Implementación de QoS

- En dispositivos intermedios: casi invariablemente en routers IP, switches ATM o dispositivos MPLS (que veremos más adelante).
- En Linux, mediante el módulo del Kernel llamado “tc” (Traffic Control) y una una herramienta de control de línea de comandos homónima.
- Viene “por defecto” en la mayoría de los sistemas.

Conceptos clave de TC

- **QDISC o Disciplina de encolado –**

  Representan los distintos tipos de “cola” que posee cada interfaz de red. Una *qdisc* realiza el scheduling de los paquetes que serán enviados por una interfaz de red. Existen distintas qdisc para distintos usos. Algunas permiten priorizar el tráfico, otras permiten clasificarlo el tráfico, otras marcarlo, otras hacen drop según ciertos parámetros, etc.

- **CLASES –** Hay *qdiscs* para las cuales todo el tráfico es igual, pero otras *qdiscs* permiten definir clasificar el tráfico y darle tratamiento distinto a cada uno.

- **FILTERS –** En las disciplinas que permiten clasificación, los filtros son reglas que determinan la clasificación de cada paquete según ciertas condiciones, tales como dirección IP origen o destino, puertos, tipo de protocolo, etc.

Algunas QDiscs en TC

- Simples, sin posibilidad de definir clases (classless)
  - First In, First Out (FIFO)
    - Packet limited First In, First Out (pfifo)
    - Three-band First In, First Out (pfifo_fast) - por defecto en linux antiguos
  - Con múltiples colas
    - basadas en hash
      - Stochastic Fairness Queuing (SFQ) - round robin, cada flujo udp o sesión tcp tiene su chance
      - Fair Queuing with Controlled Delay (fq_codel) - por defecto en linux modernos
  - Token Bucket Filter (TBF) - ralentiza una interfaz, hace shaping o policing según las opciones
  - Generalized Random Early Detection (GRED) - hace Random Early Detection para evitar congestión
- Con posibilidad de definir clases (classfull)
  - Hierarchical Token Bucket (HTB) - permite definir distintas clases con tasas de tráfico diferentes
  - Deficit Round Robin (DRR) - aplica el algoritmo visto previamente
  - ...

Sintaxis de comandos

```
# para ver configuración actual y estadísticas
# (tokens disponibles, paquetes descartados, reordenados, etc)
tc -s [ qdisc | class | filter ] show dev INTERFAZ

# para agregar / cambiar / quitar / ver disciplinas de encolado
tc qdisc [ add | change | del ] dev INTERFAZ root nombre_qdisc parametros_qdisc

# para agregar / cambiar / quitar / ver clases de tráfico
tc class [ add | change | show ] dev INTERFAZ parent id_qdisc_padre nombre_qdisc

# para agregar / cambiar / quitar / ver reglas de filtrado de tráfico
tc filter [ add | change | show ] dev INTERFAZ root protocolo proto prio prioridad tipo_filtro
parametros_filtro

# para volver todo a la normalidad
tc qdisc del dev eth0 root
```

Ejemplo - Qdisc actual

Consultar la disciplina de encolado actual

```
root@router:~# tc -s qdisc
qdisc pfifo_fast 0: dev eth0 root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
Sent 468 bytes 6 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
qdisc pfifo_fast 0: dev eth1 root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
Sent 468 bytes 6 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
```

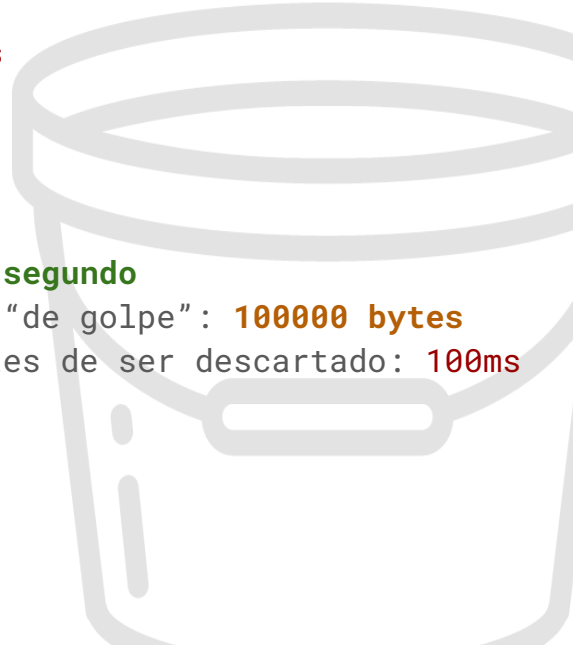
qdisc pfifo_fast → disciplina de encolado pfifo_fast
dev eth0 → dispositivo en el que está configurada
Sent 486 bytes 6 pkt (dropped 0, overlimits 0 requeues 0) → estadísticas de pqt encolados

Token Bucket con TC

Token Bucket con tasa de 1 mbit/s, balde de 100 kbits y latencia máxima de 100ms

```
root@router:~# tc qdisc add dev eth0 root tbf rate 1mbit burst 100kb latency 100ms
root@router:~# tc -s qdisc dev eth0
qdisc tbf 8002: root refcnt 2 rate 1000Kbit burst 100Kb lat 100.0ms
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

parámetros:
rate 1mbit → tasa de arribo de tokens al balde: **1000000 tokens por segundo**
burst 100kb → tamaño del balde o máxima cantidad de bytes a enviar "de golpe": **100000 bytes**
latency → tiempo máximo que puede esperar un paquete por tokens antes de ser descartado: 100ms



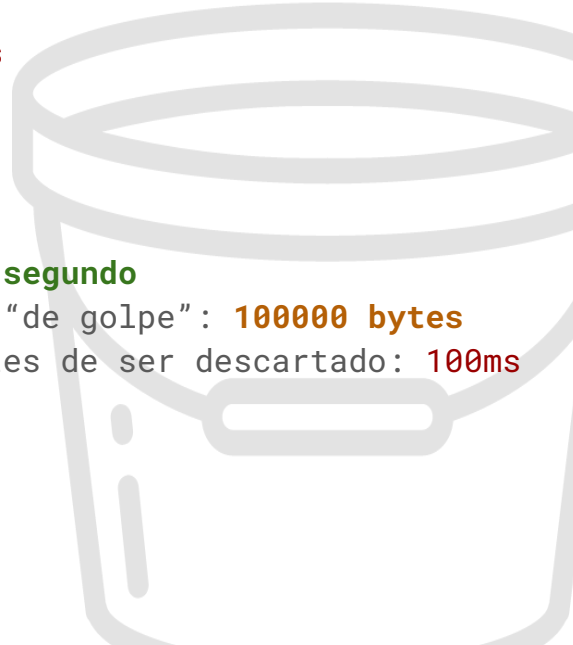
Así como está, esta disciplina ¿hace *policing* o *shaping*?

Token Bucket con TC

Token Bucket con tasa de 1 mbit/s, balde de 100 kbits y latencia máxima de 100ms

```
root@router:~# tc qdisc add dev eth0 root tbf rate 1mbit burst 100kb latency 100ms
root@router:~# tc -s qdisc dev eth0
qdisc tbf 8002: root refcnt 2 rate 1000Kbit burst 100Kb lat 100.0ms
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

parámetros:
rate 1mbit → tasa de arribo de tokens al balde: **1000000 tokens por segundo**
burst 100kb → tamaño del balde o máxima cantidad de bytes a enviar "de golpe": **100000 bytes**
latency → tiempo máximo que puede esperar un paquete por tokens antes de ser descartado: 100ms

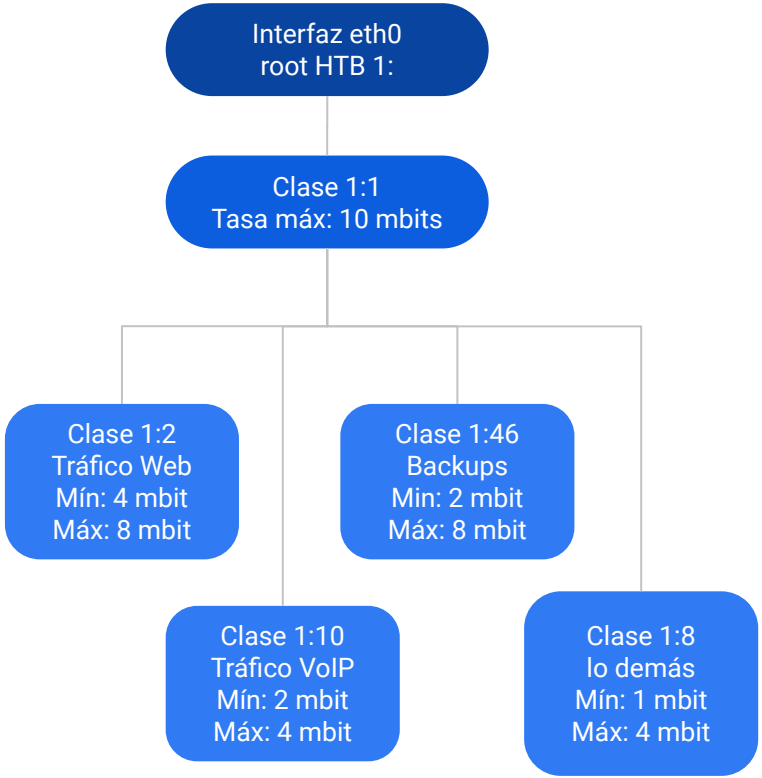


Depende del parámetro *latency*. Si tiende a 0ms → *policer* ?

Hierarchical Token Bucket

Hierarchical Token Bucket (HTB)

HTB es una ***disciplina classful*** muy útil cuando se dispone de un enlace de tasa de transferencia fijo que se desea repartir entre distintas clases de tráfico, **dando a cada clase una tasa de transferencia garantizada** si el enlace está saturado, pero a la vez permitiendo que ésos u otros flujos aprovechen todo o parte del enlace si éste estuviera libre.



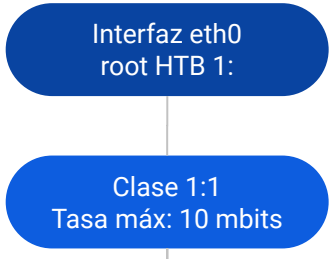
Hierarchical Token Bucket

```
tc qdisc add dev eth0 root handle 1:0 htb default 8
```

Establece la qdisc a **HTB**, con nombre **1:0** y establece que todo paquete que no se asigne explícitamente a una clase, se asignará a la **clase 8**

Hierarchical Token Bucket

```
tc qdisc add dev eth0 root handle 1:0 htb default 8

tc class add dev eth0 parent 1:0 classid 1:1 htb rate
10mbit ceil 10mbit burst 100k
```



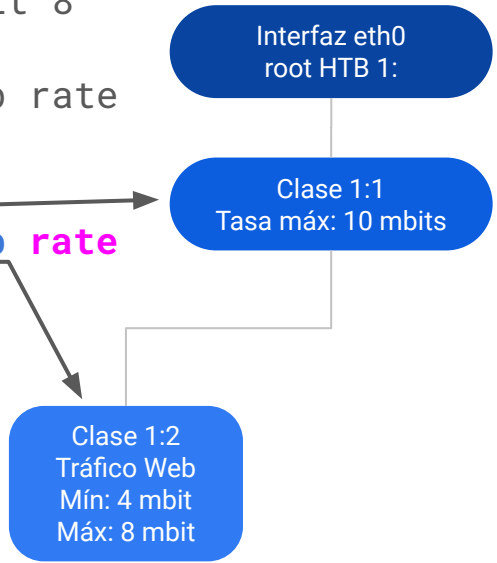
Crea una clase llamada **1:1** que desciende de la raíz **1:0** para limitar el tráfico en todo el enlace.

Le asigna una qdisc **htb**, con una **tasa mínima y máxima de 10 mbits** y tamaño de **bucket de 100 kbytes**.

Hierarchical Token Bucket

```
tc qdisc add dev eth0 root handle 1:0 htb default 8

tc class add dev eth0 parent 1:0 classid 1:1 htb rate
10mbit ceil 10mbit burst 100k

tc class add dev eth0 parent 1:1 classid 1:2 htb rate
4mbit burst 100k ceil 8mbit
```

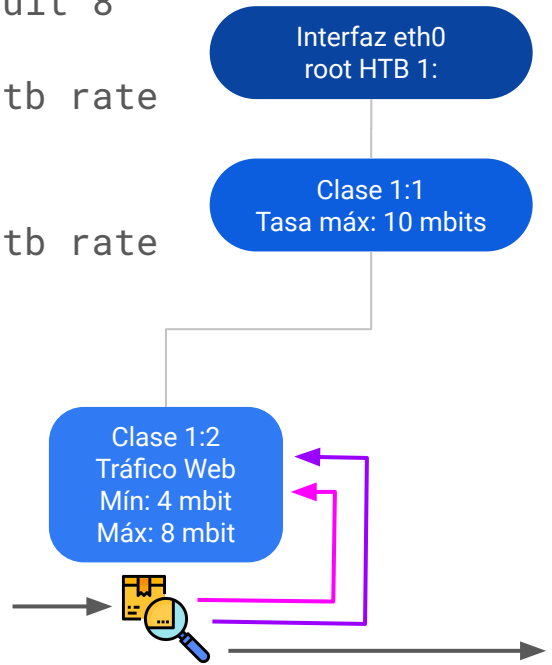
Crea una clase llamada **1:2** que desciende de la clase **1:1**. Le asigna una qdisc htb, con una **tasa mínima de 4 mbits**, tamaño de **bucket de 100 kbytes** y **tasa máxima de 8 mbit**.



Hierarchical Token Bucket

```
tc qdisc add dev eth0 root handle 1:0 htb default 8

tc class add dev eth0 parent 1:0 classid 1:1 htb rate
10mbit ceil 10mbit burst 100k

tc class add dev eth0 parent 1:1 classid 1:2 htb rate
4mbit burst 100k ceil 8mbit

tc filter add dev eth0 protocol ip parent 1:
prio 0 u32 match ip sport 80 0xffff flowid 1:2

tc filter add dev eth0 protocol ip parent 1:
prio 0 u32 match ip dport 80 0xffff flowid 1:2
```

Asigna todo el tráfico desde puerto 80 o hacia puerto 80 a la clase 1:2



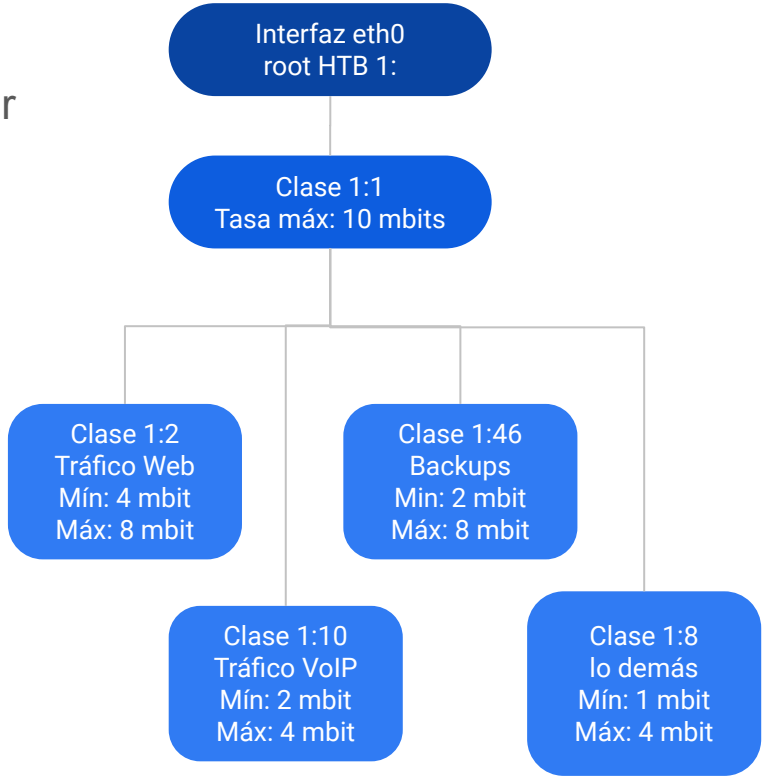
Hierarchical Token Bucket

Y así hasta crear las restantes clases...

- Crear la clase 1:10 (VoIP) con sus parámetros y asignar a ella, mediante un filtro, a todo tráfico desde o hacia el servidor VoIP.

- Crear la clase 1:46 (backups) con sus parámetros y asignar a ella, mediante un filtro, a todo tráfico…
¿cómo determinar si un paquete corresponde a backup?

- Crear la clase 1:8 (resto) para todo lo demás.
Aquí no hace falta filtro pues va por defecto
(ver en defínción de la qdisc raíz)



Bibliografía

- EVANS, J., FILSFILS, C., 2007, Deploying IP and MPLS QoS for Multiservice Networks: Theory & Practice. Morgan Kaufmann.
  - Capítulo 1. “QOS Requirements and Service Level Agreements”
  - Capítulo 2. “Introduction to QOS Mechanics and Architectures”
- MEDHI, D., RAMASAMY, K., 2007, Network Routing Algorithms Protocols and Architectures. Morgan Kaufmann.
  - Capítulo 14. “Router Architectures”
  - Capítulo 22. “Packet Queuing and Scheduling”
  - Capítulo 23. “Traffic Conditioning”
- Módulo TC en linux:
  - Manual tc Packet Filtering and netem http://tcn.hypert.net/tcmanual.pdf
  - Traffic Control HOWTO https://tldp.org/HOWTO/Traffic-Control-HOWTO/