Copyright Notice

Presentation by: Alessandro Selli <alessandroselli@linux.com>

Version 1.0.6, 2015/10/07

1

TC: Total Control

1

TC: ~~Total~~ Control

1

TC Traffic Control

1 TC Traffic Control

Contents

2 TC origins and development

- TC first appeared in kernel 2.2, developped by Alexey N. Kustnetzov
- Many additions and extensions developped ever since
- Latest addition[1] is Berkley Packet Filter “*programmable classifier and actions for ingress/egress queueing disciplines*”, available since kernel 3.18
- New qdisc `cake` is been worked on

[1] That I am aware of :-)

3

The Naming of Schedulers Is a Difficult Matter

It isn't just one of your holiday games.

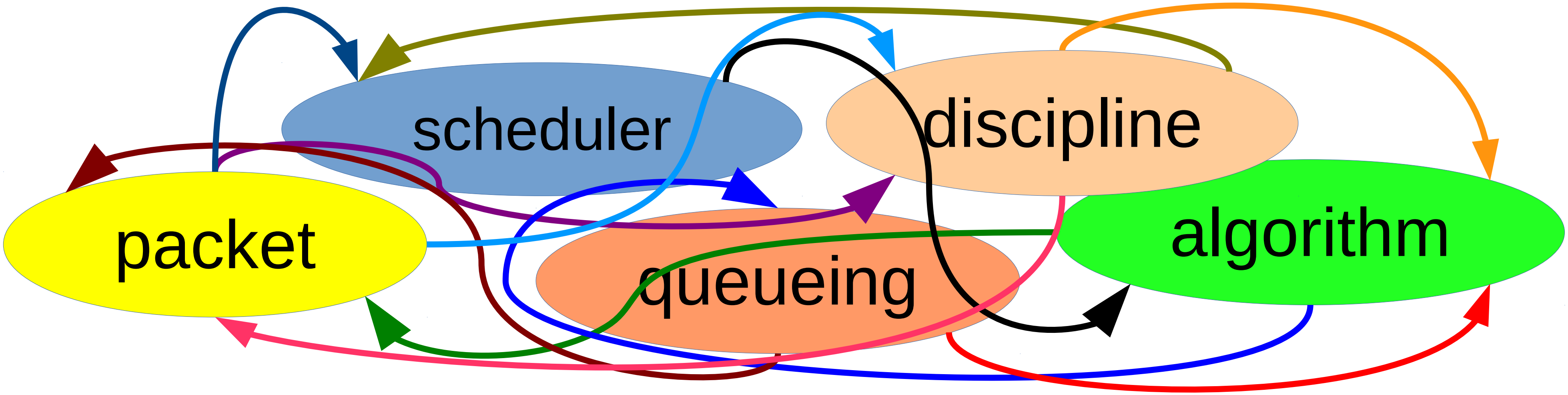
- Queueing (|Packet) (|Discipline) (Qdisc)
- (|Packet) Scheduler (|Algorithm)
- (|Packet) Queueing (|Algorithm)

3 The Naming of Schedulers Is a Difficult Matter

It isn't just one of your holiday games.

- Queueing (|Packet) (|Discipline) (Qdisc)
- (|Packet) Scheduler (|Algorithm)
- (|Packet) Queueing (|Algorithm)

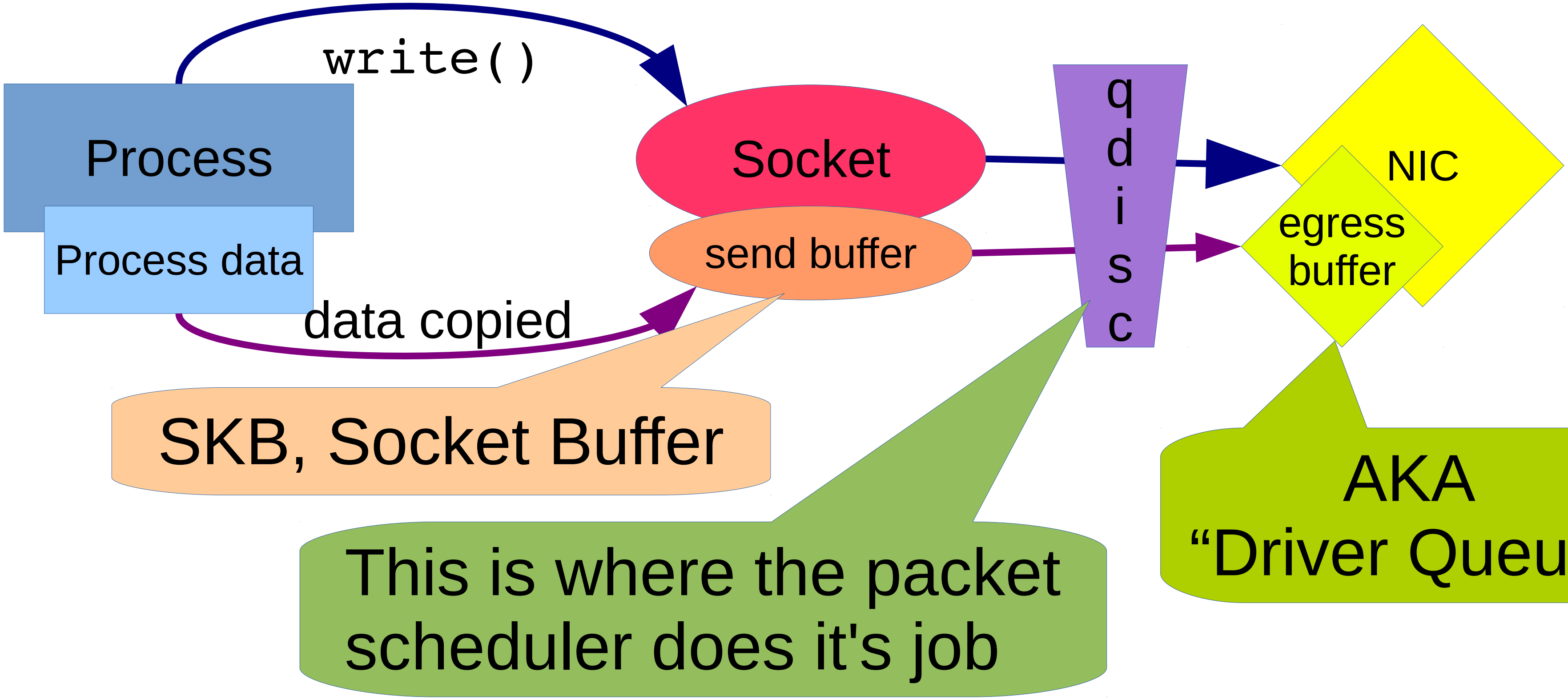
Any random combination of strings will do:



4

What traffic? Where?

Whatever goes through a socket can be scheduled:



write()

Process

Process data

data copied

Socket

send buffer

qdisc

NIC

egress buffer

SKB, Socket Buffer

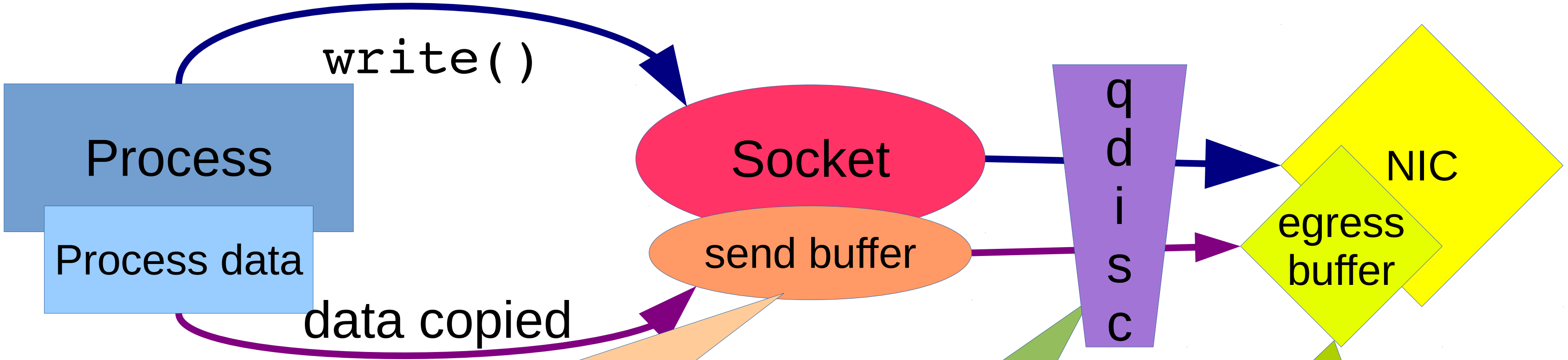
This is where the packet scheduler does it's job

AKA "Driver Queue"

Scheduling and shaping only affect outbound packets, not incoming ones

4 What traffic? Where?

Whatever goes through a socket can be scheduled:



SKB, Socket Buffer

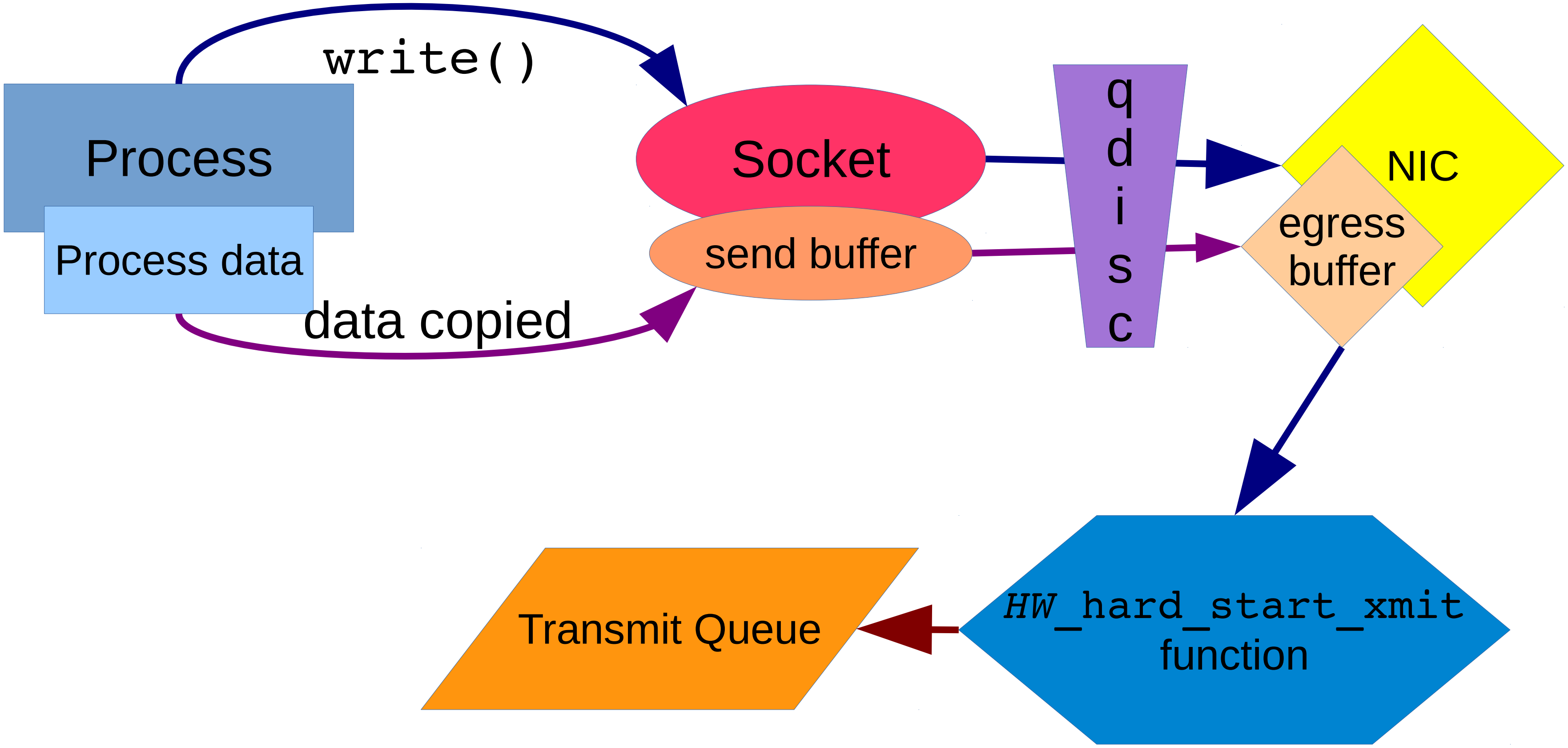
This is where the packet scheduler does it's job

AKA “Driver Queue”

Filtering, on the other hand, can affect both inbound and outbound packets

4 What traffic? Where?

Whatever goes through a socket can be scheduled:



HW = name of hardware
NIC driver

5 Sizes that Matter

Factors impacting packet transm. timings:

- Socket buffer sizes:
  - Each driver sets it's own `tx_ring`, `rx_ring`
  - Application can set `SO_SNDBUF` and `SO_RCVBUF` with `setsockopt(2)`
    - `/proc/sys/net/core/rmem_default`
    - `/proc/sys/net/core/wmem_default`
- Default transmit size: 1000 packets

  `ether_setup()`: `net/ethernet/eth.c`

```
dev->tx_queue_len    = 1000; /* Ethernet wants good queues */
```

5 Sizes that Matter

- Receiving end backlog[1] size: 1000 packets[2]: (`/proc/sys/net/core/netdev_max_backlog`)
- Queueing disciplines have their own buffer(s)
  - See `pfifo_fast` ahead, for instance
- Packet size (standard, jumbo or super sized)
- Capability of the kernel/CPU to keep up with the flux (load, jiffies…)
- Number of hops (switches, routers, …)
  - And funny hardware interactions

[1] Maximum number of input packets received before the kernel can process them

[2] For non-NAPI devices/drivers (< 2.4.20)

What Control?

Traffic Control is multifaceted:

tc

SHAPING	↑	Control traffic rate of transmission
SCHEDULING	↑	Packets transmission scheduling (reordering/prioritizing)
POLICING	↓	Rules to apply to ingress traffic
DROPPING	↓↑	Clipping of traffic exceeding a set bandwidth

↓ = ingress ↑ = egress

6

What Control?

Traffic Control is multifaceted:

`tc` {

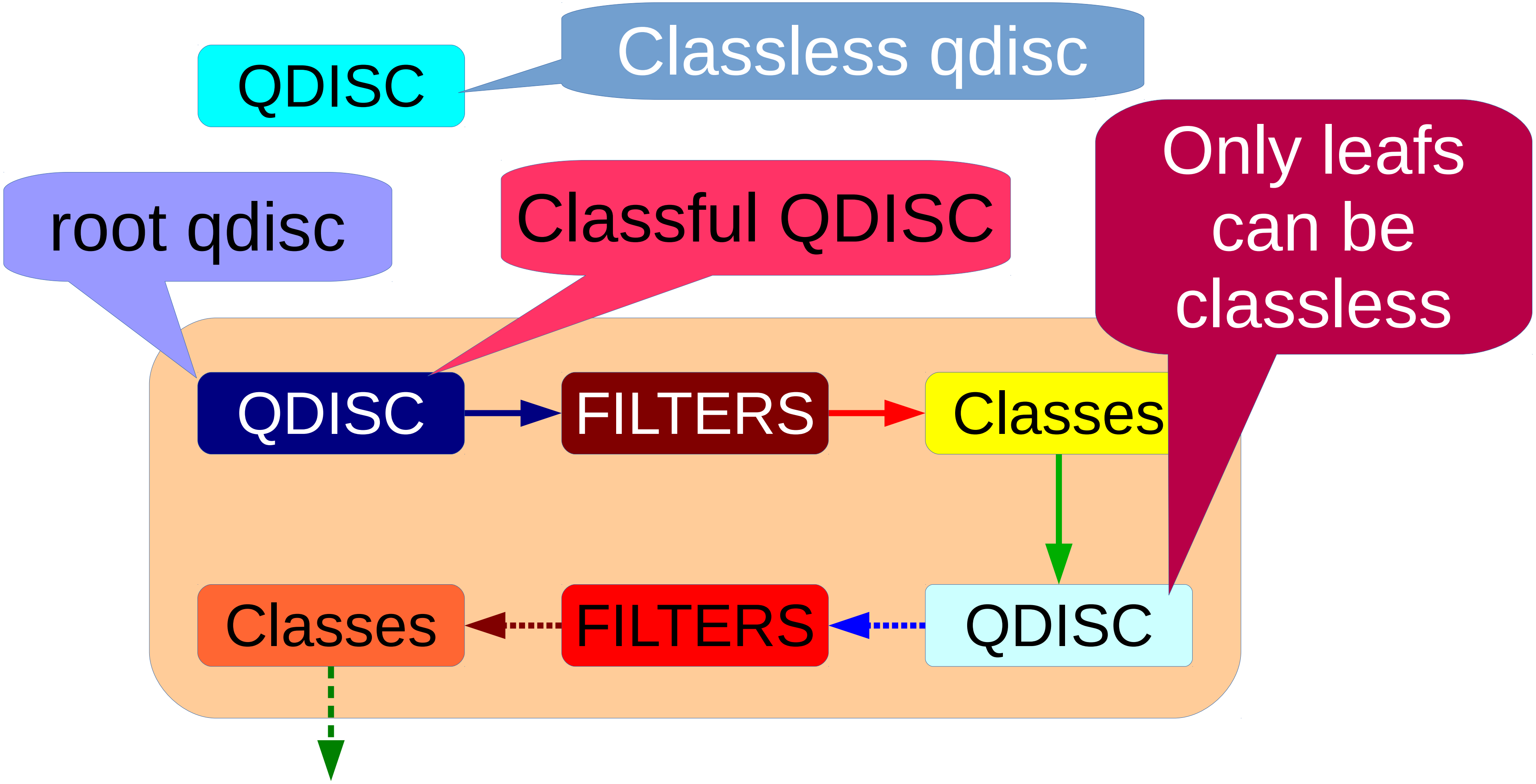
- SHAPING ↑ Control traffic rate of transmission
- SCHEDULING ↑ Packets transmission scheduling (reordering/prioritizing)
- POLICING ↓ Rules to apply to ingress traffic
- DROPPING ↓↑ Clipping of traffic a set bandwidth

Not covered!

↓ = ingress ↑ = egress

7 How is Traffic Controlled?

Traffic Control uses Queue Disciplines:



8 Queueing Schedulers

1★	**pfifo_fast**	three-band packet-FIFO (default classless qdisc)
2	**prio**	priority queueing discipline (classful)
3	**pfifo**	packet-limited FIFO
4	**bfifo**	byte-limited FIFO
5	**cbq**	Class Based Queueing
6	**htb**	Hierarchical Token Bucket (replacement for CBQ, 2.4.20)
7	**tbf**	Token Bucket Filter
8	**red**	Random Early Detection
9	**choke**	Choose and Keep for (un)responsive flow
10	**codel**	Controlled-Delay Active Queue Management
11	**drr**	Deficit Round Robin scheduler

★/proc/sys/net/core/default_qdisc

8 Queueing Schedulers

12	**fq_codel**	Fair Queuing (FQ) with Controlled Delay
13	**hfsc**	Hierarchical Fair Service Curve
15	**mqprio**	Multiqueue Priority Qdisc
15	**sfb**	Stochastic Fair Blue
16	**sfq**	Stochastic Fairness Queueing
17	**stab**	Generic size table manipulations
18	**mq**	Multiqueue dummy scheduler, aka RSS (Receive-Side-Scaling)
19	**cake** EXP!	Common Applications Kept Enhanced (enhanced htb, fq_codel)

8 Queueing Schedulers

Can be attached to qdiscs for filtering:

1	**ematch**		Extended matches for use with "basic" or "flow" filters
2	**bpf**	NEW!	BPF programmable classifier and actions (3.18)
	2a	**cBPF**	Classic Berkeley Packet Filter
	2b	**eBPF**	Extended Berkeley Packet Filter

cBPF actually always executes **eBPF**

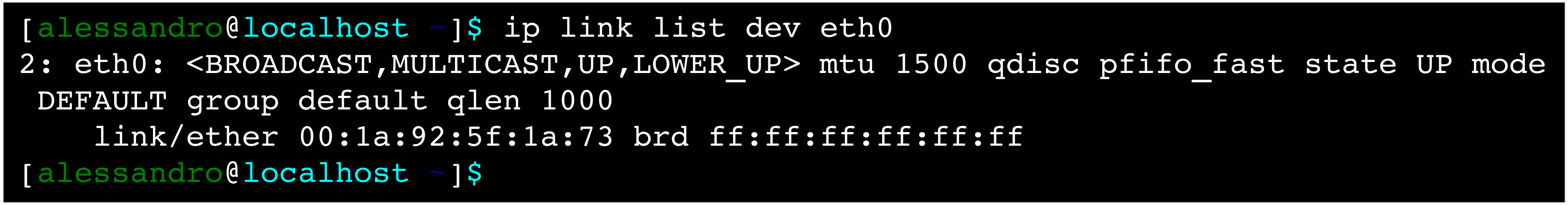
8 Queueing Schedulers

Classfull qdiscs use one of three methods to classify packets:

1) Type Of Service/Differentiated Services
2) filters
3) `skb->priority` field, i.e. SO_PRIORITY option set by application

8 Queueing Schedulers

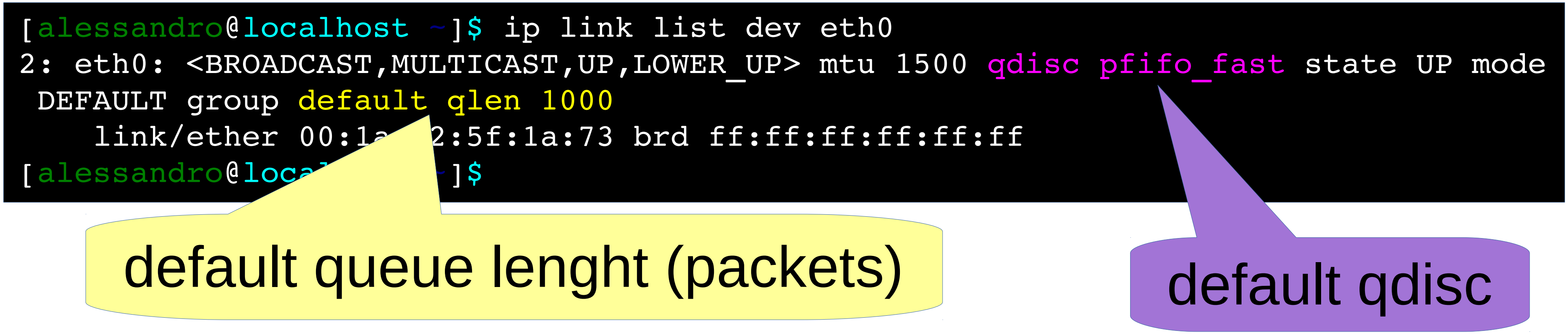
Root qdisc and default queue length:



```
[alessandro@localhost ~]$ ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[alessandro@localhost ~]$
```

8 Queueing Schedulers

Root qdisc and default queue length:



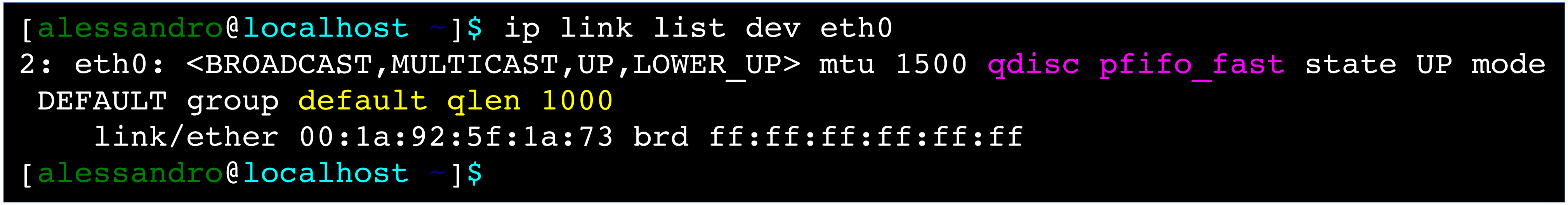
```
[alessandro@localhost ~]$ ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:1a    2:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[alessandro@loca        ~]$
```

default queue lenght (packets)

default qdisc

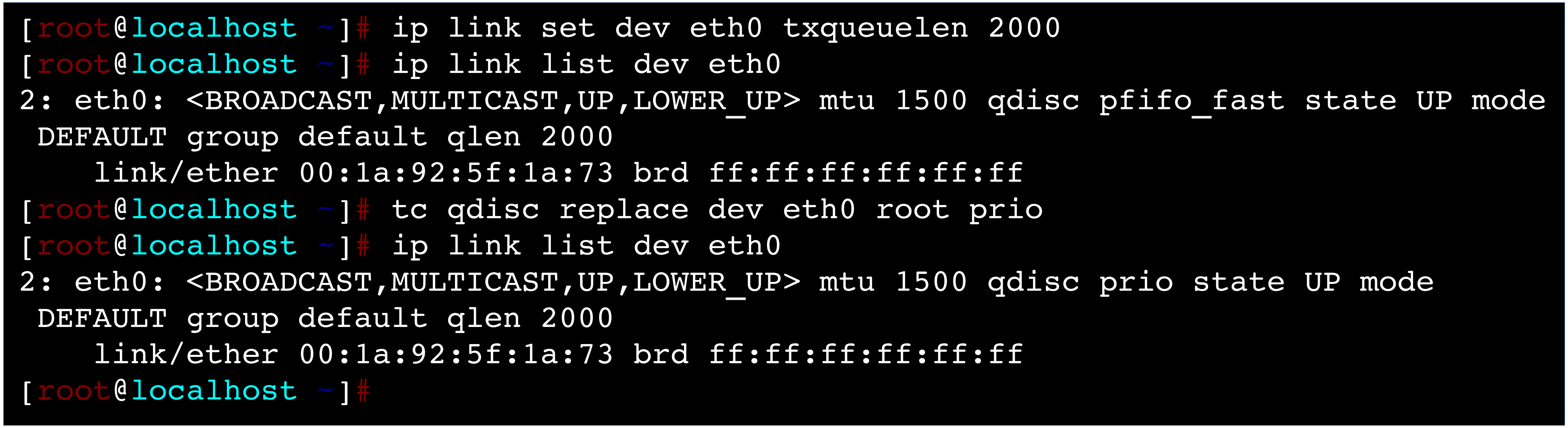
8 Queueing Schedulers

Root qdisc and default queue length:



```
[alessandro@localhost ~]$ ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[alessandro@localhost ~]$
```

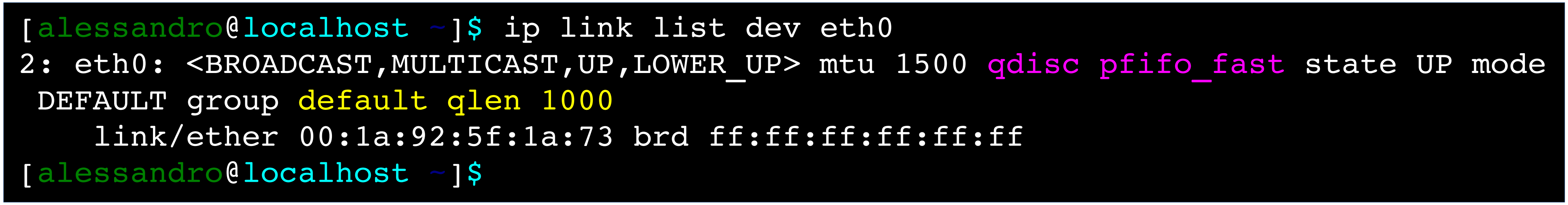
They can be changed this way:



```
[root@localhost ~]# ip link set dev eth0 txqueuelen 2000
[root@localhost ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 2000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[root@localhost ~]# tc qdisc replace dev eth0 root prio
[root@localhost ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc prio state UP mode
 DEFAULT group default qlen 2000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[root@localhost ~]#
```

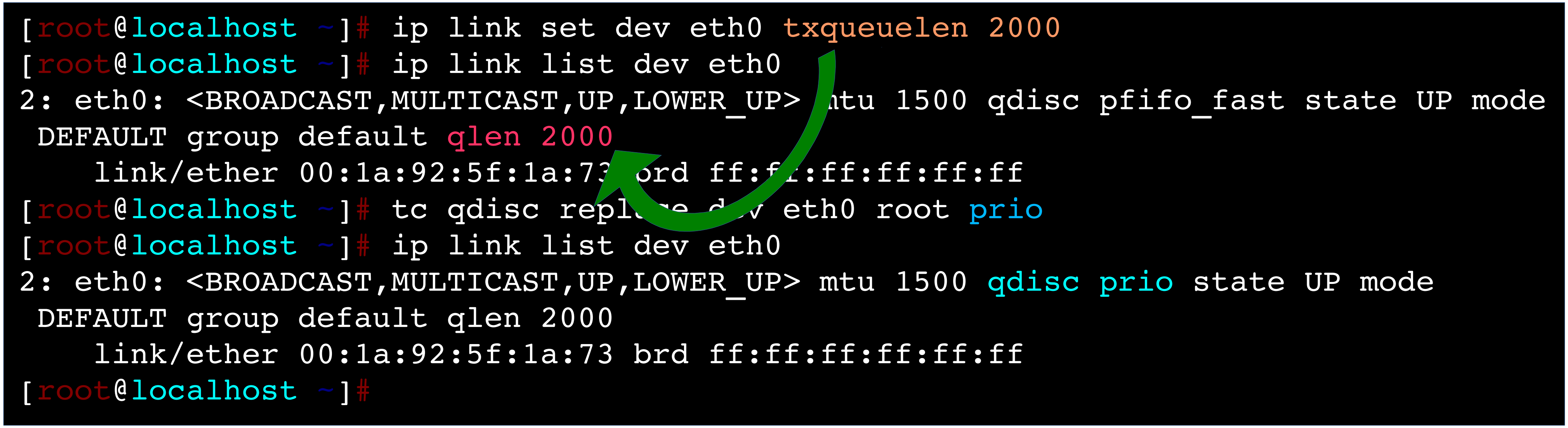
8 Queueing Schedulers

Root qdisc and default queue length:



```
[alessandro@localhost ~]$ ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[alessandro@localhost ~]$
```

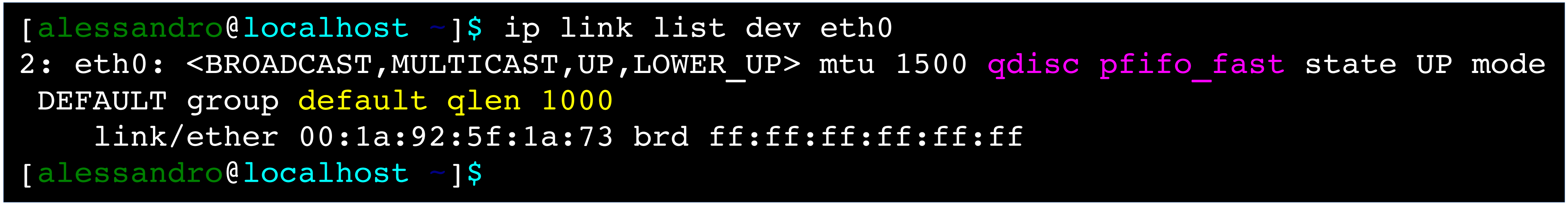
They can be changed this way:



```
[root@localhost ~]# ip link set dev eth0 txqueuelen 2000
[root@localhost ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 2000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[root@localhost ~]# tc qdisc replace dev eth0 root prio
[root@localhost ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc prio state UP mode
 DEFAULT group default qlen 2000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[root@localhost ~]#
```

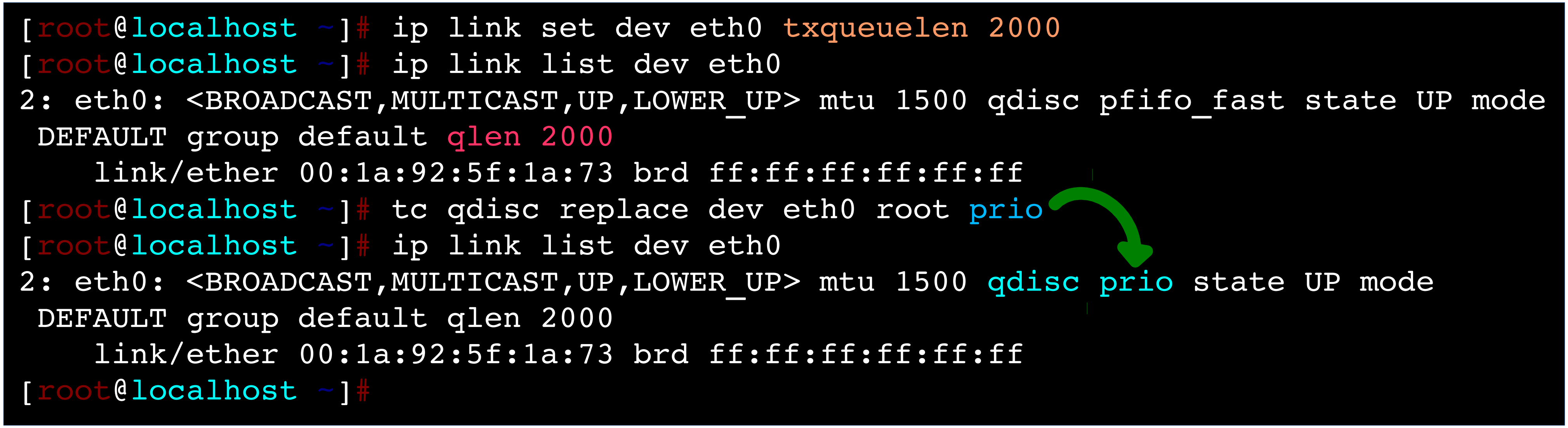
8 Queueing Schedulers

Root qdisc and default queue length:



```
[alessandro@localhost ~]$ ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[alessandro@localhost ~]$
```

They can be changed this way:



```
[root@localhost ~]# ip link set dev eth0 txqueuelen 2000
[root@localhost ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 DEFAULT group default qlen 2000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[root@localhost ~]# tc qdisc replace dev eth0 root prio
[root@localhost ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc prio state UP mode
 DEFAULT group default qlen 2000
    link/ether 00:1a:92:5f:1a:73 brd ff:ff:ff:ff:ff:ff
[root@localhost ~]#
```

8 Queueing Schedulers

Queues are run by kernel at each jiffy

Jiffies are set to:

- **100** Hz, fixed value, up to all kernels **2.4**
- **1000** Hz, kernels **2.6.0** to **2.6.12**
- Selectable among values **100**, **250** (default) and **1000**, from kernel **2.6.13**
- Beginning kernel **2.6.20**, selectable among values **100**, **250**, **300** and **1000**

```
# CONFIG_HZ_PERIODIC is not set
# CONFIG_HZ_100 is not set
# CONFIG_HZ_250 is not set
```

```
CONFIG_HZ_300=y
# CONFIG_HZ_1000 is not set
CONFIG_HZ=300
```

8 Queueing Schedulers

Queues are run by kernel at each jiffy

!

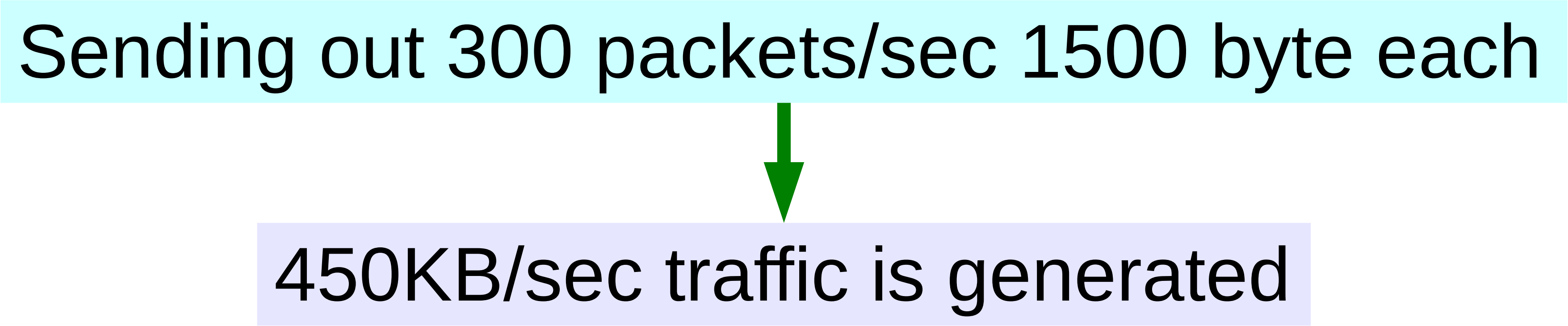
Jiffies are set to:

- **100** Hz, fixed value, up to all kernels **2.4**
- **1000** Hz, kernels **2.6.0** to **2.6.12**
- Selectable among values **100**, **250** (default) and **1000**, from kernel **2.6.13**
- Beginning kernel **2.6.20**, selectable among values **100**, **250**, **300** and **1000**

```
# CONFIG_HZ_PERIODIC is not set
# CONFIG_HZ_100 is not set
# CONFIG_HZ_250 is not set
```

```
CONFIG_HZ_300=y
# CONFIG_HZ_1000 is not set
CONFIG_HZ=300
```

9 Throughput vs Lag



Sending out 300 packets/sec 1500 byte each

450KB/sec traffic is generated

How do I get more?

Of course, first thing that comes to mind is:

- At each jiffie, flush all ready to go packets queued in buffer

Throughput vs Lag

Ways to use more bandwith/lower load:

- Jumbo frames (9000 byte packets, old idea, did not became a standard)
- LRO, Large Receive Offload[1] (and friends: TSO or LSO, UFO, GSO, since 2.6.18)
- Qdiscs queue not single packets data, but descriptors to SKB that hold several packets

1) Available in NAPI drivers
2) **T**CP **S**egmentation **O**ffload, aka **L**arge **S**egmentation **O**ffload, **U**DP **F**ragmentation **O**ffload, **G**eneric **S**egmentation **O**ffload.

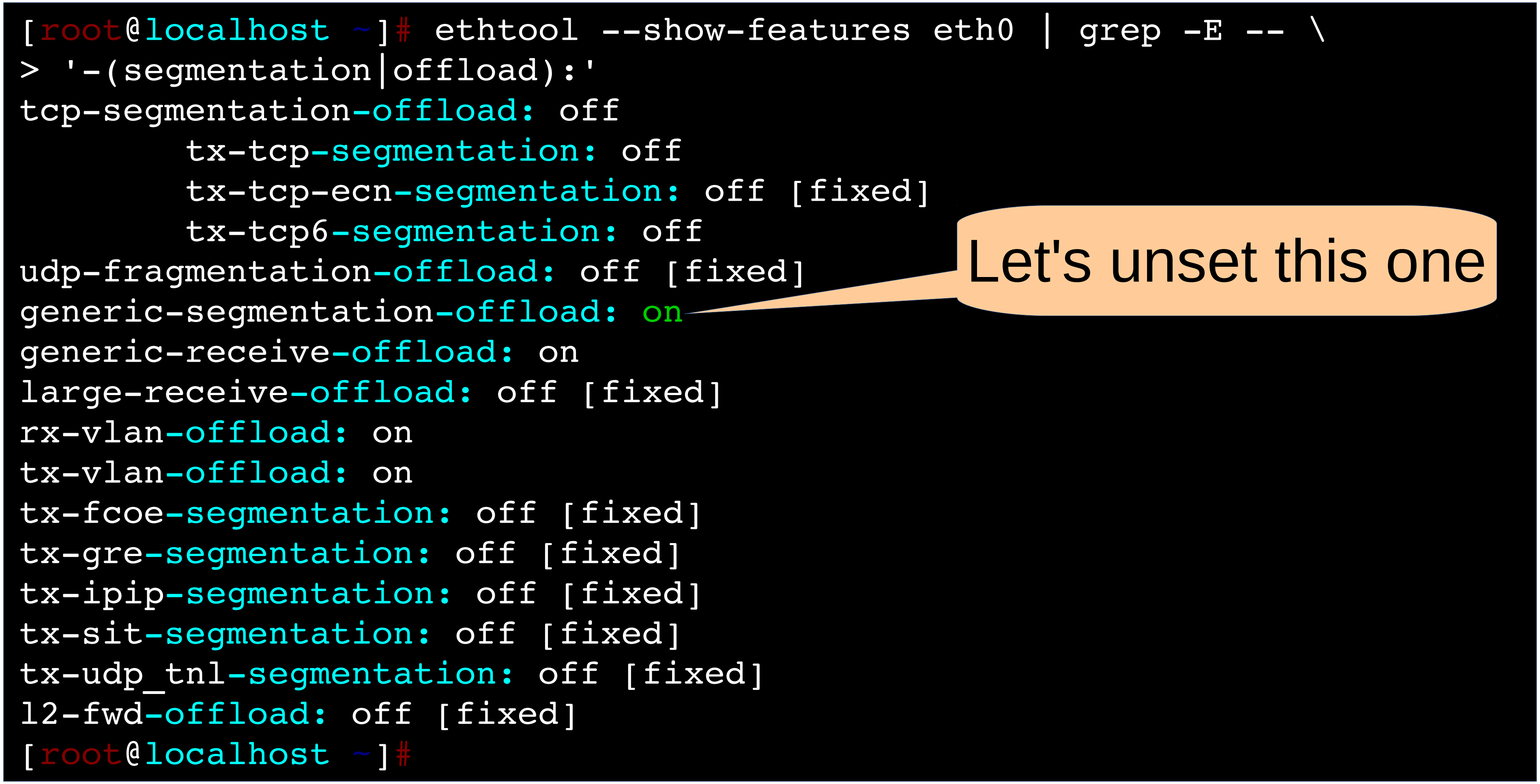
Throughput vs Lag

How large a packet can a SKB hold?

- SKB can hold larger than 1500 byte packets
- For Ipv4, the top limit is **65,536** bytes (Total Lenght header field is 16bit)
- NIC hardware splits this into <= MTU units
  - Qdisc queues SKB descriptors of super-packets sized > 1500 bytes
  - Software or hardware splits them before putting them on the wire

9 Throughput vs Lag

Settings visible/settable with `ethtool`:

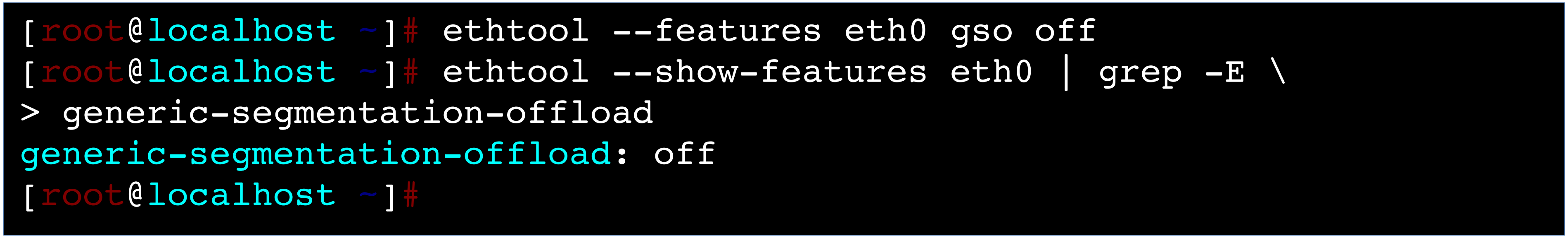


```
[root@localhost ~]# ethtool --show-features eth0 | grep -E -- \
> '-(segmentation|offload):'
tcp-segmentation-offload: off
        tx-tcp-segmentation: off
        tx-tcp-ecn-segmentation: off [fixed]
        tx-tcp6-segmentation: off
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off [fixed]
rx-vlan-offload: on
tx-vlan-offload: on
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: off [fixed]
tx-ipip-segmentation: off [fixed]
tx-sit-segmentation: off [fixed]
tx-udp_tnl-segmentation: off [fixed]
l2-fwd-offload: off [fixed]
[root@localhost ~]#
```

Let's unset this one

Throughput vs Lag

Settings visible/settable with `ethtool`:



```
[root@localhost ~]# ethtool --features eth0 gso off
[root@localhost ~]# ethtool --show-features eth0 | grep -E \
> generic-segmentation-offload
generic-segmentation-offload: off
[root@localhost ~]#
```

9 Throughput vs Lag

The larger the queue, the higher the lag...

- Take a 100Mbit link = 12,500,000 bytes/sec
- Take a qdisc buffer of 128 descriptors
- Let's assume 1 descriptor per 1500B packet
- Queue holds 127 high-throughput packets
- One small low-delay UDP packet arrives
  - How long shall it wait before it is sent out?

```
1,500*127/12,500,000=0.01524sec
```

15.24ms! *Far from Real Time!*

9 Throughput vs Lag

The larger the queue, the higher the lag...

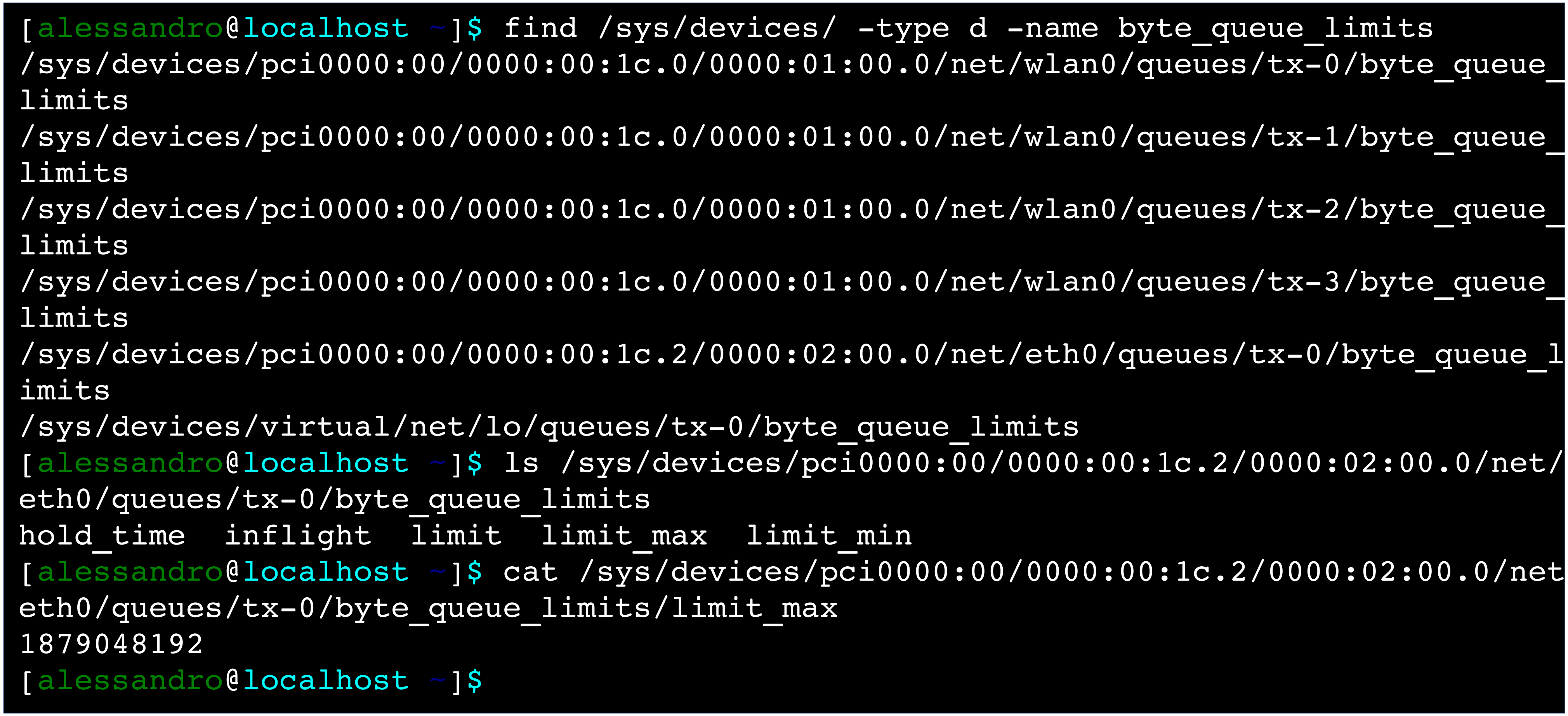
- BQL, Byte Queue Limit, designed (kernel 3.3.0) to dinamically limit the amount of data queued into driver's queue
- It does not resize the buffer size, it regulates it's use
- `/sys` interface directory:

  ```
  find /sys/devices -name byte_queue_limits
  ```
- Available on a limited set of drivers

More about this on http://www.bufferbloat.net/

10 /sys and /proc

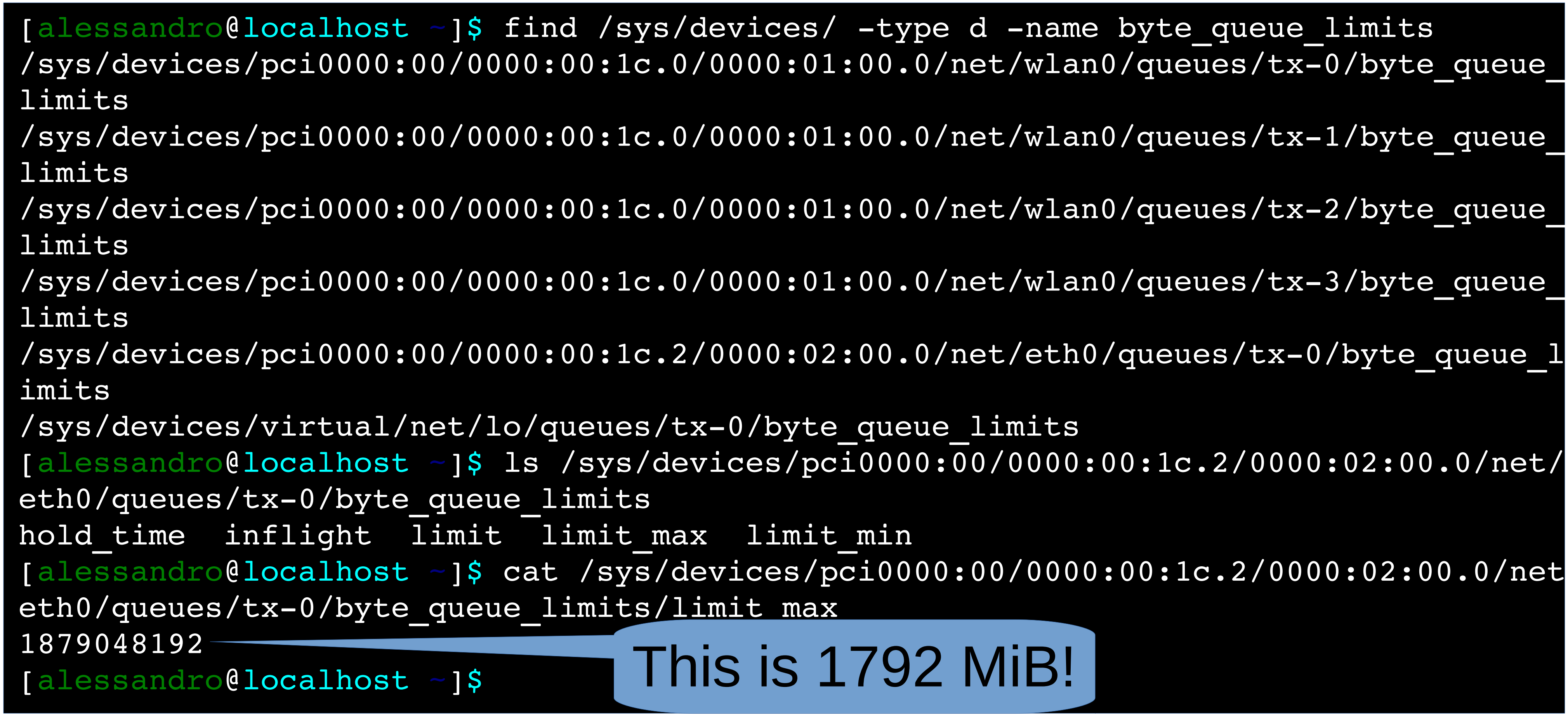
BQL /sys interface directory:



```
[alessandro@localhost ~]$ find /sys/devices/ -type d -name byte_queue_limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-0/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-1/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-2/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-3/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.2/0000:02:00.0/net/eth0/queues/tx-0/byte_queue_l
imits
/sys/devices/virtual/net/lo/queues/tx-0/byte_queue_limits
[alessandro@localhost ~]$ ls /sys/devices/pci0000:00/0000:00:1c.2/0000:02:00.0/net/
eth0/queues/tx-0/byte_queue_limits
hold_time  inflight  limit  limit_max  limit_min
[alessandro@localhost ~]$ cat /sys/devices/pci0000:00/0000:00:1c.2/0000:02:00.0/net
eth0/queues/tx-0/byte_queue_limits/limit_max
1879048192
[alessandro@localhost ~]$
```

10 /sys and /proc

BQL /sys interface directory:

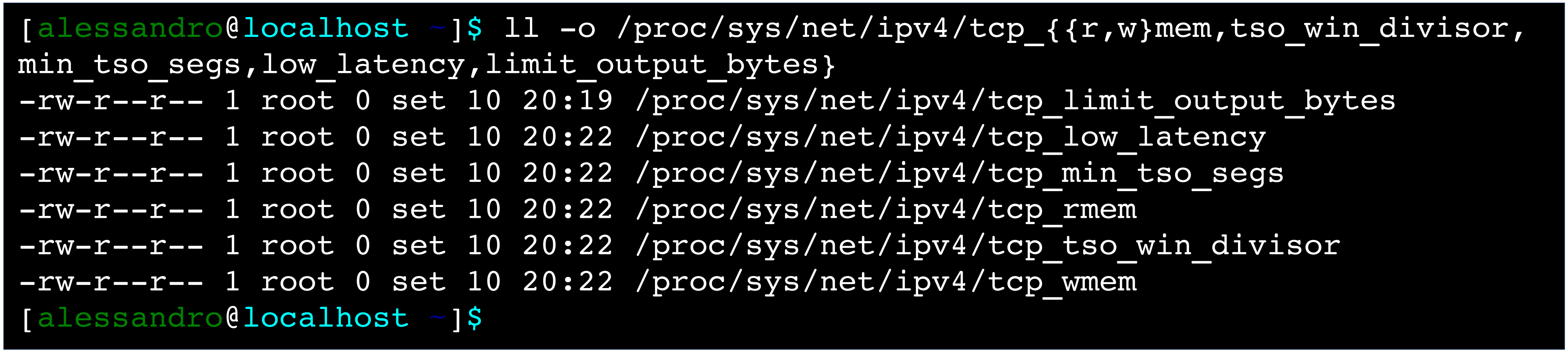


```
[alessandro@localhost ~]$ find /sys/devices/ -type d -name byte_queue_limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-0/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-1/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-2/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0/net/wlan0/queues/tx-3/byte_queue_
limits
/sys/devices/pci0000:00/0000:00:1c.2/0000:02:00.0/net/eth0/queues/tx-0/byte_queue_l
imits
/sys/devices/virtual/net/lo/queues/tx-0/byte_queue_limits
[alessandro@localhost ~]$ ls /sys/devices/pci0000:00/0000:00:1c.2/0000:02:00.0/net/
eth0/queues/tx-0/byte_queue_limits
hold_time  inflight  limit  limit_max  limit_min
[alessandro@localhost ~]$ cat /sys/devices/pci0000:00/0000:00:1c.2/0000:02:00.0/net
eth0/queues/tx-0/byte_queue_limits/limit_max
1879048192
[alessandro@localhost ~]$
```

This is 1792 MiB!

10 /sys and /proc

/proc interface files to take note of:



```
[alessandro@localhost ~]$ ll -o /proc/sys/net/ipv4/tcp_{{r,w}mem,tso_win_divisor,
min_tso_segs,low_latency,limit_output_bytes}
-rw-r--r-- 1 root 0 set 10 20:19 /proc/sys/net/ipv4/tcp_limit_output_bytes
-rw-r--r-- 1 root 0 set 10 20:22 /proc/sys/net/ipv4/tcp_low_latency
-rw-r--r-- 1 root 0 set 10 20:22 /proc/sys/net/ipv4/tcp_min_tso_segs
-rw-r--r-- 1 root 0 set 10 20:22 /proc/sys/net/ipv4/tcp_rmem
-rw-r--r-- 1 root 0 set 10 20:22 /proc/sys/net/ipv4/tcp_tso_win_divisor
-rw-r--r-- 1 root 0 set 10 20:22 /proc/sys/net/ipv4/tcp_wmem
[alessandro@localhost ~]$
```

...just in case you didn't have enough of knobs, levers, dials, buttons, switches, throttles, valves, levees, readings, metres, settings, options, queues, limits, buffers, rings, warnings, lights, sirens, signs, tables, alarms, interfaces, keys, ...

`pfifo_fast` qdisc

Simple, fast and default Queueing Discipline:

- `pfifo_fast`

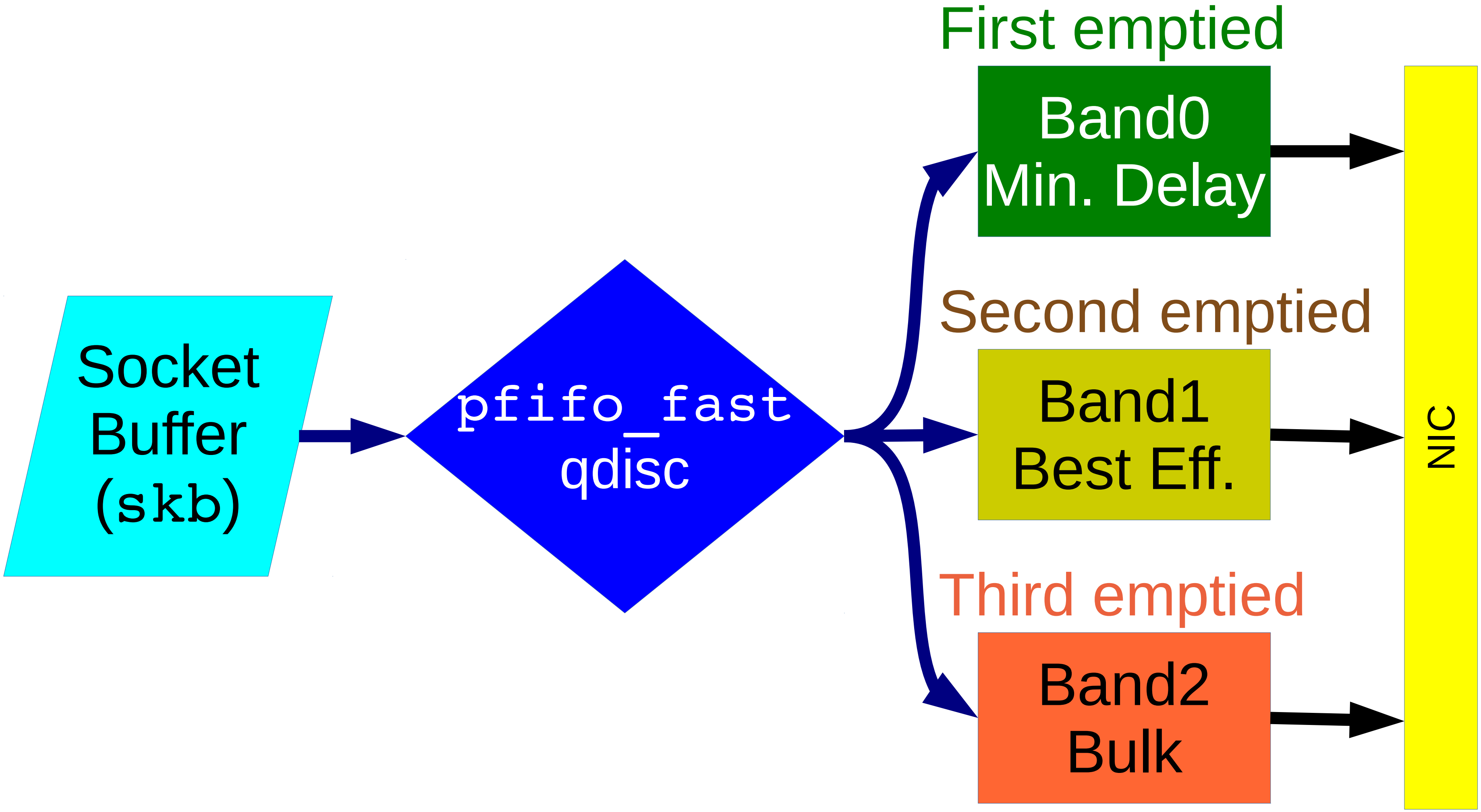
FIFO queue: first packet arrived is the first served

Three-band FIFO queue organization:

0 = Minimum Delay/Interactive
1 = Best Effort
2 = Bulk

Kernel maps them to DSCP (prev. TOS) bits

11 pfifo_fast qdisc



pfifo_fast qdisc

RFC 1349 (1992) defined TOS like this:

0	1	2	3	4	5	6	7
			D	T	R	MC	0
Precedence			TOS				MBZ[1]

MMC	Min. Monetary Cost	MT	Max. Throughput
MR	Max. Reliability	MD	Min. Delay

[1] MBZ = Must Be Zero

pfifo_fast qdisc

RFC 1349 (1992) defined TOS like this:

0	1	2	3	4	5	6	7
			D	T	R	MC	0
Precedence			TOS				MBZ

In Tuxese:

bit6	Filler	bit4	Bulk
bit5	Best Effort	bit3	Interactive

pfifo_fast qdisc

Linux mapped TOS bits into bands:

Bits	TOS	Band	Bits	TOS	Band
0000	Normal Service	1	1000	Min. Delay	0
0001	Min. Monetary Cost	2	1001	mmc+md	0
0010	Max. Reliability	1	1010	mr+md	0
0011	mmc+mr	1	1011	mmc+mr+md	0
0100	Max. Throughput	2	1100	mt+md	1
0101	mmc+mt	2	1101	mmc+mt+md	1
0110	mr+mt	2	1110	mr+mt+md	1
0111	mmc+mr+mt	2	1111	mmc+mr+mt+md	1

pfifo_fast qdisc

RFC 2474 (1998) turned TOS into DS and
RFC 3168 (2001) added ECN:

0	1	2	3	4	5	6	7
Diff. Services Code Point						X	X
Differentiated Services (traffic classes)						ECN	

- DSCP indexes up to 64 distinct Per Hop Behaviours
- Default Forwarding PHB is the only mandatory one

ToS/DS-Prio Mappings

linux/net/sched/sch_generic.c:

```
static const u8 prio2band[TC_PRIO_MAX + 1] = {
        1, 2, 2, 2, 1, 2, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1
};
```

```
static int pfifo_fast_enqueue(struct sk_buff *skb, struct Qdisc *qdisc)
{
        if (skb_queue_len(&qdisc->q) < qdisc_dev(qdisc)->tx_queue_len) {
                int band = prio2band[skb->priority & TC_PRIO_MAX];
                struct pfifo_fast_priv *priv = qdisc_priv(qdisc);
                struct sk_buff_head *list = band2list(priv, band);

                priv->bitmap |= (1 << band);
                qdisc->q.qlen++;
                return __qdisc_enqueue_tail(skb, qdisc, list);
        }

        return qdisc_drop(skb, qdisc);
}
```

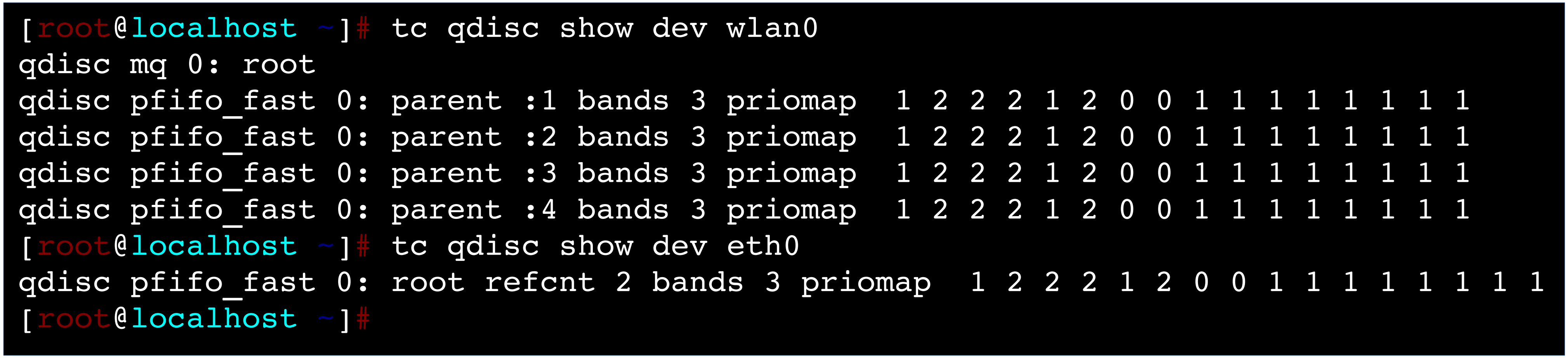
12 ToS/DS-Prio Mappings

DS-to-traffic class mappings are listed in `linux/net/sched/sch_dsmark.c:`

```
/*
 * classid      class           marking
 * -------      -----           -------
 *   n/a          0             n/a
 *   x:0          1             use entry [0]
 *   ...         ...            ...
 *   x:y y>0     y+1            use entry [y]
 *   ...         ...            ...
 * x:indices-1  indices         use entry [indices-1]
 *   ...         ...            ...
 *   x:y         y+1            use entry [y & (indices-1)]
 *   ...         ...            ...
 * 0xffff       0x10000         use entry [indices-1]
 */
```

13

Example: Display Qdisc

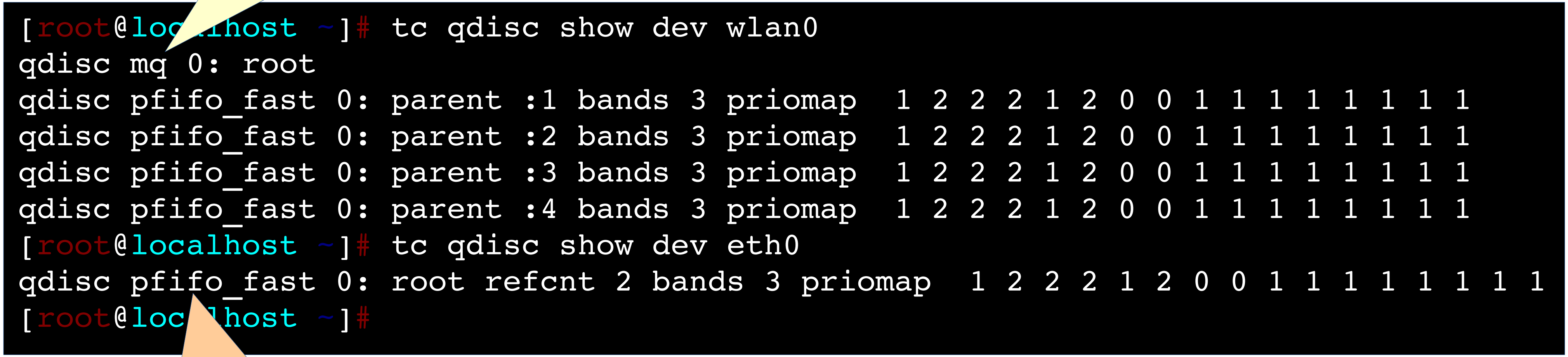


```
[root@localhost ~]# tc qdisc show dev wlan0
qdisc mq 0: root
qdisc pfifo_fast 0: parent :1 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :3 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :4 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@localhost ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@localhost ~]#
```

Priority to band mapping

13 Example: Display Qdisc

Multi-Queue qdisc



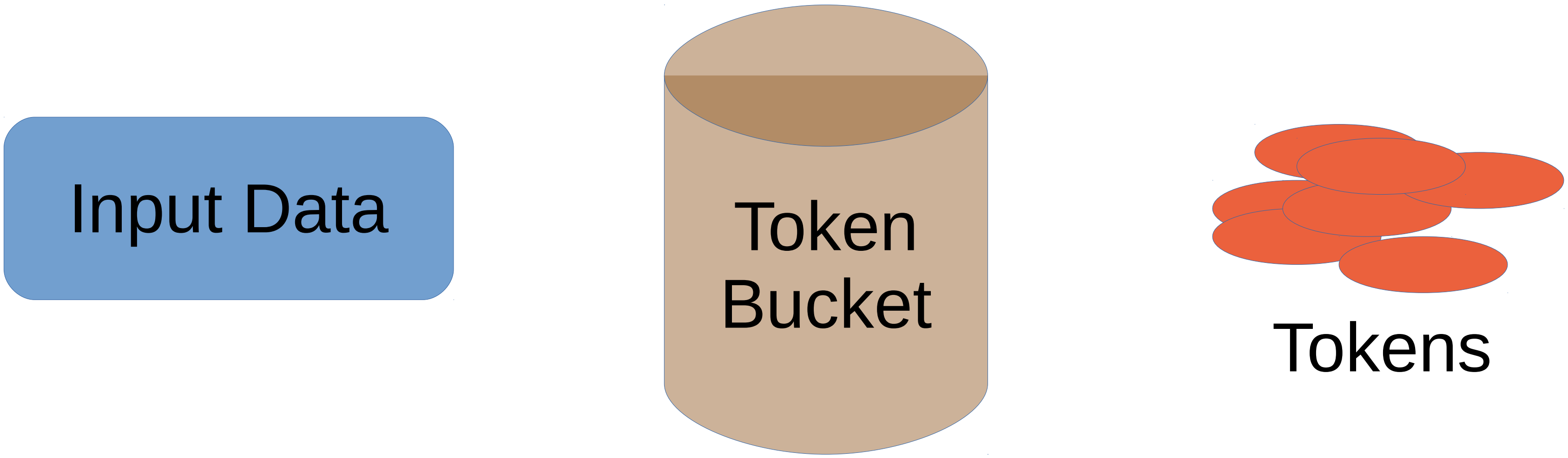
```
[root@localhost ~]# tc qdisc show dev wlan0
qdisc mq 0: root
qdisc pfifo_fast 0: parent :1 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :3 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :4 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@localhost ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@localhost ~]#
```

Default `pfifo_fast` qdisc

Priority to band mapping

The TBF Qdisc

- **TBF**, **Token Bucket Filter**: only shapes traffic, does no scheduling
- Easy qdisk to limit egress traffic
- Simple, low overhead



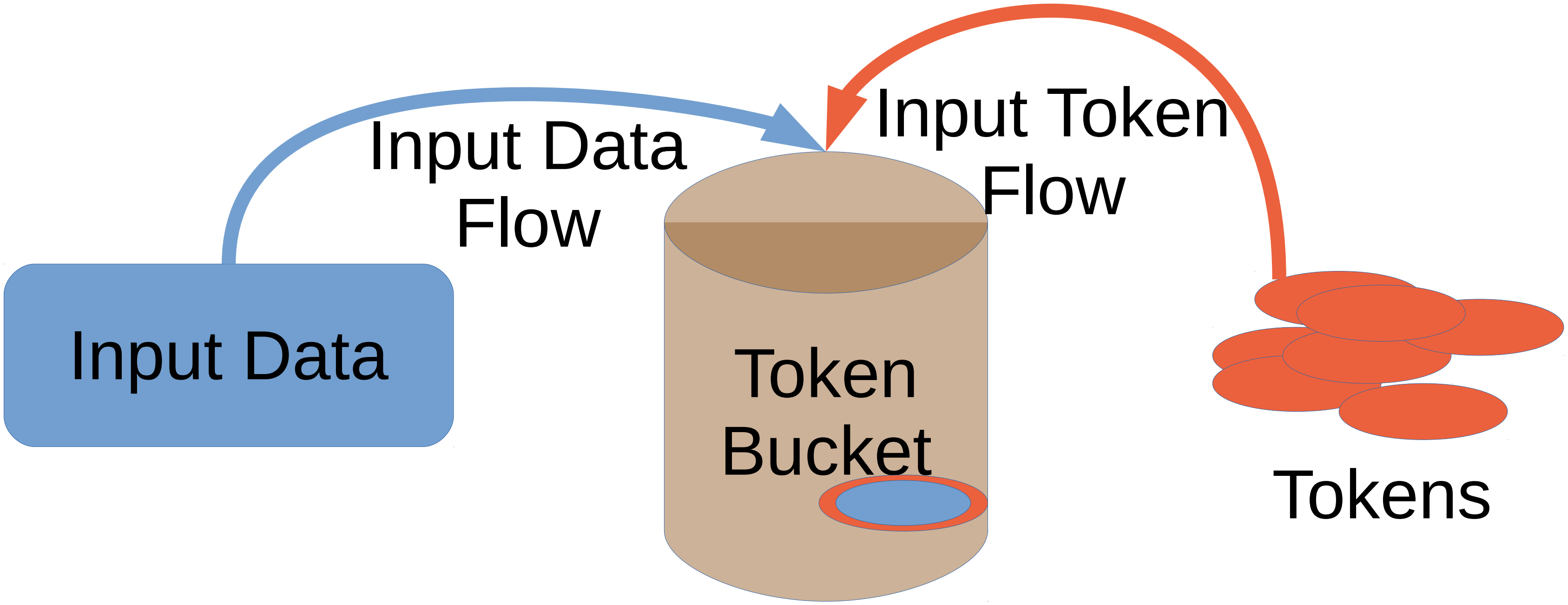
Input Data

Token Bucket

Tokens

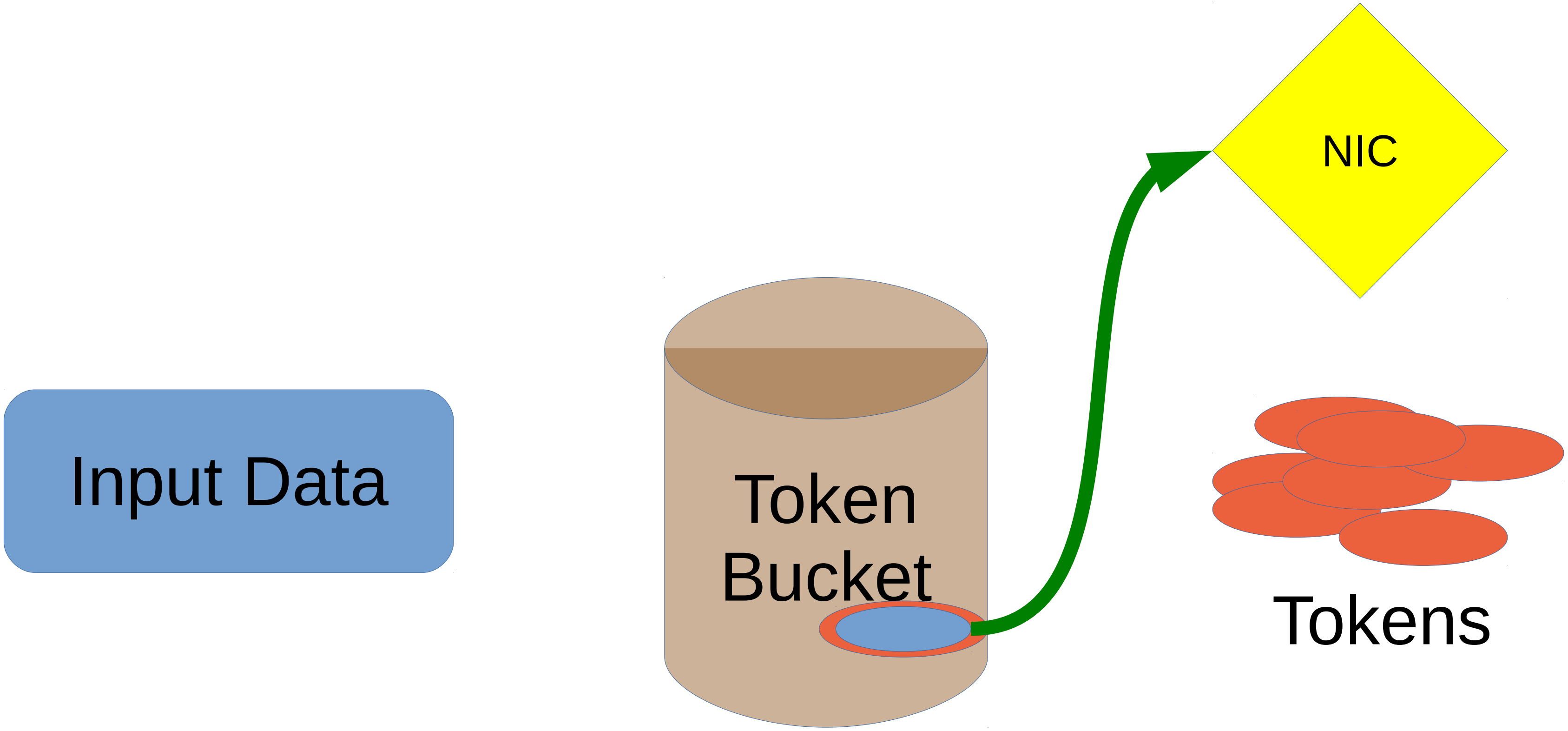
The TBF Qdisc

Token Bucket: it's the qdisc buffer
Tokens: virtual unit of data managed
Token Flow: set number of tokens per unit of time that replenish the bucket



14 The TBF Qdisc

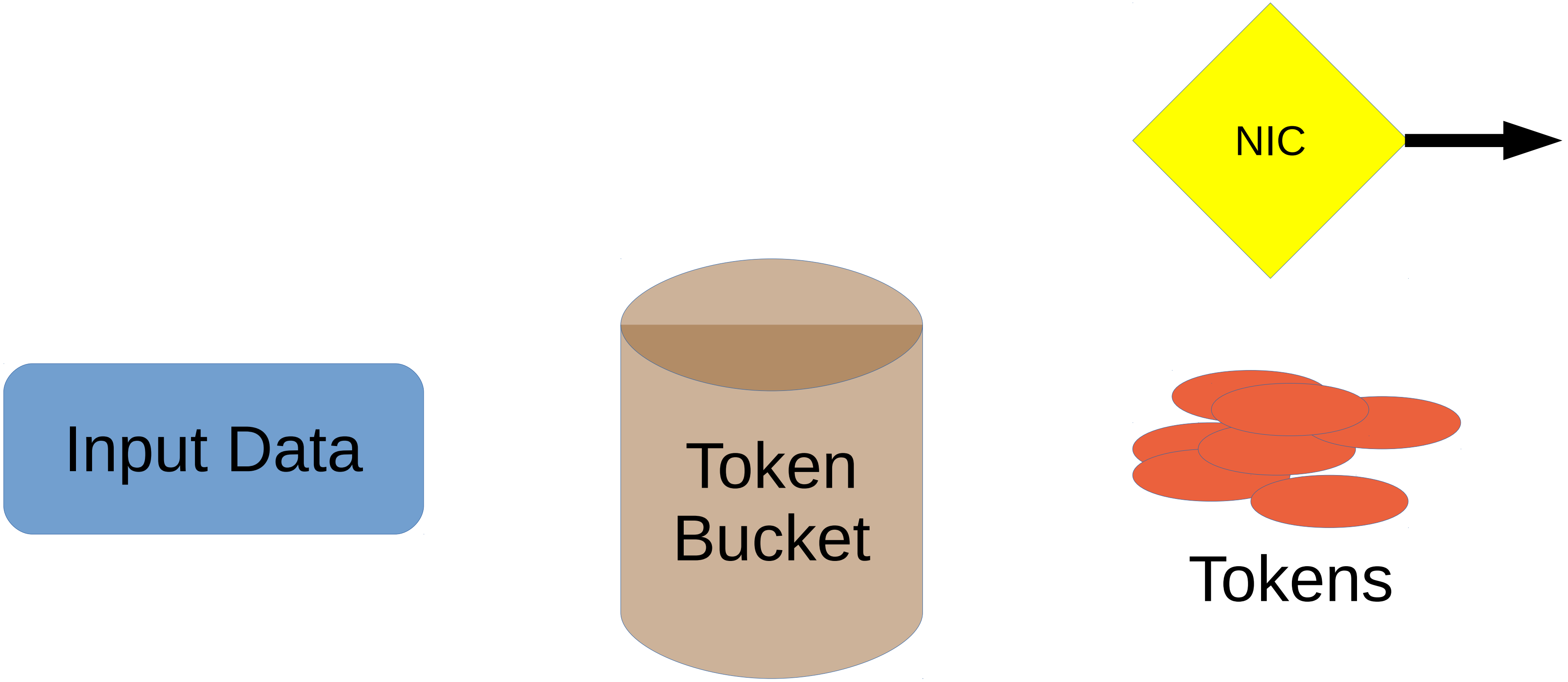
Tokens are removed from bucket when packet is sent



14

The TBF Qdisc

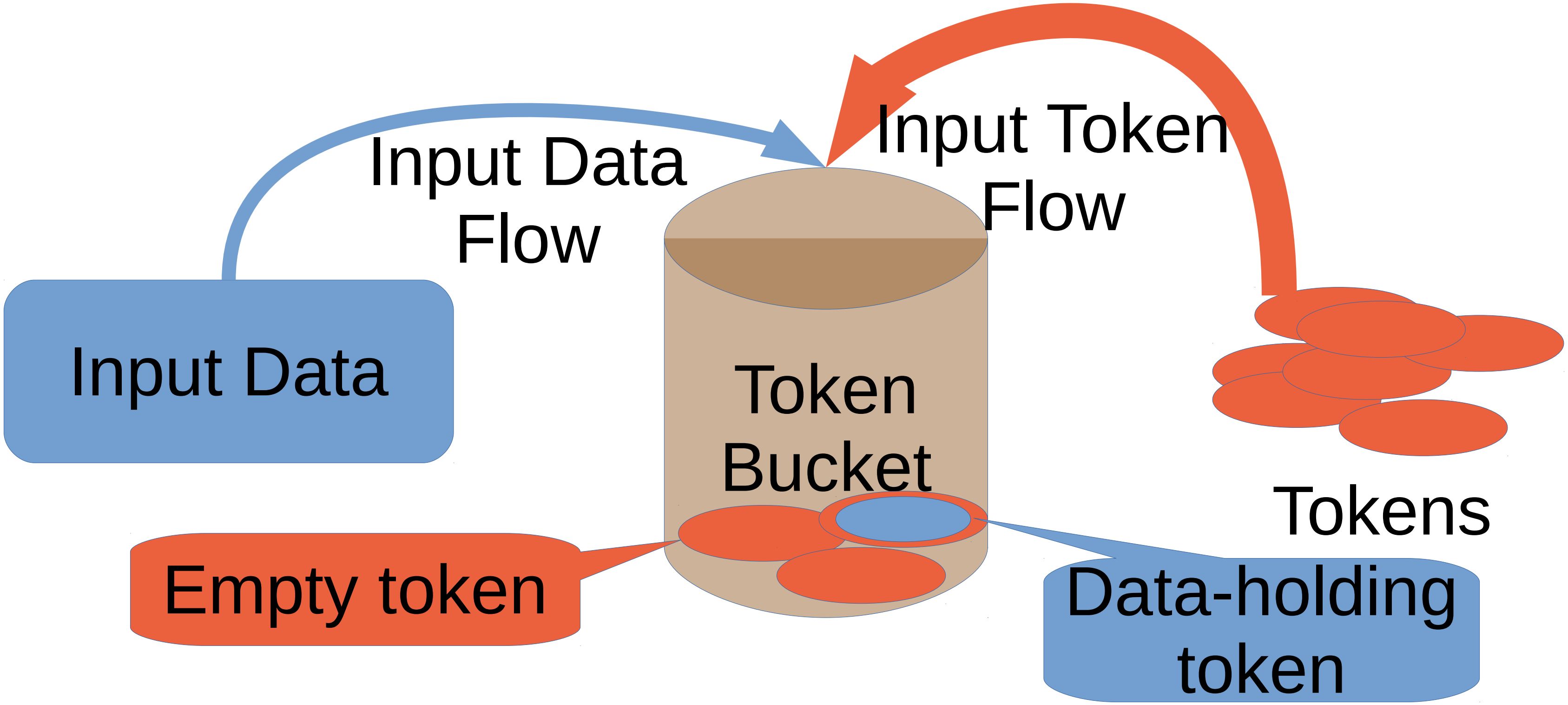
Tokens are removed from bucket when packet is sent



14 The TBF Qdisc

Scenario I

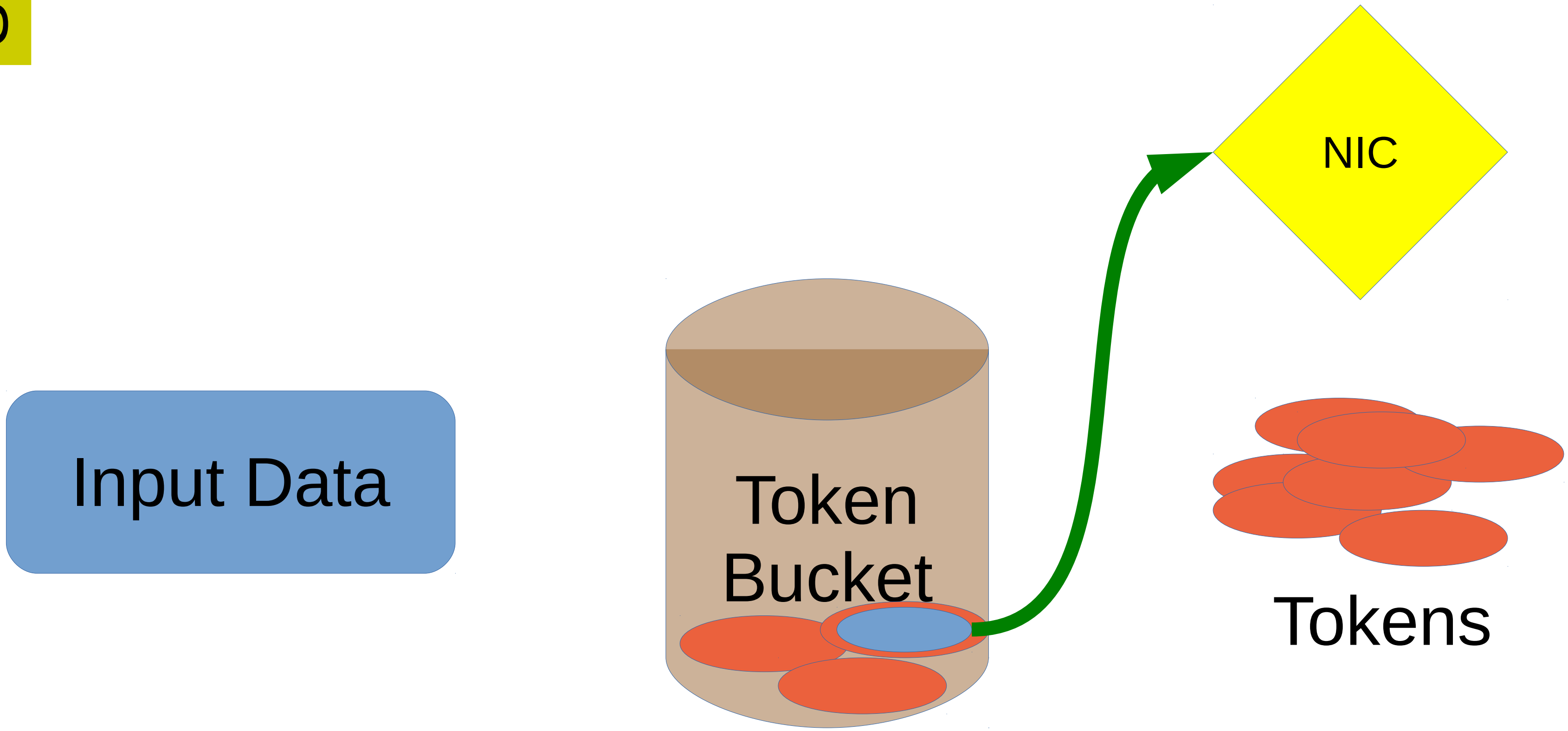
- **Input Token Flow** is faster than **Input Data Flow**
- **Bucket** contains both empty tokens and tokens that hold data to output



The TBF Qdisc

Scenario I

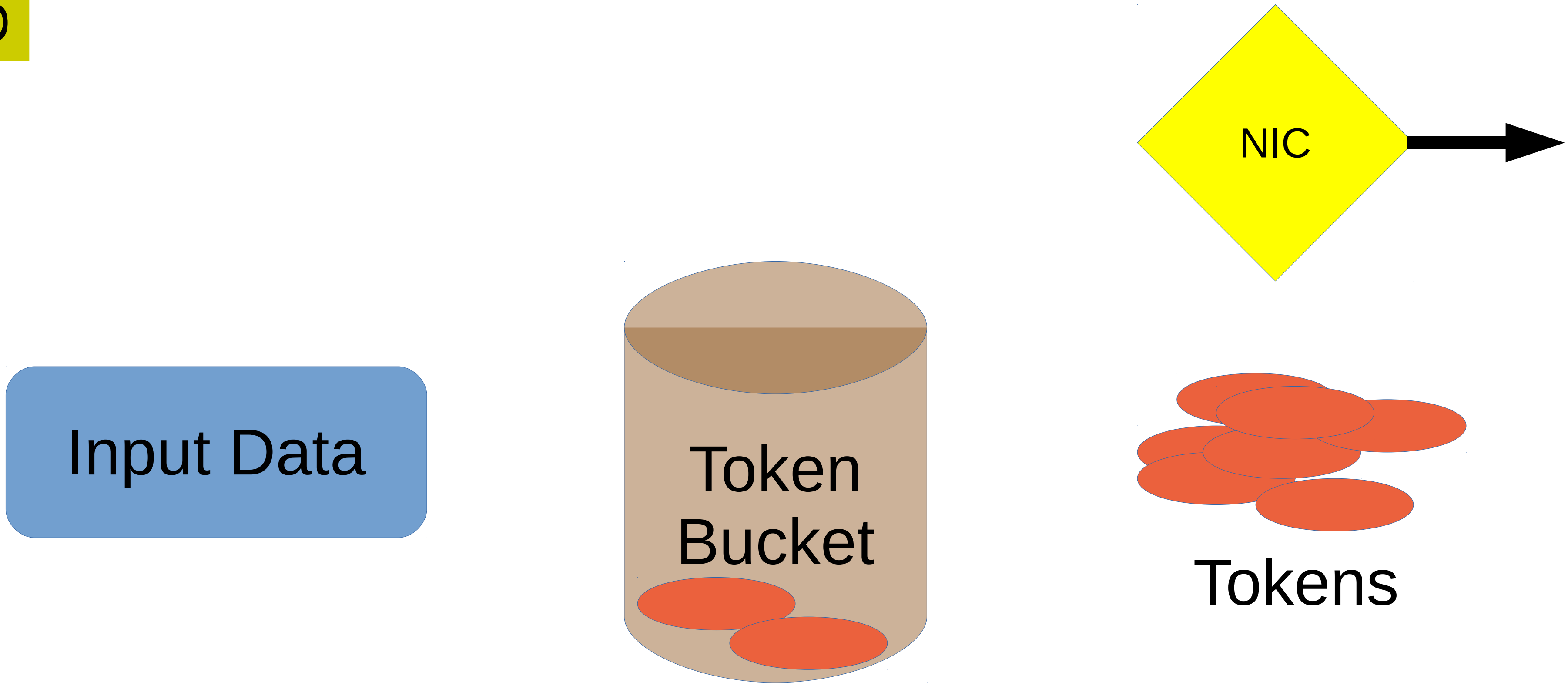
- Data-holding tokens are output and deleted from bucket



The TBF Qdisc

Scenario I

- Data-holding tokens are output and deleted from bucket



NIC

Input Data

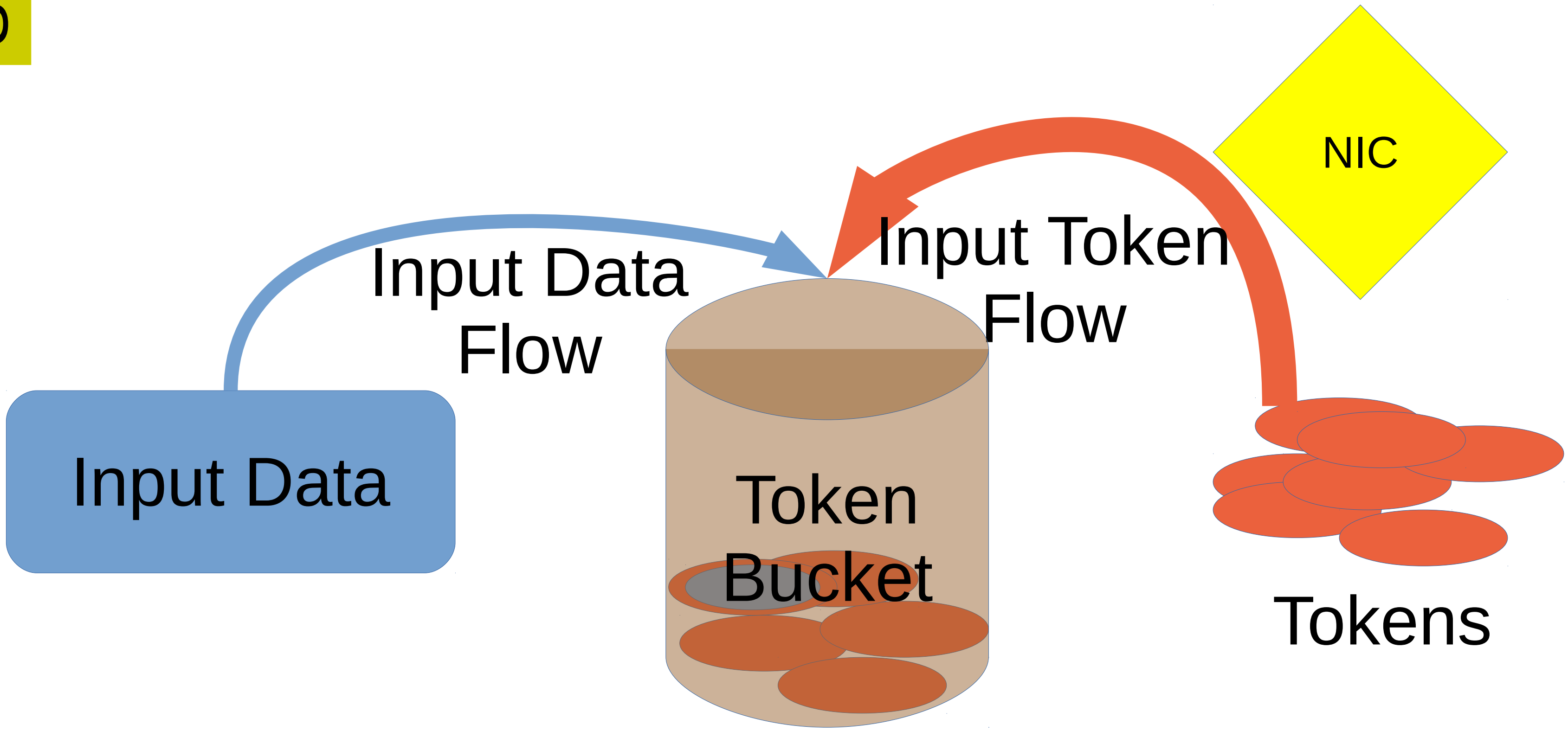
Token Bucket

Tokens

14 The TBF Qdisc

Scenario I

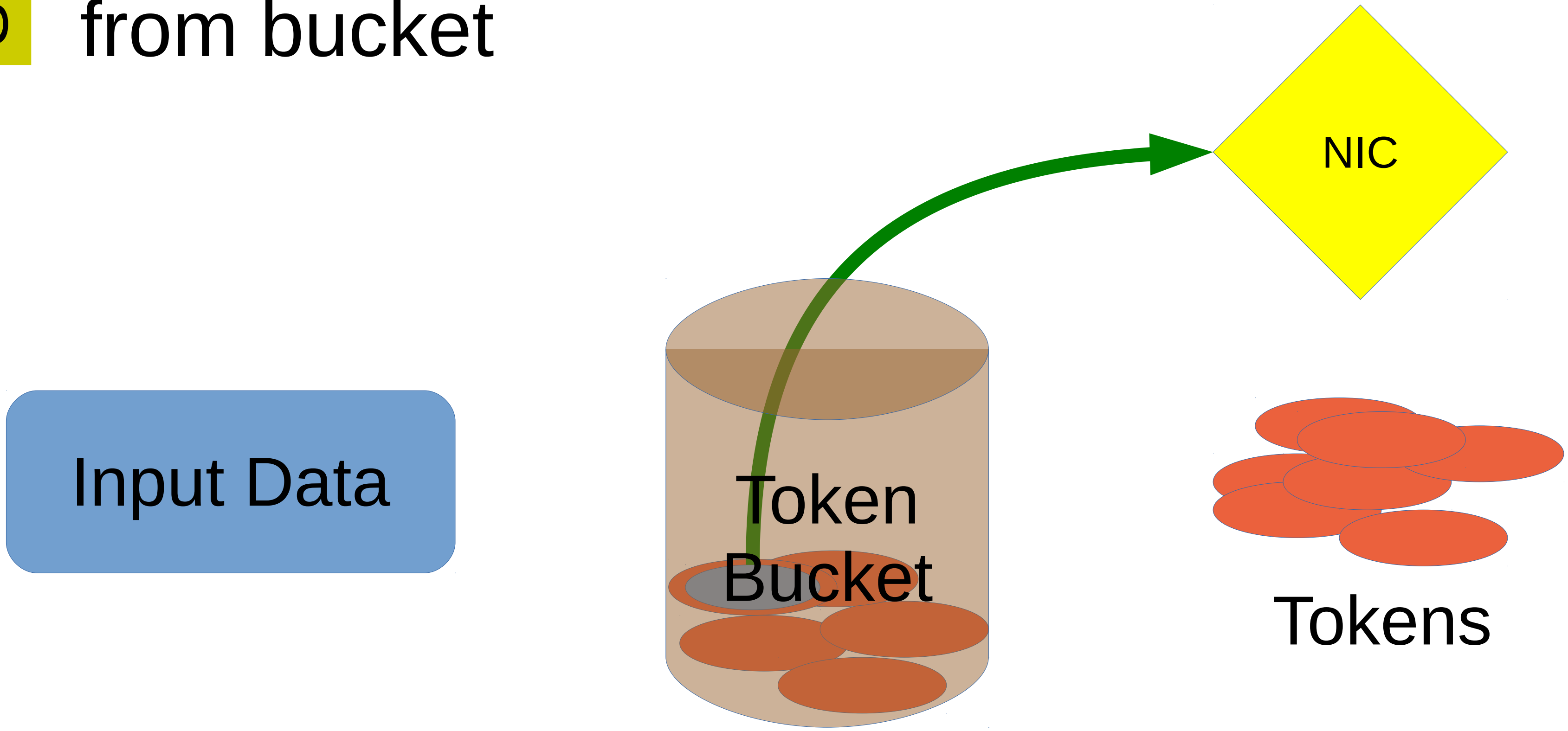
- Empty tokens and data-holding tokens keep filling the bucket



14 The TBF Qdisc

Scenario I

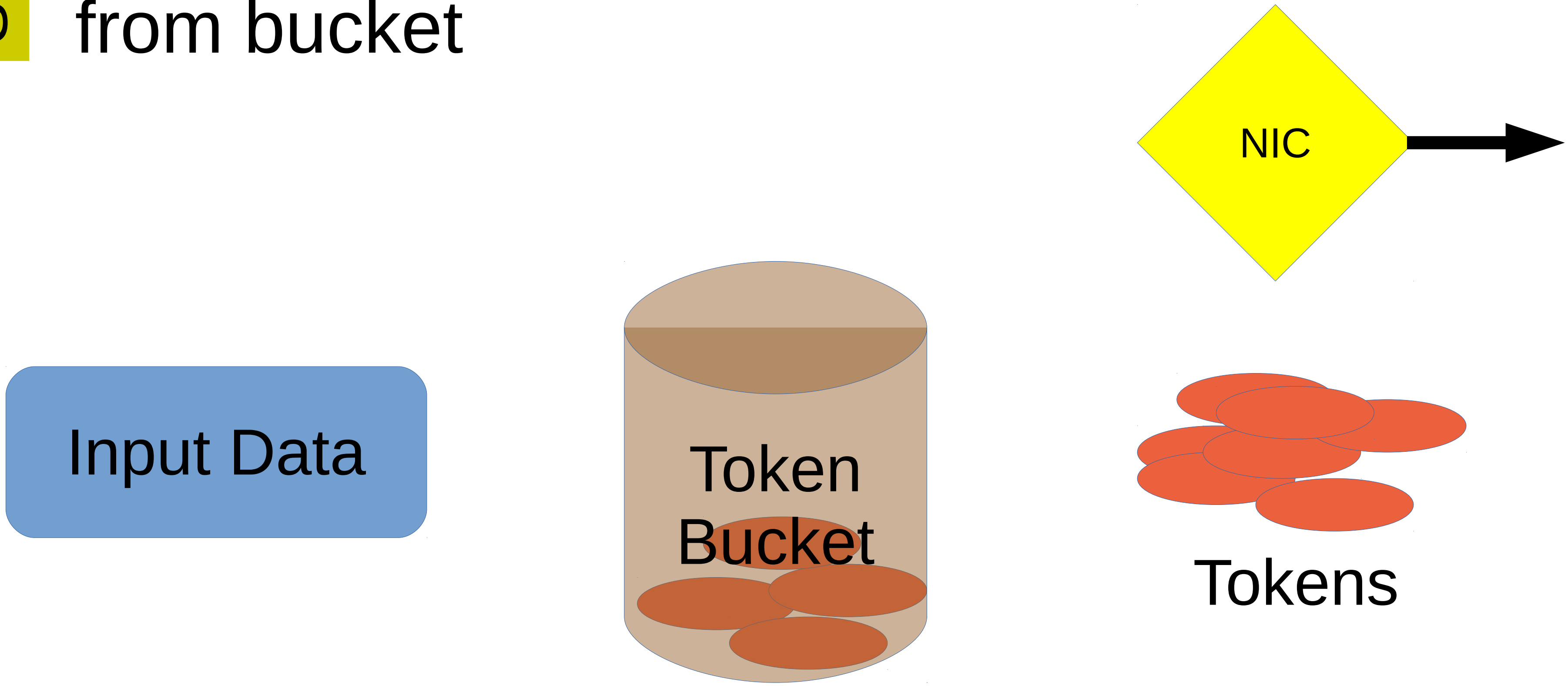
- Empty tokens and data-holding tokens keep filling the bucket
- Data-holding tokens are output and deleted from bucket



14 The TBF Qdisc

Scenario I

- Empty tokens and data-holding tokens keep filling the bucket
- Data-holding tokens are output and deleted from bucket

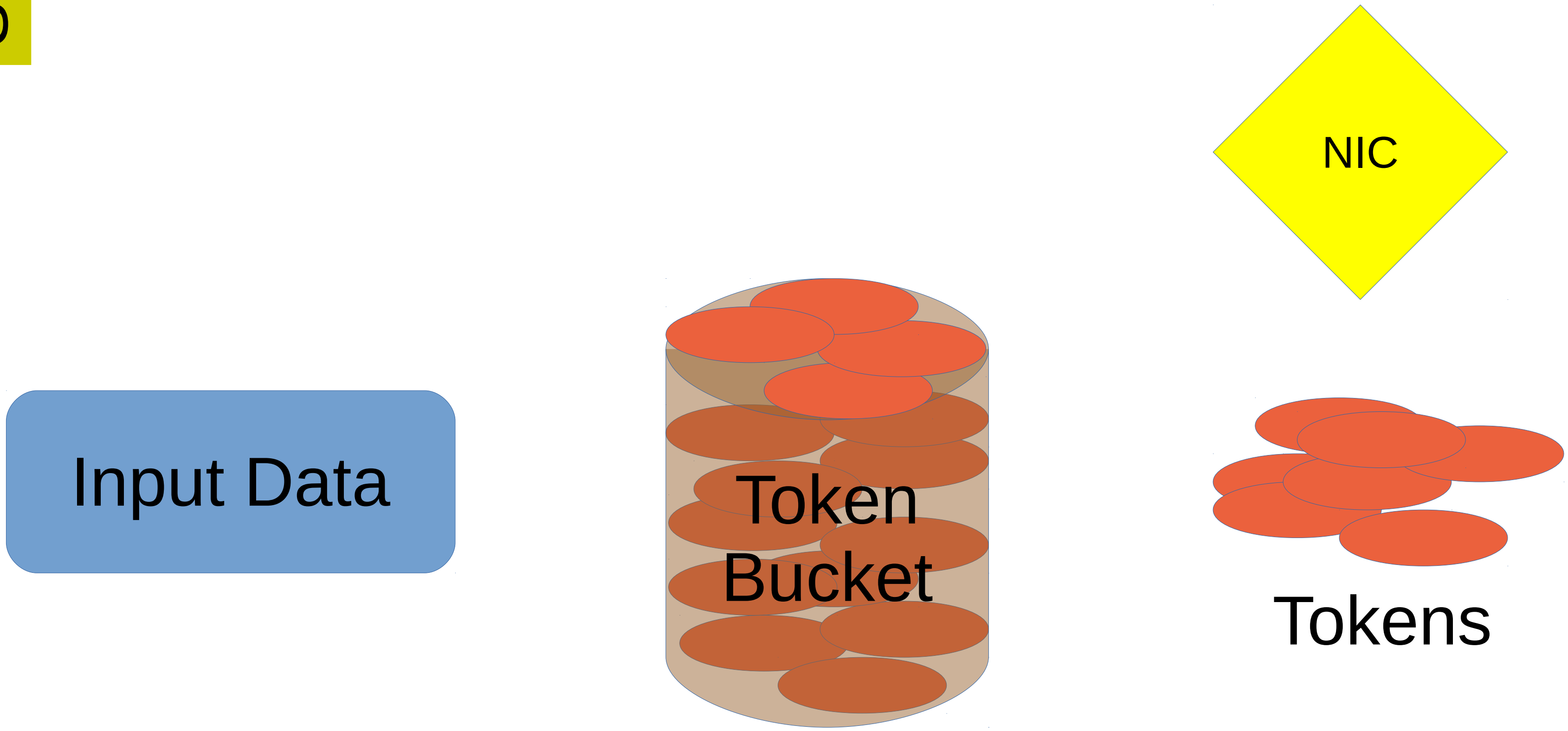


14

The TBF Qdisc

Scenario I

- Eventually the bucket is full of empty tokens.

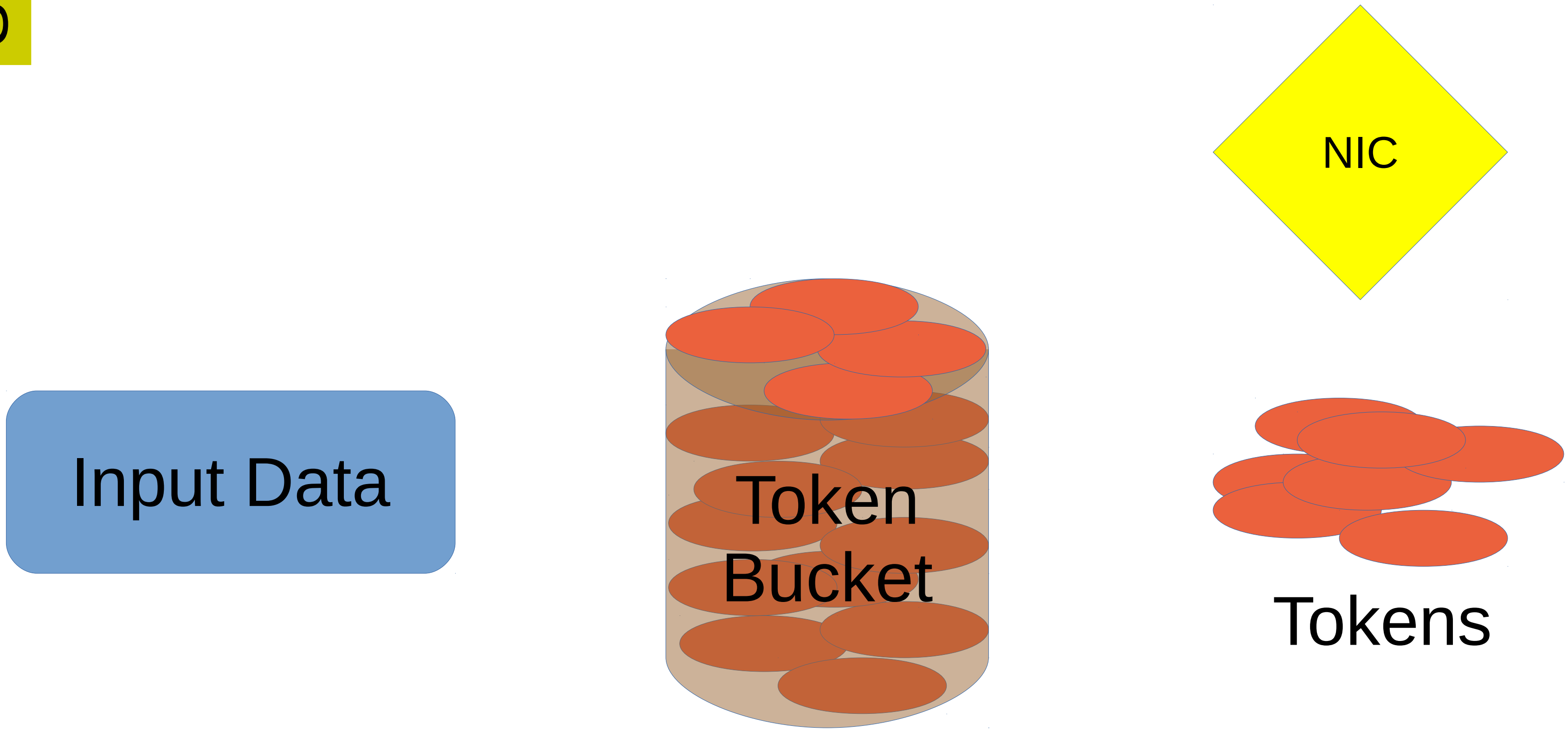


14

The TBF Qdisc

Scenario I

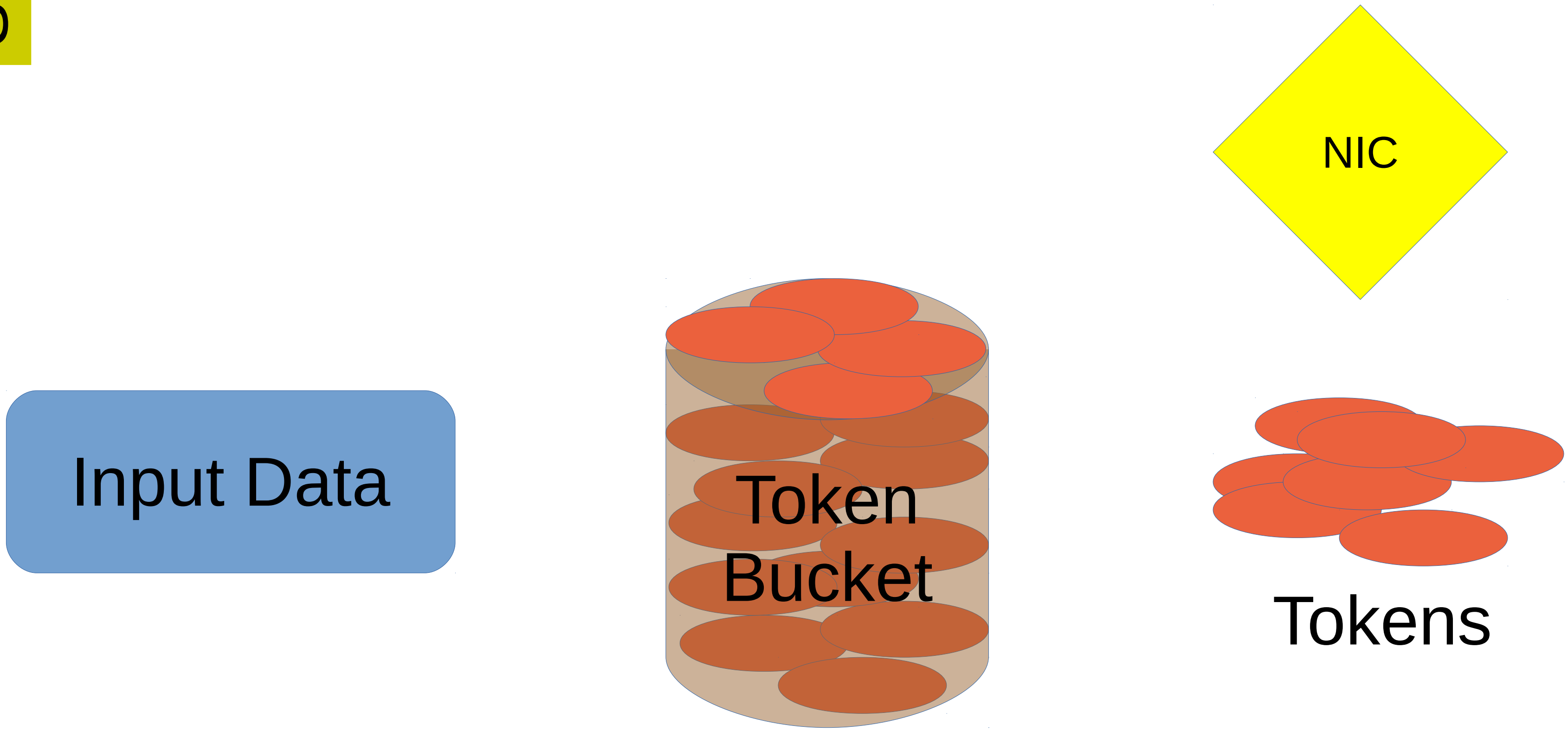
- Eventually the bucket is full of empty tokens.
- Token flow slows down



The TBF Qdisc

Scenario I

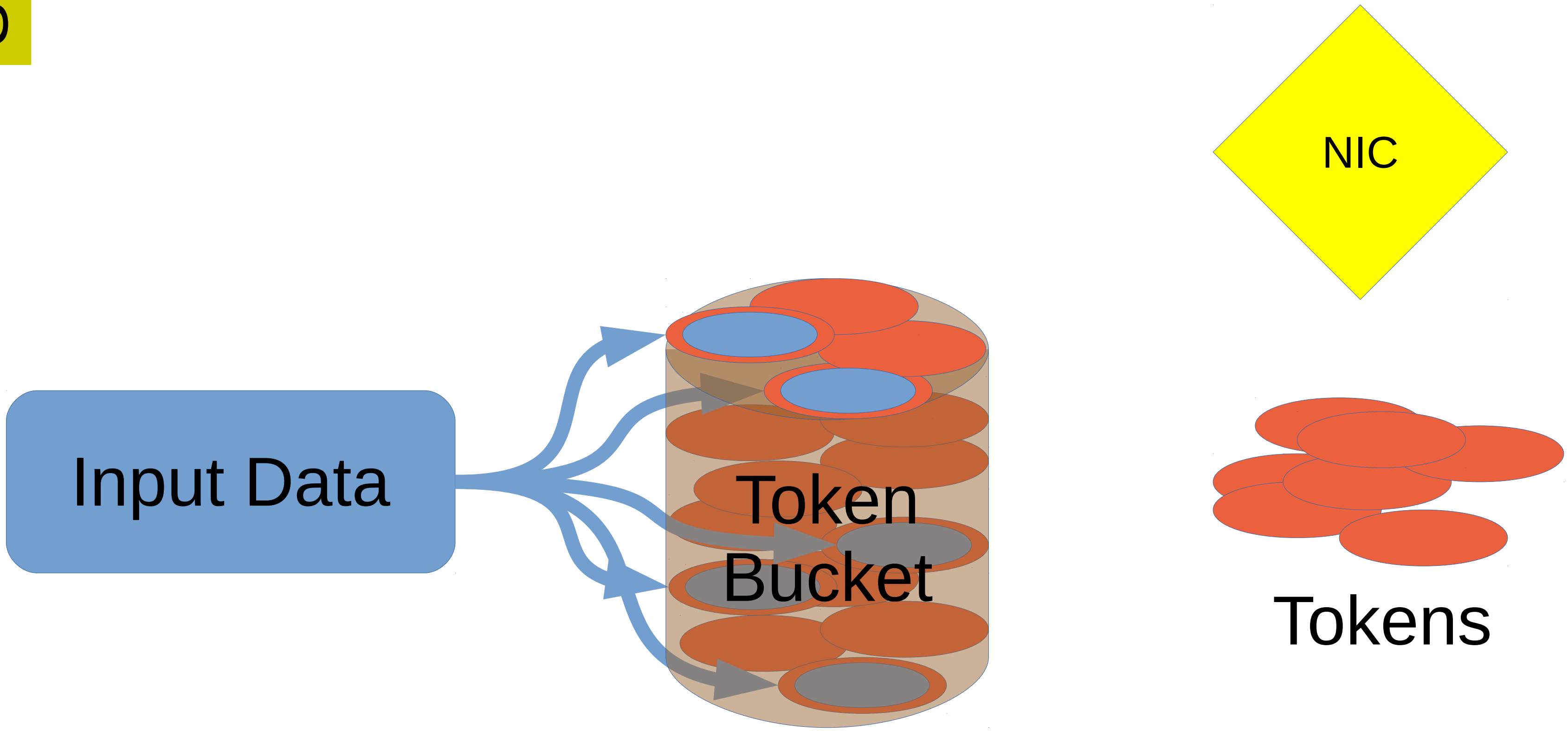
Empty tockens in bucket can be used to burst data out faster than token-rate



14 The TBF Qdisc

Scenario I

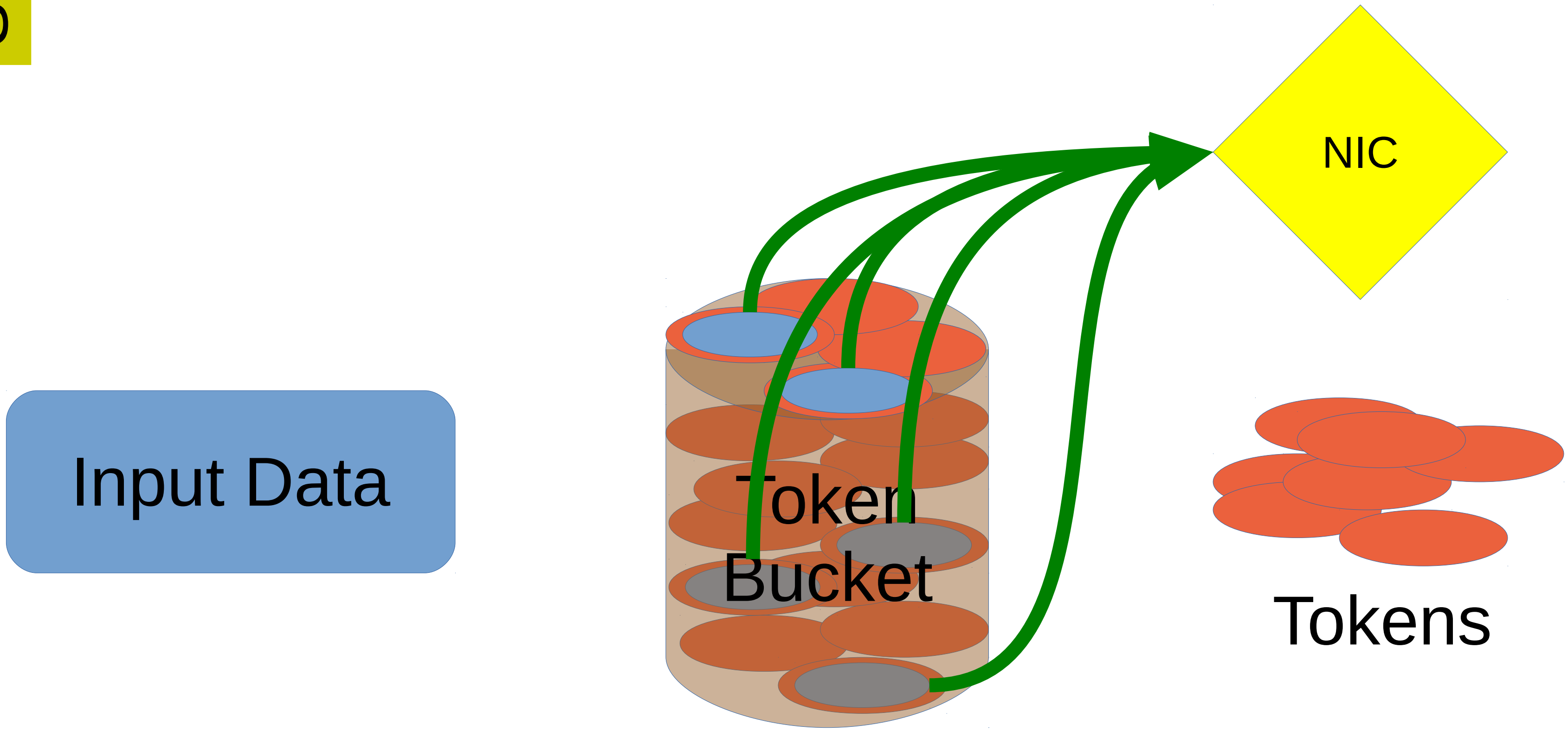
Empty tockens in bucket can be used to burst data out faster than token-rate



14 The TBF Qdisc

Scenario I

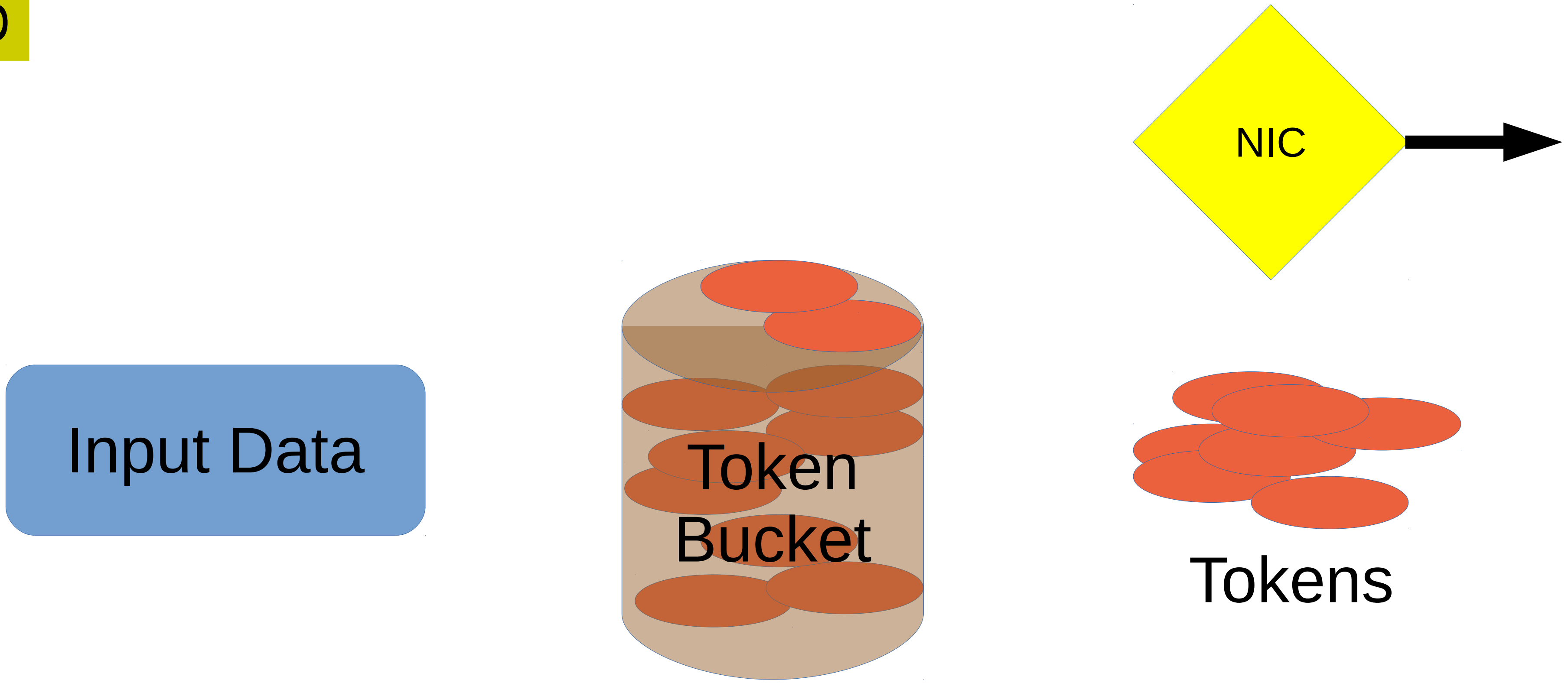
Empty tockens in bucket can be used to burst data out faster than token-rate



14 The TBF Qdisc

Scenario I

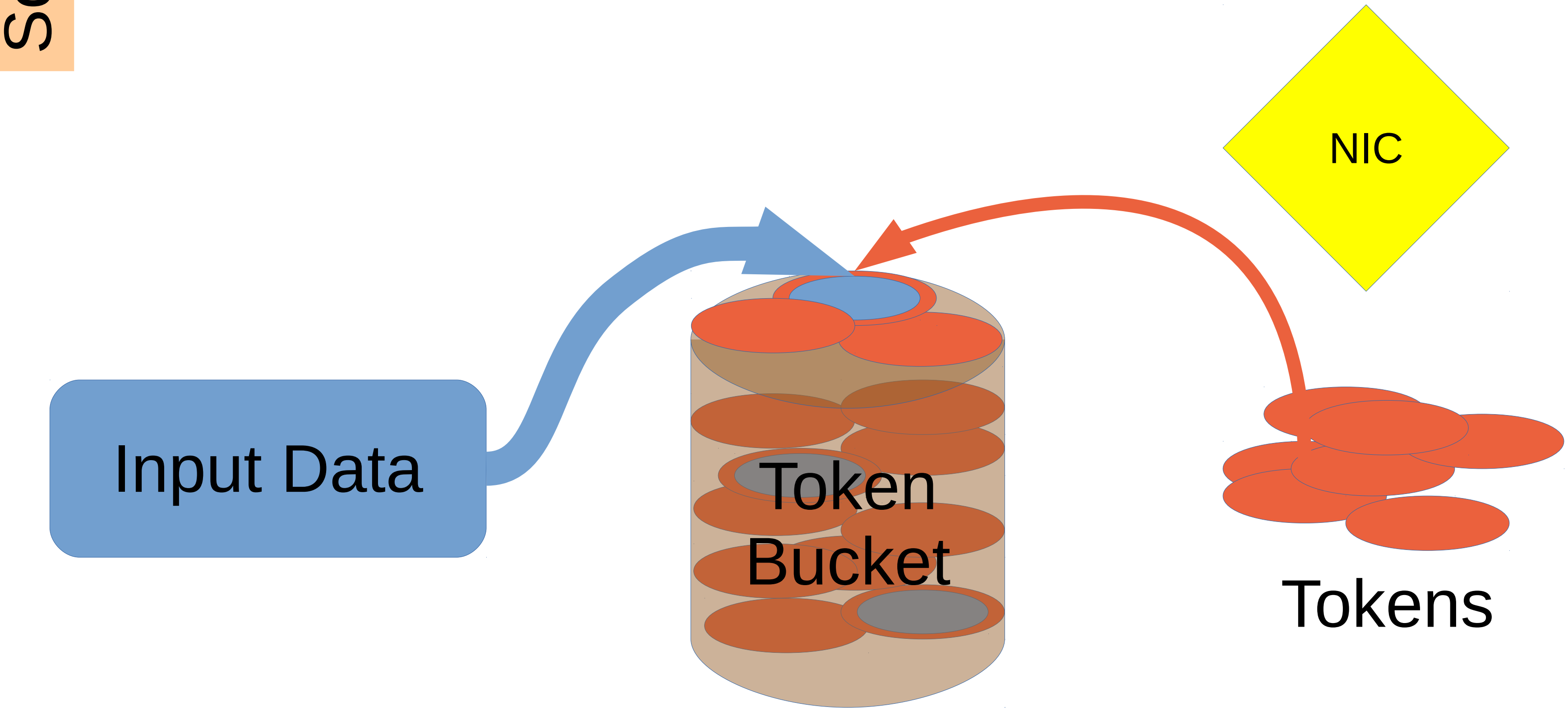
Empty tockens in bucket can be used to burst data out faster than token-rate



14 The TBF Qdisc

Scenario II

- **Token Flow** slower than **Input Data Flow**

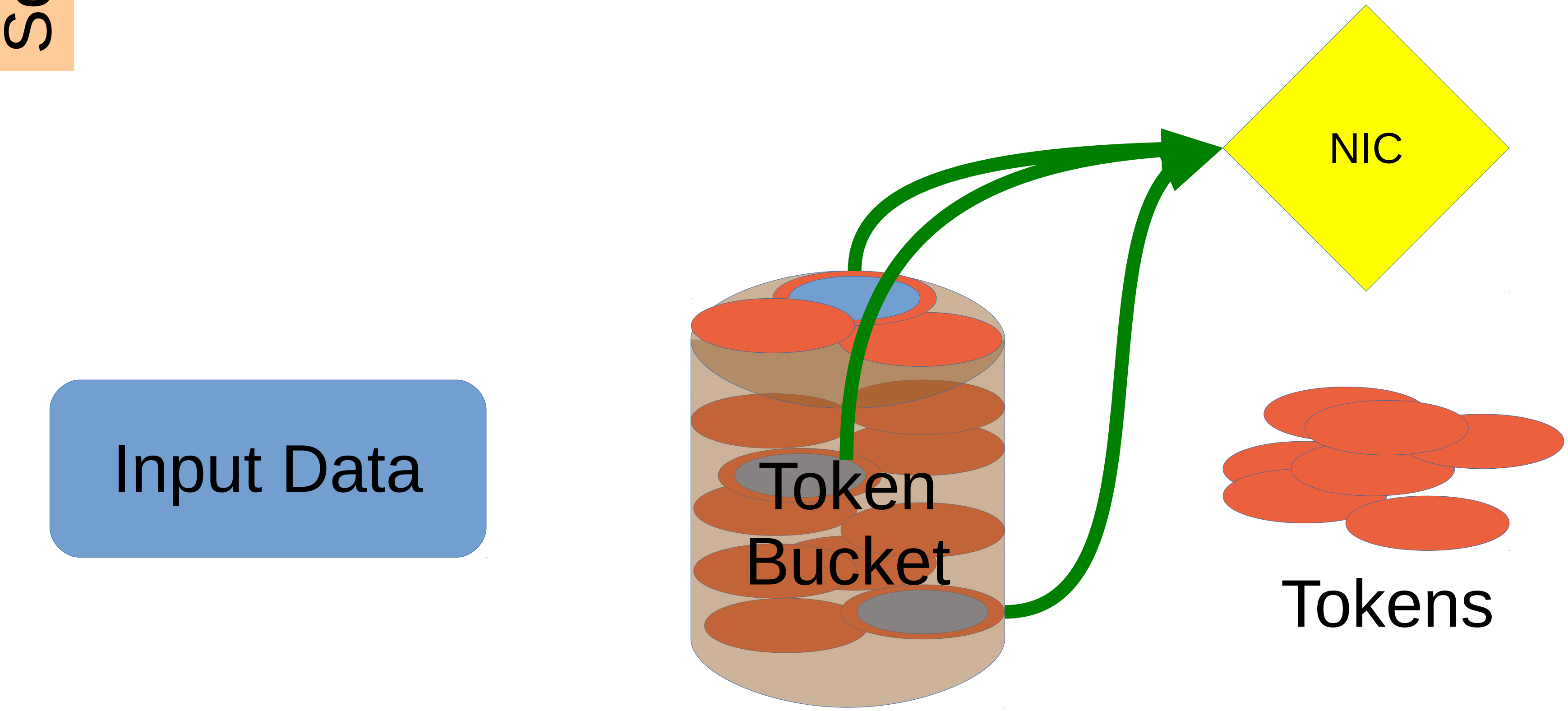


14

The TBF Qdisc

Scenario II

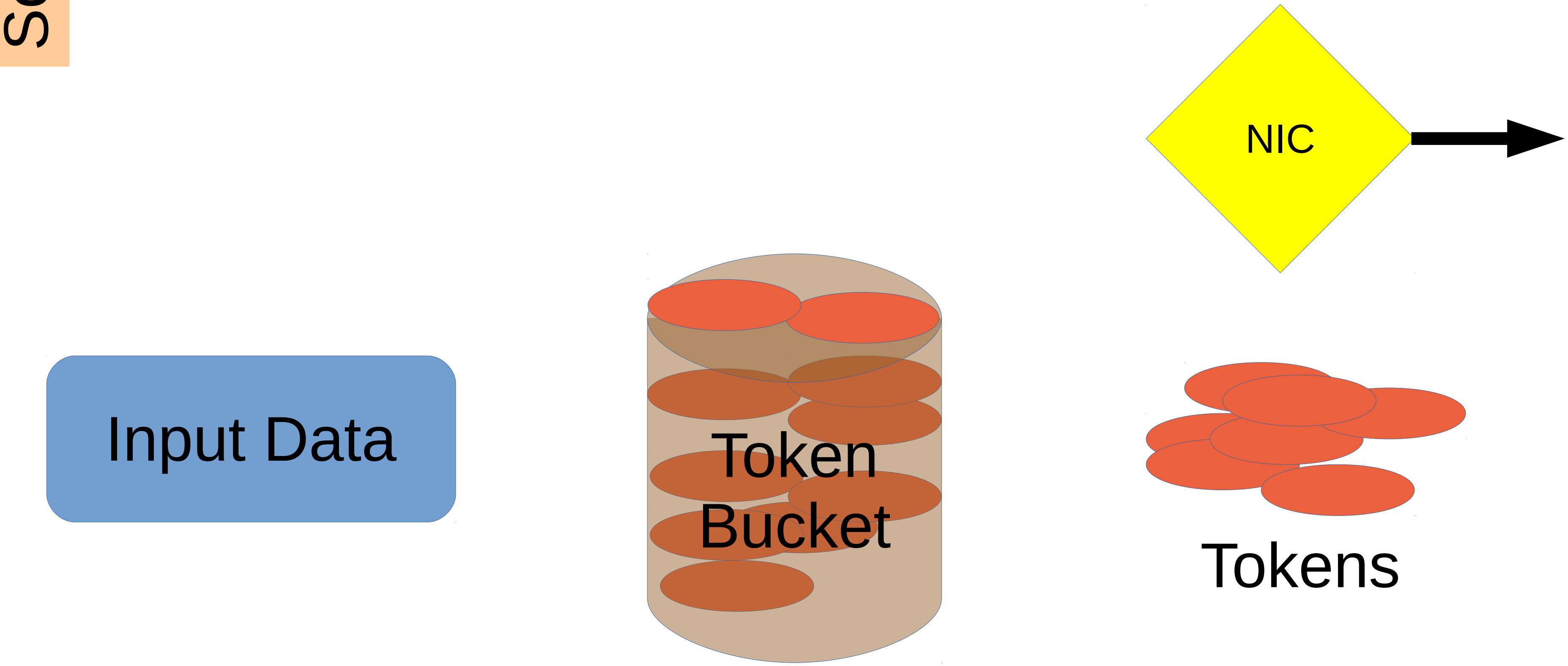
- **Token Flow** slower than **Input Data Flow**



14 The TBF Qdisc

Scenario II

- **Token Flow** slower than **Input Data Flow**

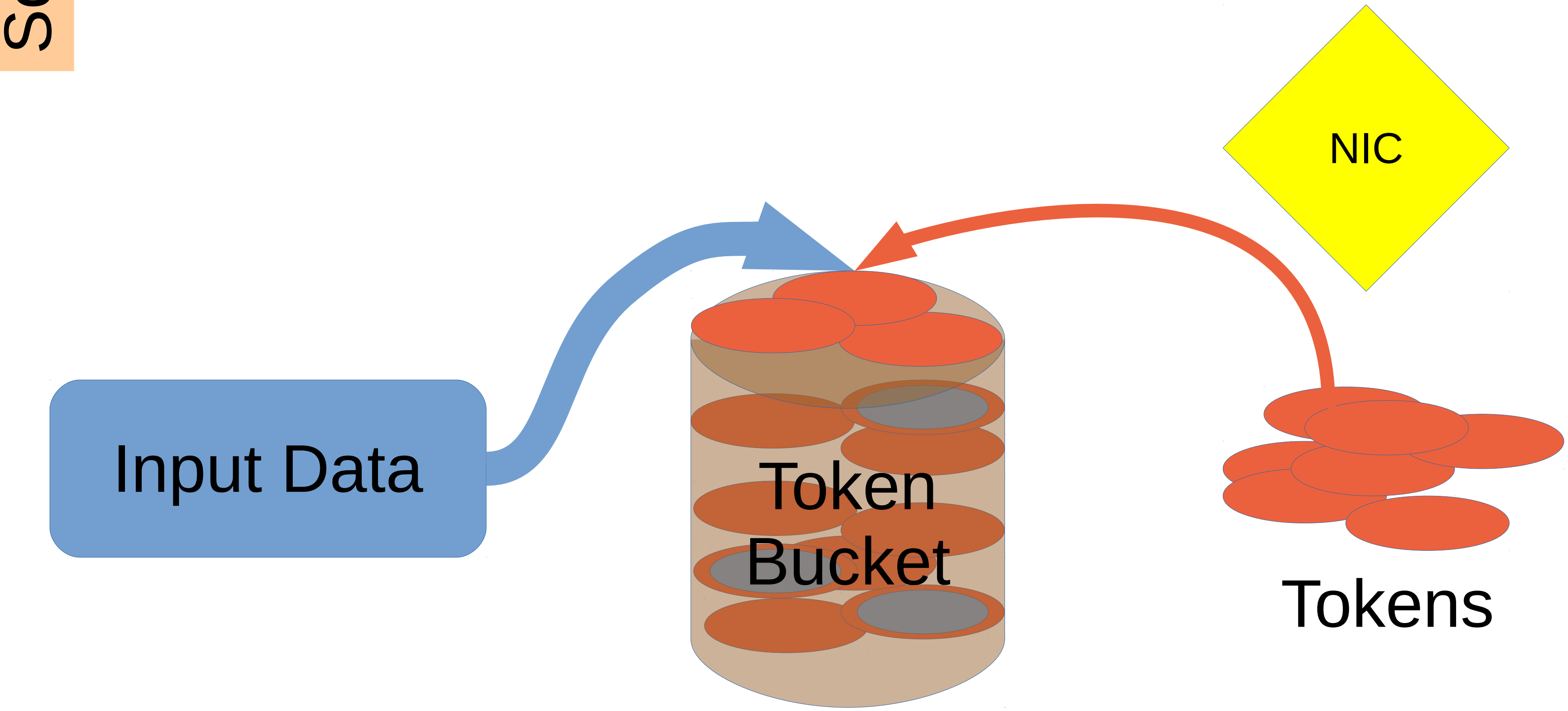


14

The TBF Qdisc

Scenario II

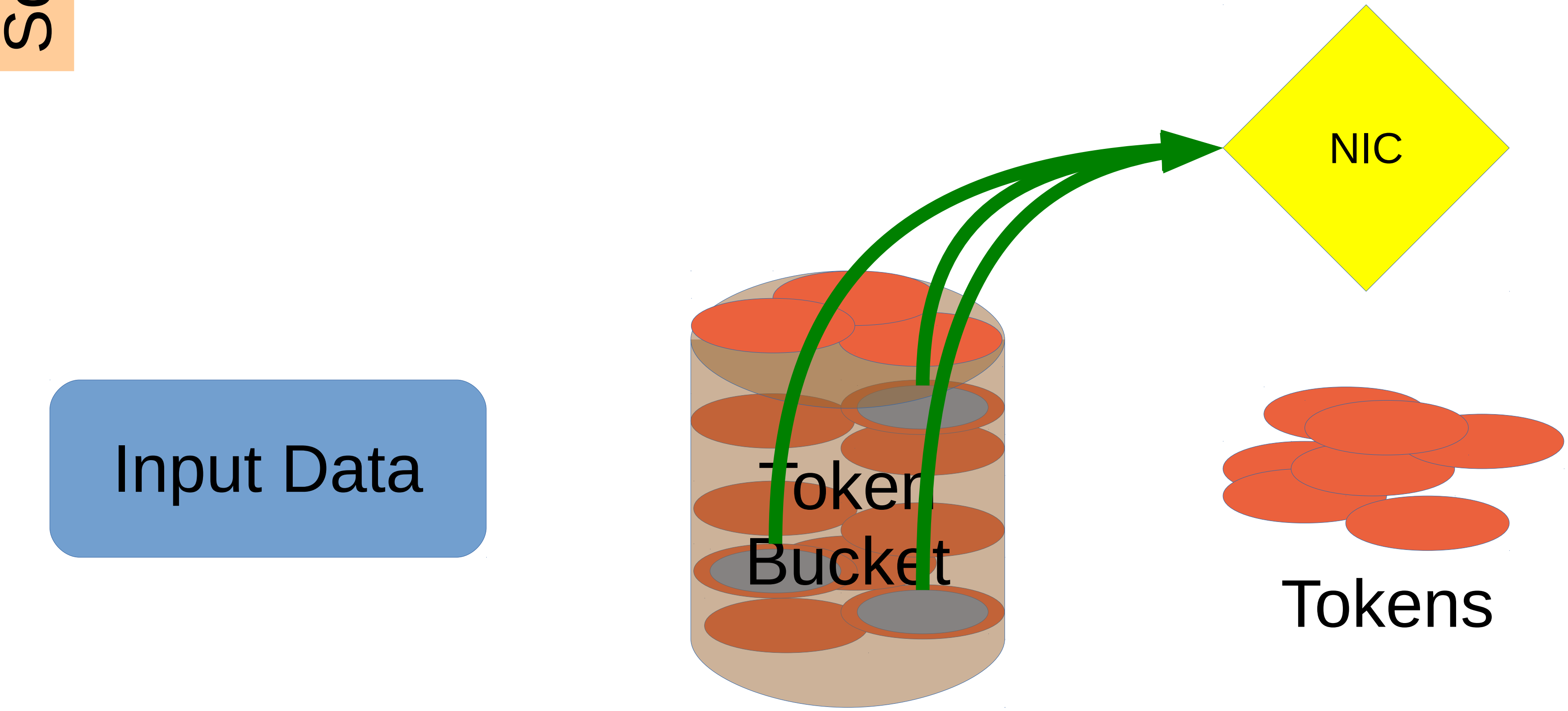
- **Token Flow** slower than **Input Data Flow**



14 The TBF Qdisc

Scenario II

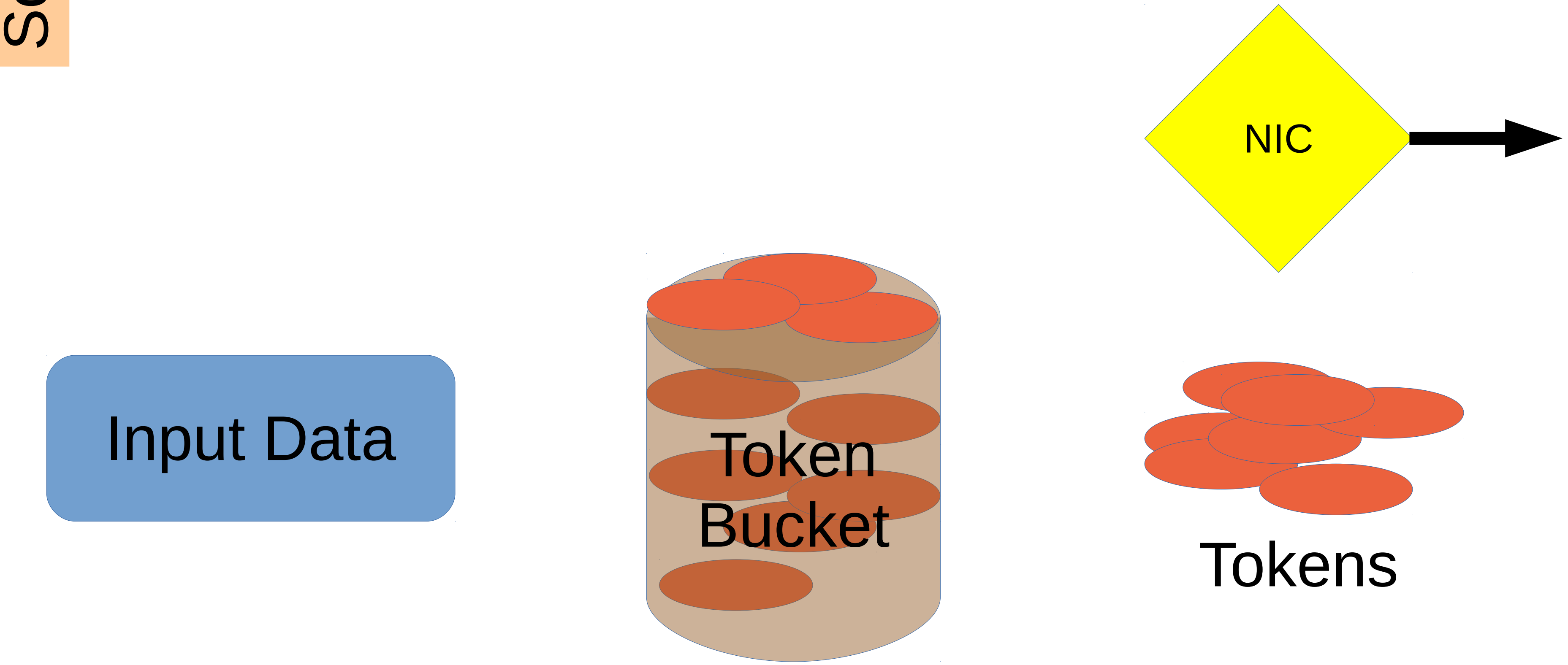
- **Token Flow** slower than **Input Data Flow**



14 The TBF Qdisc

Scenario II

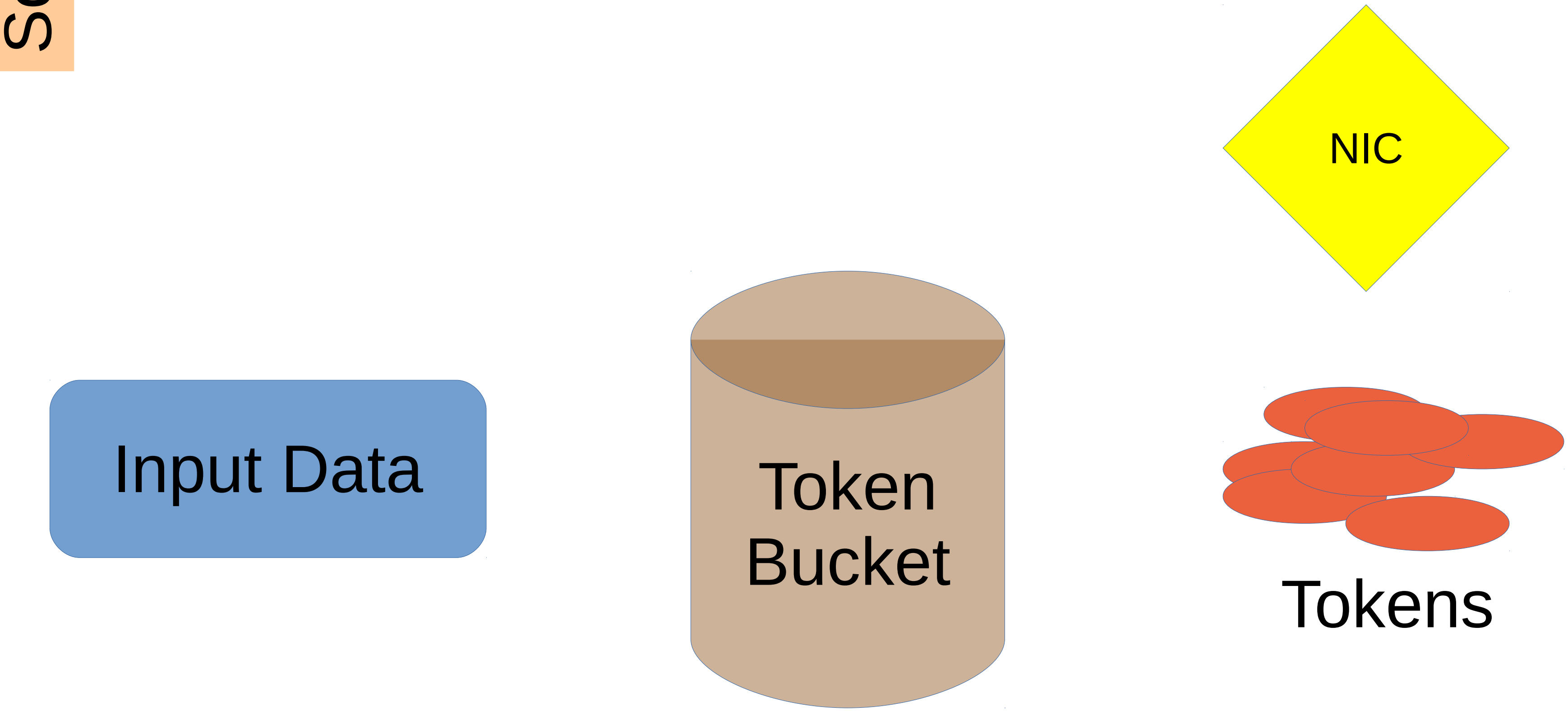
- **Token Flow** slower than **Input Data Flow**



14 The TBF Qdisc

Scenario II

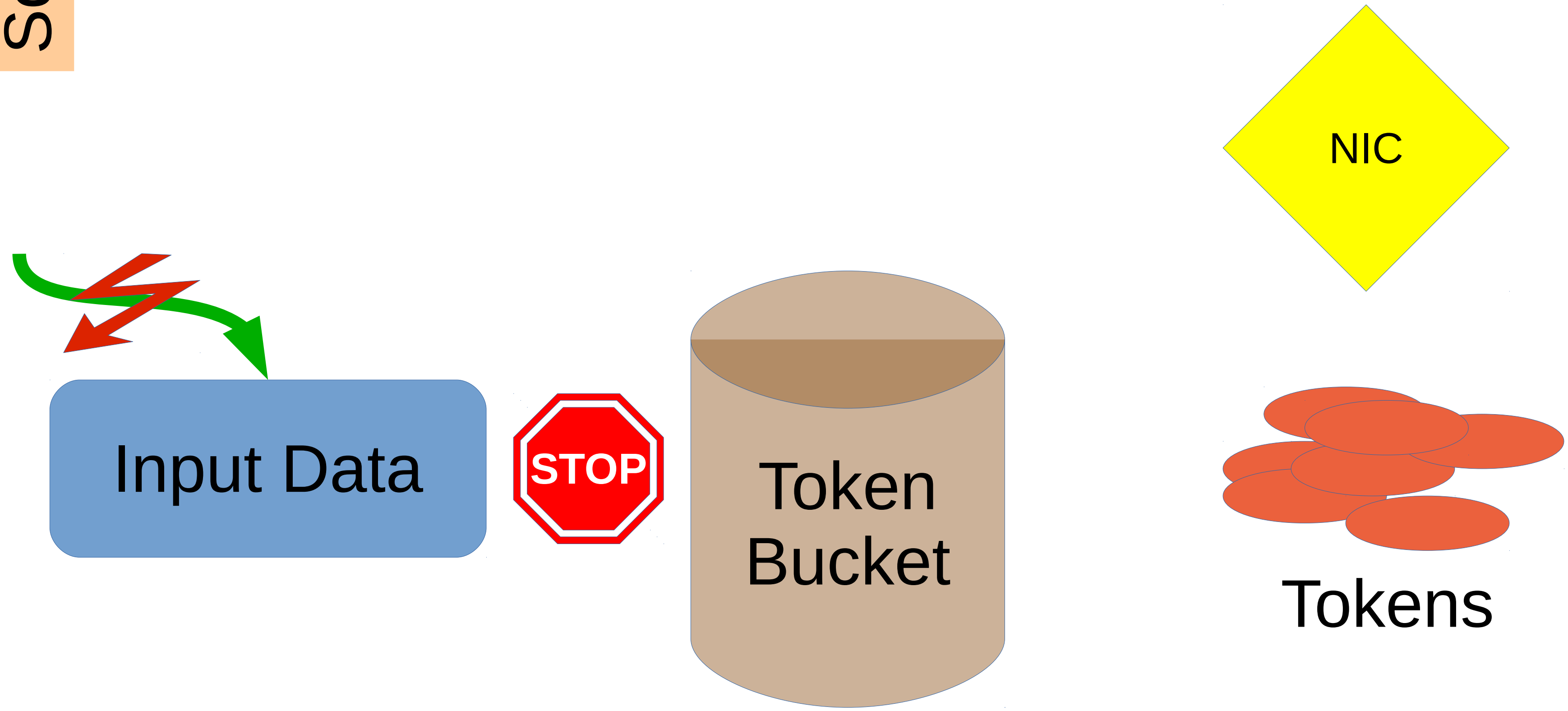
- **Token Flow** slower than **Input Data Flow**
- Eventually the bucket will be empty



14 The TBF Qdisc

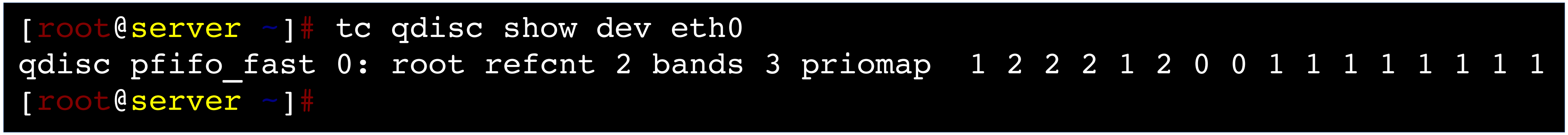
Scenario II

- **Token Flow** slower than **Input Data Flow**
- Eventually the bucket will be empty
- When SKB is full, packets start being dropped



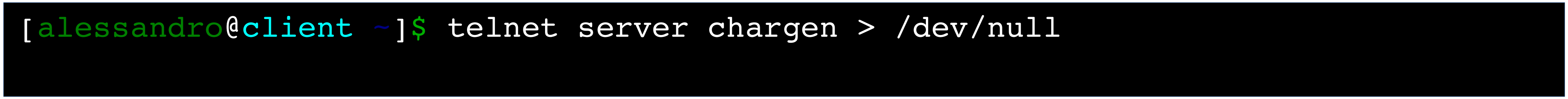
15 Classless TBF Qdisc At Work

TBF, **Token Bucket Filter**: only shapes traffic, does no scheduling



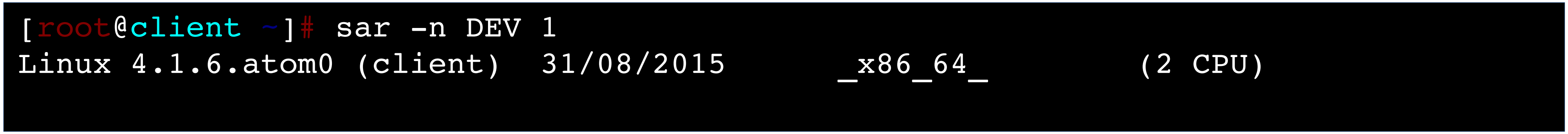
```
[root@server ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@server ~]#
```

Client creates as much TCP traffic as possible:



```
[alessandro@client ~]$ telnet server chargen > /dev/null
```

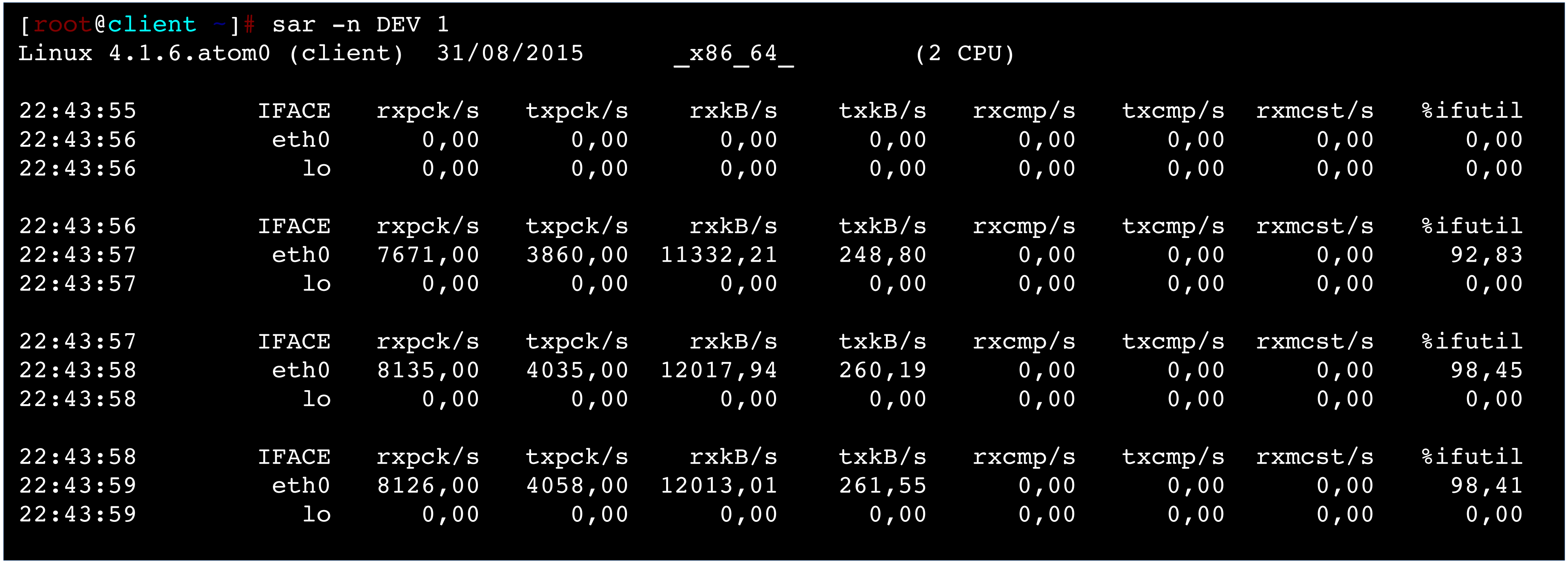
Traffic is monitored on the client:



```
[root@client ~]# sar -n DEV 1
Linux 4.1.6.atom0 (client)  31/08/2015      _x86_64_        (2 CPU)
```

15 Classless TBF Qdisc At Work

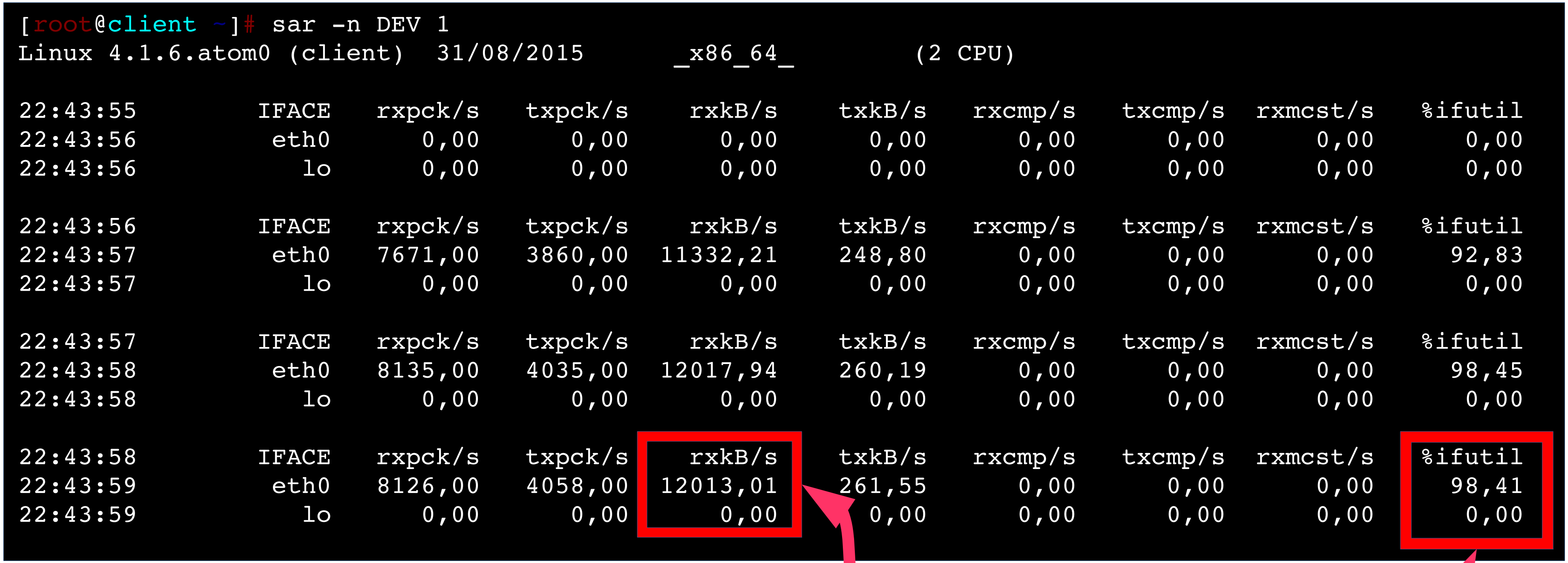
Let's follow the traffic:



```
[root@client ~]# sar -n DEV 1
Linux 4.1.6.atom0 (client)  31/08/2015      _x86_64_        (2 CPU)

22:43:55        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:56         eth0      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00
22:43:56           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

22:43:56        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:57         eth0   7671,00   3860,00  11332,21    248,80      0,00      0,00      0,00     92,83
22:43:57           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

22:43:57        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:58         eth0   8135,00   4035,00  12017,94    260,19      0,00      0,00      0,00     98,45
22:43:58           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

22:43:58        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:59         eth0   8126,00   4058,00  12013,01    261,55      0,00      0,00      0,00     98,41
22:43:59           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00
```

15 Classless TBF Qdisc At Work

Let's follow the traffic:



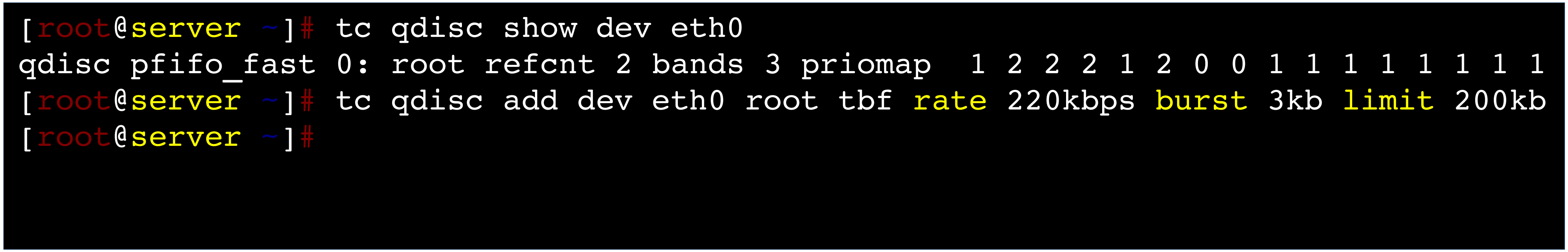
```
[root@client ~]# sar -n DEV 1
Linux 4.1.6.atom0 (client)  31/08/2015      _x86_64_        (2 CPU)

22:43:55        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:56         eth0      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00
22:43:56           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

22:43:56        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:57         eth0   7671,00   3860,00  11332,21    248,80      0,00      0,00      0,00     92,83
22:43:57           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

22:43:57        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:58         eth0   8135,00   4035,00  12017,94    260,19      0,00      0,00      0,00     98,45
22:43:58           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

22:43:58        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
22:43:59         eth0   8126,00   4058,00  12013,01    261,55      0,00      0,00      0,00     98,41
22:43:59           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00
```

Regime values

100mbps = 12,500kBps

15 Classless TBF Qdisc At Work

We can only shape traffic from the origin, that is on the server

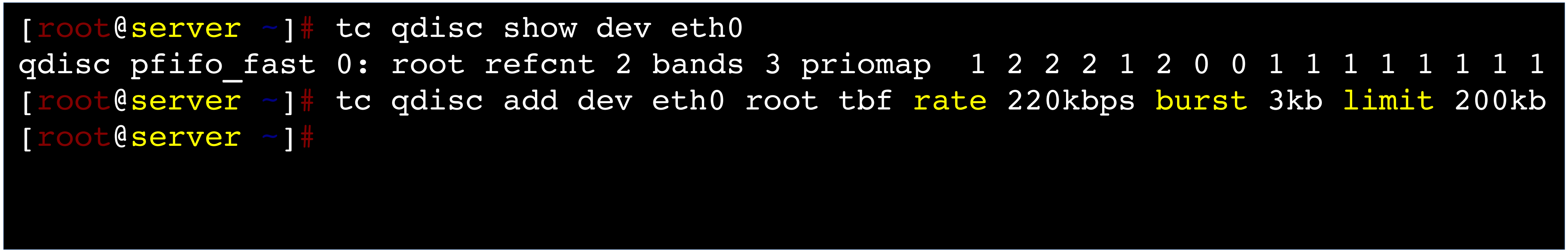


```
[root@server ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@server ~]# tc qdisc add dev eth0 root tbf rate 220kbps burst 3kb limit 200kb
[root@server ~]#
```

- **limit** (bucket buffer size [bytes]) or **latency** (time data can sit in bucket) must be set (MBS)
- **burst** (aka **buffer** or **maxburst** [bytes]) MBS
- **rate** MBS

15 Classless TBF Qdisc At Work

We can only shape traffic from the origin, that is on the server



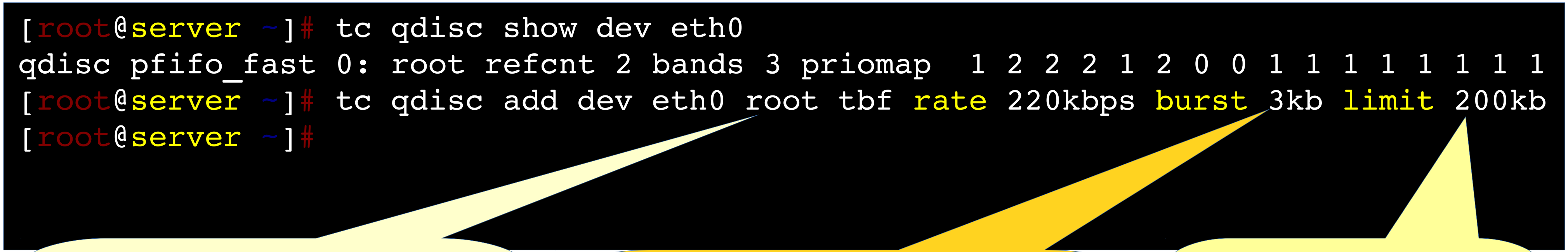
```
[root@server ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@server ~]# tc qdisc add dev eth0 root tbf rate 220kbps burst 3kb limit 200kb
[root@server ~]#
```

Rant: *Why did they choose these names?!*

- **`limit`** (bucket **buffer** size [bytes]) or **`latency`** (time data can sit in bucket) must be set (MBS)
- **`burst`** (aka **`buffer`** or **`maxburst`** [bytes]) MBS
- **`rate`** MBS

15 Classless TBF Qdisc At Work

We can only shape traffic from the origin, that is on the server



```
[root@server ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@server ~]# tc qdisc add dev eth0 root tbf rate 220kbps burst 3kb limit 200kb
[root@server ~]#
```

ID of parent QDisc

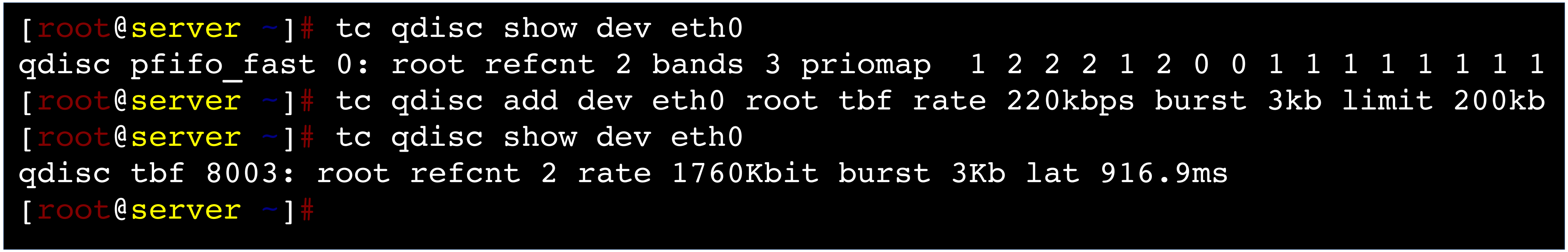
rate/HZ minimum
Never < MTU!

128 1,500
byte packets

- **`limit`** (bucket buffer size [bytes]) or **`latency`** (time data can sit in bucket) must be set (MBS)
- **`burst`** (aka **`buffer`** or **`maxburst`** [bytes]) MBS
- **`rate`** MBS

15 Classless TBF Qdisc At Work

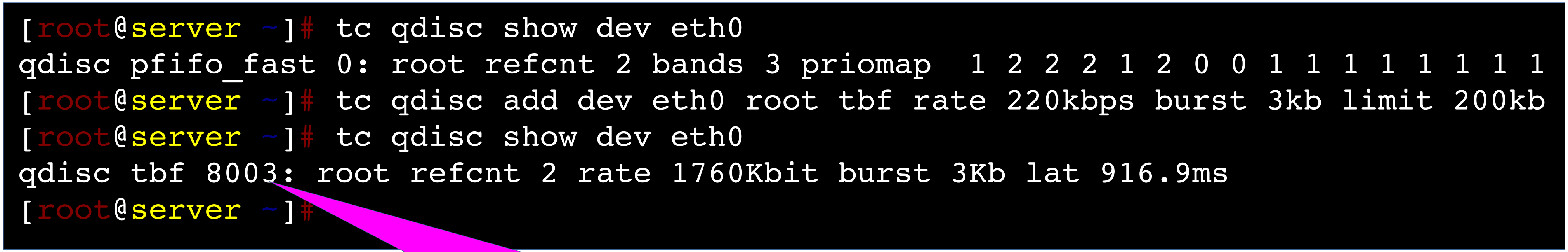
We can only shape traffic from the origin, that is on the server



```
[root@server ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@server ~]# tc qdisc add dev eth0 root tbf rate 220kbps burst 3kb limit 200kb
[root@server ~]# tc qdisc show dev eth0
qdisc tbf 8003: root refcnt 2 rate 1760Kbit burst 3Kb lat 916.9ms
[root@server ~]#
```

15 Classless TBF Qdisc At Work

We can only shape traffic from the origin, that is on the server

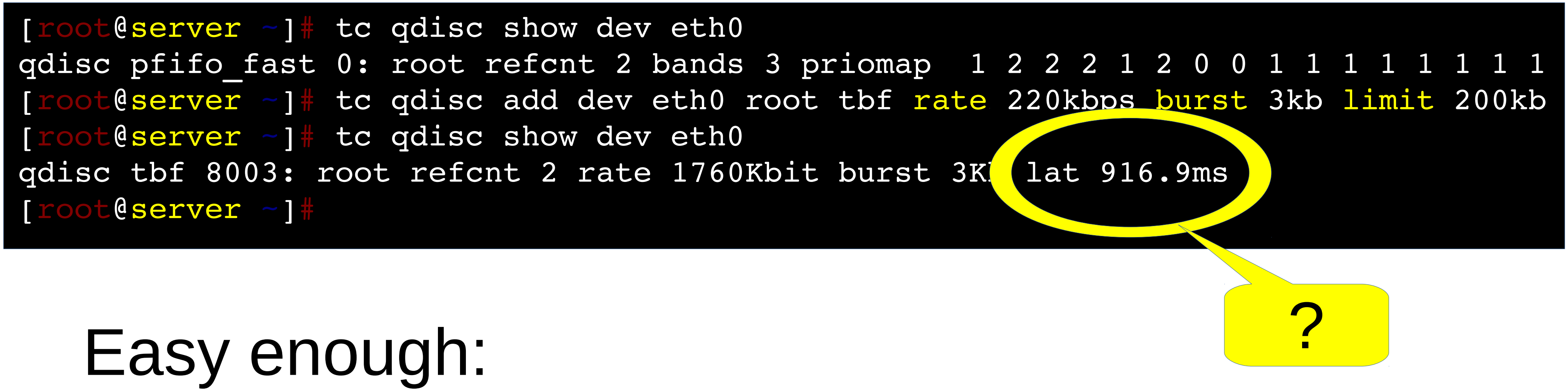


```
[root@server ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@server ~]# tc qdisc add dev eth0 root tbf rate 220kbps burst 3kb limit 200kb
[root@server ~]# tc qdisc show dev eth0
qdisc tbf 8003: root refcnt 2 rate 1760Kbit burst 3Kb lat 916.9ms
[root@server ~]#
```

handle to reference this qdisc

15 Classless TBF Qdisc At Work

We can only shape traffic from the origin, that is on the server



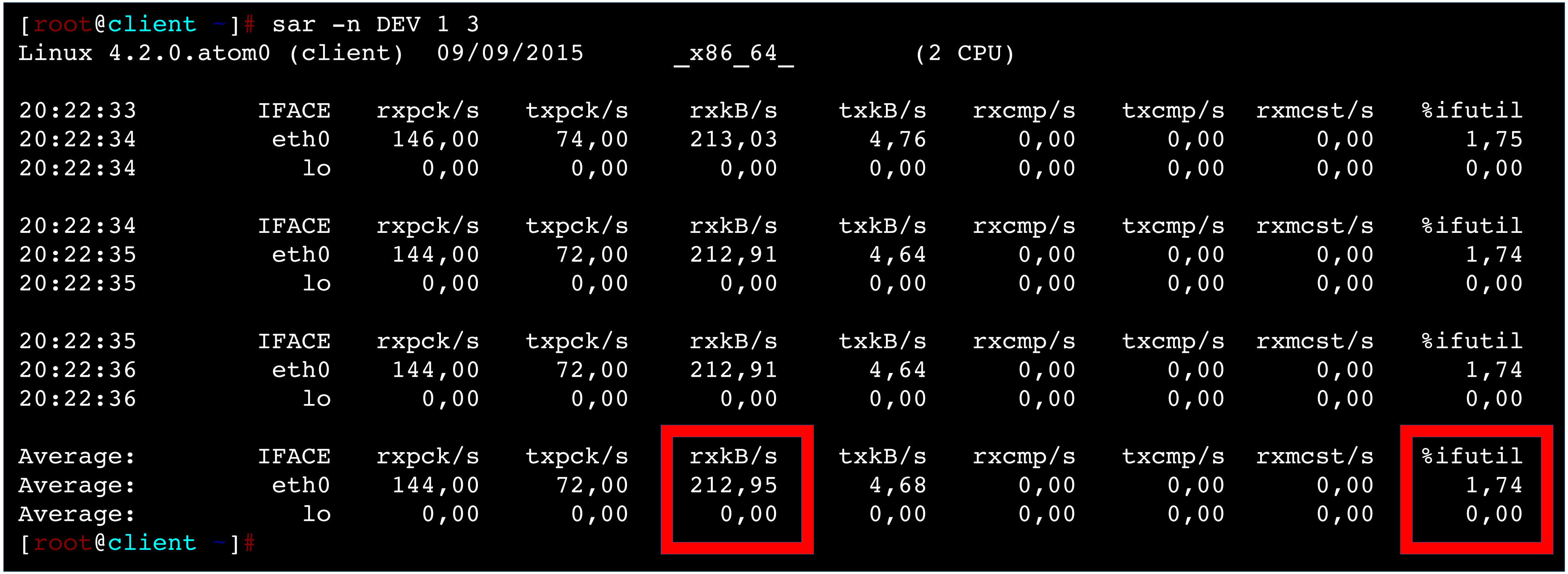
```
[root@server ~]# tc qdisc show dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
[root@server ~]# tc qdisc add dev eth0 root tbf rate 220kbps burst 3kb limit 200kb
[root@server ~]# tc qdisc show dev eth0
qdisc tbf 8003: root refcnt 2 rate 1760Kbit burst 3K  lat 916.9ms
[root@server ~]#
```

Easy enough:

- qdisc buffer size: 200KB
- at 220KB/sec rate, buffer empties in 200/220=0.909 seconds
- The difference is due to burstiness

15 Classless TBF Qdisc At Work

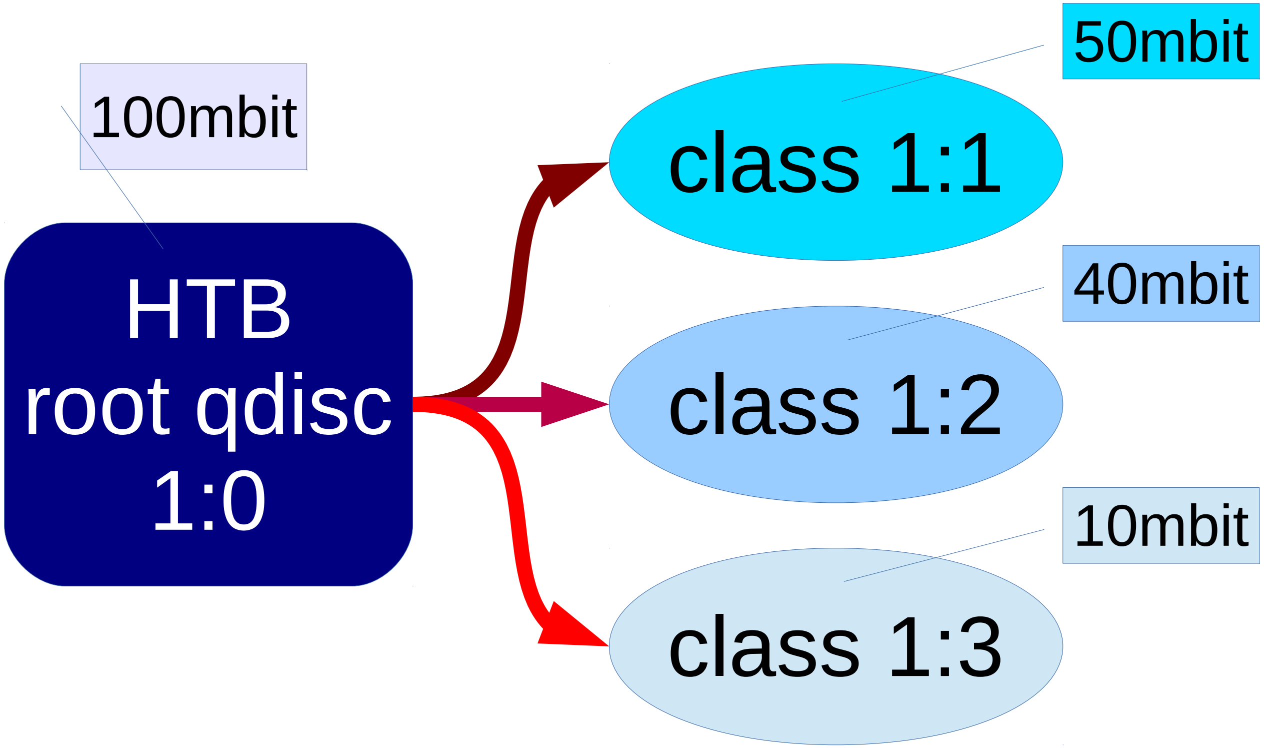
On the client:



```
[root@client ~]# sar -n DEV 1 3
Linux 4.2.0.atom0 (client)  09/09/2015      _x86_64_        (2 CPU)

20:22:33        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
20:22:34         eth0    146,00     74,00    213,03      4,76      0,00      0,00      0,00      1,75
20:22:34           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

20:22:34        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
20:22:35         eth0    144,00     72,00    212,91      4,64      0,00      0,00      0,00      1,74
20:22:35           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

20:22:35        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
20:22:36         eth0    144,00     72,00    212,91      4,64      0,00      0,00      0,00      1,74
20:22:36           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

Average:        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
Average:         eth0    144,00     72,00    212,95      4,68      0,00      0,00      0,00      1,74
Average:           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00
[root@client ~]#
```

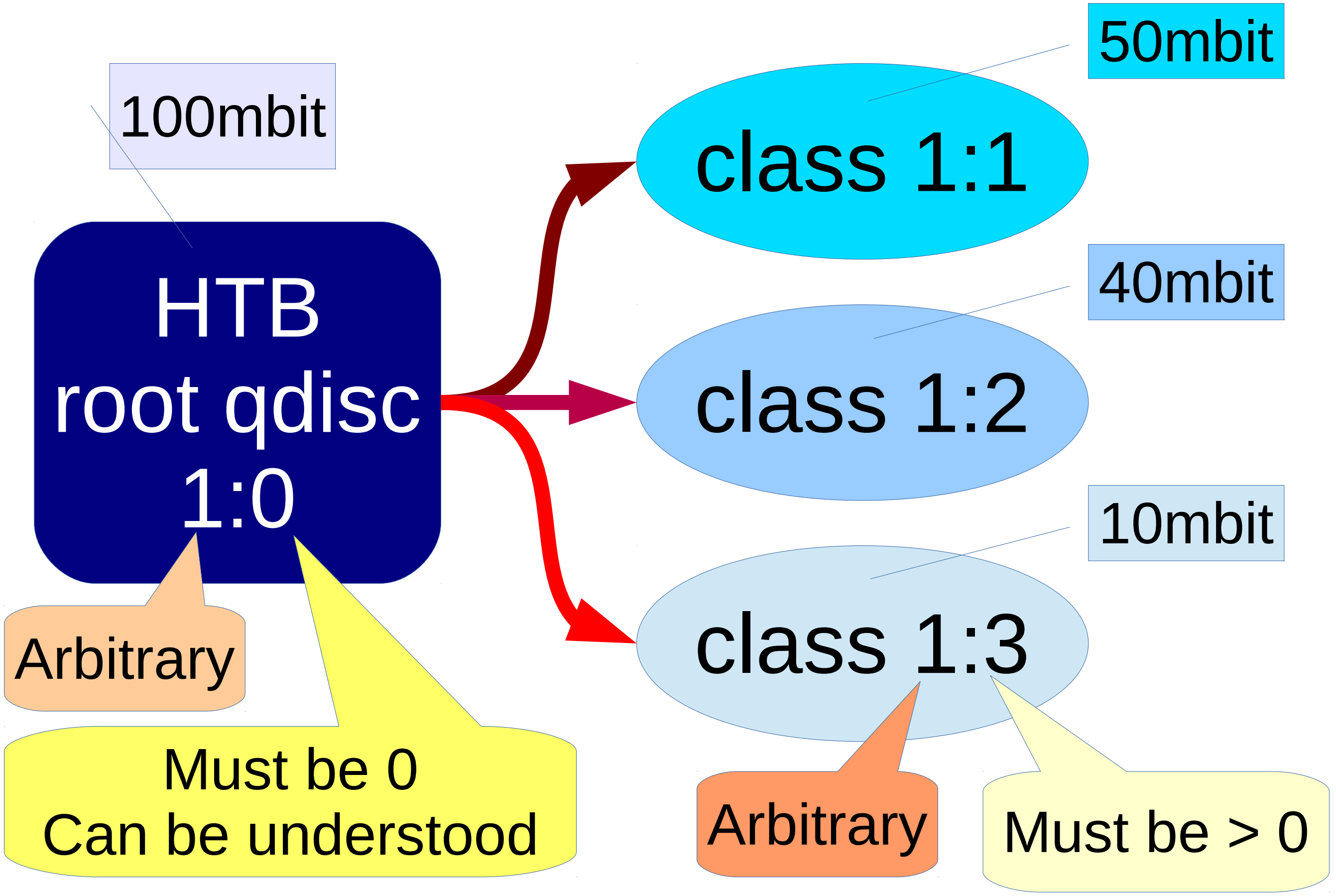
16 Classful HTB Qdisc

Hierarchical Token Bucket: a qdisc that splits data flow into several TBF-like throttled classes



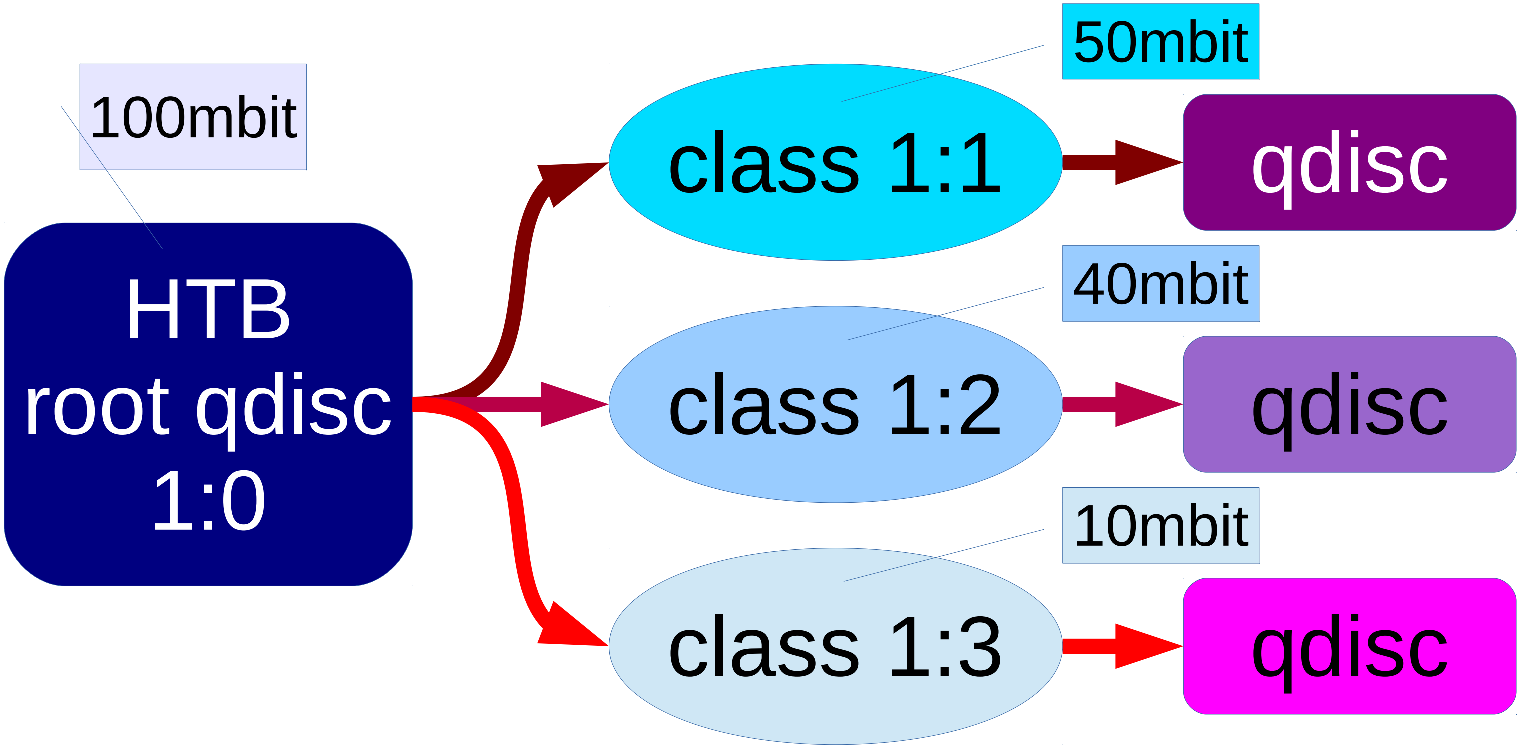
Classful HTB Qdisc

Hierarchical Token Bucket: a qdisc that splits data flow into several TBF-like throttled classes



Classful HTB Qdisc

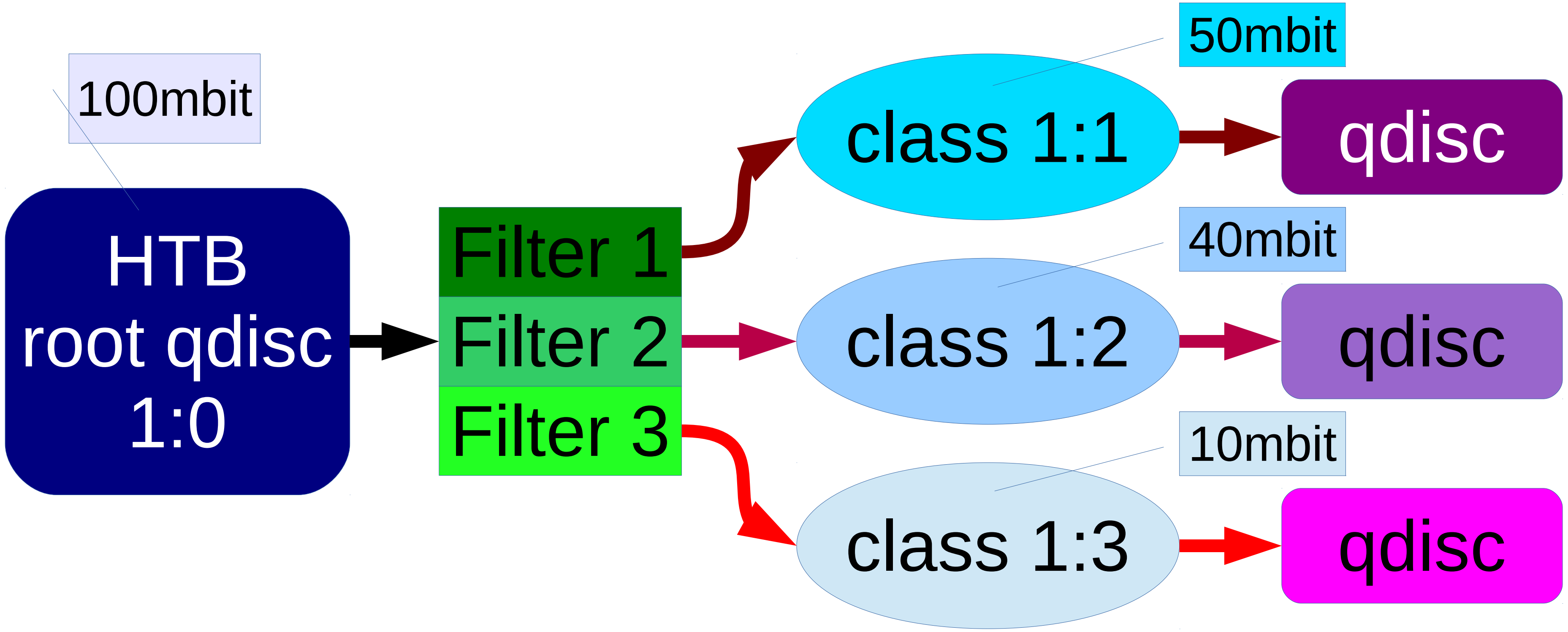
Hierarchical Token Bucket: a qdisc that splits data flow into several TBF-like throttled classes



Qdiscs are attached to classes

16 Classful HTB Qdisc

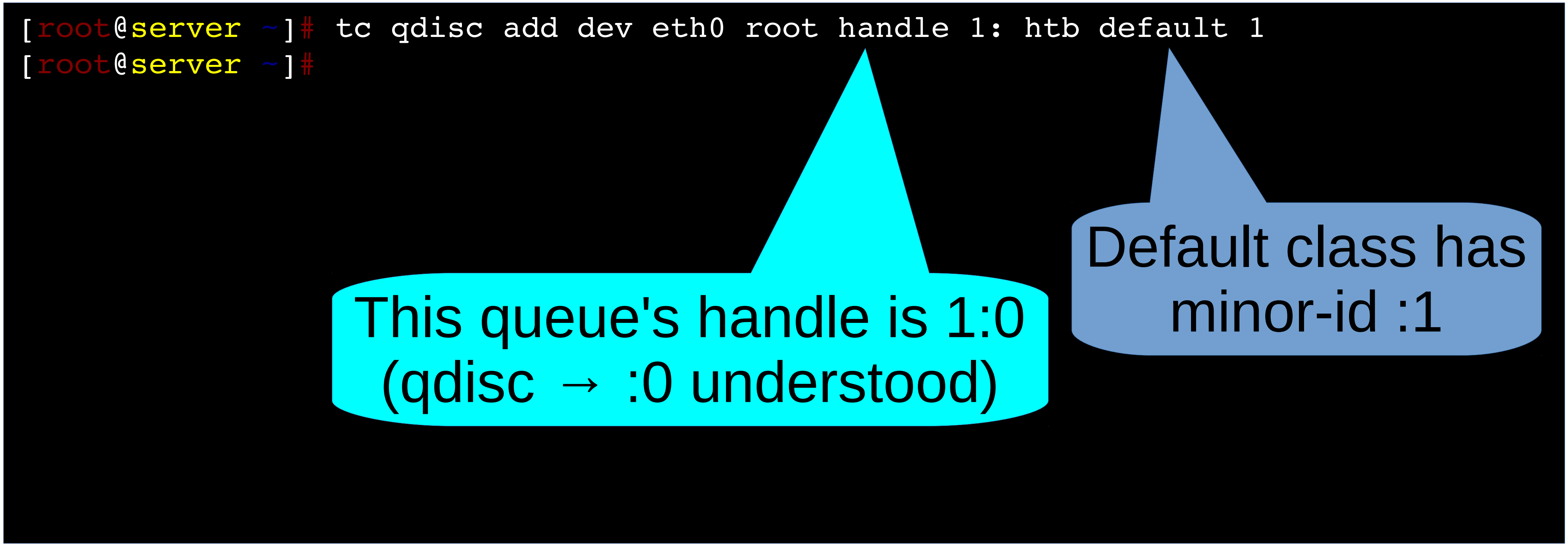
Hierarchical Token Bucket: a qdisc that splits data flow into several TBF-like throttled classes



We need filters to associate traffic to each class

17 Classful HTB Qdisc At Work

HTB: setup of root qdisc



```
[root@server ~]# tc qdisc add dev eth0 root handle 1: htb default 1
[root@server ~]#
```

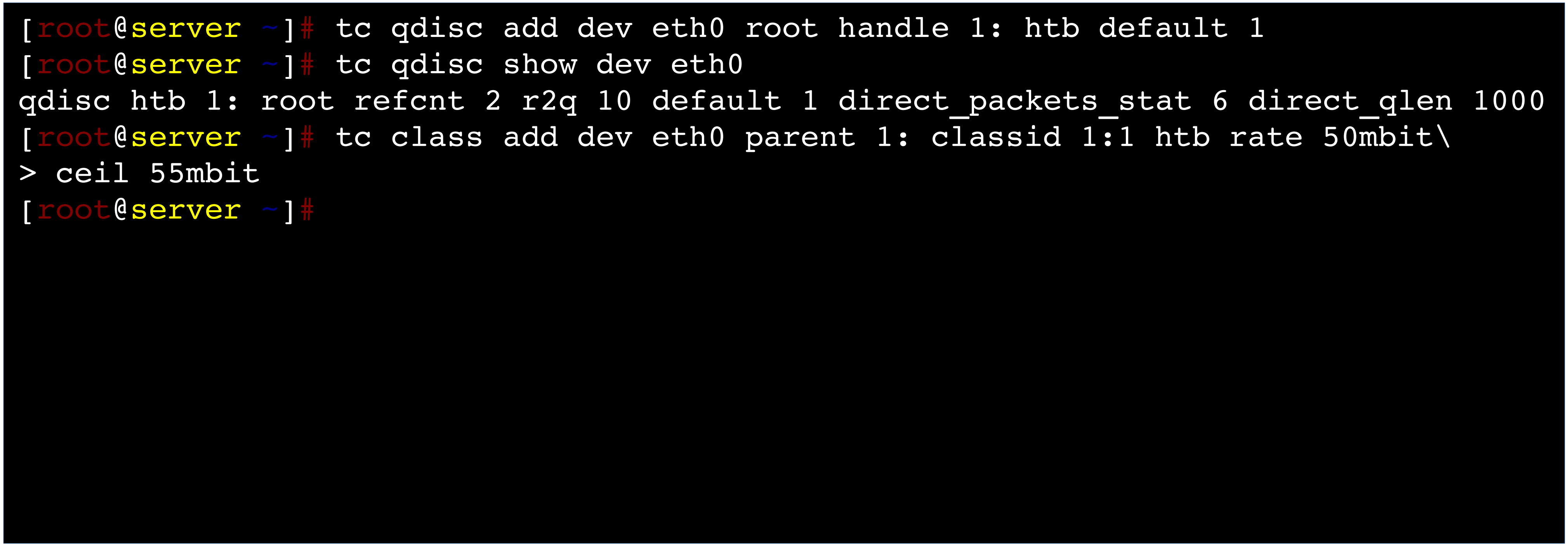
This queue's handle is 1:0 (qdisc → :0 understood)

Default class has minor-id :1

Handle identifier: `X:Y`, `Y` = 0 qdisc, `Y` > 0 class

17 Classful HTB Qdisc At Work

HTB: classes are attached to root qdisc

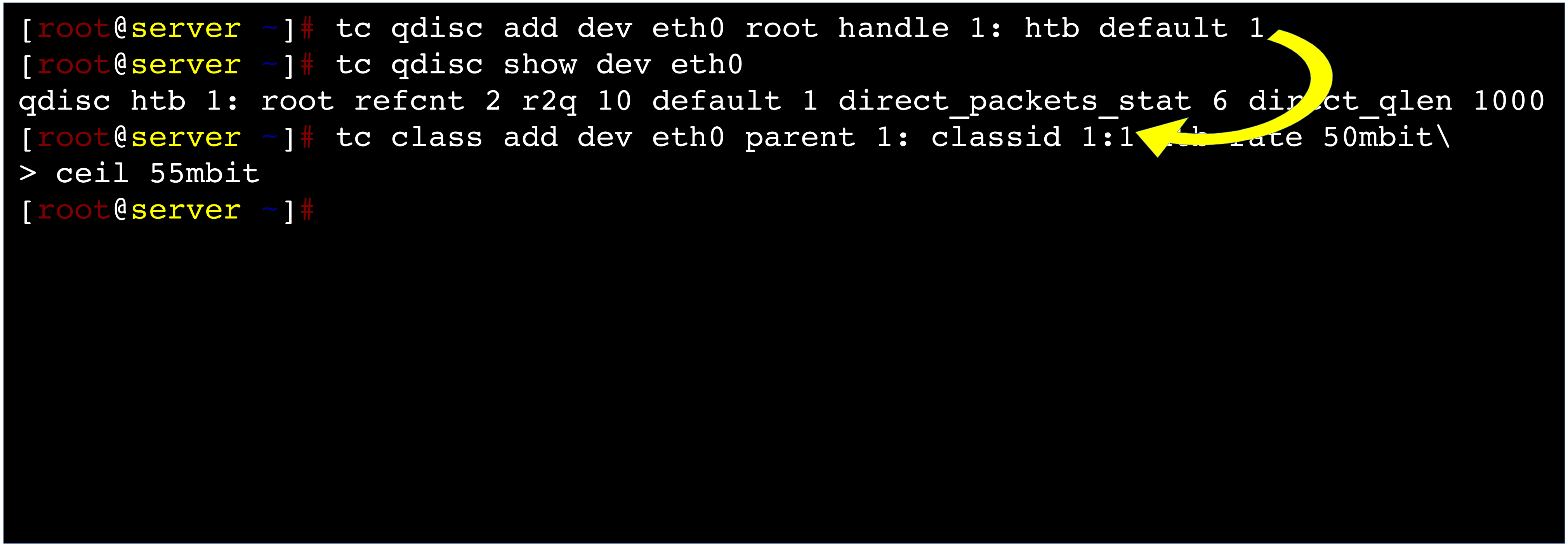


```
[root@server ~]# tc qdisc add dev eth0 root handle 1: htb default 1
[root@server ~]# tc qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 6 direct_qlen 1000
[root@server ~]# tc class add dev eth0 parent 1: classid 1:1 htb rate 50mbit\
> ceil 55mbit
[root@server ~]#
```

Handle identifier: `X:Y`, `Y` = 0 qdisc, `Y` > 0 class

17 Classful HTB Qdisc At Work

HTB: classes are attached to root qdisc

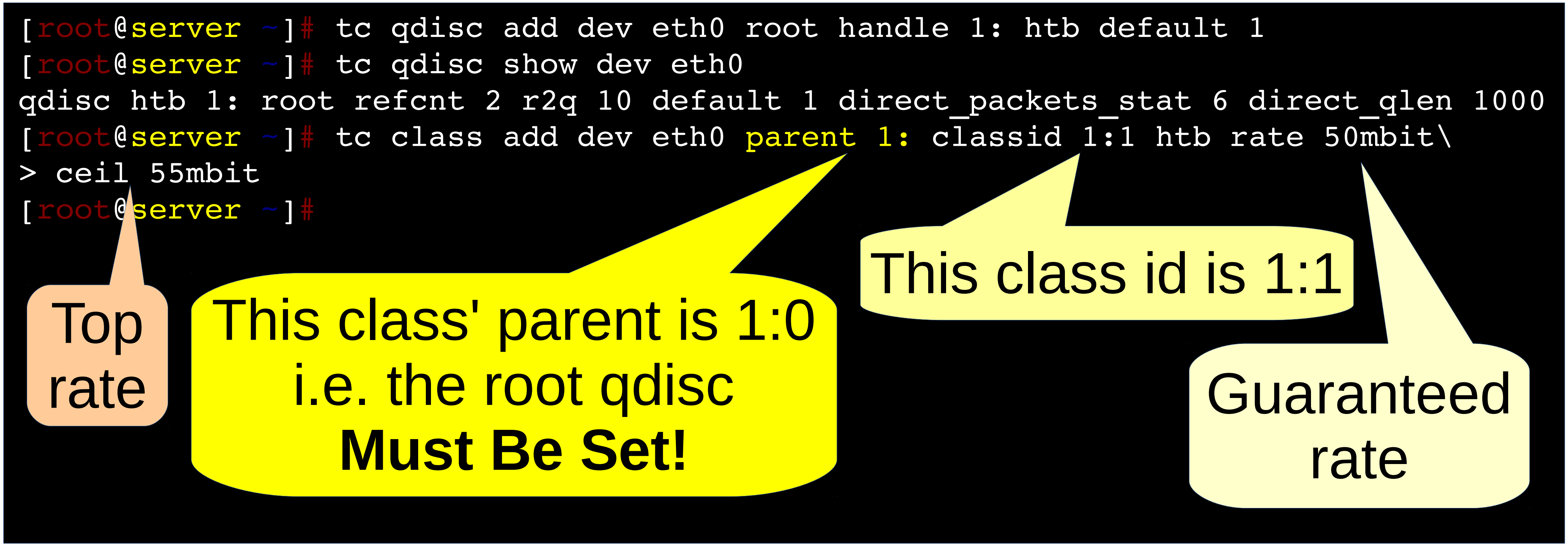


```
[root@server ~]# tc qdisc add dev eth0 root handle 1: htb default 1
[root@server ~]# tc qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 6 direct_qlen 1000
[root@server ~]# tc class add dev eth0 parent 1: classid 1:1 htb rate 50mbit\
> ceil 55mbit
[root@server ~]#
```

Handle identifier: `X:Y`, `Y` = 0 qdisc, `Y` > 0 class

Classful HTB Qdisc At Work

HTB: classes are attached to root qdisc



```
[root@server ~]# tc qdisc add dev eth0 root handle 1: htb default 1
[root@server ~]# tc qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 6 direct_qlen 1000
[root@server ~]# tc class add dev eth0 parent 1: classid 1:1 htb rate 50mbit\
> ceil 55mbit
[root@server ~]#
```

Top rate

This class' parent is 1:0
i.e. the root qdisc
Must Be Set!

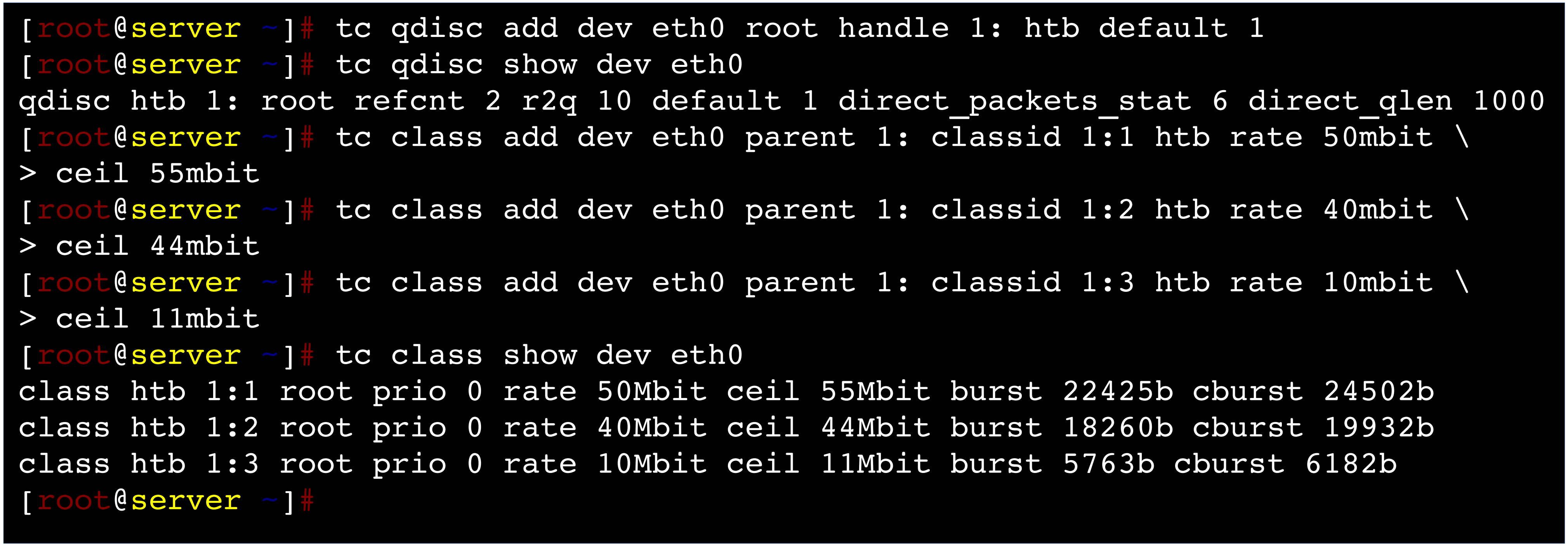
This class id is 1:1

Guaranteed rate

Handle identifier: `X:Y`, `Y` = 0 qdisc, `Y` > 0 class

Classful HTB Qdisc At Work

HTB: classes are attached to root qdisc and are checked

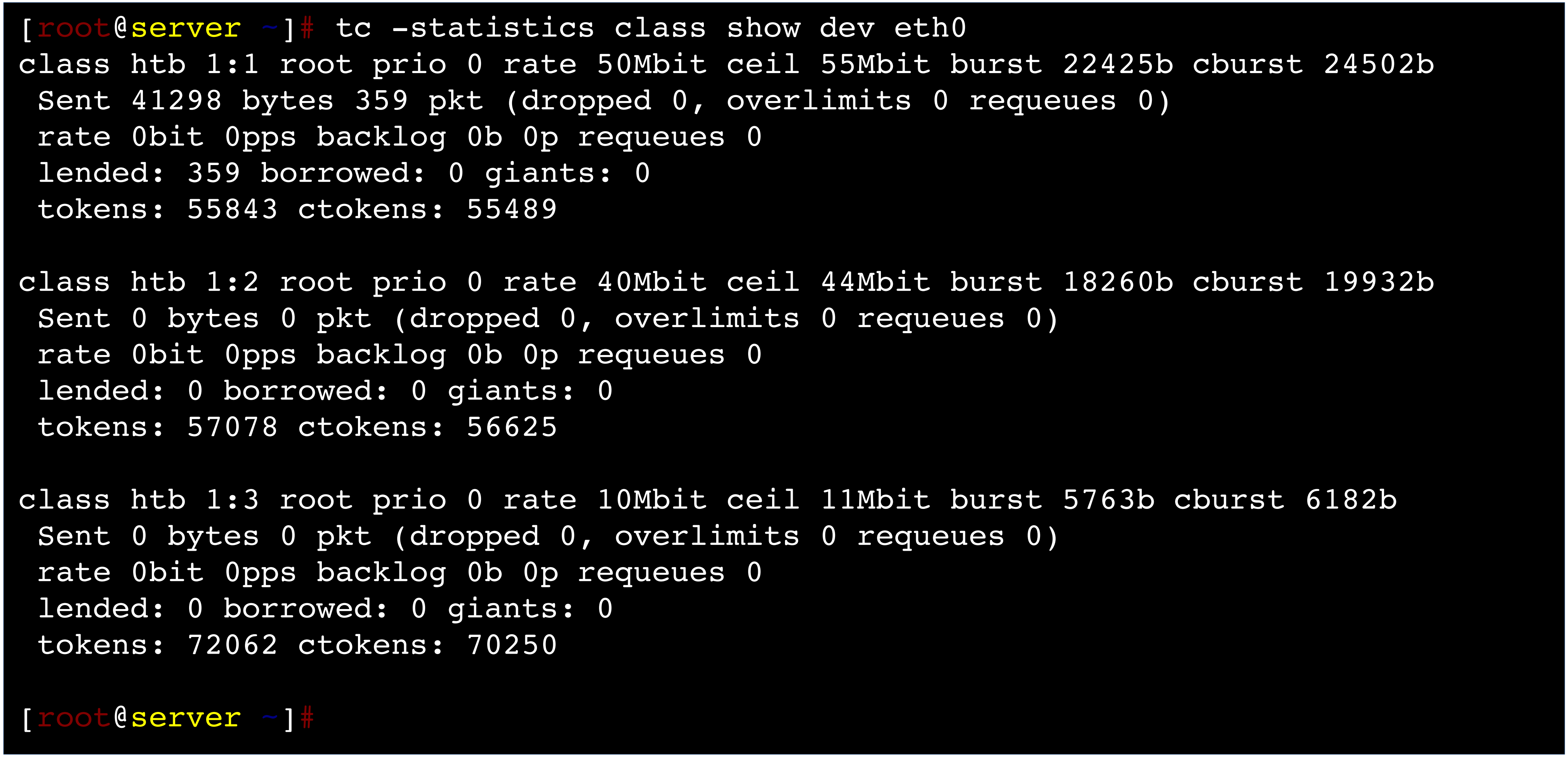


```
[root@server ~]# tc qdisc add dev eth0 root handle 1: htb default 1
[root@server ~]# tc qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 6 direct_qlen 1000
[root@server ~]# tc class add dev eth0 parent 1: classid 1:1 htb rate 50mbit \
> ceil 55mbit
[root@server ~]# tc class add dev eth0 parent 1: classid 1:2 htb rate 40mbit \
> ceil 44mbit
[root@server ~]# tc class add dev eth0 parent 1: classid 1:3 htb rate 10mbit \
> ceil 11mbit
[root@server ~]# tc class show dev eth0
class htb 1:1 root prio 0 rate 50Mbit ceil 55Mbit burst 22425b cburst 24502b
class htb 1:2 root prio 0 rate 40Mbit ceil 44Mbit burst 18260b cburst 19932b
class htb 1:3 root prio 0 rate 10Mbit ceil 11Mbit burst 5763b cburst 6182b
[root@server ~]#
```

Handle identifier: `X:Y`, `Y` = 0 qdisc, `Y` > 0 class

Classful HTB Qdisc At Work

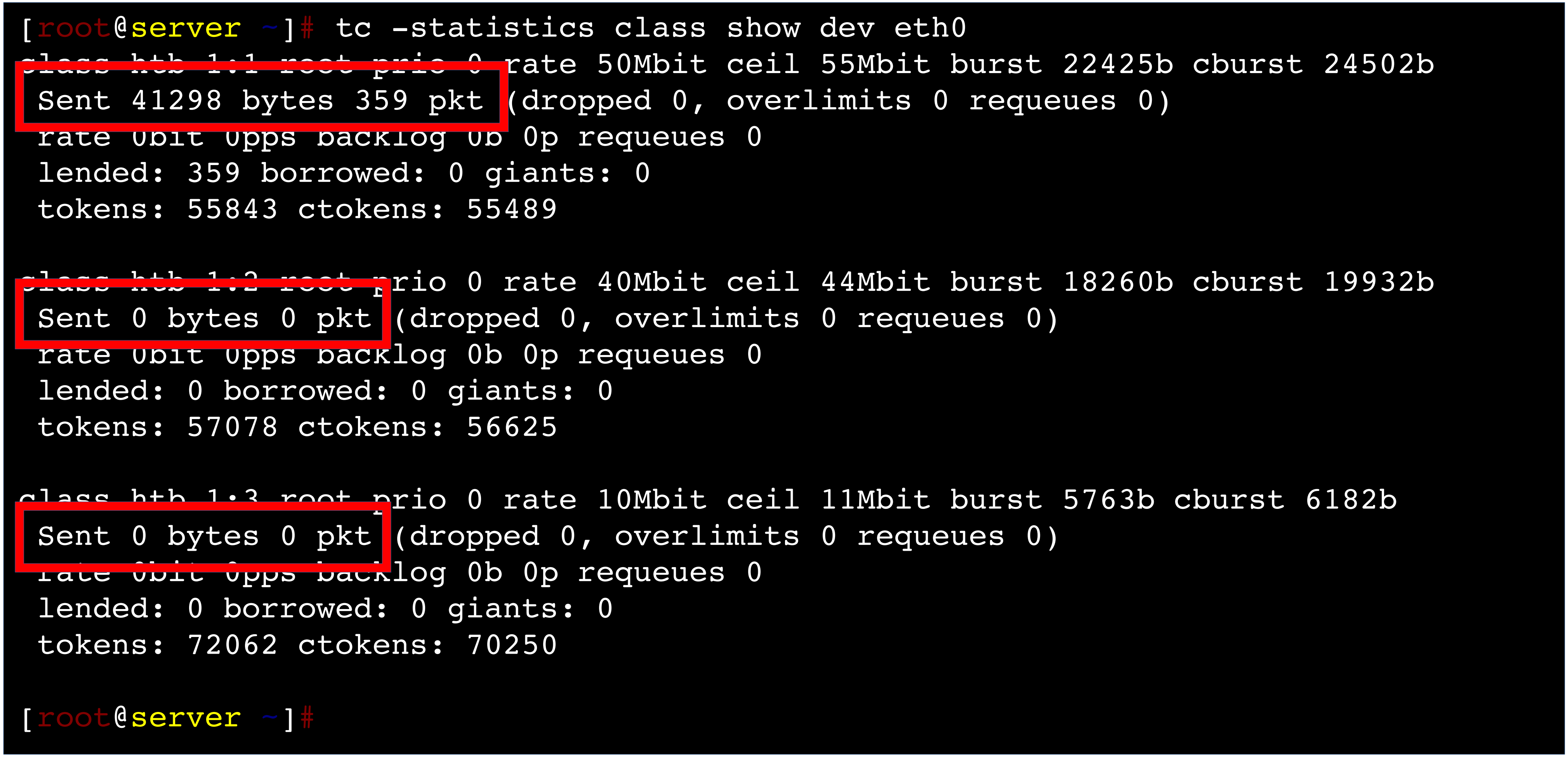
All traffic is now handled by default class 1:1



```
[root@server ~]# tc -statistics class show dev eth0
class htb 1:1 root prio 0 rate 50Mbit ceil 55Mbit burst 22425b cburst 24502b
 Sent 41298 bytes 359 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 359 borrowed: 0 giants: 0
 tokens: 55843 ctokens: 55489

class htb 1:2 root prio 0 rate 40Mbit ceil 44Mbit burst 18260b cburst 19932b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 57078 ctokens: 56625

class htb 1:3 root prio 0 rate 10Mbit ceil 11Mbit burst 5763b cburst 6182b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 72062 ctokens: 70250

[root@server ~]#
```

17 Classful HTB Qdisc At Work

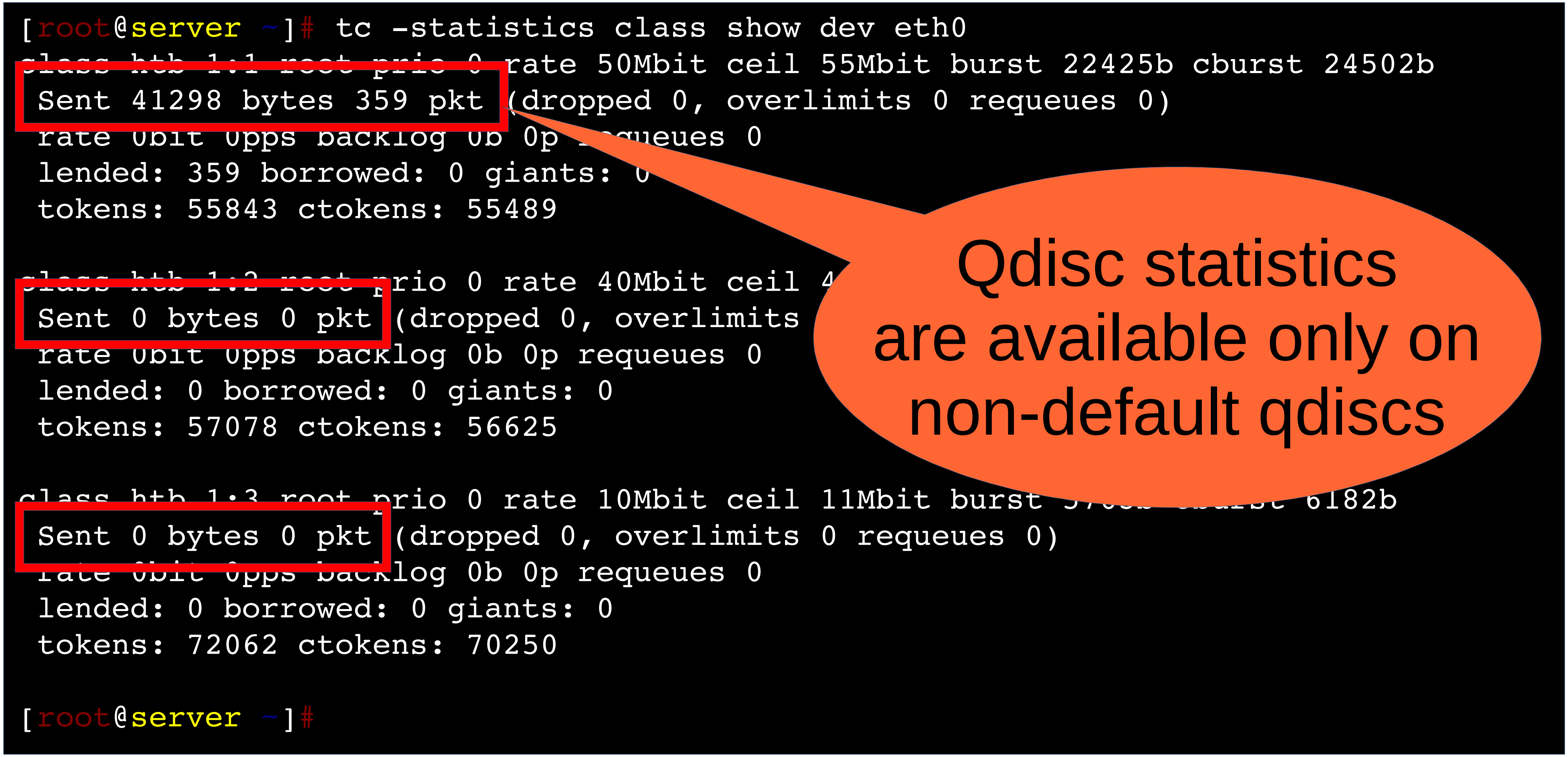
All traffic is now handled by default class 1:1



```
[root@server ~]# tc -statistics class show dev eth0
class htb 1:1 root prio 0 rate 50Mbit ceil 55Mbit burst 22425b cburst 24502b
 Sent 41298 bytes 359 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 359 borrowed: 0 giants: 0
 tokens: 55843 ctokens: 55489

class htb 1:2 root prio 0 rate 40Mbit ceil 44Mbit burst 18260b cburst 19932b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 57078 ctokens: 56625

class htb 1:3 root prio 0 rate 10Mbit ceil 11Mbit burst 5763b cburst 6182b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 72062 ctokens: 70250

[root@server ~]#
```

17 Classful HTB Qdisc At Work

All traffic is now handled by default class 1:1

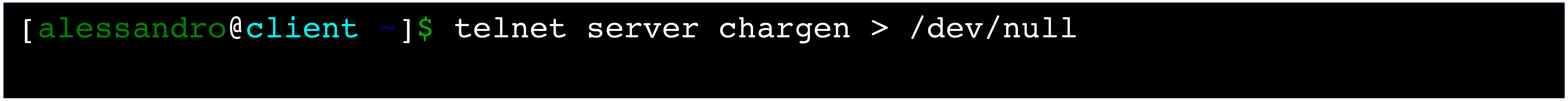


```
[root@server ~]# tc -statistics class show dev eth0
class htb 1:1 root prio 0 rate 50Mbit ceil 55Mbit burst 22425b cburst 24502b
 Sent 41298 bytes 359 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 359 borrowed: 0 giants: 0
 tokens: 55843 ctokens: 55489

class htb 1:2 root prio 0 rate 40Mbit ceil 4
 Sent 0 bytes 0 pkt (dropped 0, overlimits
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 57078 ctokens: 56625

class htb 1:3 root prio 0 rate 10Mbit ceil 11Mbit burst 5705b cburst 6182b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 72062 ctokens: 70250

[root@server ~]#
```

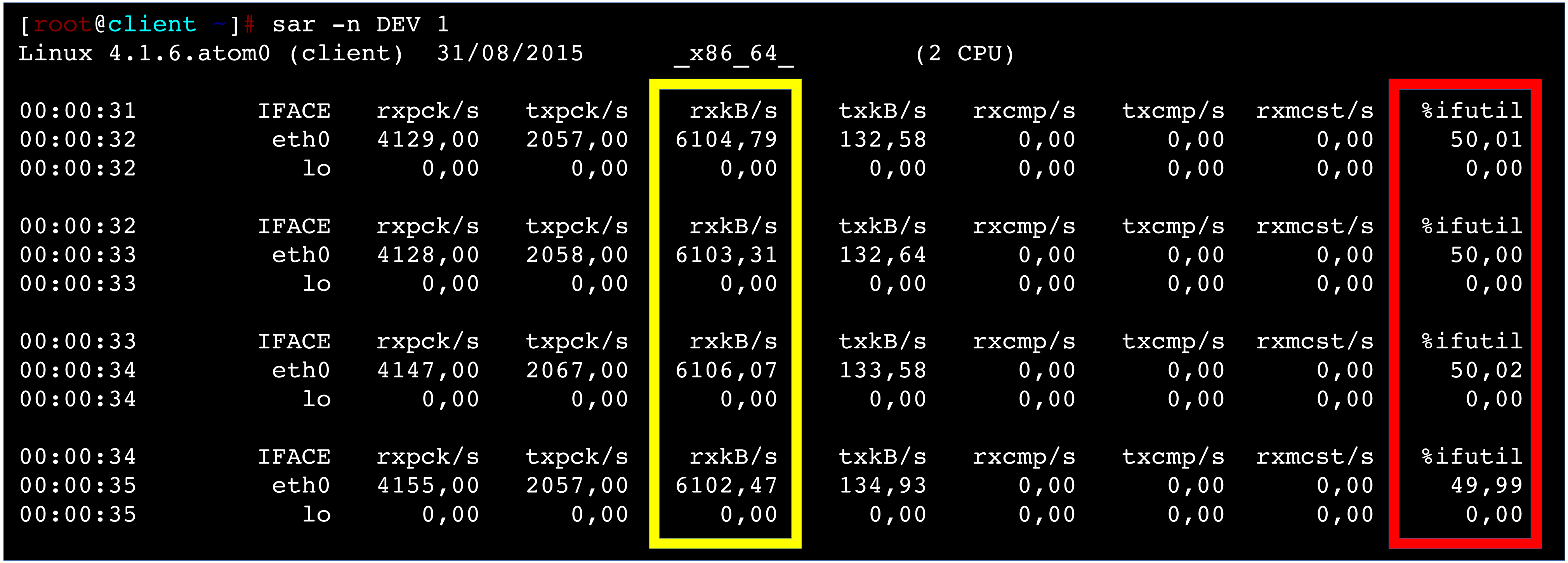
Qdisc statistics
are available only on
non-default qdiscs

17 Classful HTB Qdisc At Work

Class 1:1 traffic limit 50mbps, 50% of 100mbps



```
[alessandro@client ~]$ telnet server chargen > /dev/null
```

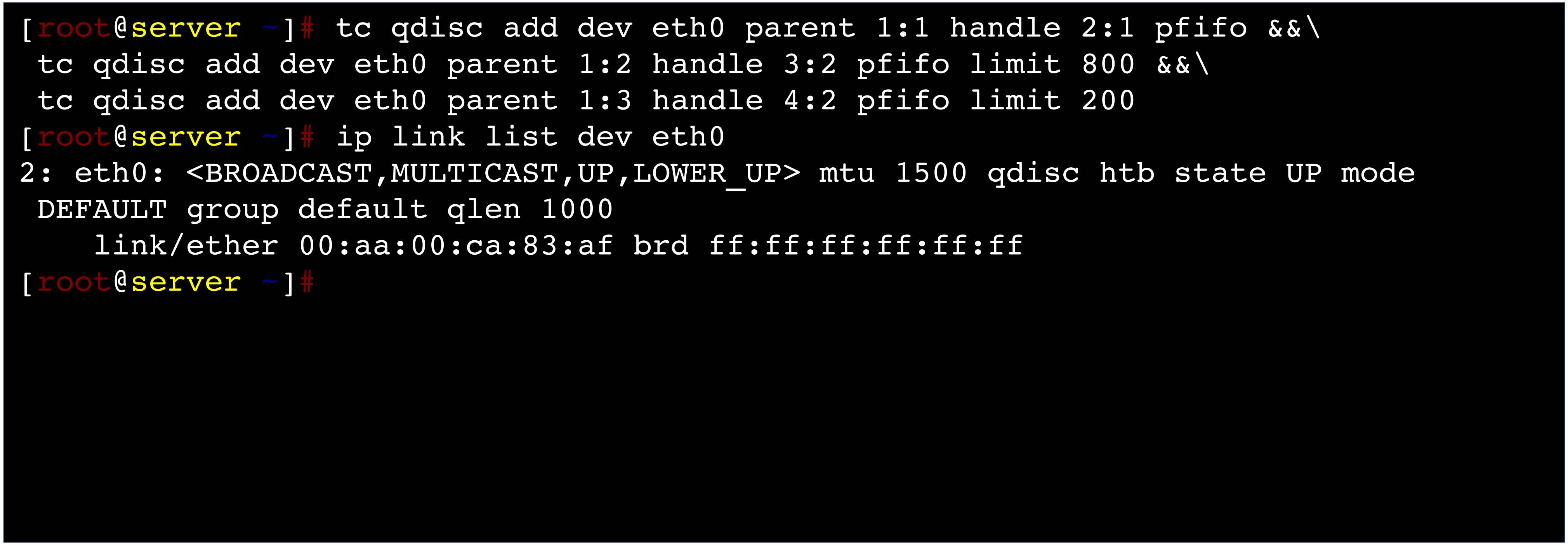


```
[root@client ~]# sar -n DEV 1
Linux 4.1.6.atom0 (client)  31/08/2015      _x86_64_        (2 CPU)

00:00:31        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:00:32         eth0   4129,00   2057,00   6104,79    132,58      0,00      0,00      0,00     50,01
00:00:32           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

00:00:32        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:00:33         eth0   4128,00   2058,00   6103,31    132,64      0,00      0,00      0,00     50,00
00:00:33           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

00:00:33        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:00:34         eth0   4147,00   2067,00   6106,07    133,58      0,00      0,00      0,00     50,02
00:00:34           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

00:00:34        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:00:35         eth0   4155,00   2057,00   6102,47    134,93      0,00      0,00      0,00     49,99
00:00:35           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00
```

50mbps = 6,250kBps

Classful HTB Qdisc At Work

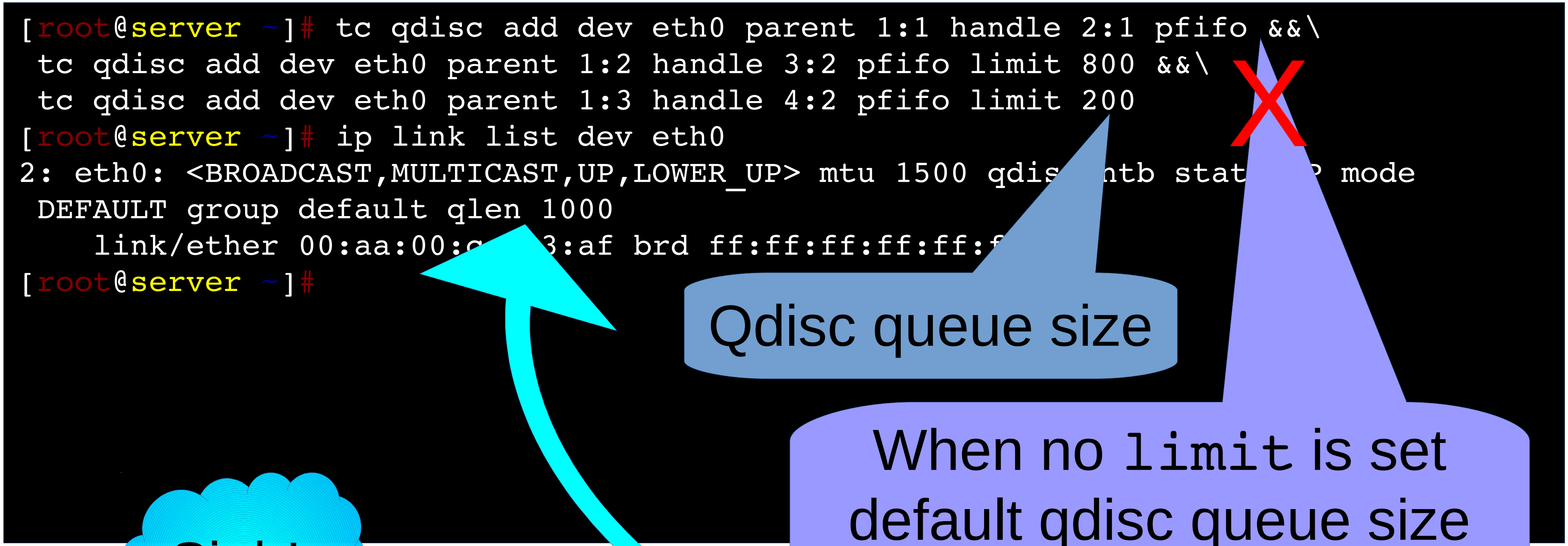
Classes can be given qdiscs:



```
[root@server ~]# tc qdisc add dev eth0 parent 1:1 handle 2:1 pfifo &&\
 tc qdisc add dev eth0 parent 1:2 handle 3:2 pfifo limit 800 &&\
 tc qdisc add dev eth0 parent 1:3 handle 4:2 pfifo limit 200
[root@server ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc htb state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:aa:00:ca:83:af brd ff:ff:ff:ff:ff:ff
[root@server ~]#
```

17 Classful HTB Qdisc At Work

Classes can be given qdiscs:



```
[root@server ~]# tc qdisc add dev eth0 parent 1:1 handle 2:1 pfifo &&\
 tc qdisc add dev eth0 parent 1:2 handle 3:2 pfifo limit 800 &&\
 tc qdisc add dev eth0 parent 1:3 handle 4:2 pfifo limit 200
[root@server ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc htb state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:aa:00:c  3:af brd ff:ff:ff:ff:f
[root@server ~]#
```

X

Qdisc queue size

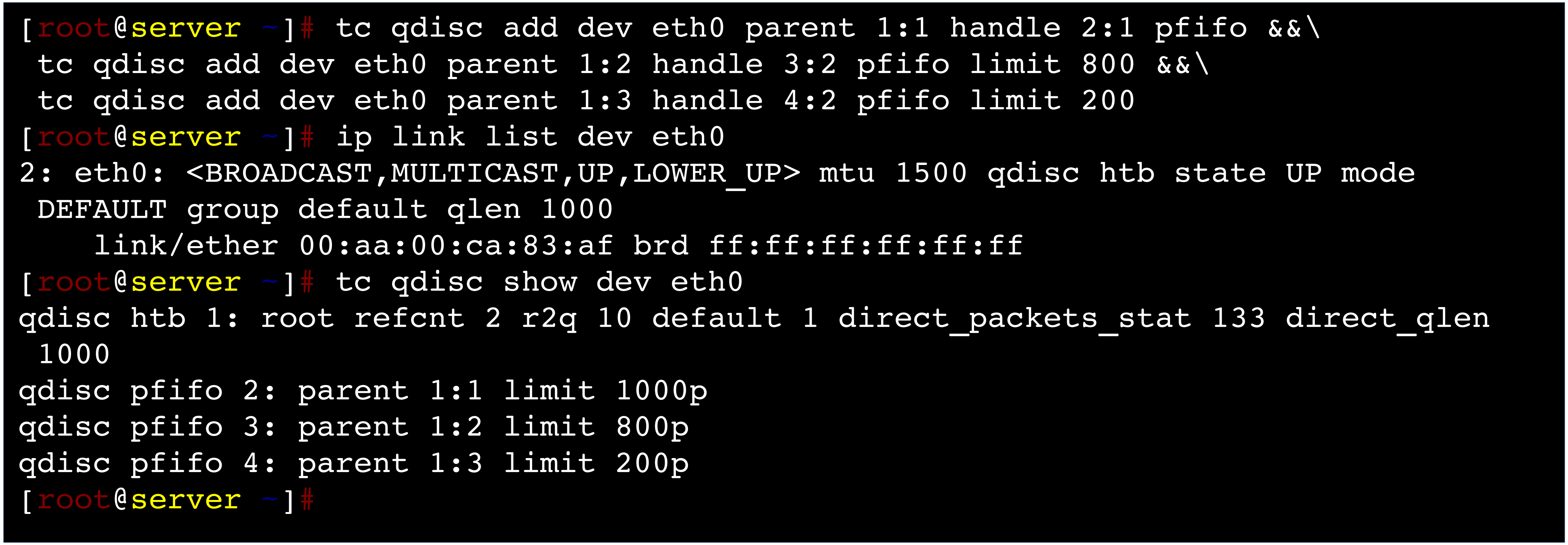
When no `limit` is set default qdisc queue size is interface `txqueuelen`

Sigh!

What `ip link set` calls `txqueuelen` is what `ip link list` calls `qlen` which is what `pfifo` calls `limit`

17 Classful HTB Qdisc At Work

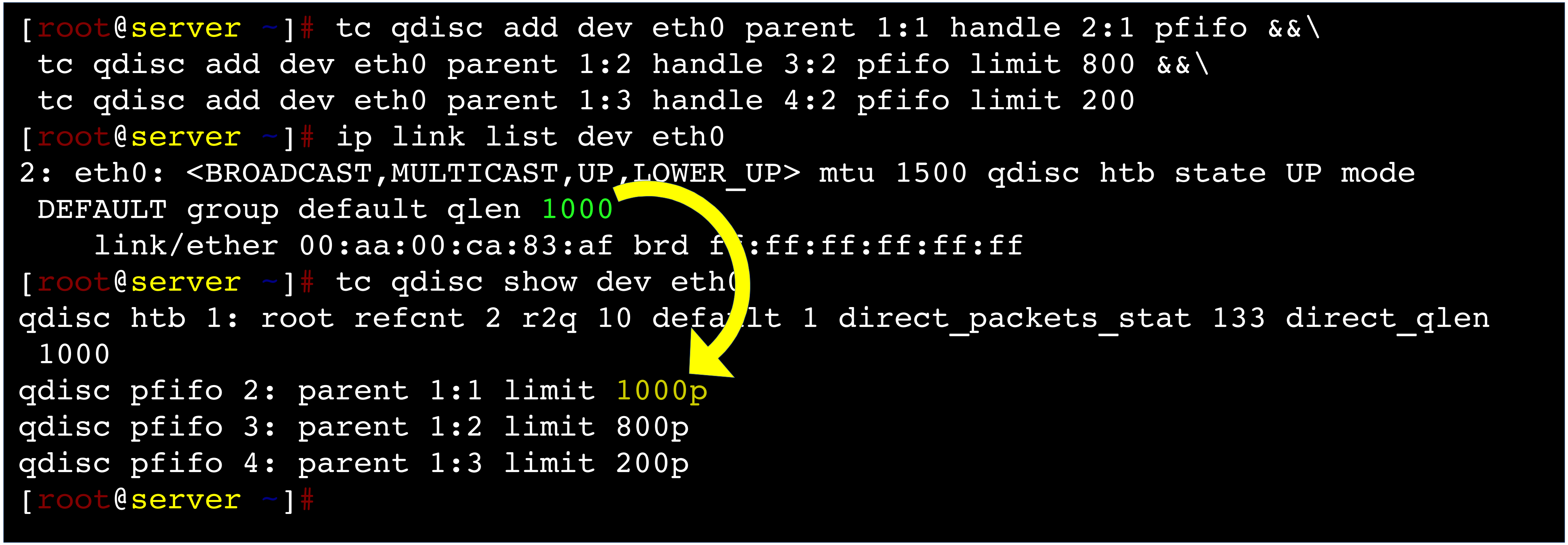
Qdiscs check:



```
[root@server ~]# tc qdisc add dev eth0 parent 1:1 handle 2:1 pfifo &&\
 tc qdisc add dev eth0 parent 1:2 handle 3:2 pfifo limit 800 &&\
 tc qdisc add dev eth0 parent 1:3 handle 4:2 pfifo limit 200
[root@server ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc htb state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:aa:00:ca:83:af brd ff:ff:ff:ff:ff:ff
[root@server ~]# tc qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 133 direct_qlen
 1000
qdisc pfifo 2: parent 1:1 limit 1000p
qdisc pfifo 3: parent 1:2 limit 800p
qdisc pfifo 4: parent 1:3 limit 200p
[root@server ~]#
```

17 Classful HTB Qdisc At Work

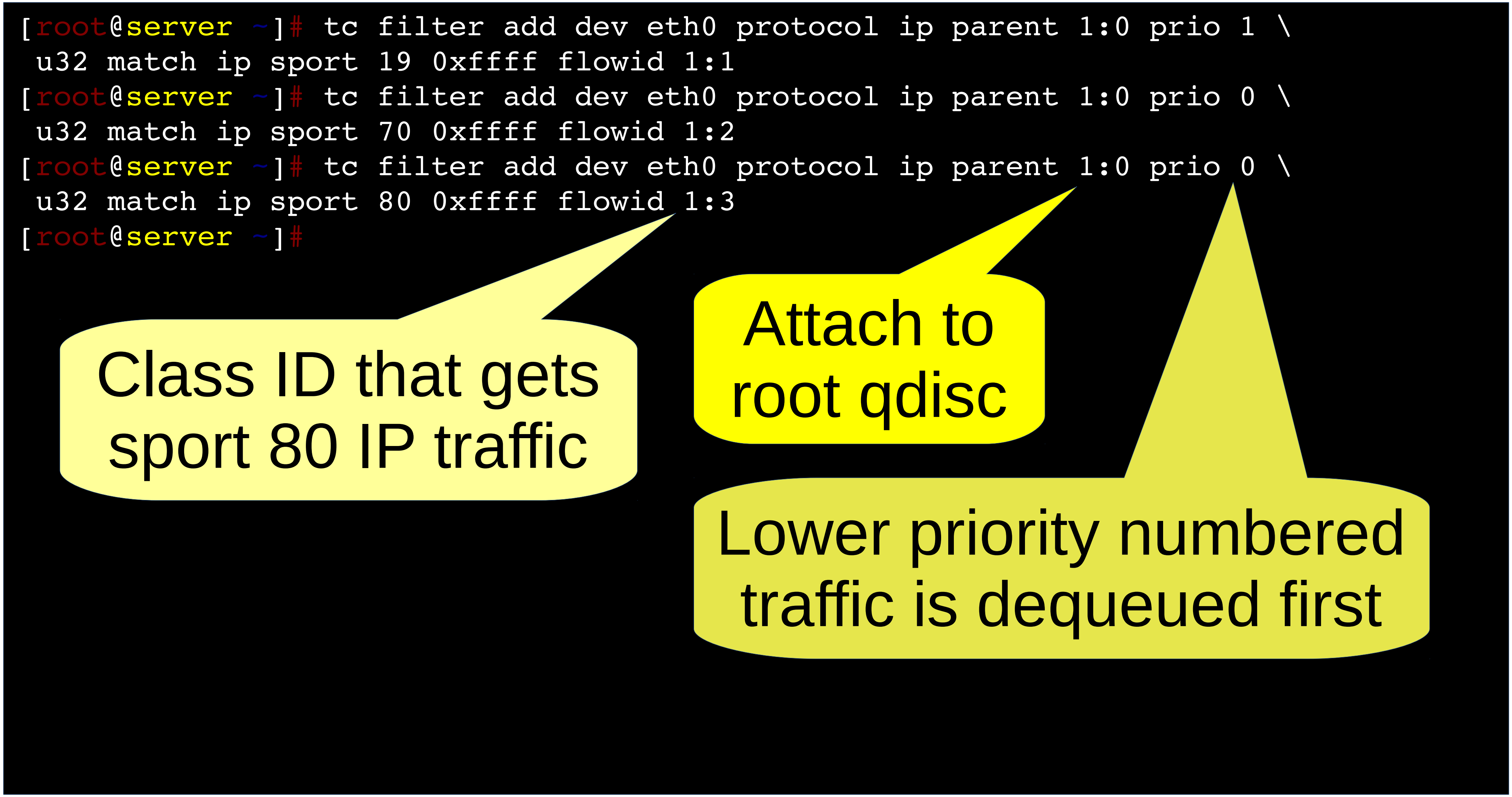
Qdiscs check:



```
[root@server ~]# tc qdisc add dev eth0 parent 1:1 handle 2:1 pfifo &&\
 tc qdisc add dev eth0 parent 1:2 handle 3:2 pfifo limit 800 &&\
 tc qdisc add dev eth0 parent 1:3 handle 4:2 pfifo limit 200
[root@server ~]# ip link list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc htb state UP mode
 DEFAULT group default qlen 1000
    link/ether 00:aa:00:ca:83:af brd ff:ff:ff:ff:ff:ff
[root@server ~]# tc qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 133 direct_qlen
 1000
qdisc pfifo 2: parent 1:1 limit 1000p
qdisc pfifo 3: parent 1:2 limit 800p
qdisc pfifo 4: parent 1:3 limit 200p
[root@server ~]#
```

17 Classful HTB Qdisc At Work

Filters tell `tc` to what class direct what traffic:



```
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 \
 u32 match ip sport 19 0xffff flowid 1:1
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 0 \
 u32 match ip sport 70 0xffff flowid 1:2
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 0 \
 u32 match ip sport 80 0xffff flowid 1:3
[root@server ~]#
```

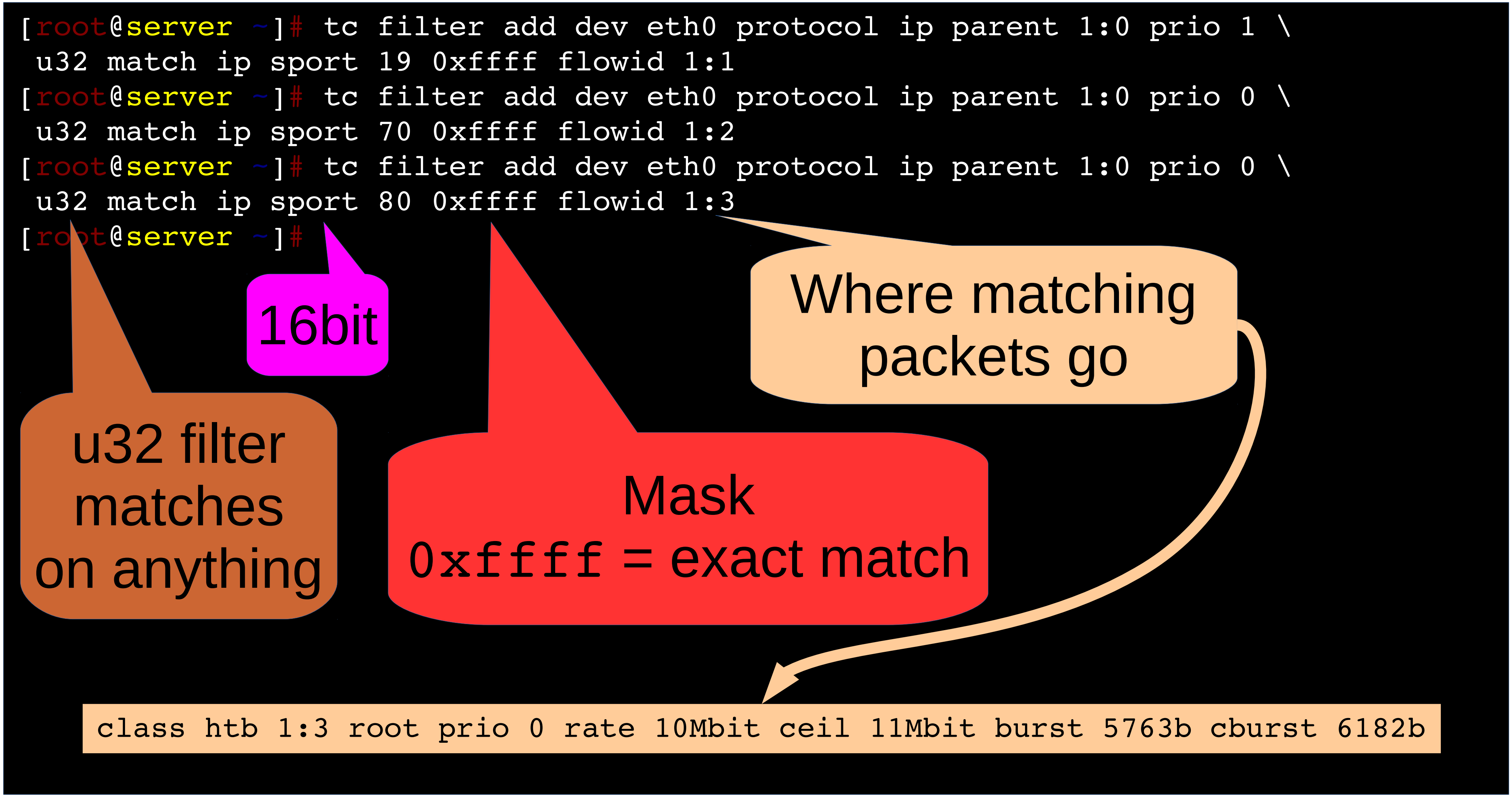
Class ID that gets sport 80 IP traffic

Attach to root qdisc

Lower priority numbered traffic is dequeued first

17 Classful HTB Qdisc At Work

Filters tell `tc` to what class direct what traffic:



```
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 \
 u32 match ip sport 19 0xffff flowid 1:1
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 0 \
 u32 match ip sport 70 0xffff flowid 1:2
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 0 \
 u32 match ip sport 80 0xffff flowid 1:3
[root@server ~]#
```

16bit

Where matching packets go

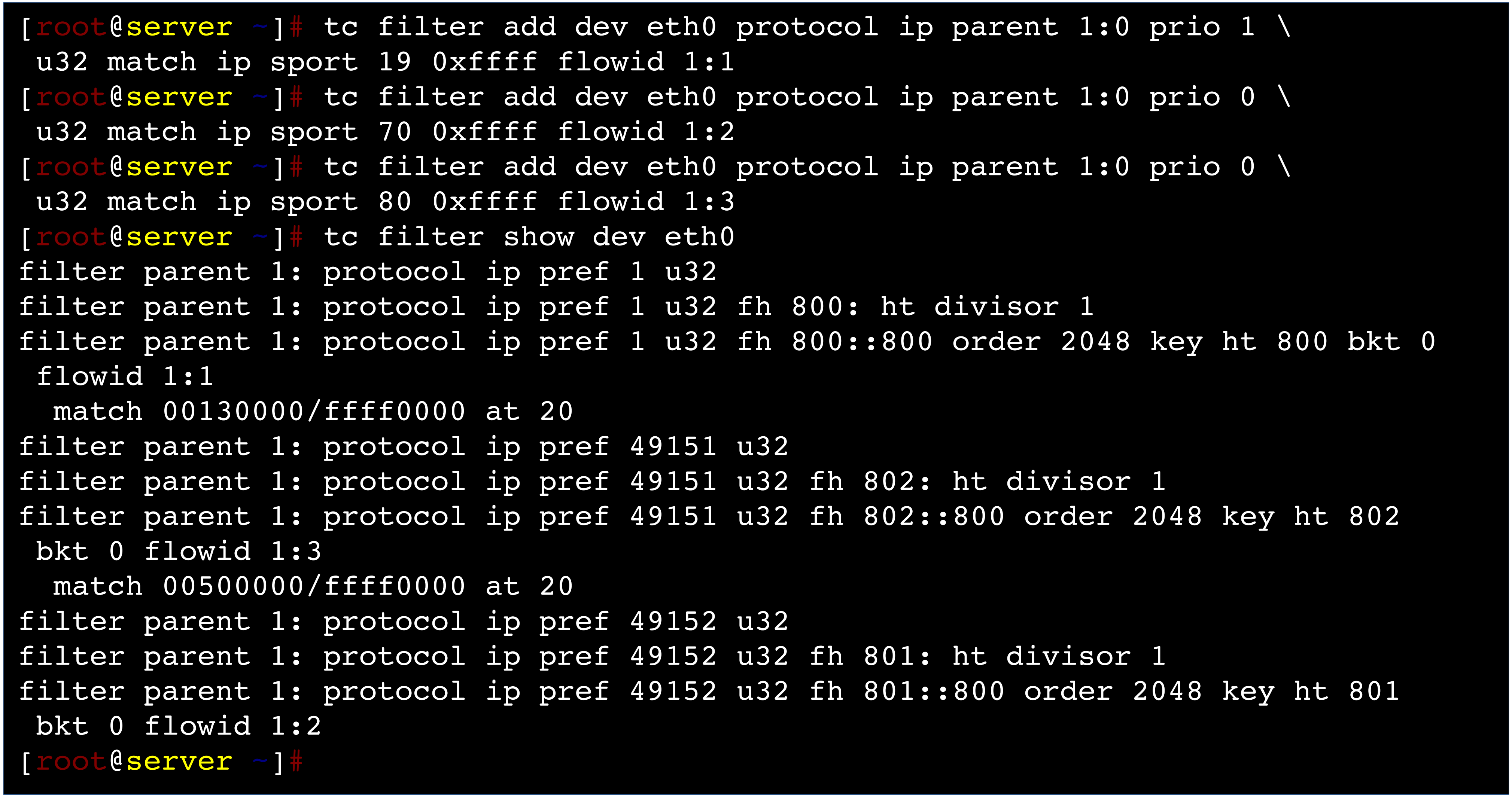
u32 filter matches on anything

Mask `0xffff` = exact match

```
class htb 1:3 root prio 0 rate 10Mbit ceil 11Mbit burst 5763b cburst 6182b
```

Classful HTB Qdisc At Work

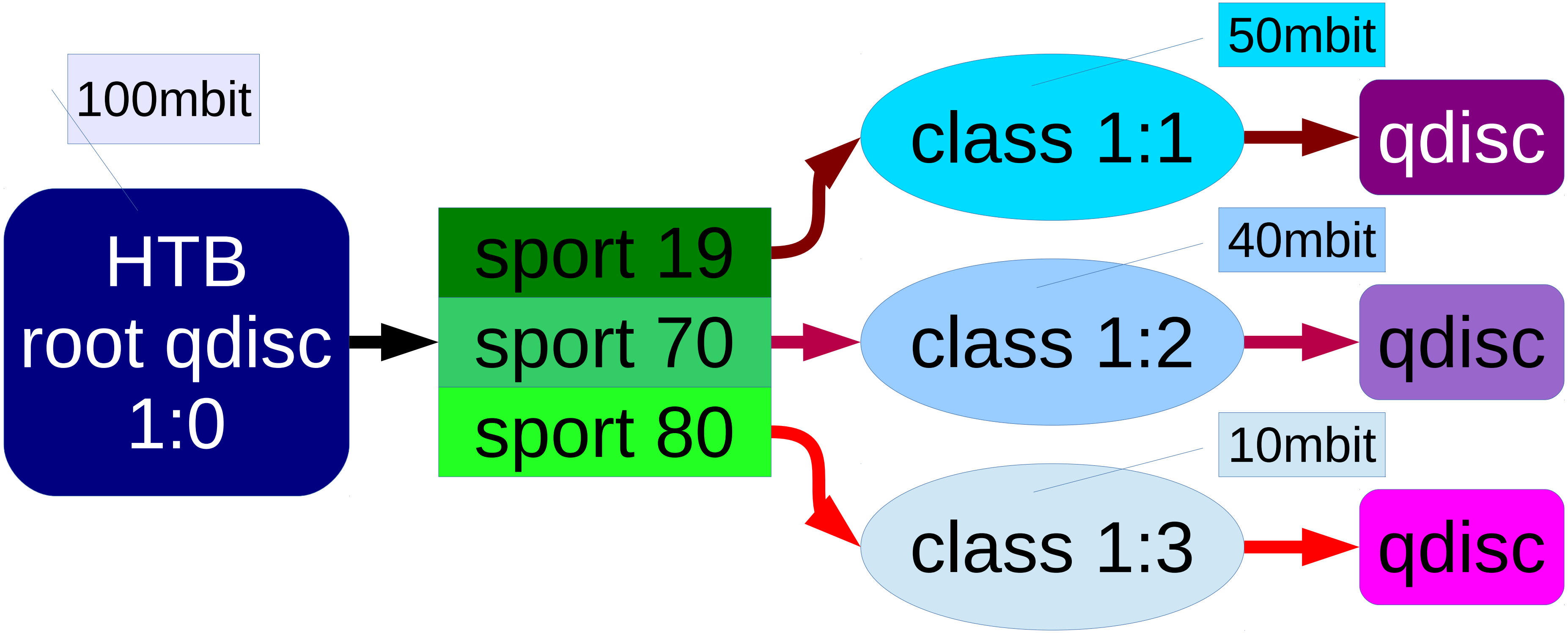
Filter check:



```
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 \
 u32 match ip sport 19 0xffff flowid 1:1
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 0 \
 u32 match ip sport 70 0xffff flowid 1:2
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 0 \
 u32 match ip sport 80 0xffff flowid 1:3
[root@server ~]# tc filter show dev eth0
filter parent 1: protocol ip pref 1 u32
filter parent 1: protocol ip pref 1 u32 fh 800: ht divisor 1
filter parent 1: protocol ip pref 1 u32 fh 800::800 order 2048 key ht 800 bkt 0
 flowid 1:1
  match 00130000/ffff0000 at 20
filter parent 1: protocol ip pref 49151 u32
filter parent 1: protocol ip pref 49151 u32 fh 802: ht divisor 1
filter parent 1: protocol ip pref 49151 u32 fh 802::800 order 2048 key ht 802
 bkt 0 flowid 1:3
  match 00500000/ffff0000 at 20
filter parent 1: protocol ip pref 49152 u32
filter parent 1: protocol ip pref 49152 u32 fh 801: ht divisor 1
filter parent 1: protocol ip pref 49152 u32 fh 801::800 order 2048 key ht 801
 bkt 0 flowid 1:2
[root@server ~]#
```

17 Classful HTB Qdisc At Work

Recap: How we are set now on the server:

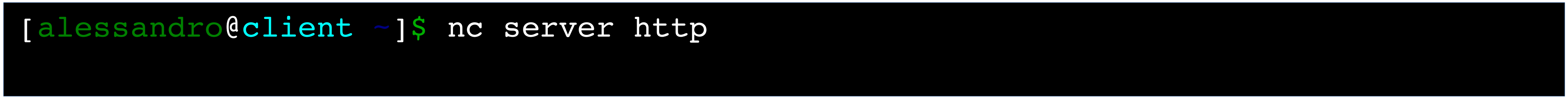


Classful HTB Qdisc At Work

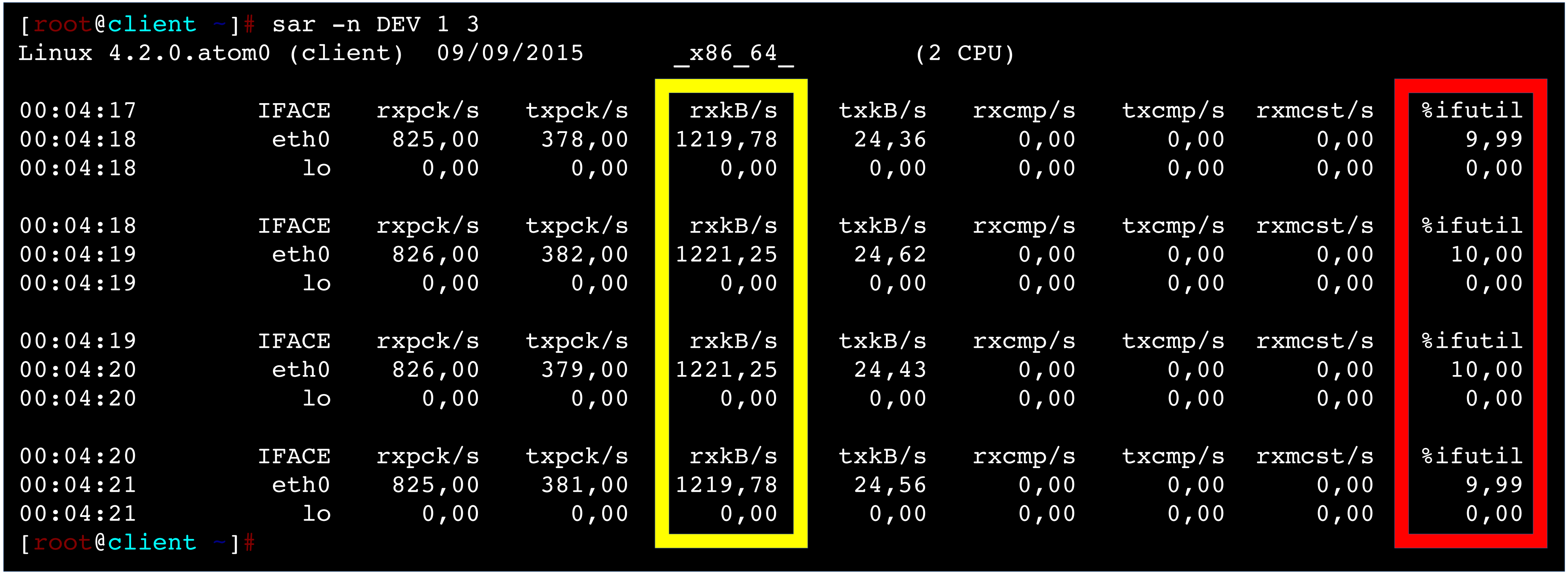
- Traffic from server:19 (chargen) to client: no changes.
- Same 50%, ~6100kBps measured on client as before.

17 Classful HTB Qdisc At Work

Class 1:3 traffic limit 10mbps, 10% of 100mbps



```
[alessandro@client ~]$ nc server http
```

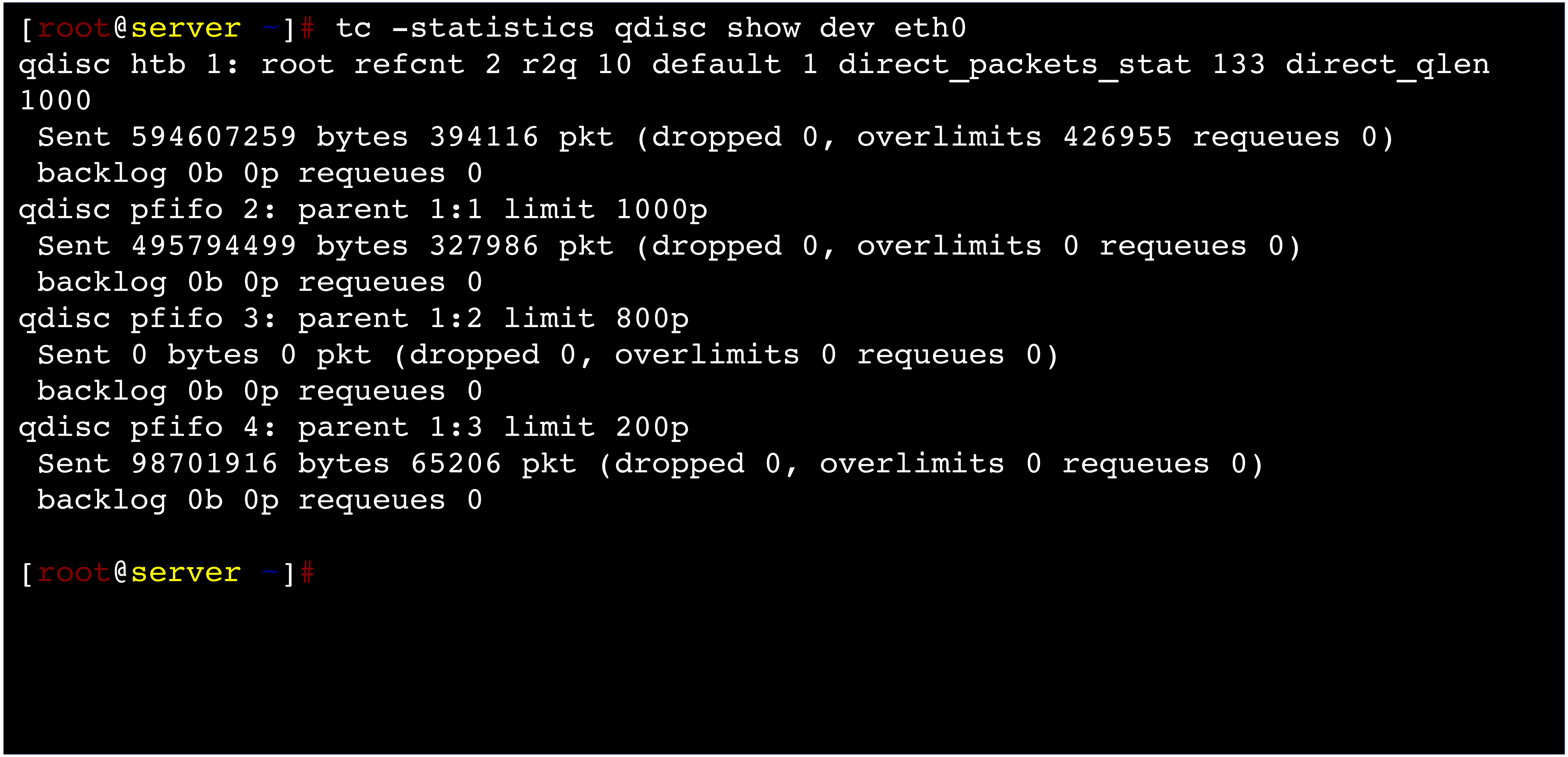


```
[root@client ~]# sar -n DEV 1 3
Linux 4.2.0.atom0 (client)  09/09/2015      _x86_64_        (2 CPU)

00:04:17        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:04:18         eth0    825,00    378,00   1219,78     24,36      0,00      0,00      0,00      9,99
00:04:18           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

00:04:18        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:04:19         eth0    826,00    382,00   1221,25     24,62      0,00      0,00      0,00     10,00
00:04:19           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

00:04:19        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:04:20         eth0    826,00    379,00   1221,25     24,43      0,00      0,00      0,00     10,00
00:04:20           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00

00:04:20        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s   %ifutil
00:04:21         eth0    825,00    381,00   1219,78     24,56      0,00      0,00      0,00      9,99
00:04:21           lo      0,00      0,00      0,00      0,00      0,00      0,00      0,00      0,00
[root@client ~]#
```

10mbps = 1,250kBps

Classful HTB Qdisc At Work

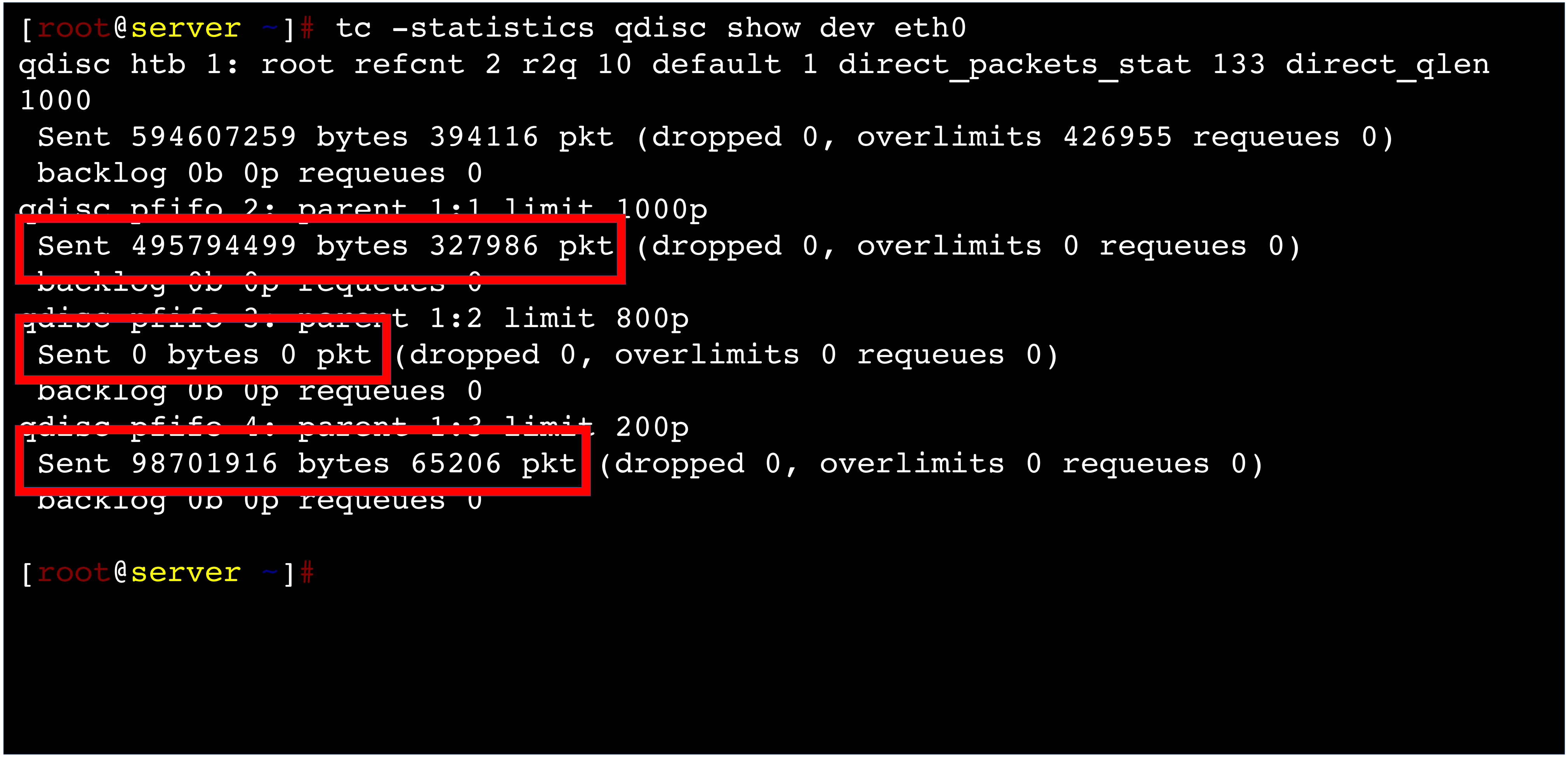
Correspondingly, server **qdisc** stats:



```
[root@server ~]# tc -statistics qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 133 direct_qlen
1000
 Sent 594607259 bytes 394116 pkt (dropped 0, overlimits 426955 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 2: parent 1:1 limit 1000p
 Sent 495794499 bytes 327986 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 3: parent 1:2 limit 800p
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 4: parent 1:3 limit 200p
 Sent 98701916 bytes 65206 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0

[root@server ~]#
```

17 Classful HTB Qdisc At Work

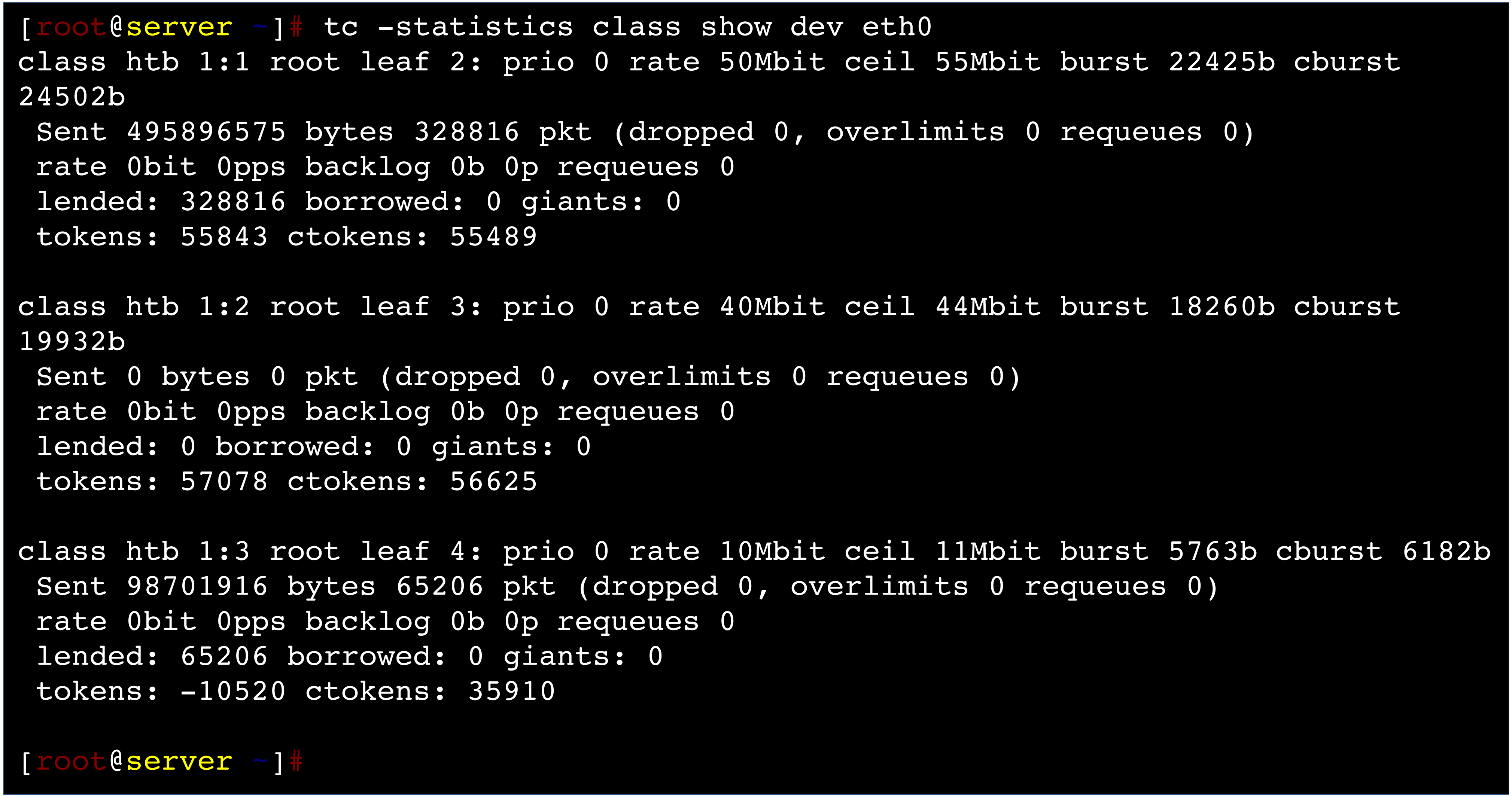
Correspondingly, server **qdisc** stats:



```
[root@server ~]# tc -statistics qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 1 direct_packets_stat 133 direct_qlen
1000
 Sent 594607259 bytes 394116 pkt (dropped 0, overlimits 426955 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 2: parent 1:1 limit 1000p
 Sent 495794499 bytes 327986 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 3: parent 1:2 limit 800p
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 4: parent 1:3 limit 200p
 Sent 98701916 bytes 65206 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0

[root@server ~]#
```

17 Classful HTB Qdisc At Work

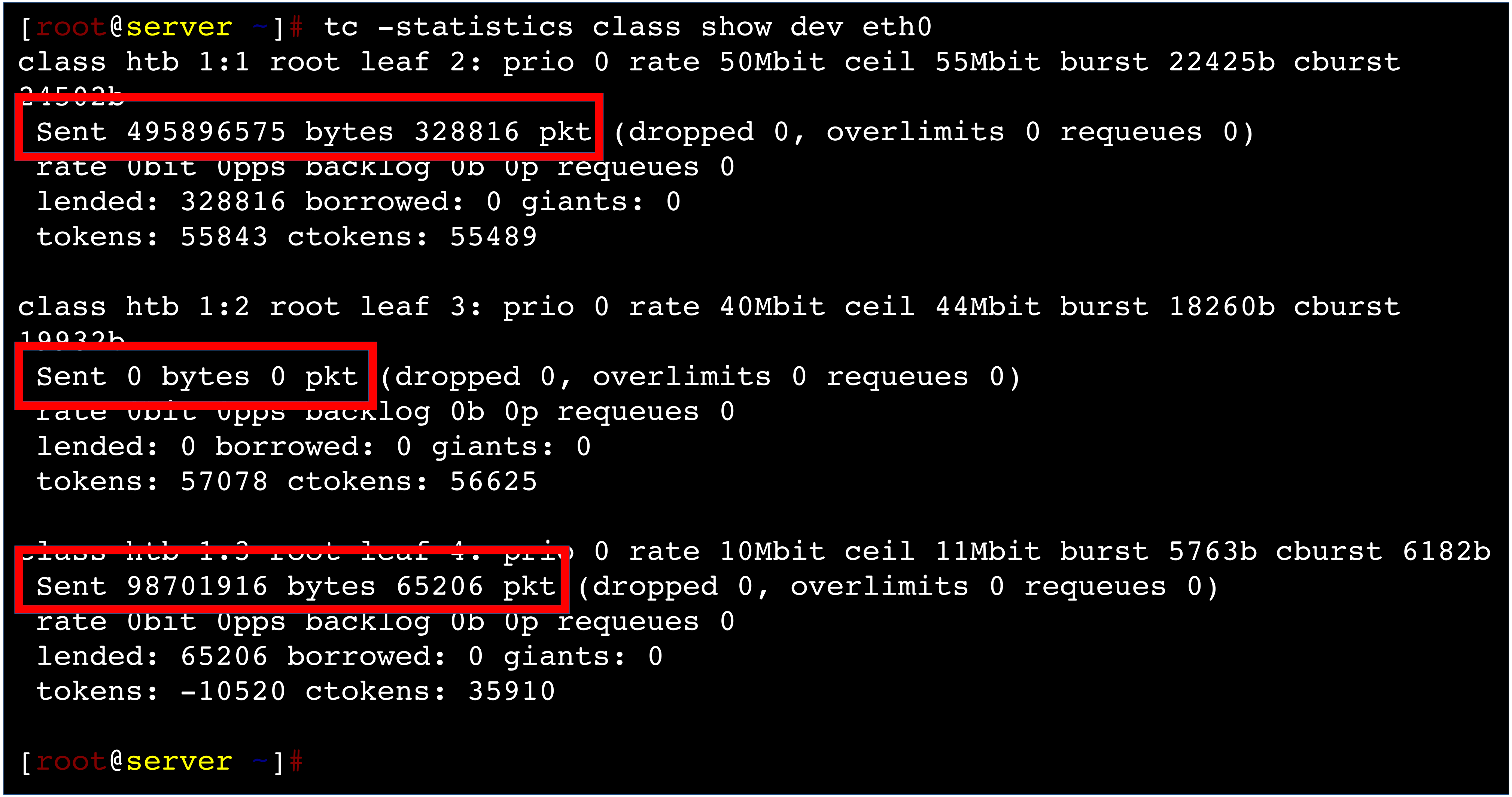
Correspondingly, server **class** stats:



```
[root@server ~]# tc -statistics class show dev eth0
class htb 1:1 root leaf 2: prio 0 rate 50Mbit ceil 55Mbit burst 22425b cburst
24502b
 Sent 495896575 bytes 328816 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 328816 borrowed: 0 giants: 0
 tokens: 55843 ctokens: 55489

class htb 1:2 root leaf 3: prio 0 rate 40Mbit ceil 44Mbit burst 18260b cburst
19932b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 57078 ctokens: 56625

class htb 1:3 root leaf 4: prio 0 rate 10Mbit ceil 11Mbit burst 5763b cburst 6182b
 Sent 98701916 bytes 65206 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 65206 borrowed: 0 giants: 0
 tokens: -10520 ctokens: 35910

[root@server ~]#
```

Classful HTB Qdisc At Work

Correspondingly, server **class** stats:



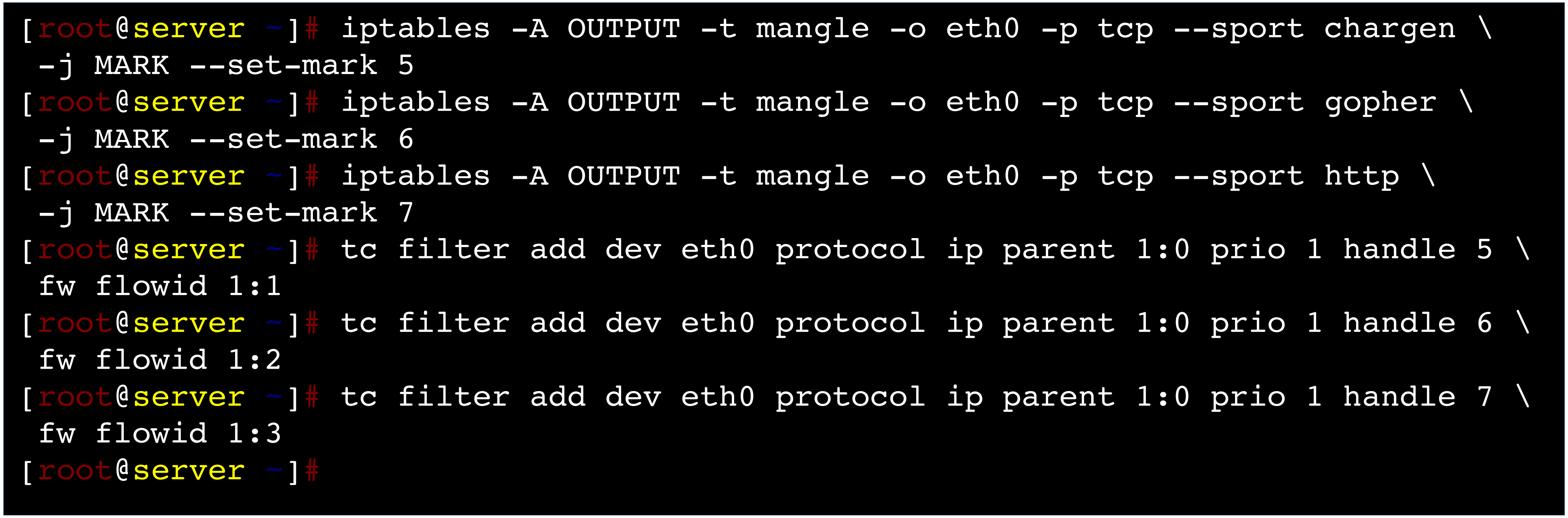
```
[root@server ~]# tc -statistics class show dev eth0
class htb 1:1 root leaf 2: prio 0 rate 50Mbit ceil 55Mbit burst 22425b cburst
24502b
 Sent 495896575 bytes 328816 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 328816 borrowed: 0 giants: 0
 tokens: 55843 ctokens: 55489

class htb 1:2 root leaf 3: prio 0 rate 40Mbit ceil 44Mbit burst 18260b cburst
19932b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 57078 ctokens: 56625

class htb 1:3 root leaf 4: prio 0 rate 10Mbit ceil 11Mbit burst 5763b cburst 6182b
 Sent 98701916 bytes 65206 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 65206 borrowed: 0 giants: 0
 tokens: -10520 ctokens: 35910

[root@server ~]#
```

Classful HTB Qdisc At Work

- Similar thing regarding traffic from server:70 (gopher) to client.
- We get 39.99%, 4881.78kBps measured on client.
- If client opens multiple connections to separate ports on the server, traffick adds up
  - I.e. chargen+http = 50%+10%=60%
  - chargen+gopher = 50%+40%=90%
  - gopher+http = 40%+10%=50%
  - chargen+gopher+http = 50%+40%+10%=100%

Filter classifiers

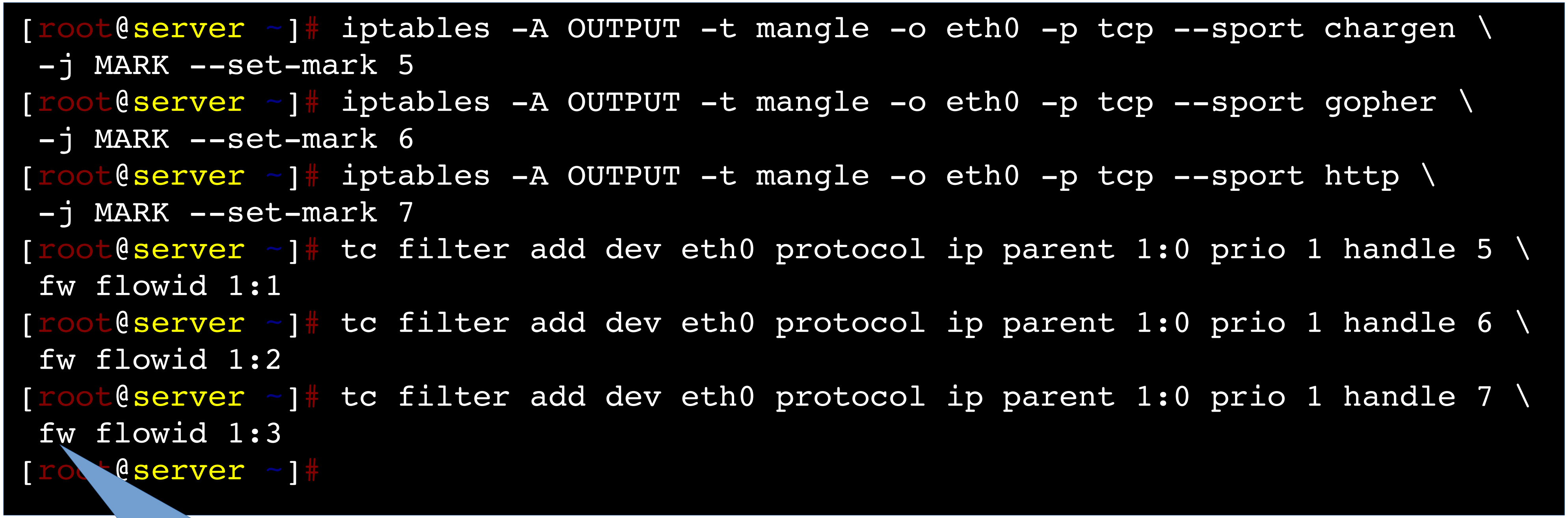
Same thing can be done with `iptables` marks instead of `u32` matches:



```
[root@server ~]# iptables -A OUTPUT -t mangle -o eth0 -p tcp --sport chargen \
 -j MARK --set-mark 5
[root@server ~]# iptables -A OUTPUT -t mangle -o eth0 -p tcp --sport gopher \
 -j MARK --set-mark 6
[root@server ~]# iptables -A OUTPUT -t mangle -o eth0 -p tcp --sport http \
 -j MARK --set-mark 7
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 handle 5 \
 fw flowid 1:1
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 handle 6 \
 fw flowid 1:2
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 handle 7 \
 fw flowid 1:3
[root@server ~]#
```

Filter classifiers

Same thing can be done with `iptables` marks instead of `u32` matches:



```
[root@server ~]# iptables -A OUTPUT -t mangle -o eth0 -p tcp --sport chargen \
 -j MARK --set-mark 5
[root@server ~]# iptables -A OUTPUT -t mangle -o eth0 -p tcp --sport gopher \
 -j MARK --set-mark 6
[root@server ~]# iptables -A OUTPUT -t mangle -o eth0 -p tcp --sport http \
 -j MARK --set-mark 7
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 handle 5 \
 fw flowid 1:1
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 handle 6 \
 fw flowid 1:2
[root@server ~]# tc filter add dev eth0 protocol ip parent 1:0 prio 1 handle 7 \
 fw flowid 1:3
[root@server ~]#
```

Not a `u32` filter

18 Filter classifiers

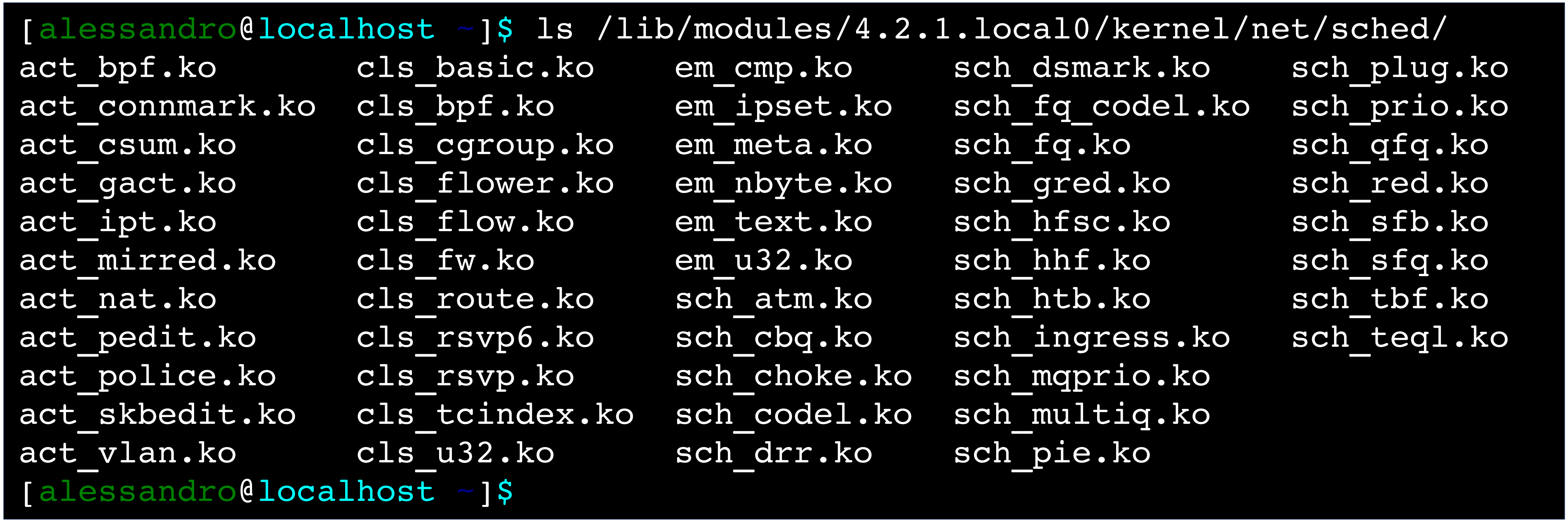
Filters can use several classifiers, including:

1) `fw`, firewall
2) `route`, route
3) `tcindex`, tcindex
4) `u32`, u32
5) `basic`, basic
6) `cgroup`, Control Group Classifier

They allow `tc` to do many things that can be done with netfilter.

Filter classifiers

List of kernel supported modules:



```
[alessandro@localhost ~]$ ls /lib/modules/4.2.1.local0/kernel/net/sched/
act_bpf.ko       cls_basic.ko    em_cmp.ko     sch_dsmark.ko    sch_plug.ko
act_connmark.ko  cls_bpf.ko      em_ipset.ko   sch_fq_codel.ko  sch_prio.ko
act_csum.ko      cls_cgroup.ko   em_meta.ko    sch_fq.ko        sch_qfq.ko
act_gact.ko      cls_flower.ko   em_nbyte.ko   sch_gred.ko      sch_red.ko
act_ipt.ko       cls_flow.ko     em_text.ko    sch_hfsc.ko      sch_sfb.ko
act_mirred.ko    cls_fw.ko       em_u32.ko     sch_hhf.ko       sch_sfq.ko
act_nat.ko       cls_route.ko    sch_atm.ko    sch_htb.ko       sch_tbf.ko
act_pedit.ko     cls_rsvp6.ko    sch_cbq.ko    sch_ingress.ko   sch_teql.ko
act_police.ko    cls_rsvp.ko     sch_choke.ko  sch_mqprio.ko
act_skbedit.ko   cls_tcindex.ko  sch_codel.ko  sch_multiq.ko
act_vlan.ko      cls_u32.ko      sch_drr.ko    sch_pie.ko
[alessandro@localhost ~]$
```

`act` = action

`cls` = classifier

`em` = extended match

`sch` = scheduler

Filter classifiers

But:

1) netfilter does more things
2) netfilter is faster than packet scheduling
   - except when eBPF is used[1]
3) packet flow inside Linux networking stack must be kept in mind, expecially when natting

[1] According to Michael Holzheu, “eBPF on the Mainframe - Packet Filtering and More”, LinuxCon2015, Mon. Oct. 5th

19 Wish list

- Documentation!
- `lartc.org` stopped in 2006, kernel 2.4
- man tc-filter(8) missing
- Tutorials lacking
- Wider:
  - awareness of `tc`
  - user base
  - mention in certifications (LF and LPI)
- Naming consistency (txqueuelen = qlen, limit = buffer, burst ≠ buffer etc.)

20 Docs & Credits

- Alexey Kuznetsov, first/main Linux scheduler developer
- Authors and maintainers of the Linux Advanced Routing and Traffic Control HOWTO http://lartc.org/ (project needs to be revived and updated)
- OpenWRT developers: http://wiki.openwrt.org/doc/howto/packet.scheduler/packet.scheduler
- Dan Siemon: Queueing in the Linux Network Stack (2013) http://www.coverfire.com/articles/queueing-in-the-linux-network-stack/
- Advanced traffic control – ArchWiki https://wiki.archlinux.org/index.php/Advanced_traffic_control
- Linux Kernel documentation https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt
- Linux Foundation's iproute2_examples http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2_examples
- The author thanks Mr. Giovambattista Vieri for his tips and encouragement <g.vieri@ent-it.com>