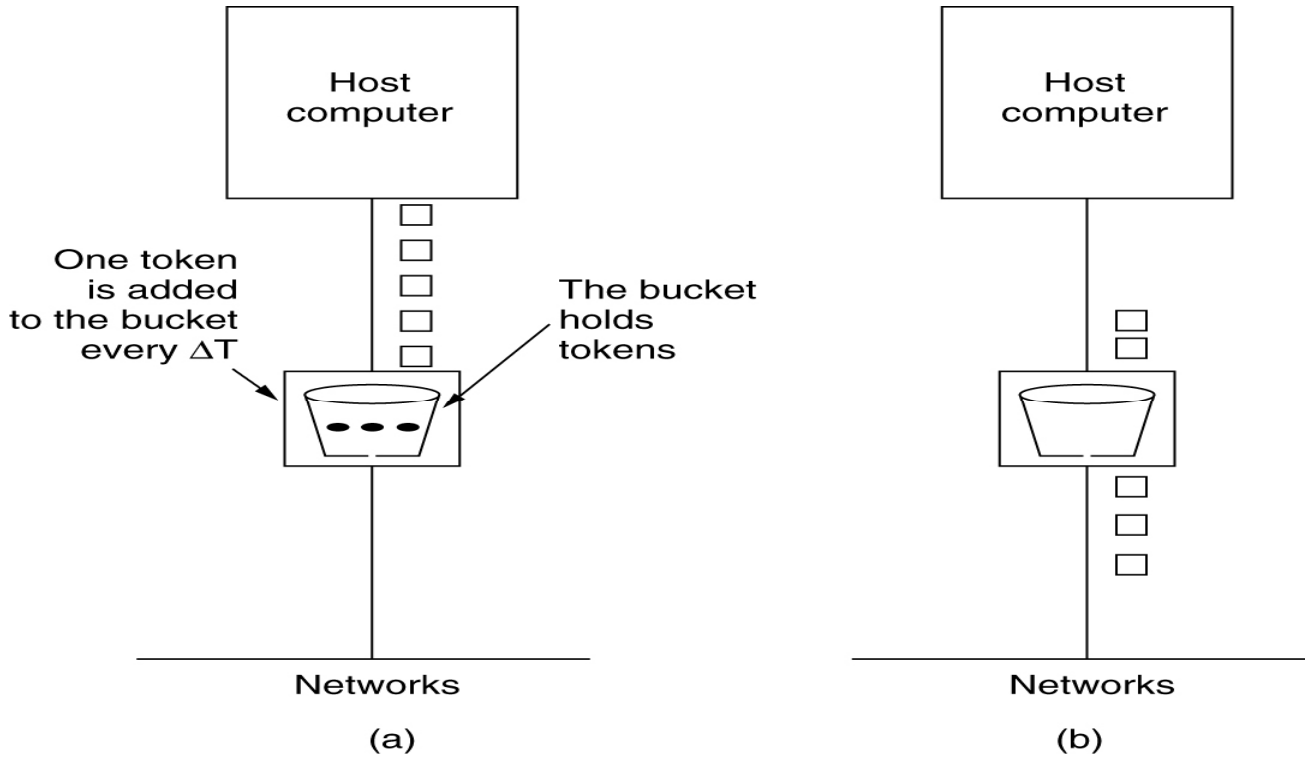


(a) A leaky bucket with water. (b) a leaky bucket with packets.

Leaky Bucket Algorithm (contd.)

- The leaky bucket enforces a constant output rate regardless of the burstiness of the input. Does nothing when input is idle.
- The host injects one packet per clock tick onto the network. This results in a uniform flow of packets, smoothing out bursts and reducing congestion.
- When packets are the same size (as in ATM cells), the one packet per tick is okay. For variable length packets though, it is better to allow a fixed number of bytes per tick.

Token Bucket Algorithm (contd.)



(a) Before (b) After

Token bucket operation

- TB accumulates fixed size tokens in a token bucket
- Transmits a packet (from data buffer, if any are there) or arriving packet if the sum of the token sizes in the bucket add up to packet size
- More tokens are periodically added to the bucket (at rate Δt). If tokens are to be added when the bucket is full, they are discarded

Token bucket properties

- Does not bound the peak rate of small bursts, because bucket may contain enough token to cover a complete burst size

- Performance depends only on the sum of the data buffer size and the token bucket size

Leaky Bucket vs Token Bucket

- LB discards packets; TB does not. TB discards tokens.
- With TB, a packet can only be transmitted if there are enough tokens to cover its length in bytes.
- LB sends packets at an average rate. TB allows for large bursts to be sent faster by speeding up the output.
- TB allows saving up tokens (permissions) to send large bursts. LB does not allow saving.

Tool administrasi traffic shaping

- Untuk mengelola dan memaintain traffic digunakan tool administrasi yang disediakan dalam bentuk perintah 'tc'.
- tc merupakan tool yang berasal dari paket software 'iproute' atau 'iproute2'

tc - show / manipulate traffic control settings

SYNOPSIS

tc qdisc [add | change | replace | link] dev DEV [parent qdisc-id |root] [handle qdisc-id] qdisc [qdisc specific parameters]

tc class [add | change | replace] dev DEV parent qdisc-id [classid class-id] qdisc [qdisc specific parameters]

tc filter [add | change | replace] dev DEV [parent qdisc-id | root] protocol protocol prio priority filtertype [filtertype specific parameters] flowid flow-id

tc [-s | -d] qdisc show [dev DEV]

tc [-s | -d] class show dev DEV

tc filter show dev DEV

Aplikasi tc dan qdisc

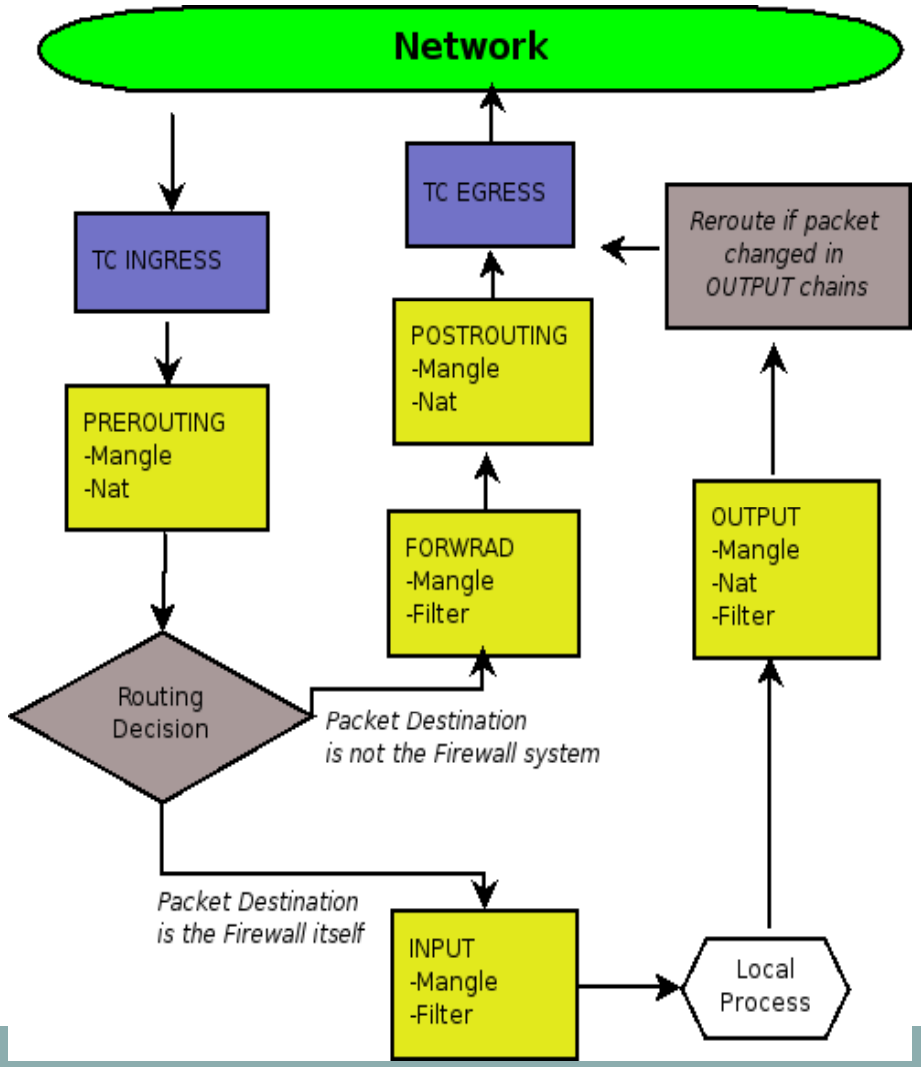
```
tc qdisc del dev eth1 root
tc qdisc add dev eth1 root handle 1 cbq bandwidth 100Mbit avpkt 1000 cell 8
tc class change dev eth1 root cbq weight 10Mbit allot 1514

tc qdisc del dev eth2 root
tc qdisc add dev eth2 root handle 1 cbq bandwidth 100Mbit avpkt 1000 cell 8
tc class change dev eth2 root cbq weight 10Mbit allot 1514

tc class add dev eth1 parent 1: classid 1:28 cbq bandwidth 100Mbit rate 64Kbit w
eight 6Kbit prio 5 allot 1514 cell 8 maxburst 20 avpkt 1000 bounded
tc filter add dev eth1 parent 1:0 protocol ip prio 100 u32 match ip dst 192.168.
10.2/24 classid 1:28

tc class add dev eth2 parent 1: classid 1:40 cbq bandwidth 100Mbit rate 128Kbit
weight 12Kbit prio 5 allot 1514 cell 8 maxburst 20 avpkt 1000 bounded
tc filter add dev eth2 parent 1:0 protocol ip prio 100 u32 match ip dst 192.168.
20.0/24 classid 1:40
```

Diagram hubungan netfilter dan TC



Spesifikasi Bandwidths / rates

kbps :Kilobytes per second

mbps :Megabytes per second

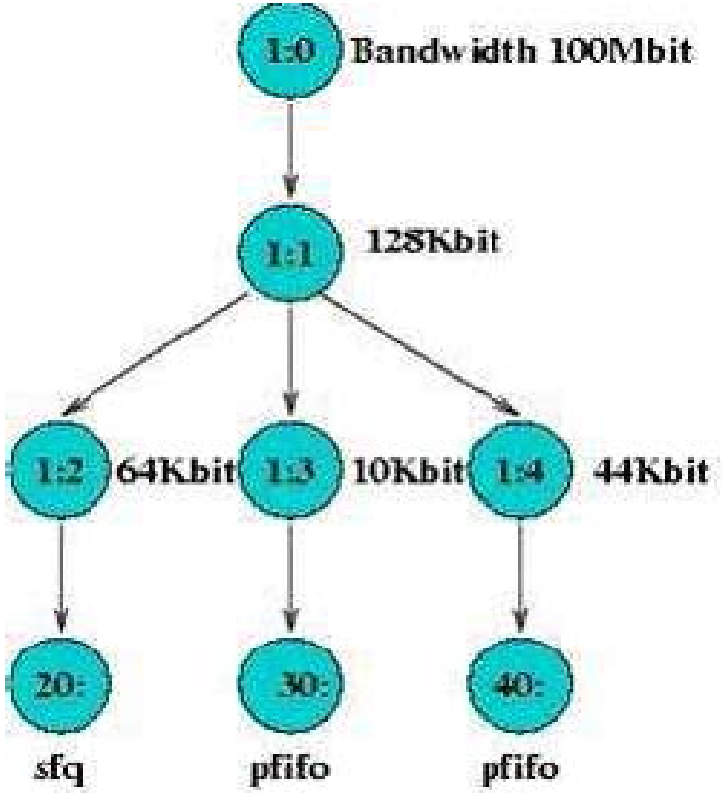
kbit :Kilobits per second

mbit :Megabits per second

BANDWIDTH MANAGEMENT MENGGUNANAN QOS

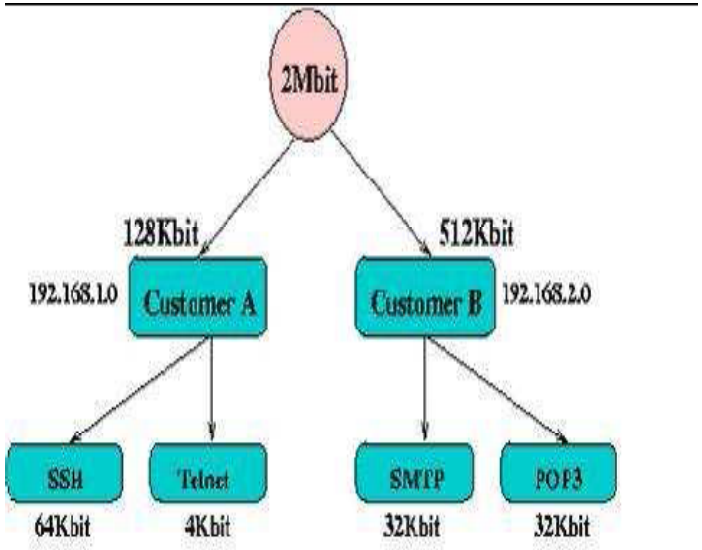
CBQ

- Teknik klasifikasi paket data yang paling terkenal adalah CBQ, mudah dikonfigurasi, memungkinkan sharing bandwidth
- antar kelas (class) dan memiliki fasilitas user interface. CBQ mengatur pemakaian bandwidth jaringan yang dialokasikan
- untuk tiap user, pemakaian bandwidth yang melebihi nilai set akan dipotong (shaping), cbq juga dapat diatur untuk sharing
- dan meminjam bandwidth antar class jika diperlukan.

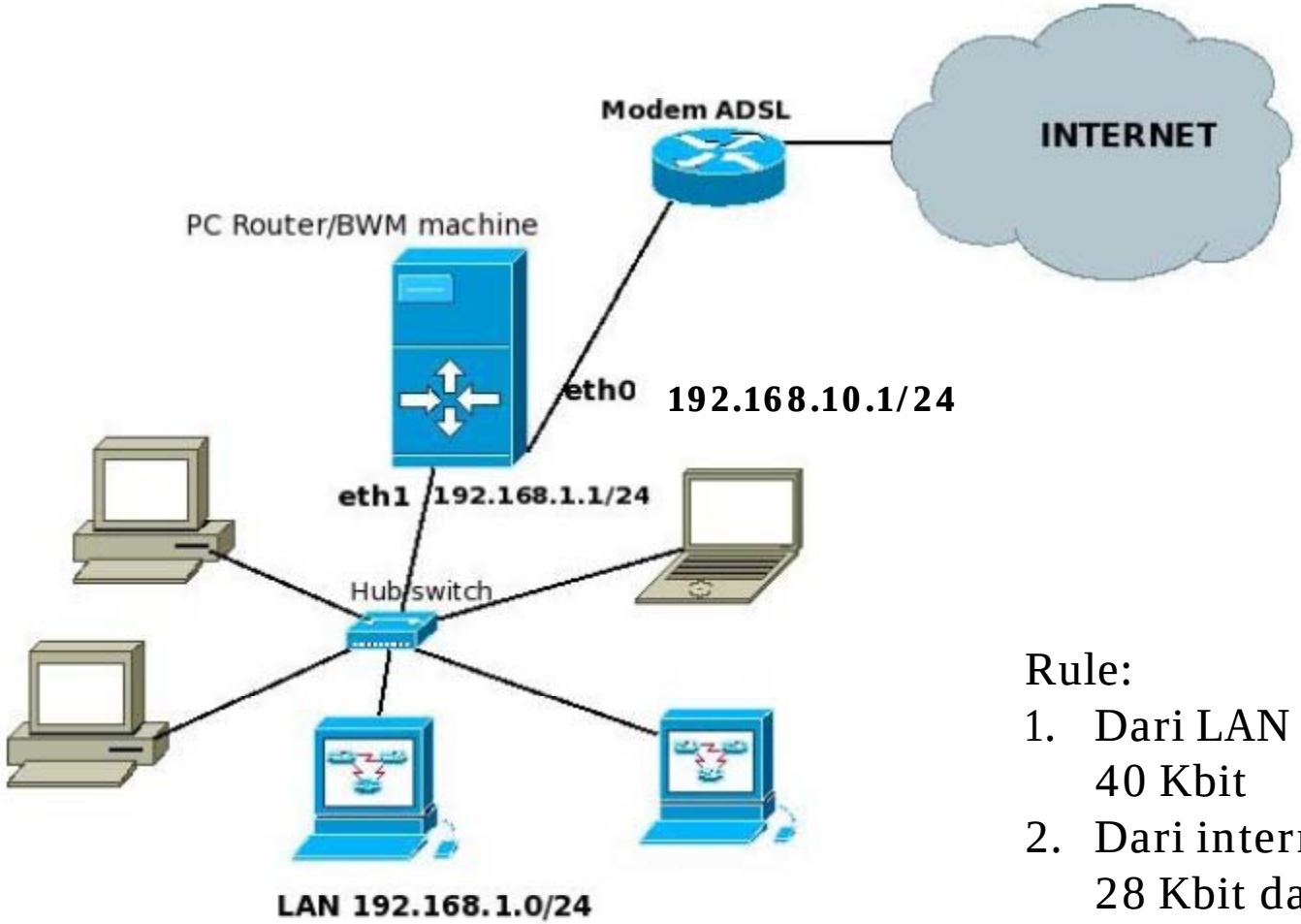


HTB

- Teknik antrian HTB mirip dengan CBQ hanya perbedaannya terletak pada opsi, HTB lebih sedikit opsi saat konfigurasi
- serta lebih presisi. Teknik antrian HTB memberikan kita fasilitas pembatasan trafik pada setiap level maupun klasifikasi,
- bandwidth yang tidak terpakai bisa digunakan oleh klasifikasi yang lebih rendah. Kita juga dapat melihat HTB seperti suatu
- struktur organisasi dimana pada setiap bagian memiliki wewenang dan mampu membantu bagian lain yang memerlukan,
- teknik antrian HTB sangat cocok diterapkan pada perusahaan dengan banyak struktur organisasi.



Network Design



Rule:

1. Dari LAN ke internet : 40 Kbit
2. Dari internet ke LAN : 28 Kbit dan email 100Kbit

Contoh aplikasi CBQ

/etc/sysconfig/cbq/cbq-028.internet-client

```
DEVICE=eth1,100Mbit,10Mbit
RATE=28Kbit
WEIGHT=2Kbit
PRIO=5
RULE=192.168.1.0/24
```

/etc/sysconfig/cbq/cbq-040.client-internet

```
DEVICE=eth0,100Mbit,10Mbit
RATE=40Kbit
WEIGHT=4Kbit
PRIO=5
RULE=192.168.10.1

DEVICE=eth0,100Mbit,1Mbit
RATE=100Kbit
WEIGHT=10Kbit
PRIO=5
RULE=192.168.10.1:25
```

Parameter

DEVICE=<ifname>,<bandwidth>[,<weight>] wajib
DEVICE=eth0,10Mbit,1Mbit

<ifname> nama dari interface yang akan di kontrol, misalnya, eth0
<bandwidth> bandwidth fisik dari device
misalnya 10Mbps atau 100Mbps
<weight> parameter tuning, harus proportional dengan <bandwidth>.
Biasanya digunakan aturan <weight> = <bandwidth> / 10

RATE=<speed> wajib
RATE=5Mbit

Alokasi bandwidth ke sebuah class.

WEIGHT=<speed> wajib
WEIGHT=500Kbit

Parameter tuning yang harus propoporsional dengan RATE.
Aturannya, WEIGHT ~= RATE/10.

Parameter

PRIO=<1-8> optional, default 5
PRIO=5

Prioritas dari class traffic. Semakin besar nomor, semakin kecil prioritas. Prioritas 5 sudah cukup.

LEAF=none|tbf|sfq optional, default "tbf"

Memberitahukan script untuk menggunakan teknik antrian (queueing) di leaf tertentu untuk sebuah kelas CBQ. Default, akan menggunakan TBF (Tocken Bucket Filter). Bila TBF digunakan, maka akan tidak mengijinkan kelas tersebut untuk meminjam bandwidth. Untuk mengijinkan sebuah kelas untuk meminjam bandwidth maka harus menset LEAF menjadi “none” atau “sfq”. SFQ = Stochastic Fairness Queuing

RULE=[[saddr[/prefix]][:port[/mask]],][daddr[/prefix]][:port[/mask]]

Parameter ini akan membuat “u32” filter yang akan memilih traffic untuk setiap class. Dapat digunakan multiple RULE per config.
Contoh:
RULE=10.1.1.0/24:80
Pilih trafik menuju port 80 di jaringan 10.1.1.0

RULE=10.2.2.5
Pilih trafik menuju ke semua port pada sebuah mesin 10.2.2.5

Contoh Aplikasi HTB

```
eth1-qos.cfg
# DOWNLOAD
class LAN_1 {
        bandwidth 256;      # garansi bandwidth yg dialokasikan untuk LAN
        limit 256;          # maksimal bandwidth yang bisa dicapai untuk LAN
        burst 2;
        priority 1;
        que sfq;
        client pc1 {
                bandwidth 128; # garansi bandwidth yang di alokasikan untuk pc1
                limit 192;     # bandwidth maksimal yg bisa di capai untuk pc1
                burst 2;
                priority 1;
                dst {
                        192.168.1.2/32;
                };
        };
        client pc2 {
                ....
        };
};
```

Contoh Aplikasi HTB

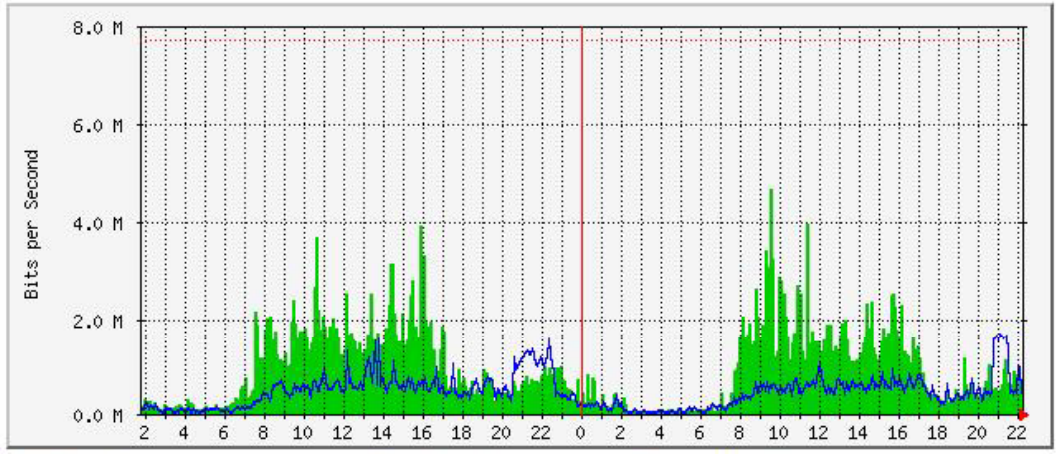
```
eth1-qos.cfg
# UPLOAD
class LAN_1 {
        bandwidth 256;      # garansi bandwidth yg dialokasikan untuk LAN
        limit 256;          # maksimal bandwidth yang bisa dicapai untuk LAN
        burst 2;
        priority 1;
        que sfq;
        client pc1 {
                bandwidth 128; # garansi bandwidth yang di alokasikan untuk pc1
                limit 192;     # bandwidth maksimal yg bisa di capai untuk pc1
                burst 2;
                priority 1;
                src {
                        192.168.10.1/32;
                };
        };
};
```

BURST: mengatur jumlah data yang akan dikirim dari satu class pada maksimum kecepatan hardware sebelum berusaha memberikan servis ke class yang lain

Monitoring Traffik Jaringan

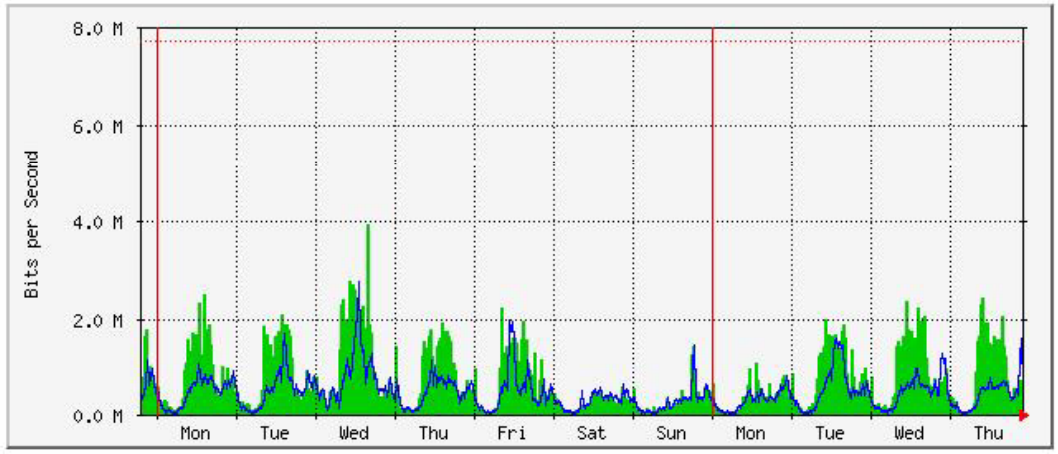
MRTG Example

`Daily' Graph (5 Minute Average)

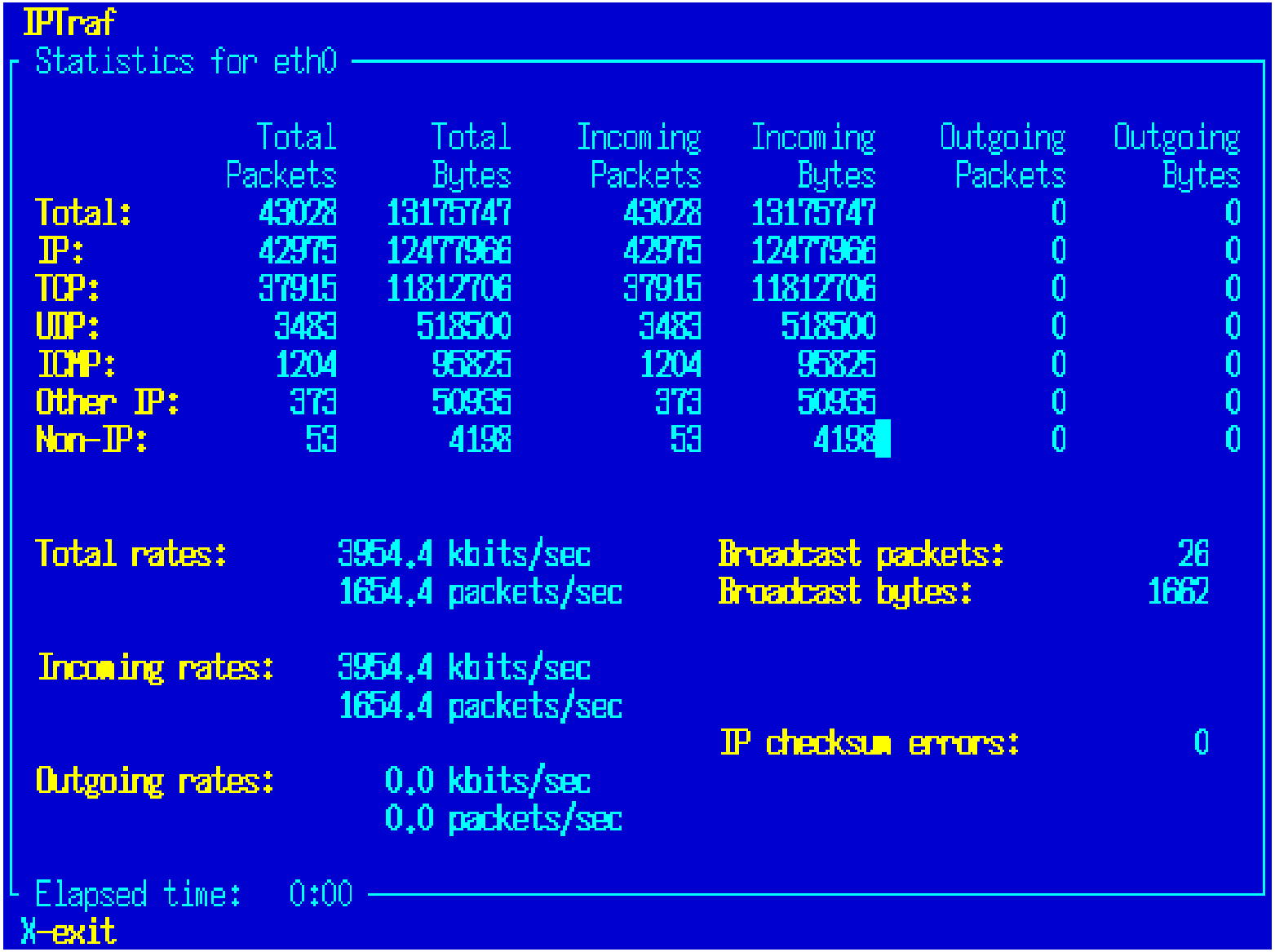


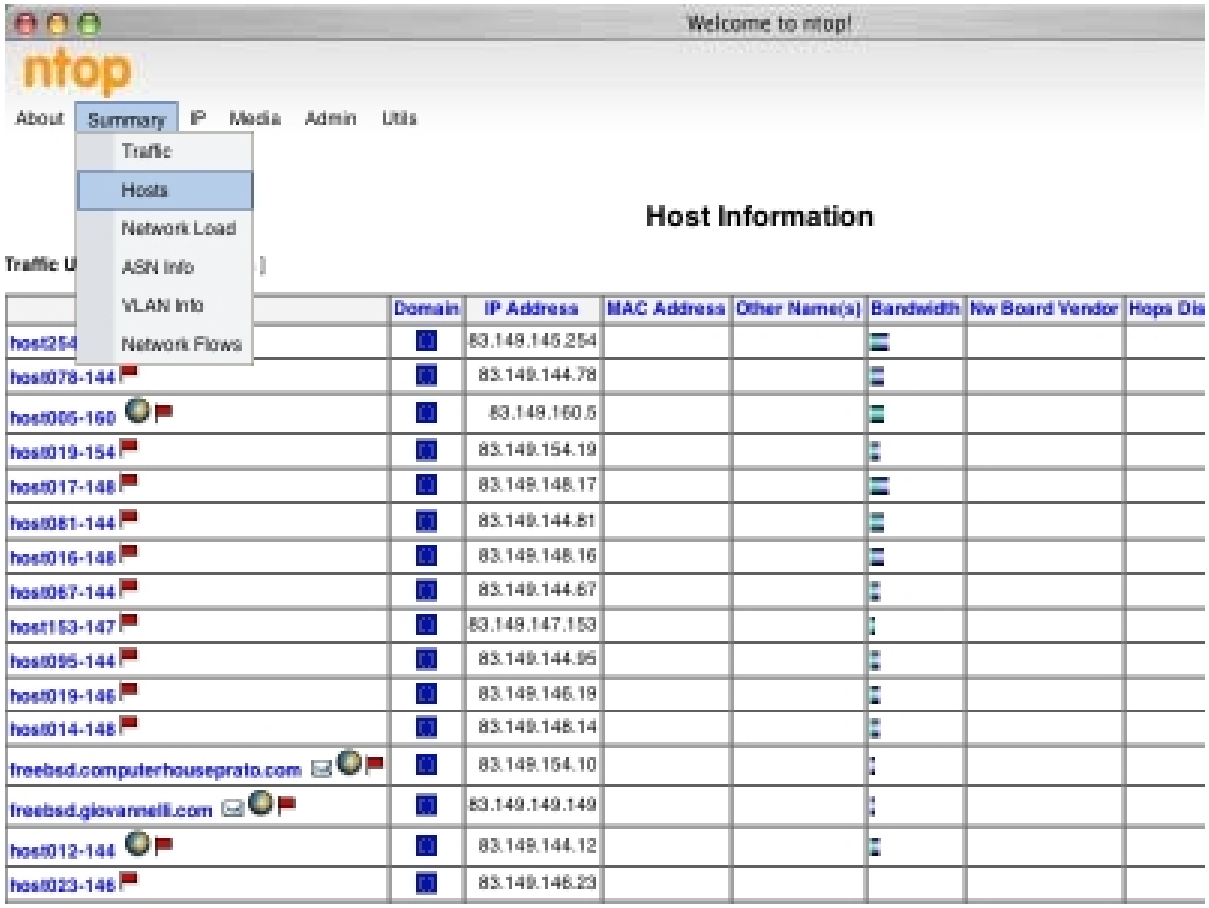
Max In:4679.7 kb/s (60.6%) Average In:917.1 kb/s (11.9%) Current In:455.3 kb/s (5.9%)
Max Out:1672.4 kb/s (21.7%) Average Out: 465.4 kb/s (6.0%) Current Out:547.1 kb/s (7.1%)

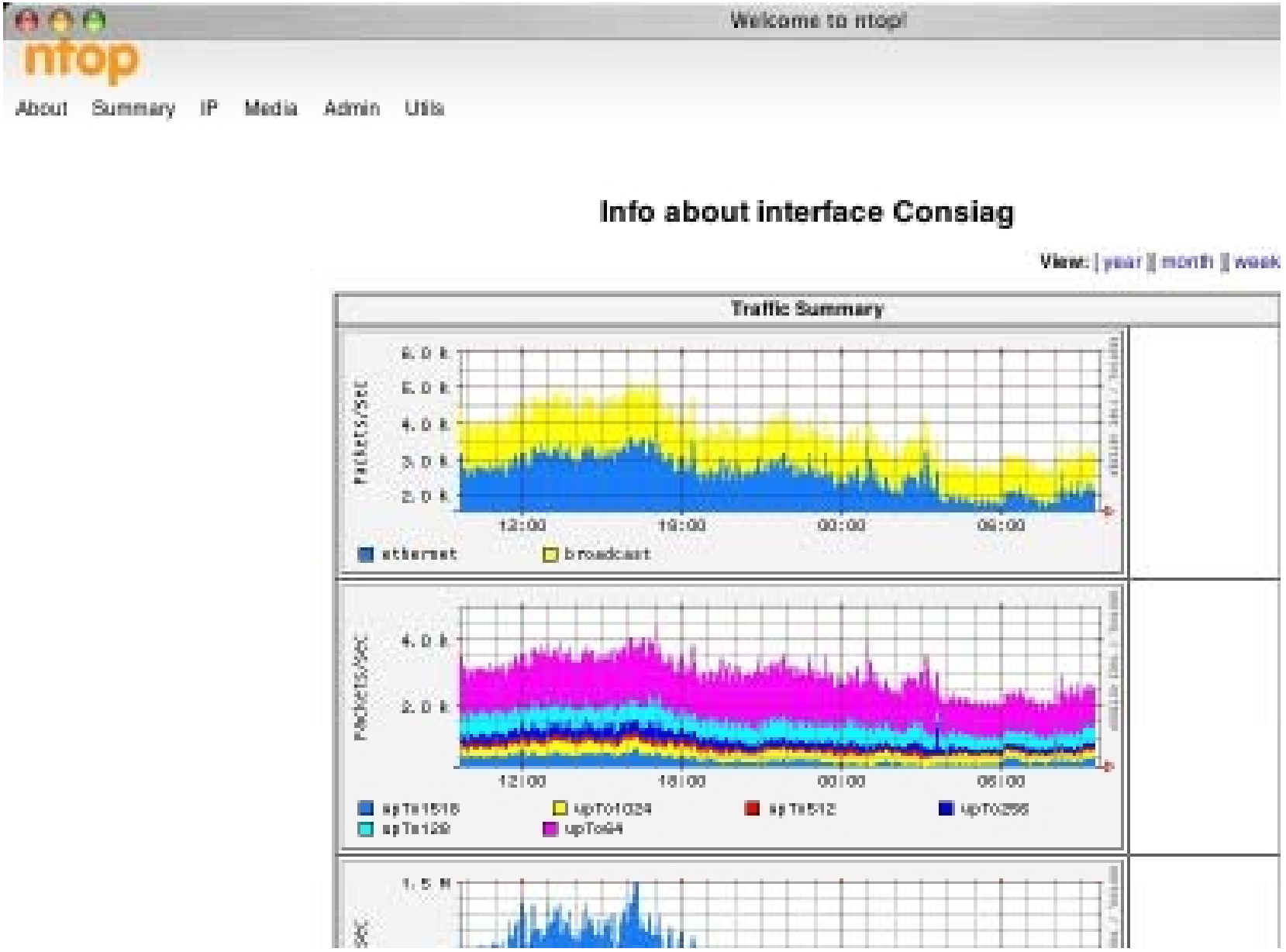
`Weekly' Graph (30 Minute Average)



Max In:3925.6 kb/s (50.8%) Average In:734.2 kb/s (9.5%) Current In:670.7 kb/s (8.7%)
Max Out:2728.5 kb/s (35.3%) Average Out:464.0 kb/s (6.0%) Current Out:559.3 kb/s (7.2%)


IPTraf
Statistics for eth0
Total Packets
Total Bytes
Incoming Packets
Incoming Bytes
Outgoing Packets
Outgoing Bytes
Total: 43028 13175747 43028 13175747 0 0
IP: 42975 12477966 42975 12477966 0 0
TCP: 37915 11812706 37915 11812706 0 0
UDP: 3483 518500 3483 518500 0 0
ICMP: 1204 95825 1204 95825 0 0
Other IP: 373 50935 373 50935 0 0
Non-IP: 53 4198 53 4198 0 0
Total rates: 3954.4 kbits/sec
1654.4 packets/sec
Broadcast packets: 26
Broadcast bytes: 1662
Incoming rates: 3954.4 kbits/sec
1654.4 packets/sec
IP checksum errors: 0
Outgoing rates: 0.0 kbits/sec
0.0 packets/sec
Elapsed time: 0:00
X-exit


Welcome to ntop!
ntop
About
Summary
IP
Media
Admin
Utils
Traffic
Hosts
Network Load
ASN Info
VLAN Info
Network Flows
Host Information
Domain
IP Address
MAC Address
Other Name(s)
Bandwidth
Nw Board Vendor
Hops Dis
host254
83.149.145.254
host078-144
83.149.144.78
host005-160
83.149.160.5
host019-154
83.149.154.19
host017-148
83.149.148.17
host081-144
83.149.144.81
host016-148
83.149.148.16
host067-144
83.149.144.67
host153-147
83.149.147.153
host095-144
83.149.144.95
host019-146
83.149.146.19
host014-148
83.149.148.14
freebsd.computerhouseprato.com
83.149.154.10
freebsd.giovannelli.com
83.149.149.149
host012-144
83.149.144.12
host023-146
83.149.146.23


Welcome to ntop!
ntop
About Summary IP Media Admin Utils
Info about interface Consiag
View: [year][month][week
Traffic Summary
Packets/Sec
6.0 k
5.0 k
4.0 k
3.0 k
2.0 k
12:00
18:00
00:00
06:00
ethernet
broadcast
Packets/Sec
4.0 k
2.0 k
12:00
18:00
00:00
06:00
upTo1024
upTo512
upTo256
upTo64
1.5 M