Mecanismos de QoS em Linux
tc – Traffic Control

Edgard Jamhour

Bibliografia

- A maior parte das figuras desta apresentação foi extraída do tutorial:

- http://www.opalsoft.net/qos/DS.htm

Arquitetura de Rede no Linux

- Um host com duas placas de rede, executando o sistema operacional linux pode ser transformado em um roteador, conforme a figura abaixo.

Figure 2.1.1

Figure 2.1.2

Implementação de QoS em Linux

- Os elementos que implementam o QoS no linux são os seguintes:

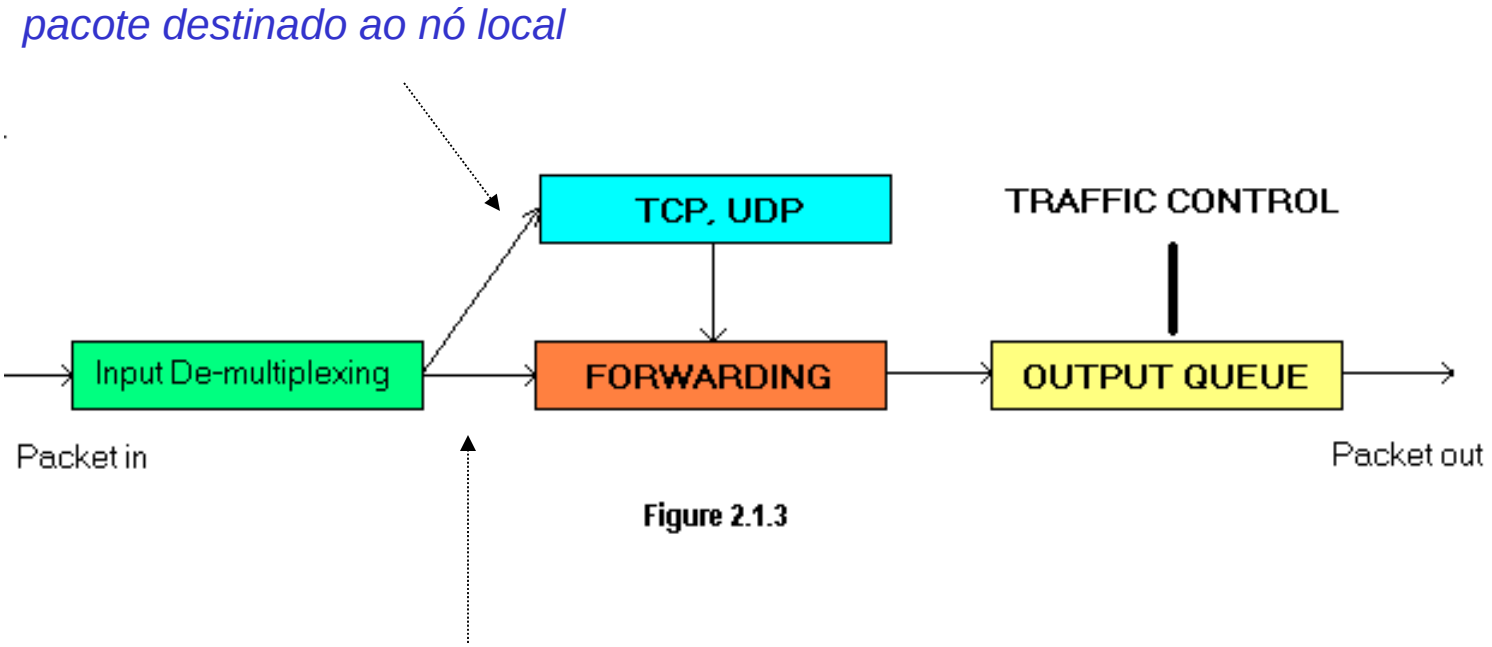
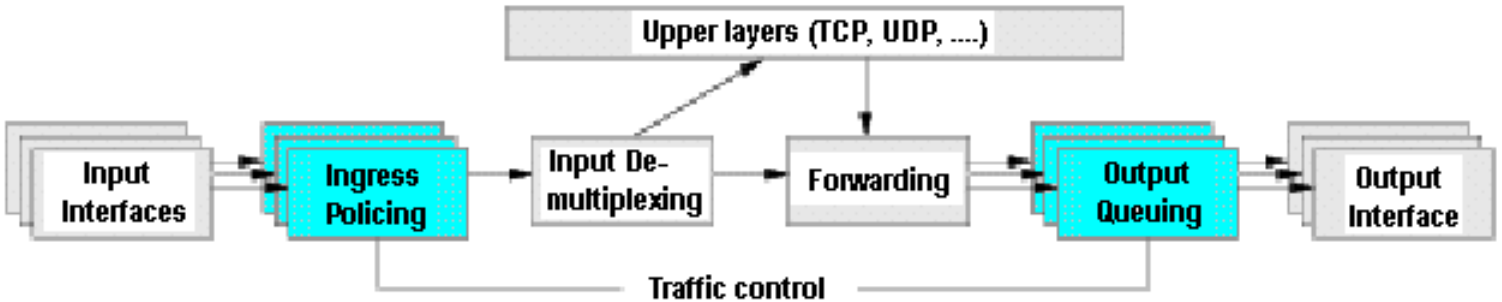


Figure 2.1.3

Controle de Tráfego

- O controle de tráfego é implementado através de dois mecanismos:
  - Pacotes são policiados na entrada
    - pacotes indesejáveis são descartados
  - Pacotes são enfileirados na respectiva interface de saída
    - pacotes podem ser atrasados, descartados ou priorizados



Processing of network data

Figure 2.1.4

Elementos do Controle de Tráfego

- O controle de tráfego é implementado internamente por 4 tipos de componentes:
  - Queuing Disciplines = qdisc
    - algoritmos que controlam o enfileiramento e envio de pacotes.
    - e.g. FIFO
  - Classes
    - representam “entidades de classificação de pacotes”.
    - cada classe pode estar associada a uma qdisc
  - Filters
    - utilizados para classificar os pacotes e atribuí-los as classes.
  - Policers
    - utilizados para evitar que o tráfego associado a cada filtro ultrapasse limites pré-definidos

Exemplo

- A Qdisc principal é obrigatória. Ela controla como os pacotes são recebidos e enviados pela interface.
- As Qdisc associadas as classes controlam apenas os pacotes da classe.

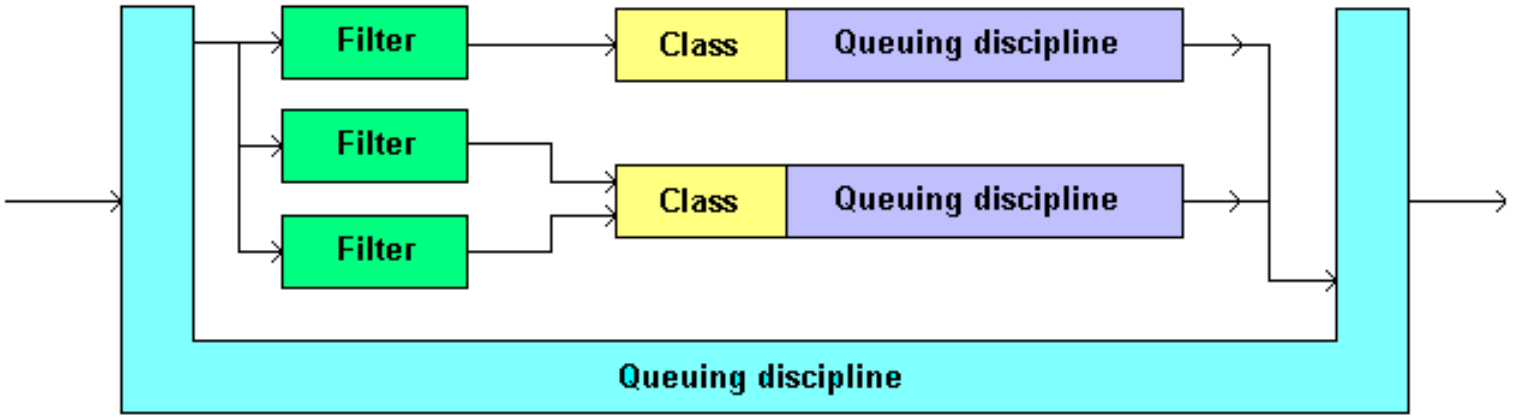


Figure 2.1.7

Comandos: Criar a qdisc principal

- 1) Cria a qdisc principal chamada 1:0
  - handle é o identificador da qdisc.
    - O id de qdisc sempre termina em :0
  - htb é o algoritmo utilizado pela qdisc.
    - no caso, o algoritmo não exige nenhum parâmetro obrigatório

```
> tc qdisc add dev eth0
  root handle 1:0
  htb
```

Comandos: Criar as classes filhas

- 2) cria duas classes com taxas diferentes
  - As classes são filhas da qdisc principal
  - O htb, quando utilizado em uma classe existe parâmetros de taxa de transmissão

```
> tc class add dev eth0
  parent 1:0 classid 1:1
  htb rate 500Kbit
> tc class add dev eth0
  parent 1:0 classid 1:2
  htb rate 300Kbit
```

Comandos: Criar as qdisc das classes

- 3) cria as qdiscs de saída, associadas a cada classe:
  - Cada qdisc é filha das classe a ela associada
  - O sfq é o algoritmo escolhido para a qdisc
    - perturb é um parâmetro do algoritmo

```
> tc qdisc add dev eth0
  parent 1:1 handle 10:
  sfq perturb 10
> tc qdisc add dev eth0
  parent 1:2 handle 20:
  sfq perturb 10
```

Comandos: Criar os filtros

- 4) cria os filtros para as classes
  - Os filtros são filhos do qdisc principal
  - O tipo de filtro utilizado é u32

```
> tc filter add dev eth0
  parent 1:0
  protocol ip u32 match ip protocol 0x06 0xff
  flowid 1:1
> tc filter add dev eth0
  parent 1:0
  protocol ip u32 match ip protocol 0x11 0xff
  flowid 1:2
```

Comandos de monitoramento

- **iplink show**
  - mostra a classe default associada a interface
- **tc [-s] qdisc/class/filter show dev eth0**
  - mosta as qdisc/class/filter associadas a interface
  - –s mostra as estatísticas do uso da qdisc/class/filter
- **tc qdisc del root dev eth0**
  - limpa as regras de QoS

Queueing Disciplines

- Sem classe (sem classificação)
  - FIFO: First In First Out
  - SFQ: Stochastic Fair Queuing
  - TBF: Token Bucket Flow
  - DS_MARK: Diff-Serv Marker
  - RED: Random Early Detection
- Com classe (classificação e priorização)
  - PRIO: Priority Queue
  - [CBQ: Class-Based Queueing]
  - HTB: Substituto do CBQ

FIFO

- Cria uma fila com capacidade para 10 pacotes
- Os pacotes são processados na ordem de chegada

```
# tc qdisc add dev eth0 root pfifo limit 10
```

cm-01

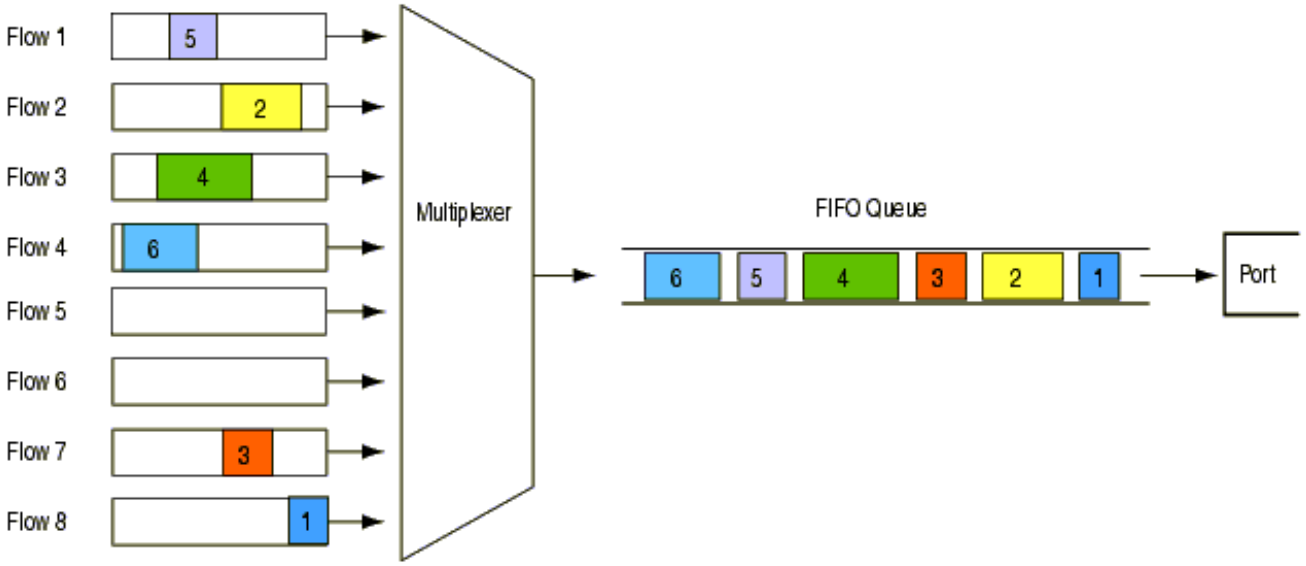


Figure 2.2.1

PRIO

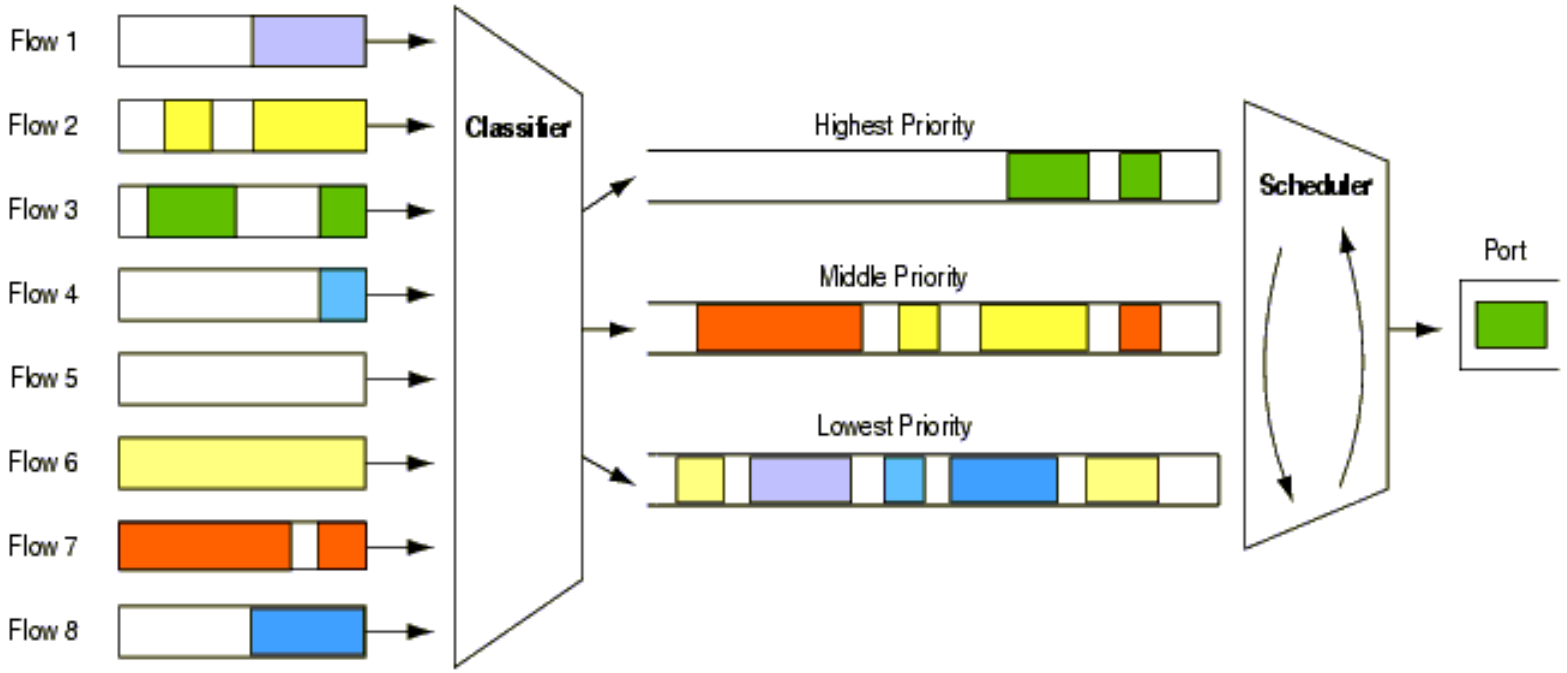


Figure 2.3.1

Classes PRIO são associadas a FIFO

```
# tc qdisc add dev eth0 root handle 1: prio
```

cm-05

- Cria automaticamente 3 classes denominadas: 1:1, 1:2 e 1:3
- Associa a qdisc PFIFO as classes
- Os filtros precisam ser adicionados posteriormente as classes.

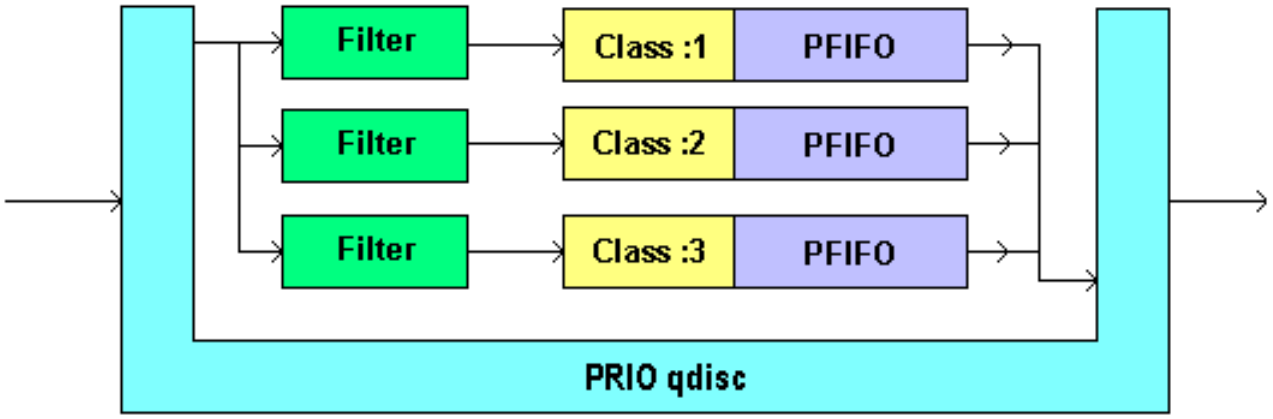


Figure 2.3.2

Exemplos de Filtros para as Classes PFIFO

- Os exemplos abaixo mostram como associar pacotes as classes baseando-se nos códigos de TOS (DSCP)

```
# tc filter add dev eth0 parent 1:0 prio 1 protocol ip u32 \
        match ip tos 0x28 0xff flowid 1:1
```

cm-06

```
# tc filter add dev eth0 parent 1:0 prio 2 protocol ip u32 \
        match ip tos 0x48 0xff flowid 1:2
# tc filter add dev eth0 parent 1:0 prio 3 protocol ip u32 \
        match ip tos 0x58 0xff flowid 1:3
```

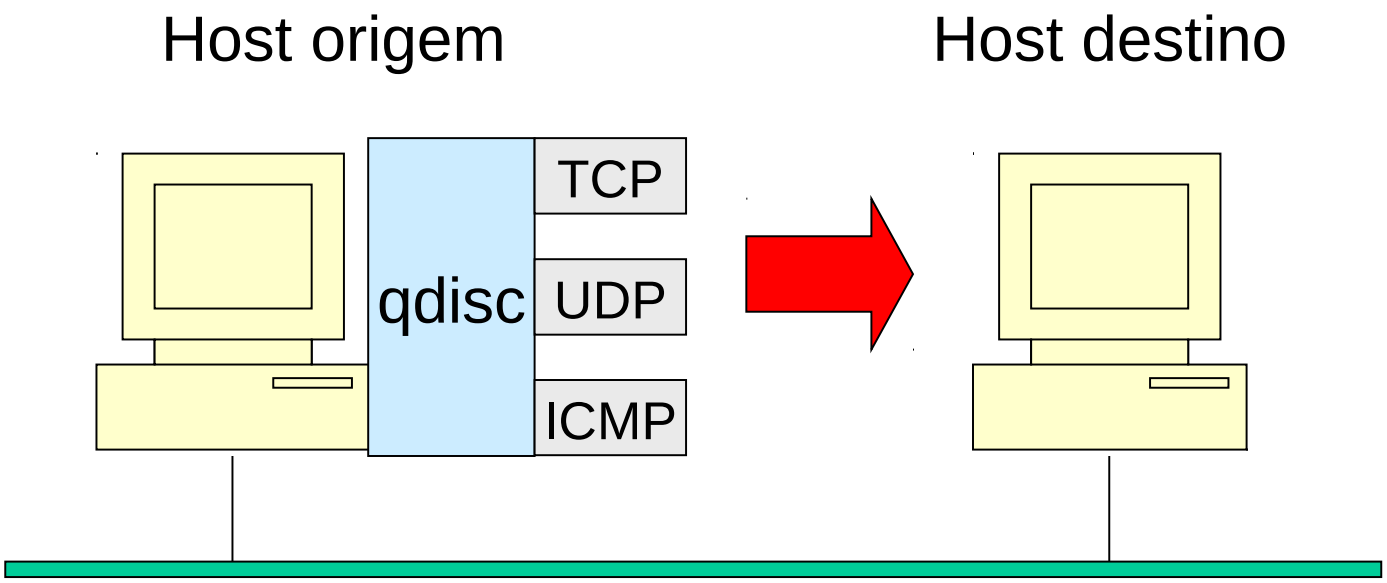
cm-07

Exercício 1

- Crie um script para classificar o tráfego recebido e enviado pelo seu computador, associando:
  - todo tráfego TCP na classe 1:1 (prioridade alta)
  - todo tráfego UDP na classe 1:2 (prioridade média)
  - todo tráfego ICMP na classe 1:3 (prioridade baixa)
- Após gerar tráfego com seu computador, verifique as estatísticas de uso dos elementos qdisc, classe e filter (se houverem).

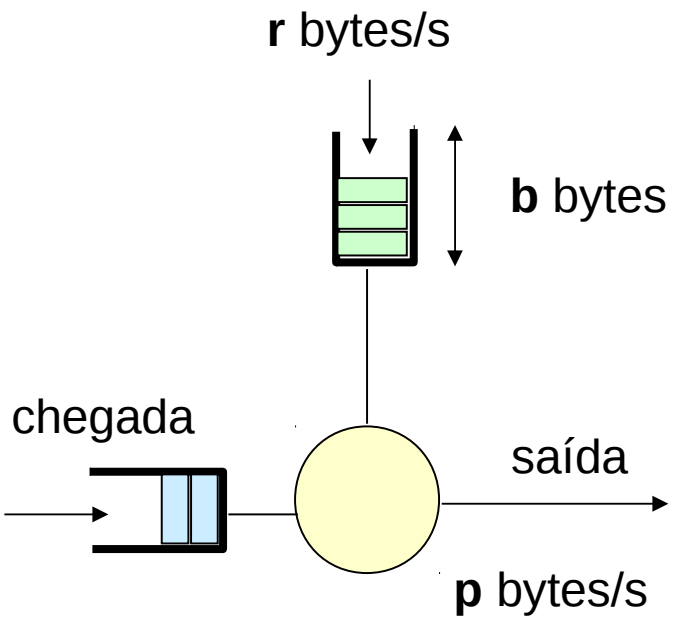
Observação

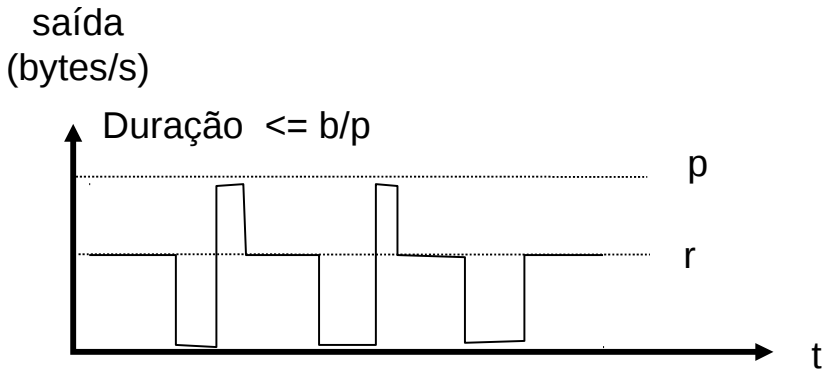
- Ao fazer os testes, lembre-se que qdisc influencia o tráfego de saída do computador e não o de entrada.
- Dessa forma, a medição deve ser feita no host de origem.



TBF: Token-Bucket Filter

- TBF é um algoritmo de condicionamento de tráfego (traffic-shaping)
- Ele permite limitar a banda associada a uma classe, associando uma taxa média e a possibilidade de envio de rajadas controladas.





Parâmetros

```
> tc qdisc add dev eth0
  parent 1:1
   tbf rate 0.5mbit burst 5k latency 70ms
   peakrate 1mbit minburst 1540
```

- Onde:
  - **rate:** taxa média transmitida
  - **burst:** tamanho do balde (em bytes)
  - **latency**: tempo máximo que um pacote pode ficar na fila aguardando o token
  - **peakrate**: taxa de pico de descarga do baldo
  - **minburst**: geralmente o mtu de um pacote

Exemplo

```
# tc qdisc add dev eth0 root handle 1: prio
# tc filter add dev eth0 parent 1:0 prio 1 protocol ip u32 \
          match ip tos 0x28 0xff flowid 1:1
# tc filter add dev eth0 parent 1:0 prio 2 protocol ip u32 \
          match ip tos 0x48 0xff flowid 1:2
# tc filter add dev eth0 parent 1:0 prio 3 protocol ip u32 \
          match ip tos 0x58 0xff flowid 1:3
# tc qdisc add dev eth0 parent 1:1 handle 10: tbf rate 0.5mbit burst 5kb \
           latency 70ms peakrate 1mbit minburst 1540
```

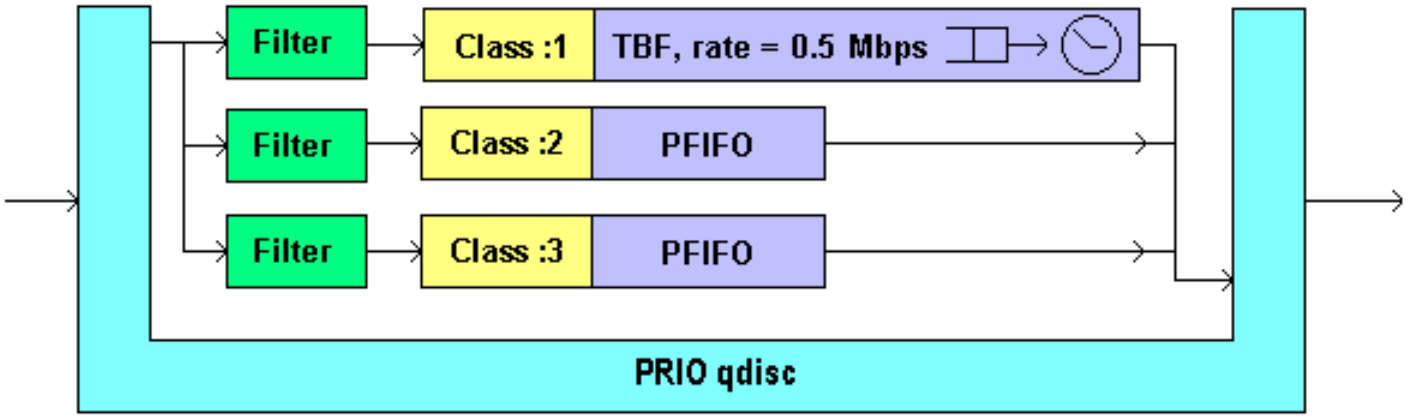


Figure 2.4.1

Exercício 2

- Altere o script do exercício 1, associando uma qdisc do tipo TBF a classe 1:1
  - **tbf** rate **0.5mbit** burst **5k** latency **70ms** peakrate **1mbit** minburst **1540**
- Crie os seguintes filtros:
  - Todo tráfego TCP é associado a classe 1:1
  - O tráfego enviado ao computador ao lado é associado a classe 1:2
  - O tráfego default é associado a classe 1:3
- Faça download de um arquivo grande em seu computador pela Internet e verifique as estatísticas associadas a classe.
- Efetue pings no computador enquanto você faz o download para avaliar o efeito do QoS.

SFQ: Stochastic Fair Queuing

- As filas são servidas um pacote de cada vez, utilizando a estratégia de round-robin

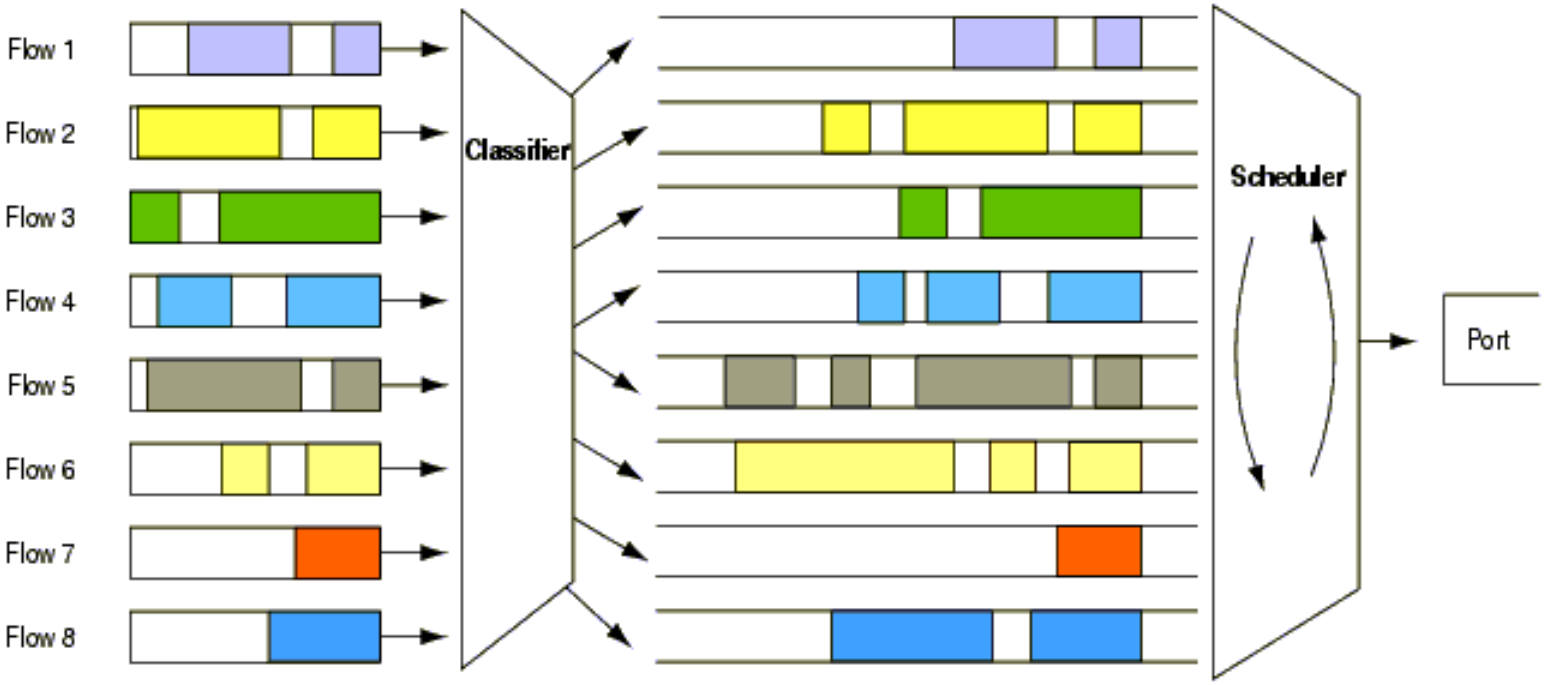


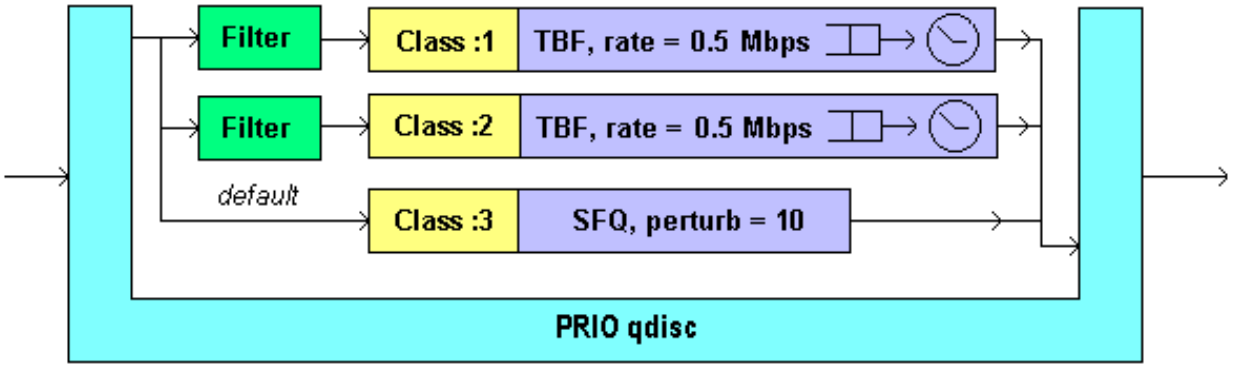
Figure 2.5.1

Parâmetros

- *perturb*
  - *Intervalo para reconfiguração de hashing.*
  - *Valor recomendado: 10s*
- *quantum*
  - *Quantidade de bytes removidos da fila por interação.*
  - *O valor default é 1 = maximum sized packet (MTU-sized).*

```
# tc qdisc add dev eth0 root sfq perturb 10
```

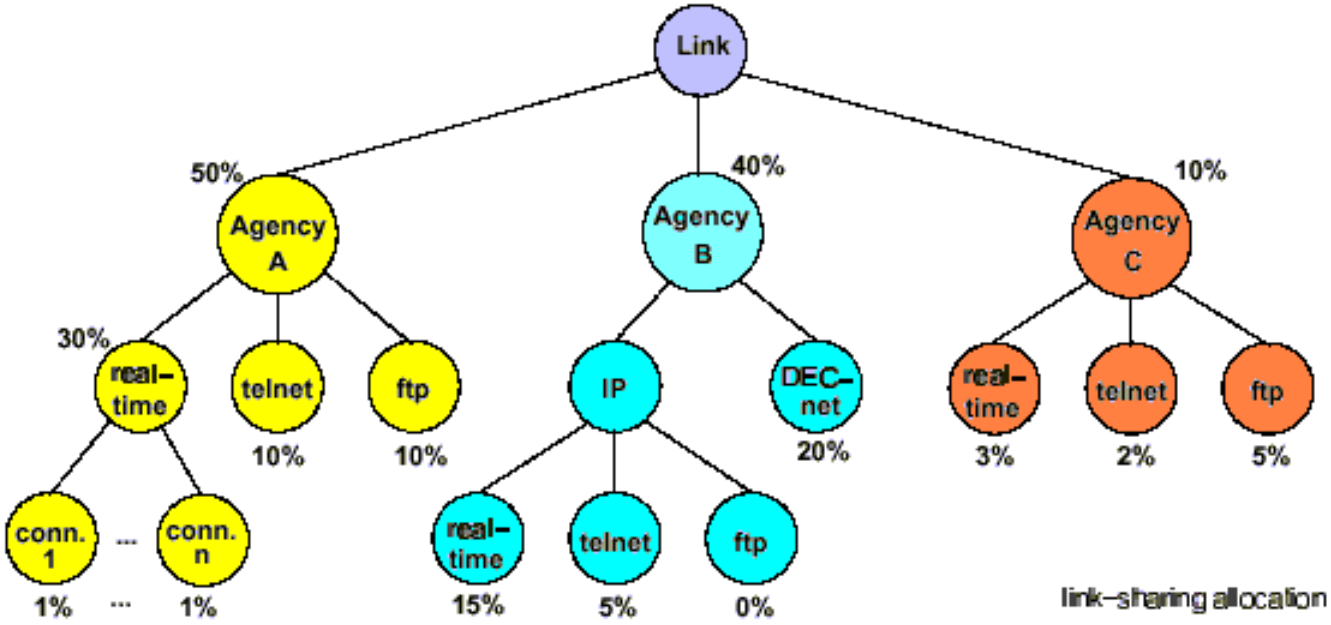
Exemplo




```
# tc qdisc add dev eth0 root handle 1: prio
# tc filter add dev eth0 parent 1:0 prio 1 protocol ip u32 \
          match ip tos 0x28 0xff flowid 1:1
# tc filter add dev eth0 parent 1:0 prio 2 protocol ip u32 \
          match ip tos 0x48 0xff flowid 1:2
# tc filter add dev eth0 parent 1:0 prio 3 protocol ip u32 \
           match ip src 0/0 flowid 1:3
# tc qdisc add dev eth0 parent 1:1 handle 10: tbf rate 0.5mbit burst 5kb \
            latency 70ms peakrate 1mbit minburst 1540
# tc qdisc add dev eth0 parent 1:2 handle 20: tbf rate 0.5mbit burst 5kb \
            latency 70ms peakrate 1mbit minburst 1540
# tc qdisc add dev eth0 parent 1:3 handle 30: sfq perturb 10
```

HTB

- Esse algoritmo é utilizado no lugar do CBQ, considerada muito complexa, para construir hierarquias de divisão de banda conforme a figura abaixo.



A hierarchical link-sharing structure.

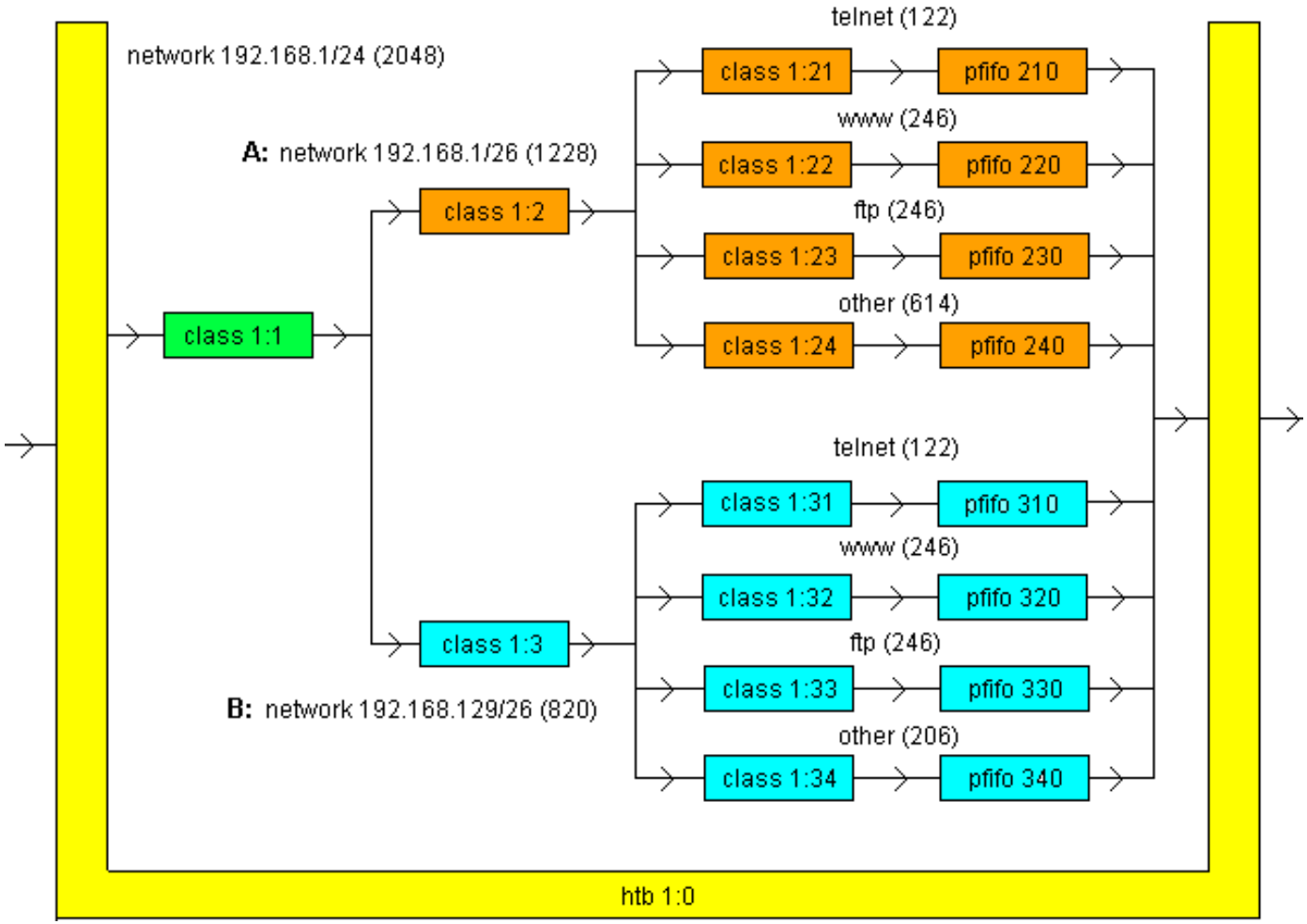
Figure 2.8.2

Parâmetros

- **rate**: taxa associada garantida para classe e suas filhas
- **ceil**: taxa máxima que pode ser emprestada da classe pai
- **burst**: quantidade máxima de bytes que pode ser enviada na taxa ceil
- **cburst**:quantidade máxima de bytes que pode ser enviada na taxa da interface (quando não houver limite imposto pela classe pai)
- **priority**: ordenamento das classes. As classes de maior prioridade recebem o excesso de banda primeiro, reduzindo sua latência (prio 0 é a maior)

```
> tc qdisc add dev eth0 root handle 1: htb
> tc class add dev eth0 parent 1:0  classid 1:1
  htb rate rate ceil rate burst bytes
  [ cburst bytes ] [ prio priority ]
```

Exemplo


network 192.168.1/24 (2048)
A: network 192.168.1/26 (1228)
class 1:1
class 1:2
class 1:3
B: network 192.168.129/26 (820)
telnet (122)
class 1:21
pfifo 210
www (246)
class 1:22
pfifo 220
ftp (246)
class 1:23
pfifo 230
other (614)
class 1:24
pfifo 240
telnet (122)
class 1:31
pfifo 310
www (246)
class 1:32
pfifo 320
ftp (246)
class 1:33
pfifo 330
other (206)
class 1:34
pfifo 340
htb 1:0

Criação da Hierarquia

```
#tc qdisc add dev eth0 root handle 1:0 htb
```

```
#tc class add dev eth0 parent 1:0 classid 1:1 htb rate 2048kbit
```

```
#tc class add dev eth0 parent 1:1 classid 1:2 htb \
      rate 1228kbit ceil 1228kbit
#tc class add dev eth0 parent 1:1 classid 1:3 htb \
      rate 820kbit ceil 820kbit
```

```
#tc class add dev eth0 parent 1:2 classid 1:21 htb \
      rate 122kbit ceil 1228kbit
#tc class add dev eth0 parent 1:2 classid 1:22 htb \
      rate 246kbit ceil 1228kbit
#tc class add dev eth0 parent 1:2 classid 1:23 htb \
      rate 246kbit ceil 1228kbit
#tc class add dev eth0 parent 1:2 classid 1:24 htb \
      rate 614kbit ceil 1228kbit
```

Criação da Qdisc de Saída

- A criação da Qdisc de saída é obrigatória

```
#tc qdisc add dev eth0 parent 1:21 handle 210: pfifo limit 10
#tc qdisc add dev eth0 parent 1:22 handle 220: pfifo limit 10
#tc qdisc add dev eth0 parent 1:23 handle 230: pfifo limit 10
#tc qdisc add dev eth0 parent 1:24 handle 240: pfifo limit 10
#tc qdisc add dev eth0 parent 1:31 handle 310: pfifo limit 10
#tc qdisc add dev eth0 parent 1:32 handle 320: pfifo limit 10
#tc qdisc add dev eth0 parent 1:33 handle 330: pfifo limit 10
#tc qdisc add dev eth0 parent 1:34 handle 340: pfifo limit 10
```

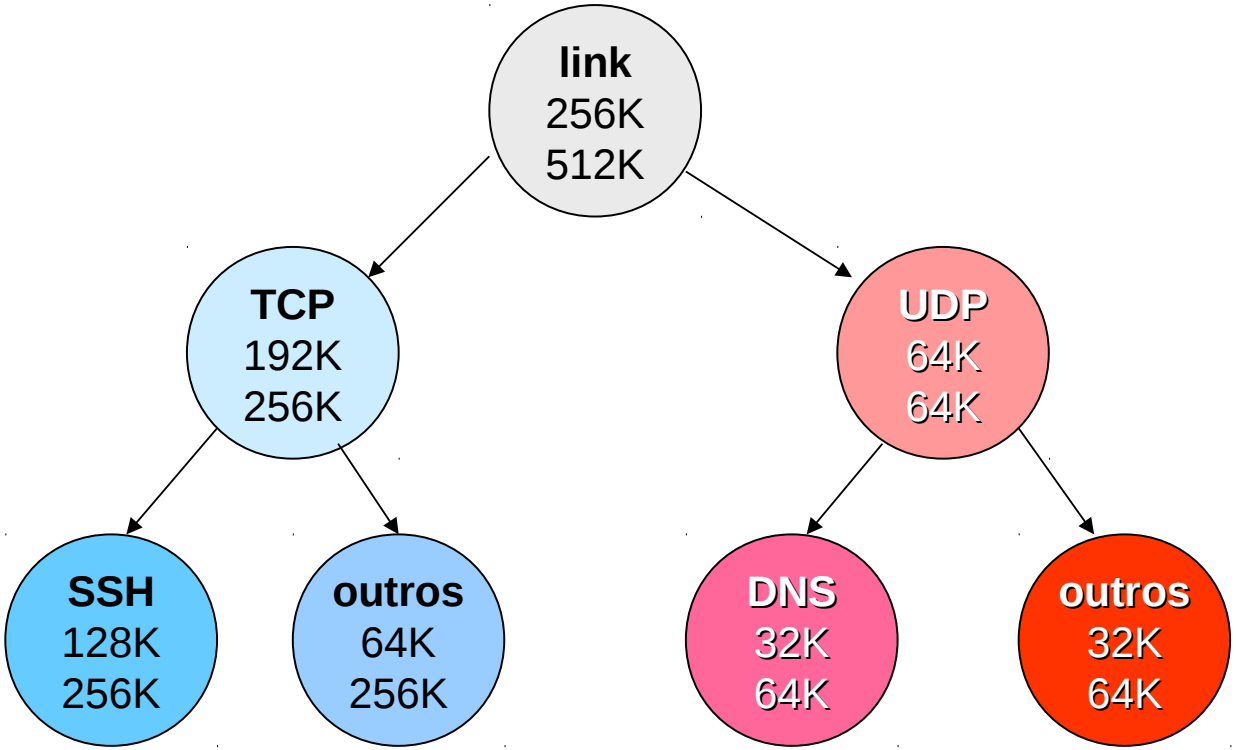
Criação dos Filtros

- Os filtros são atribuídos diretamente as classes filhas.
- A classe pai é utilizada apenas para definir os limites do empréstimo de banda compartilhada.

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip dst 192.168.1/26 match ip sport 23 0xfff flowid 1:21
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip dst 192.168.1/26 match ip sport 80 0xfff flowid 1:22
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip dst 192.168.1/26 match ip sport 20 0xfff flowid 1:23
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip dst 192.168.1/26 match ip sport 21 0xfff flowid 1:23
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip dst 192.168.1/26 flowid 1:24
```

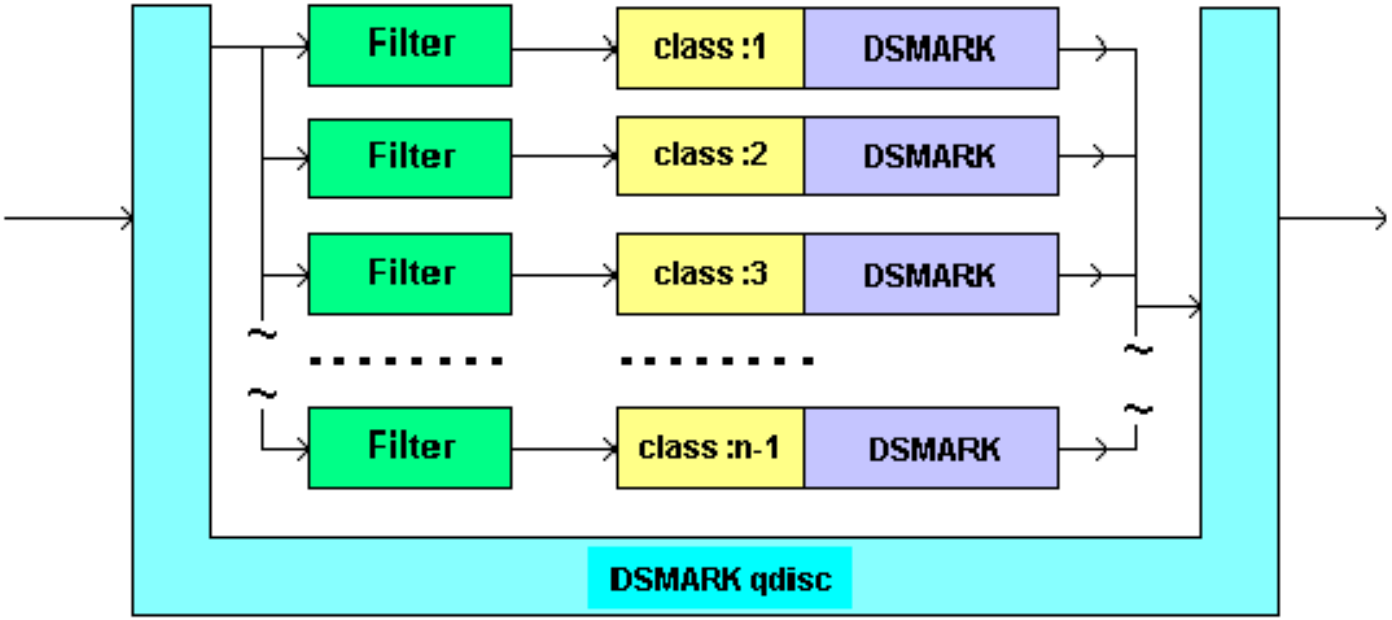
Exercício 3

- Crie um script com a hierarquia para o seu computador, adotando o desenho abaixo. Depois avalie as estatísticas da classe:



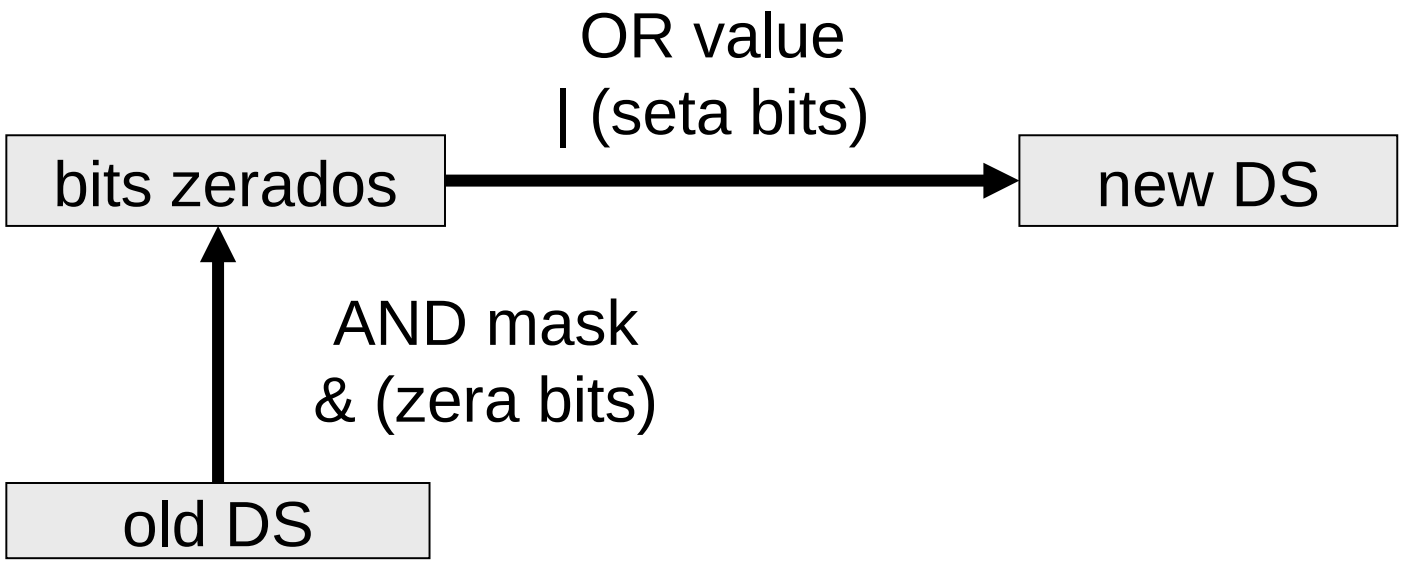
DSMARK

- A disciplina DSMARK é utilizado para fazer a marcação ou remarcação de bits do campo DS.



DSMARK

- A marcação é baseada na seguinte equação:
  - new_DS = (old_DS & mask) | value



Exemplo: Criação das classes

```
#tc add qdisc dev eth0 handle <hd> root dsmark \
      indices <id> [ default_index <did> ] [ set_tc_index ]
```

cm-50

```
#tc qdisc add dev eth0 handle 1:0 root dsmark indices 8

#tc class change dev eth0 classid 1:1 dsmark mask 0x0 value 0xb8
#tc class change dev eth0 classid 1:2 dsmark mask 0x3 value 0x58
#tc class change dev eth0 classid 1:3 dsmark mask 0xe3 value 0x10
#tc class change dev eth0 classid 1:4 dsmark mask 0x1f value 0x60
#tc class change dev eth0 classid 1:5 dsmark mask 0x0 value 0x30
#tc class change dev eth0 classid 1:6 dsmark mask 0x3 value 0x70
#tc class change dev eth0 classid 1:7 dsmark mask 0x0 value 0x0
```

cm-53

Exemplo: Criação de Classes

```
#tc qdisc add dev eth0 parent 1:1 handle 2:0 dsmark \
      indices 8 default_index 7
```

cm-52

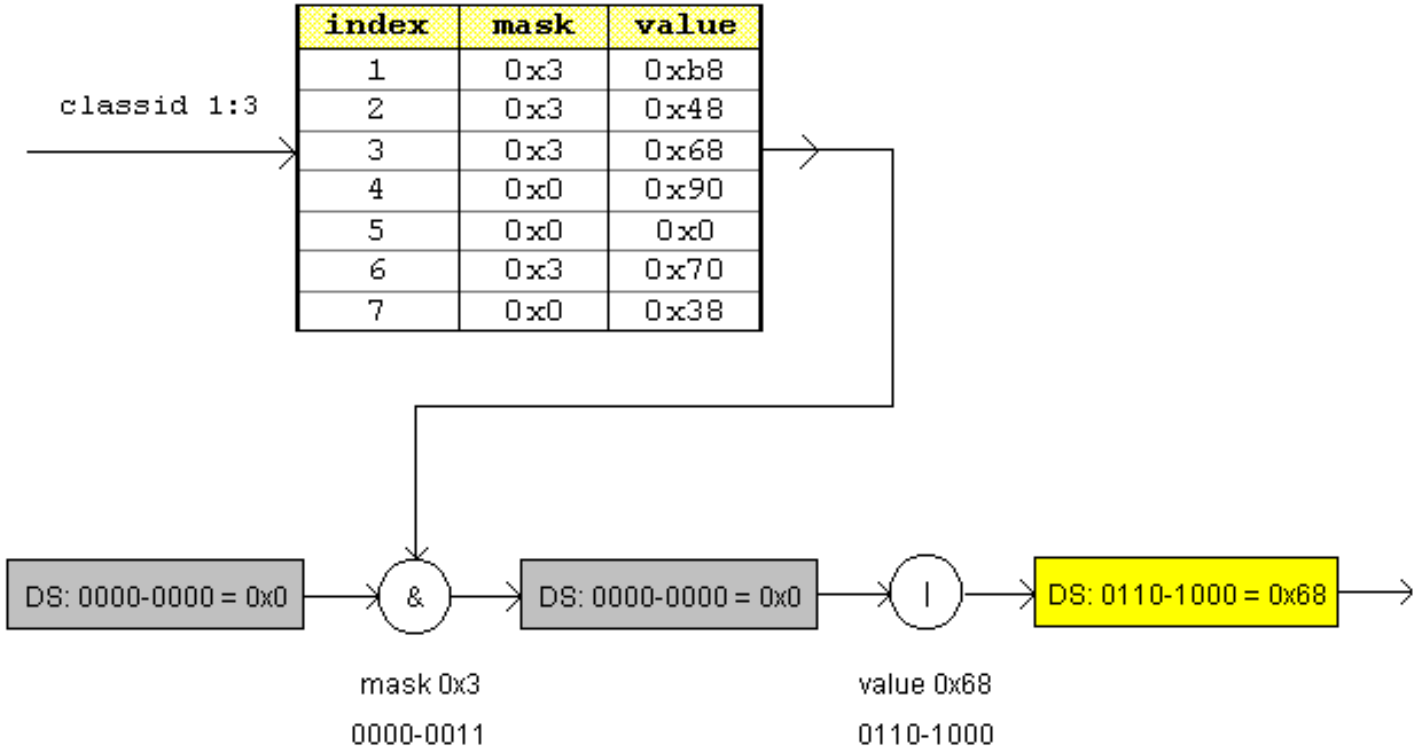
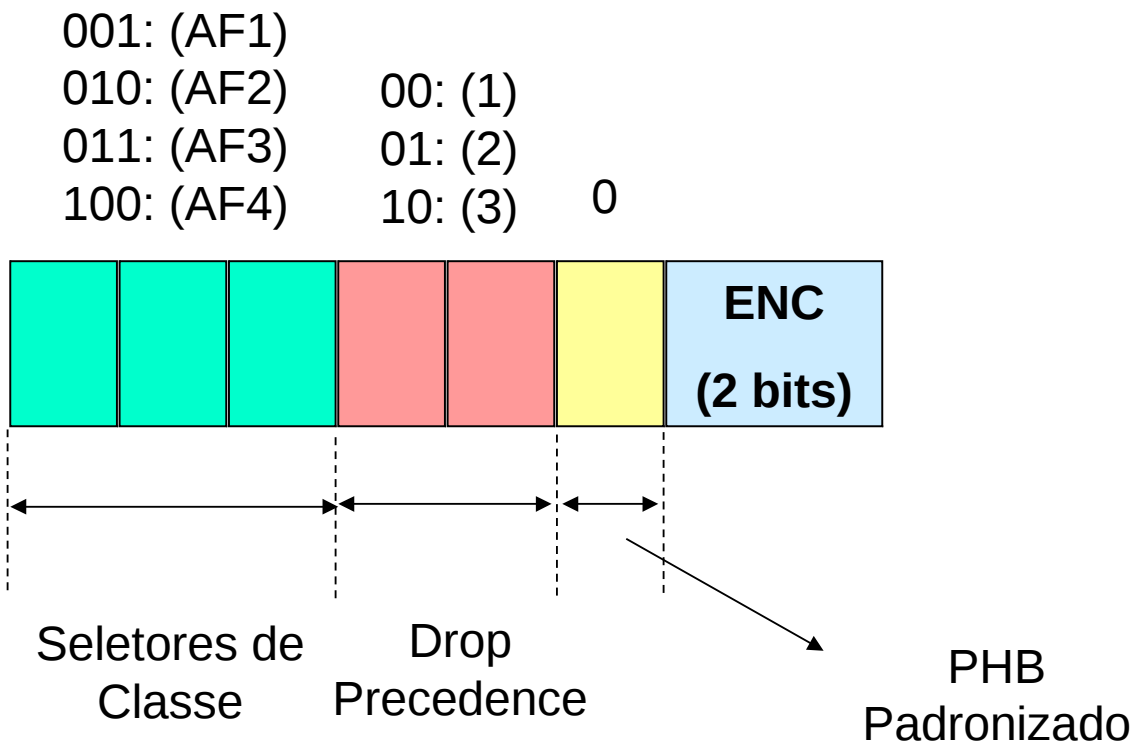


Figure 2.9.2

CodePoints de PHB

- A marcação dos pacotes deve desprezar os três últimos bits do byte de TOS (DSCP), conforme a figura abaixo.



PHB's Padronizados

Drop Precedence

DCSP em Hexa

DS em Hexa

CLASS	DP	DSCP	b-DSCP	x-DCSP	b-DS	x-DS
AF1	1	001010	0000-1010	0xa	0010-1000	0x28
	2	001100	0000-1100	0xc	0011-0000	0x30
	3	001110	0000-1110	0xe	0011-1000	0x38
AF2	1	010010	0001-0010	0x12	0100-1000	0x48
	2	010100	0001-0100	0x14	0101-0000	0x50
	3	010110	0001-0110	0x16	0101-1000	0x58
AF3	1	011010	0001-1010	0x1a	0110-1000	0x68
	2	011100	0001-1100	0x1c	0111-0000	0x70
	3	011110	0001-1110	0x1e	0111-1000	0x78
AF4	1	100010	0010-0010	0x22	1000-1000	0x88
	2	100100	0010-0100	0x24	1001-0000	0x90
	3	100110	0010-0110	0x26	1001-1000	0x98
EF		101110	0010-1110	0x2e	1011-1000	0xb8

Table 2.9.1

Exemplos

- Setar todos os pacotes para AF23:
  - mask 0x0 (*b'00000**00***) value 0x58 (*b'01011**00***)
- Setar todos os pacotes como AF12, preservando os bits ECN:
  - mask 0x3 (*b'00000**11***) value 0x30 (*b'00110**00***)
- Setar em 2 o 'drop precedence' de todos os pacotes
  - mask 0xe3 (*b'11100**11***) value 0x10 (*b'00010**00**)*
- Setar todos os pacotes para AF3, sem alterar os bites ECN e os bits de precedência.
  - mask 0x1f (b'*00011**11**)* value 0x60 (b'*01100**00**)*

Exemplo: Criação dos Filtros

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip src 192.168.1/24 flowid 1:1

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip src 192.168.2/24 flowid 1:2

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip src 192.168.3/24 flowid 1:3

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip src 192.168.4/24 flowid 1:4

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip src 192.168.5/24 flowid 1:5

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip src 192.168.6/24 flowid 1:6

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
    match ip src 192.168.7/24 flowid 1:7
```

Exercício 4:

- Crie um script para:
  - marcar os pacotes UDP com AF 11
  - marcar os pacotes TCP com AF 23
- Utilizando o *Ethereal*, capture os pacotes enviados e recebidos pelo seu computador e verifique como eles são marcados.

Policiamento: Policing

- A função do policiamento é limitar o tráfego do usuário as condições importas pelo SLA.
- O policiamento é feito normalmente na interface de entrada dos roteadores de borda.

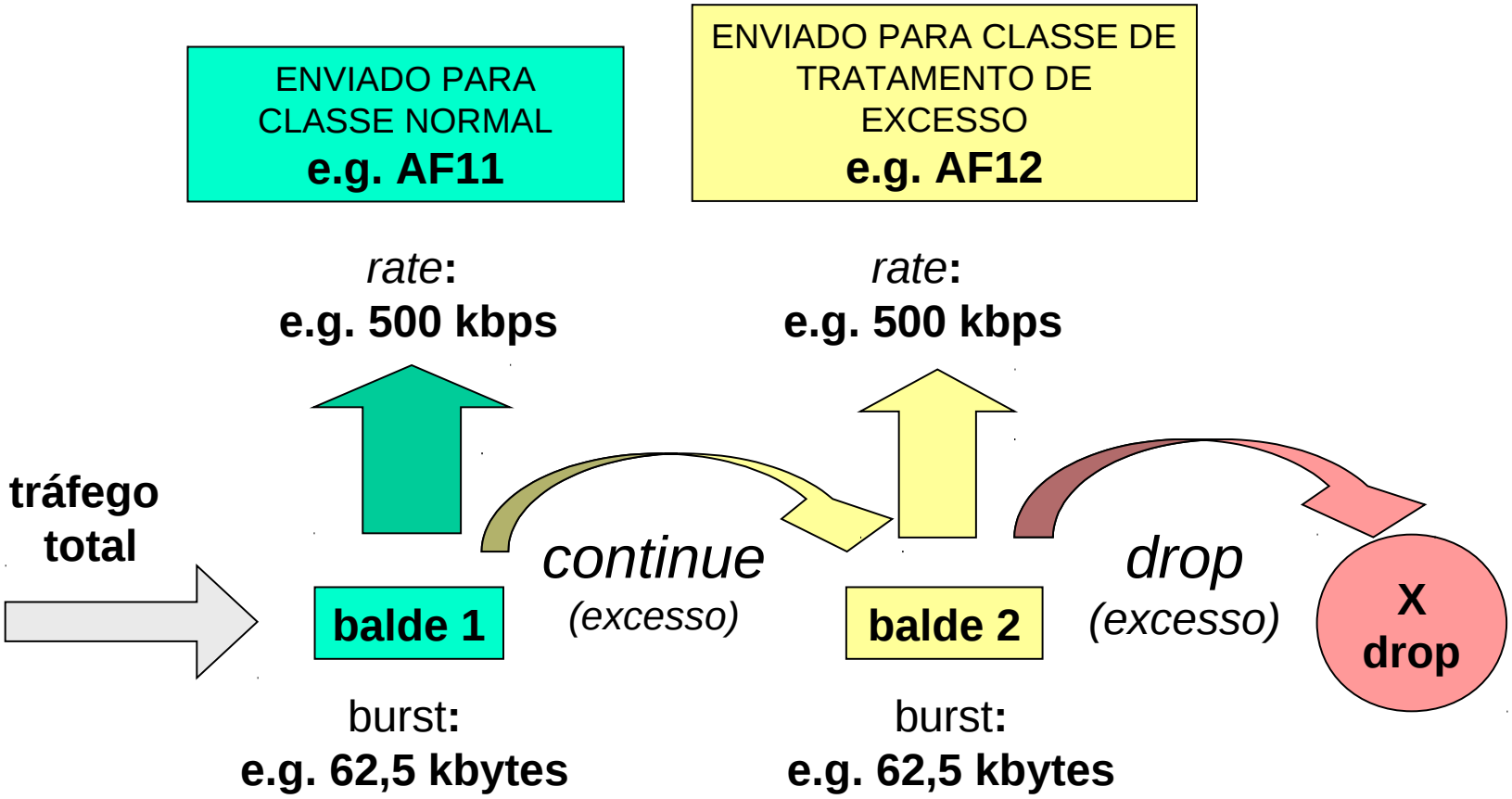
Policiamento: Policing

- O policiamento é implementado pelo cascateamento de fluxos token-bucket controlados
- O último parâmetro especifica o que deve ser feito com os pacotes que excederem o burst.
  - **drop**: os pacotes são descartados
  - **continue**: continua a classificação do pacote assumindo a regra de filtro de prioridade inferior mais próxima.
  - **classify** (apenas para CBQ): classifica o pacote como Best Effort.

> **police** rate **BPS** burst **BYTES**
> [**reclassify** | **drop** | **continue**]

Policiamento: Policing

- O tráfego que excede o balde é tratado pelo próximo filtro no qual o tráfego se encaixa.



Exemplo de script com policiamento:

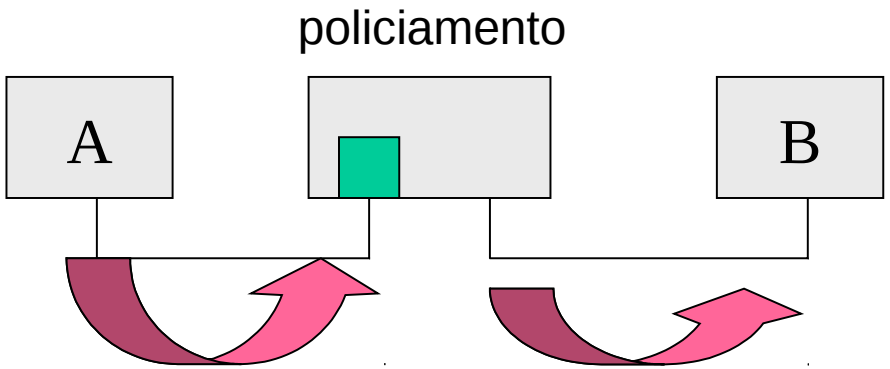
- *#!/bin/bash*
- **tc qdisc del** dev eth0 root
- *# Crias as classes dsmark*
- **tc qdisc add** dev eth0 handle 1:0 root dsmark indices 4
- *# Marcação em AF41*
- **tc class change** dev eth0 parent 1:0 classid 1:1 dsmark mask **0x0** value **0x88**
- *# Marcação AF42*
- **tc class change** dev eth0 parent 1:0 classid 1:2 dsmark mask **0x0** value **0x90**
- *# Marcação AF13*
- **tc class change** dev eth0 parent 1:0 classid 1:3 dsmark mask **0x0** value **0x38**

Continuação: Filtros

- *# Filtro para classe AF41*
- **tc filter add** dev eth0 parent 1:0
  protocol ip **prio 1** u32
  match ip dst 192.168.1.2/32
  police rate **500kbit** burst **50k** **continue** classid 1:1
- *# Filtro para classe AF42*
- **tc filter add** dev eth0 parent 1:0
  protocol ip **prio 2** u32
  match ip dst 192.168.1.2/32
  police rate **500kbit** burst **50k** **drop** classid 1:2
- *# Filtro para classe AF13*
- **tc filter add** dev eth0 parent 1:0
  protocol ip **prio 5** u32
  match ip protocol 0 0 flowid 1:3

Exercício 6:

- Configure um dos computadores da sua bancada como roteador.
  - Utilizando o script anterior como base, policie o tráfego de A para B.
  - Faça a transferência de um arquivo grande (> 5Mbytes) utilizando scp, e verifique como a marcação do campo DS dos pacotes foi feita.
  - Verifique também as estatísticas dos filtros da sua regra.



ALGORITMOS DE DESCARTE

Algoritmos de Descarte

Projeto de um Roteador Core

Criação de Filtros com IP Tables

RED: Random Early Detection

- O principal objetivo deste algoritmo é limitar o tamanho das filas, controlando o atraso médio introduzido na transmissão de pacotes.

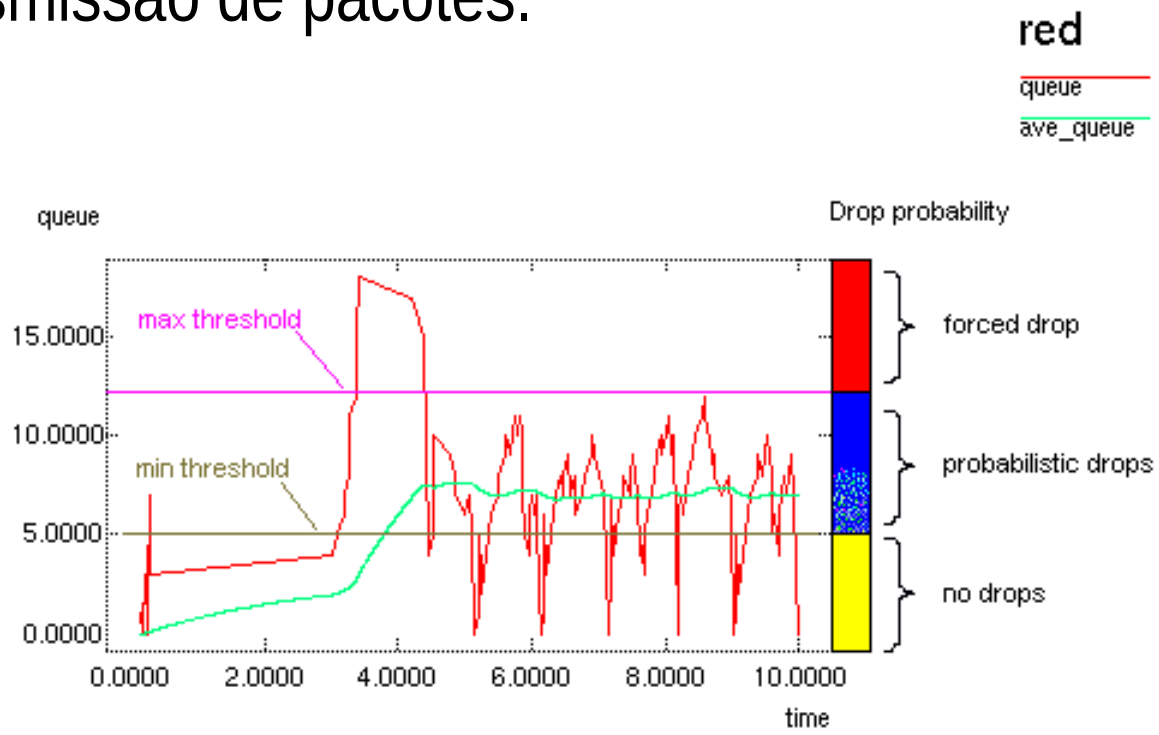


Figure 2.6.2

Parâmetros

```
# tc qdisc add dev eth0 root limit <bytes> min <bytes> max <bytes> \
    avpkt <bytes> burst <packets> probability <number> bandwidth <kbps> \
    [ecn]
```


- **probability** : probabilidade de descarte (de 0.0 a 1.0)
  - Recomendado: 0.01 ou 0.02
  - Entre min e max, a probabilidade de descarte é proporcional ao tamanho médio da fila
- **max** : tamanho de fila médio com probabilidade de descarte máxima
  - Cálculo: (largura de banda) * (delay máximo desejado)
- **min**: tamanho de fila médio que inicia o descarte
  - Recomendado: 1/3 * max

Parâmetros

- limit: tamanho máximo da fila
  - Recomendado: >> max + burst ou 8 * max
- burst : tolerância para tamanho instantâneo da fila
  - Recomendado (min+min+max)/(3*avpkt).
- avpkt :
  - Tamanho médio do pacote em bytes
- ecn: Explicit Congestion Notification
  - Bits menos significativos do DSCP
  - Usado como alternativa ao descarte
- bandwidth: usado para calcular o tamanho médio da fila na ausência de tráfego (velocidade do link).

Exemplo: Dimensionamento do RED

- Considere que:
  - **<bandwidth>** = 512 kbps ~ 512000 bps = 64000 bytes / sec
  - Latência máxima desejada = 500 ms
- Então:
  - **<max>**
    - 64000 bytes / sec * 0.5 sec = 32000 bytes
  - **<min>**
    - ~ 1/3 <max> = 12000 bytes
  - **<limit>**
    - ~ 8 * <max> = 256000 bytes.
  - **<avpkt>**
    - = 1000 bytes.
  - **<burst>**
    - = (2 * <min> + <max>) / (3 * <avpkt>)
    - = (2 * 12000 + 32000) / (3 * 1000) = 18.67 ~ 20.

Exemplos

- O comando abaixo define a seguinte política de descarte:
  - Iniciar o descarte em 12 Kbytes
  - A probabilidade máxima de descarte é 2%
  - Atingir a probabilidade máxima de descarte em 32 Kbytes
  - Admitir burst de 20 pacotes (20 Kbytes)
  - Descartar tudo acima de 256 Kbytes
- Obs.
  - Quando ECN é usado, os pacotes abaixo de "limit" são marcados com ECN ao invés de descartados.

```
# tc qdisc add dev eth0 root red limit 256000 min 12000 max 32000
      avpkt 1000 burst 20 probability 0.02 bandwidth 512
```

```
# tc qdisc add dev eth0 root red limit 256000 min 12000 max 32000
      avpkt 1000 burst 20 probability 0.02 bandwidth 512 ecn
```

GRED

- GRED é um algoritmo de descarte que permite tratar múltiplos níveis de prioridade de descarte.
- Cada nível de prioridade é associado a uma fila virtual.
- São possíveis até 16 níveis de prioridade (1 a 16), sendo 1 o nível mais alto.

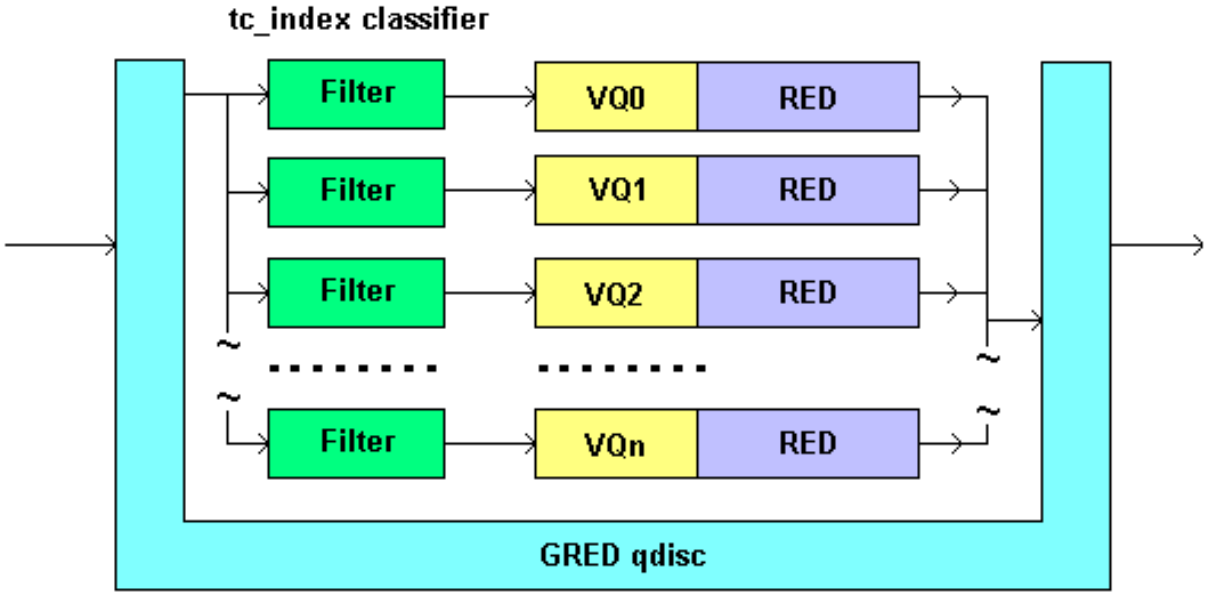


Figure 2.7.1

Exemplos de Comandos GRED

```
# tc qdisc add dev eth0 root gred setup DPs <VQs> default <VQ> [grio]
```

```
# tc qdisc add dev eth0 root gred setup DPs 3 default 3 grio
```

```
# tc qdisc change dev eth0 root gred limit <bytes> min <bytes> \
      max <bytes> burst <packets> avpkt <bytes> bandwidth <kbps> \
      DP <vq> probability <number> [ prio <number> ]
```

```
# tc qdisc change dev eth0 root gred limit 60KB min 15K   \
      max 45KB burst 20 avpkt 1000 bandwidth 10Mbit \
      DP 1 probability 0.02 prio 2
```

Tcindex

- Tcindex é um parâmetro opcional de DSMARK
  - **tc qdisc add** dev eth0 handle 1:0 root dsmark indices 4 **set_tc_index**

- A opção permite criar um filtro de entrada que efetua operações de mascaramento e deslocamento no byte DS:
  - **tc filter add** dev eth0 parent 1:0 protocol ip prio 1 tcindex **mask** <*mask*> **shift** <*shift*> pass_on

- O resultado pode ser explorado pelos demais filtros:
  - Resultado = (byte_DS & **p**.mask) >> **p**.shift

Exemplo

- tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex mask **0xfc** shift **2** pass_on

- Suponha que o código recebido foi:
  - Tratamento de prioridade numa mesma classe
    - AF11: 0x28 & 0xfc >> 2 =
    - (00101000 & 11111100) >> 2 =
    - 00001010 = 0xA (10 em decimal)
  - Similarmente:
    - AF11 = 10 e AF12 = 12 e AF13 = 14
    - AF21 = 18 e AF22 = 20 e AF23 = 22
    - AF31 = 26 e AF32 = 28 e AF33 = 30
    - AF41 = 34 e AF42 = 36 e AF43 = 38

TcIndex

- Os filtros subsequentes podem ser criados usando o atributo handle X tcindex, conforme o exemplo abaixo:
  - AF11 = 10, AF12 = 12 e AF13 = 14

```
#tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 \
      set_tc_index

#tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex \
      mask 0xfc shift 2 pass_on

#tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 10 tcindex classid 1:111

#tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 12 tcindex classid 1:112

#tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 14 tcindex classid 1:113
```

=

(

0

x

3

0

&

Tcindex e RED

```
# tc qdisc add dev eth0 handle 1:0 root dsmark indices 16 \
      set_tc_index

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex \
      mask 0xfc shift 2 pass_on

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 10 tcindex classid 1:1

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 12 tcindex classid 1:2

# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 14 tcindex classid 1:3

# tc qdisc add dev eth0 parent 1:0 gred setup DPs 3 default 3

# tc qdisc change dev eth0 parent 1:0 gred limit 3000 min 400 \
      max 800 burst 5 avpkt 128 bandwidth 10Mbit DP 1 \
      probability 0.01

# tc qdisc change dev eth0 parent 1:0 gred limit 6000 min 800 \
      max 1600 burst 12 avpkt 256 bandwidth 10Mbit DP 2 \
      probability 0.02

# tc qdisc change dev eth0 parent 1:0 gred limit 12000 min 1600 \
      max 3200 burst 20 avpkt 512 bandwidth 10Mbit DP 3 \
      probability 0.03
```

A prioridade é setada de acordo com o índice minor da classe.

Exercício

- Crie um roteador de core com suporte as classes
  - AF11, AF12 e AF13
- Supondo que a interface de saída do roteador é de 1Mbps, imponha os seguintes limites de delay:
  - Classe AF11:
    - delay máximo: 10 ms
    - probabilidade de descarte: 10%
  - Classe AF12
    - delay máximo: 30 ms
    - probabilidade de descarte: 20%
  - Classe AF13
    - delay máximo: 50 ms
    - probabilidade de descarte: 30%

Exemplo de Script

- #!/bin/bash
- **tc qdisc del** root dev eth0
- # criar a qdisc principal
- **tc qdisc add** dev eth0 root **handle 1: prio**
- # criar as qdiscs de saída
- **tc qdisc add** dev eth0 **handle 2:** parent 1:3 **htb**
- **tc class add** dev eth0 parent 2:0 **classid 2:1** htb rate 1000kbit ceil 1000kbit burst 1k
- **tc qdisc add** dev eth0 parent 2:1 **handle 3: red** \
- limit 10000 min 1000 max 2000 avpkt 1000 burst 1 probability 0.5
- #criar os filtros
- **tc filter add** dev eth0 **parent 2:0** protocol ip prio 1 u32 \
- match ip protocol 0x06 0xff flowid 2:1

Modos de Criação das Regras

- Um filtro pode ser implementado de duas formas:
  - classificador u32
  - classificador fw
- A classificação baseada em fw permite utilizar regras de filtro iptables para classificar os pacotes.
- A ação do iptables é do tipo:
  - -j MARK – set mark id
- Onde id é definido pelo parâmetro handle do tc filter.

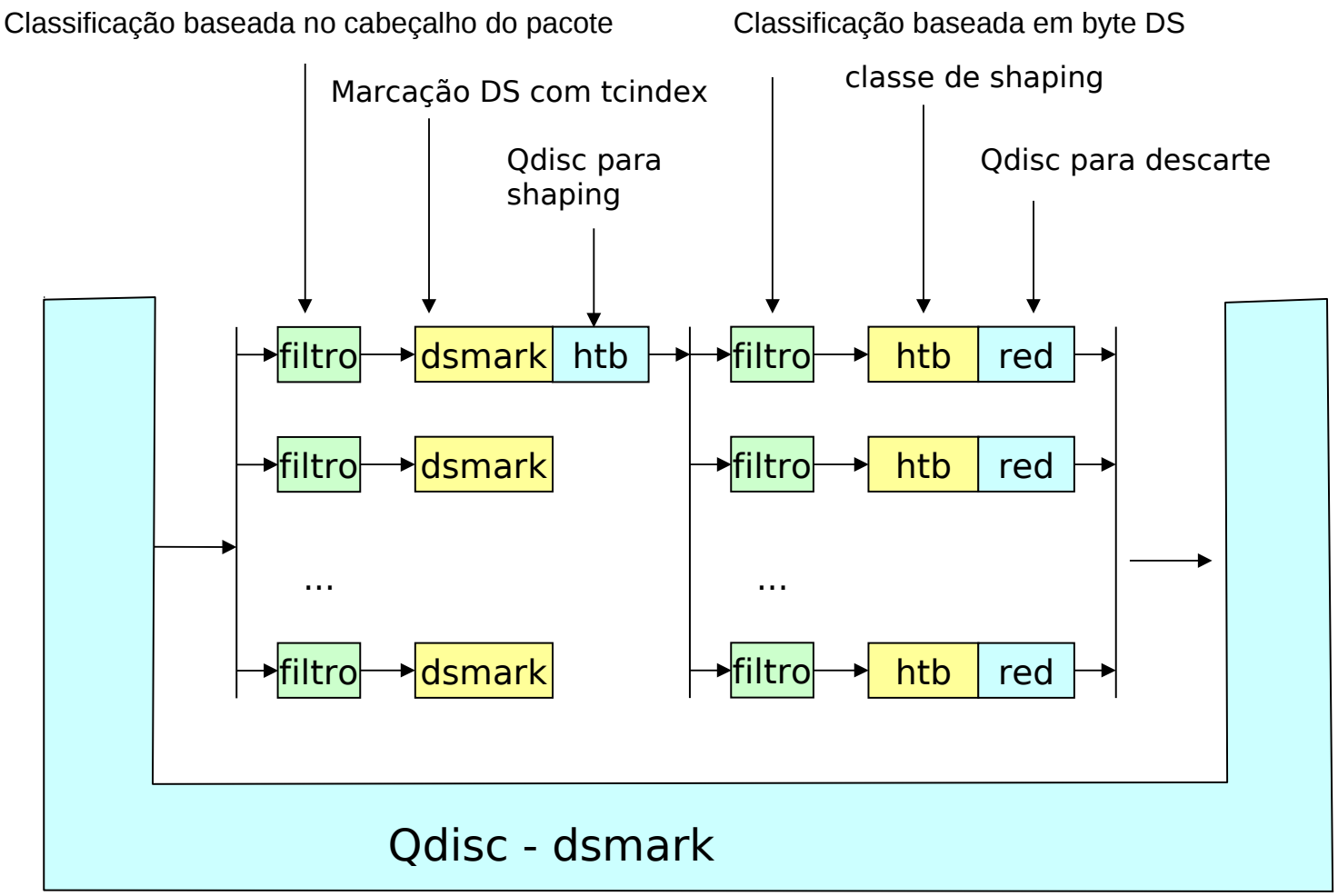
Exemplo de Filtro com IPTables

```
# iptables -t mangle -A FORWARD -i eth1 -s 0/0 \
          -j MARK --set-mark 2

# iptables -t mangle -A FORWARD -i eth1 -s 10.2.0.0/24 \
          -j MARK --set-mark 1

# tc qdisc add dev eth1 handle ffff: ingress

# tc filter add dev eth1 parent ffff: protocol ip prio 1 \
          handle 1 fw flowid :1

# tc filter add dev eth1 parent ffff: protocol ip prio 1 \
          handle 2 fw flowid :2

# tc qdisc add dev eth0 handle 1:0 root dsmark indices 16

# tc class change dev eth0 classid 1:1 dsmark \
          mask 0x3 value 0x88

# tc class change dev eth0 classid 1:2 dsmark \
          mask 0x3 value 0x90
```

Exemplo com Policiamento

```
# iptables -t mangle -A FORWARD -i eth1 -s 0/0 \
            -j MARK --set-mark 2

# iptables -t mangle -A FORWARD -i eth1 -s 10.2.0.0/24 \
            -j MARK --set-mark 1

# tc qdisc add dev eth1 handle ffff: ingress

# tc filter add dev eth1 parent ffff: protocol ip prio 4 \
     handle 1 fw police rate 1500kbit burst 90k \
     continue flowid 4:1

# tc filter add dev eth1 parent ffff: protocol ip prio 5 \
     handle 1 fw police rate 1500kbit burst 90k \
     continue flowid 4:2

# tc filter add dev eth1 parent ffff: protocol ip prio 6 \
     handle 1 fw police rate 1000kbit burst 60k \
     drop flowid 4:3

# tc filter add dev eth1 parent ffff: protocol ip prio 6 \
     handle 2 fw police rate 1000kbit burst 60k \
     drop flowid 4:4
```

Exemplo: Marcação, Shaping e Descarte com tcindex



Script

- **tc qdisc** del root dev eth0
- *# cria a qdisc e as classes de marcação*
- **tc qdisc** add dev eth0 handle 1:0 root dsmark indices 4 set_tc_index
- **tc class** change dev eth0 classid 1:1 dsmark mask 0x0 value 0x31
- *# associa o filtro a qdisc principal*
- **tc filter** add dev eth0 **parent 1:0** \

  protocol ip prio 1 u32 match ip protocol 0x06 0xff flowid 1:1

Continuação

- *# cria a qdisc e as classes de shaping*
- **tc qdisc** add dev eth0 parent 1:1 handle 2:0 htb
- **tc class** add dev eth0 classid 2:1 htb rate 1000kbit ceil 1000kbit
- *# associa o filtro a qdisc de shaping, baseada no código dsmark*
- **tc filter** add dev eth0 parent 2:0 protocol ip prio 2 handle 0x31 tcindex mask 0xff classid 2:1
- *# cria a qdisc de descarte*
- **tc qdisc** add dev eth0 parent 2:1 handle 3: red limit 50000 min 10000 max 40000 avpkt 1000 burst 10 probability 0.1