Compromising Linux the Right Way: 0-days, Novel Techniques, and Lessons from Failure

Pumpkin Chang (@u1f383)

August 16, 2025

DEVCORE

Hacks In Taiwan Conference

$ whoami

- Pumpkin 🎃 (@u1f383)
- Security researcher in DEVCORE
- Focus on Linux Kernel & Virtual Machine

This is a story of failure — none of the mentioned CVEs are mine 😣.

What You Can Learn from This Talk

- Evolution of bug types in the Linux net/sched subsystem
- Analysis and exploitation of three net/sched vulnerabilities
- Novel or lesser-known exploitation techniques
- kernelCTF experience
- The mindset for Linux kernel research 😎

Introduction

$ kernelCTF

security-research

kernelCTF rules

kernelCTF is a part of the Google VRP and is focused on making exploiting Linux kernel vulnerabilities harder by inviting security researchers to demonstrate their exploitation techniques on 0-day and 1-day vulnerabilities in various kernel versions. This includes the kernel version with our experimental mitigations; we'd like to see if and how researchers can bypass these mitigations.

We are asking researchers to publish their submissions, helping the community to learn from each other's techniques.

https://google.github.io/security-research/kernelctf/rules.html

$ kernelCTF

- A bounty program for Linux 0-day and LPE exploits
- Goal: escalate to root user and read the /flag
- Targets:
  1. LTS (Long Term Support)
  2. Mitigation
  3. COS (Container-Optimized OS)

$ kernelCTF

- Finding 0-days is harder than on typical Linux distros
  - Runs in nsjail with limited resources & restricted device access
  - Few feature-based kernel configs enabled

➡ Greatly reduced attack surface

$ kernelCTF

- Exploit is easier and more feasible!
  - KASLR bypass via hardware side-channels
  - Less memory noise
- In short: exploit-friendly!

$ kernelCTF

- Hot attack surfaces in LTS:
  - netfilter (*banned since March 15, 2024*)
  - net/sched
  - bpf
- Chose net/sched as my first research as a kernel beginner

$ kernelCTF

- Hot attack surfaces in LTS:
  - netfilter (*banned since March 15, 2024*)
  - net/sched
  - bpf
- Chose net/sched as my first research as a kernel beginner
- *Unprivileged userns have been disabled since July 1, 2025*

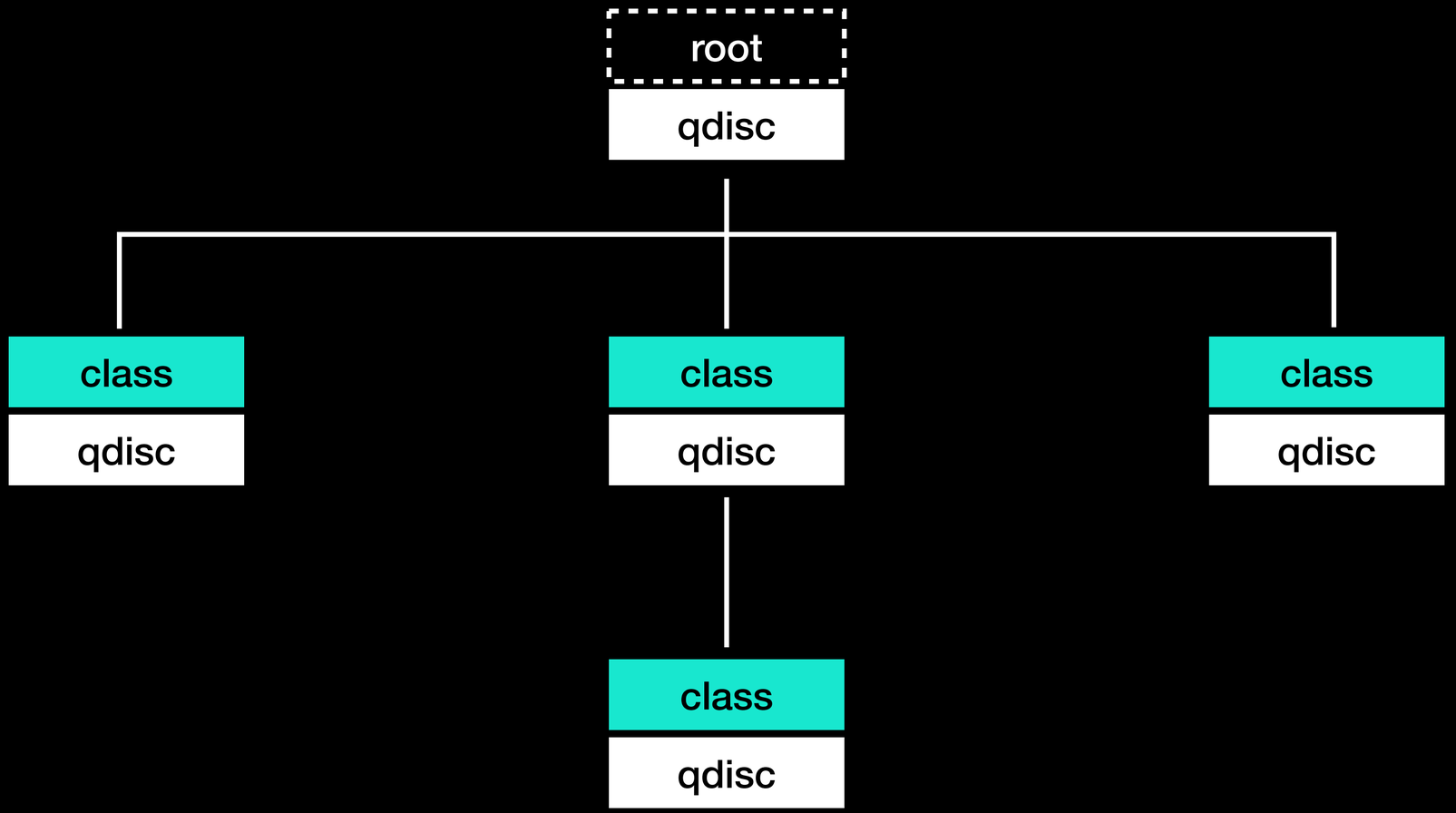
$ net/sched 101

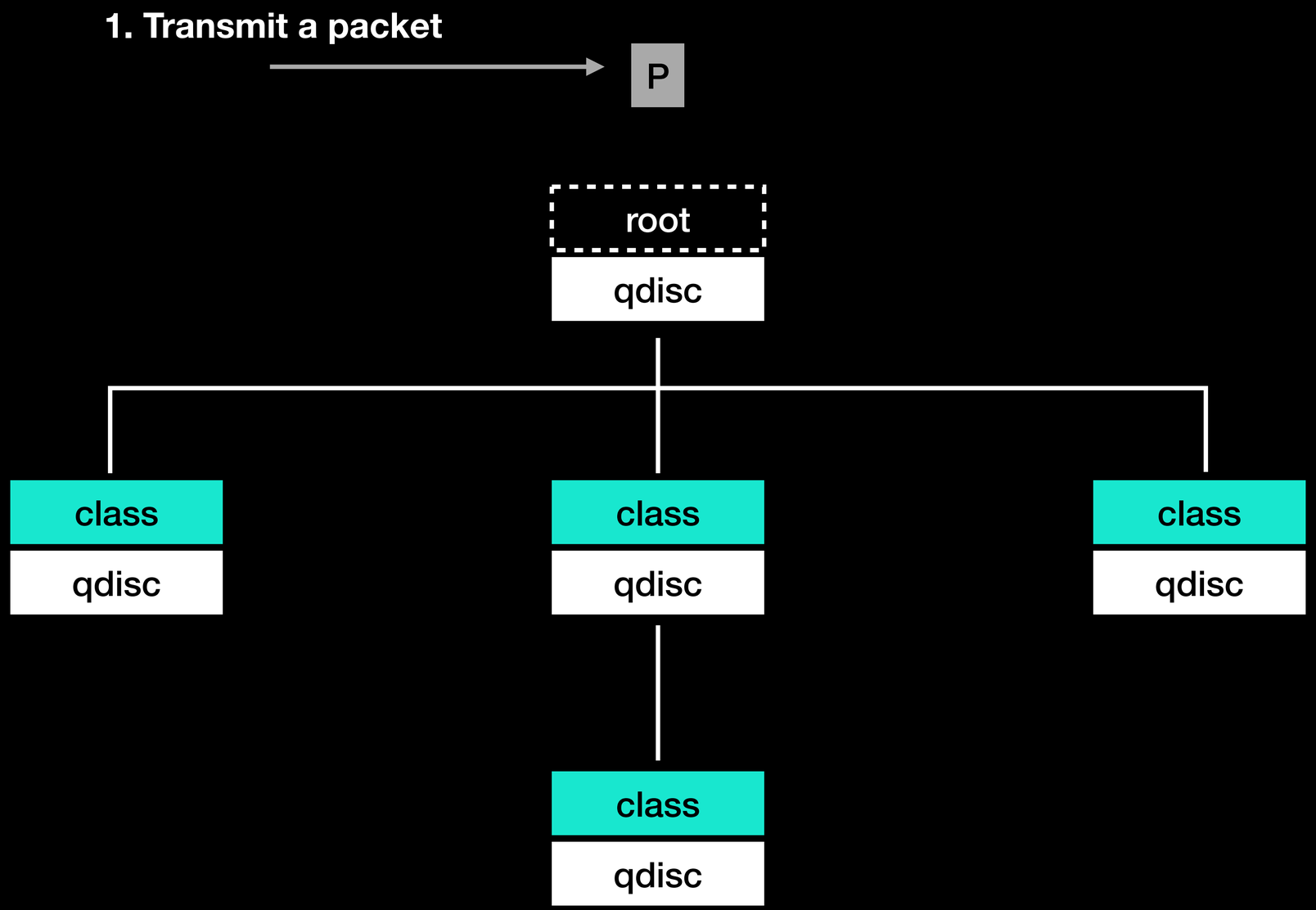
- net/sched: network scheduling subsystem
  - Manages packet flow & packet scheduling
  - Supports multiple schedulers
- Accessed via AF_NETLINK socket or the tc (traffic control) tool

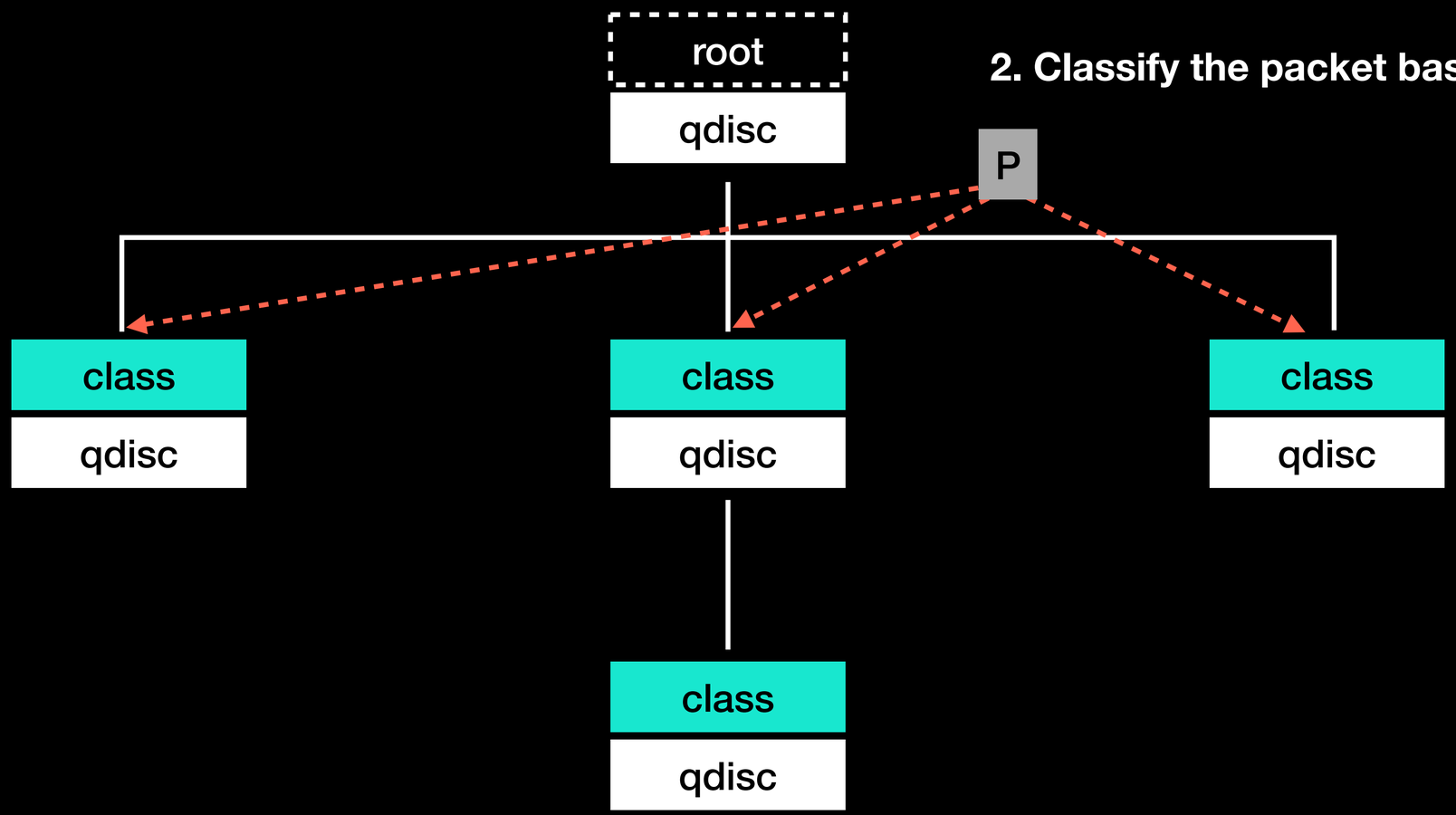
$ net/sched 101

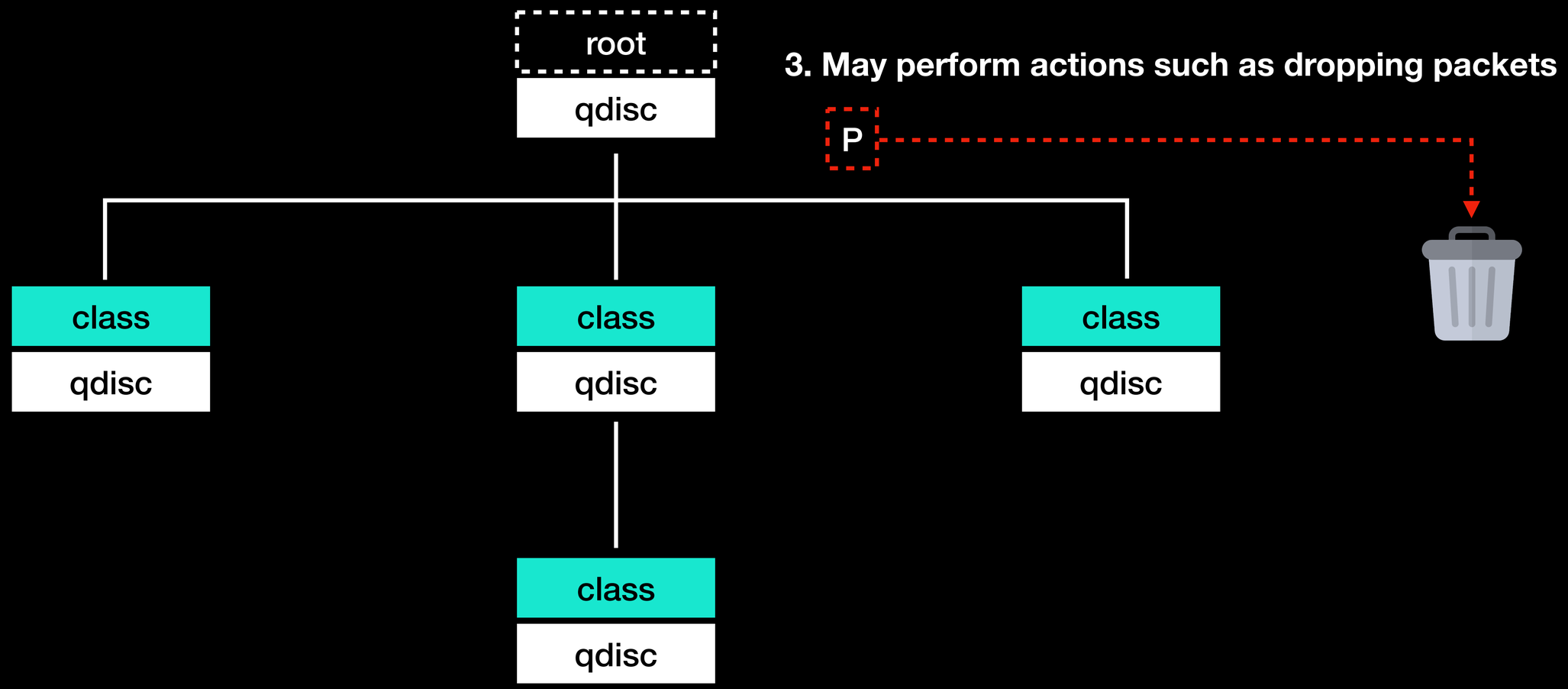
- Qdisc: Queueing discipline that schedules packets
- Class: Subdivision of a qdisc for grouping packets with custom parameters
- Filter: Matches packets by rules (e.g., IP, port)
- Action: Handles matched packets (e.g., drop, redirect)

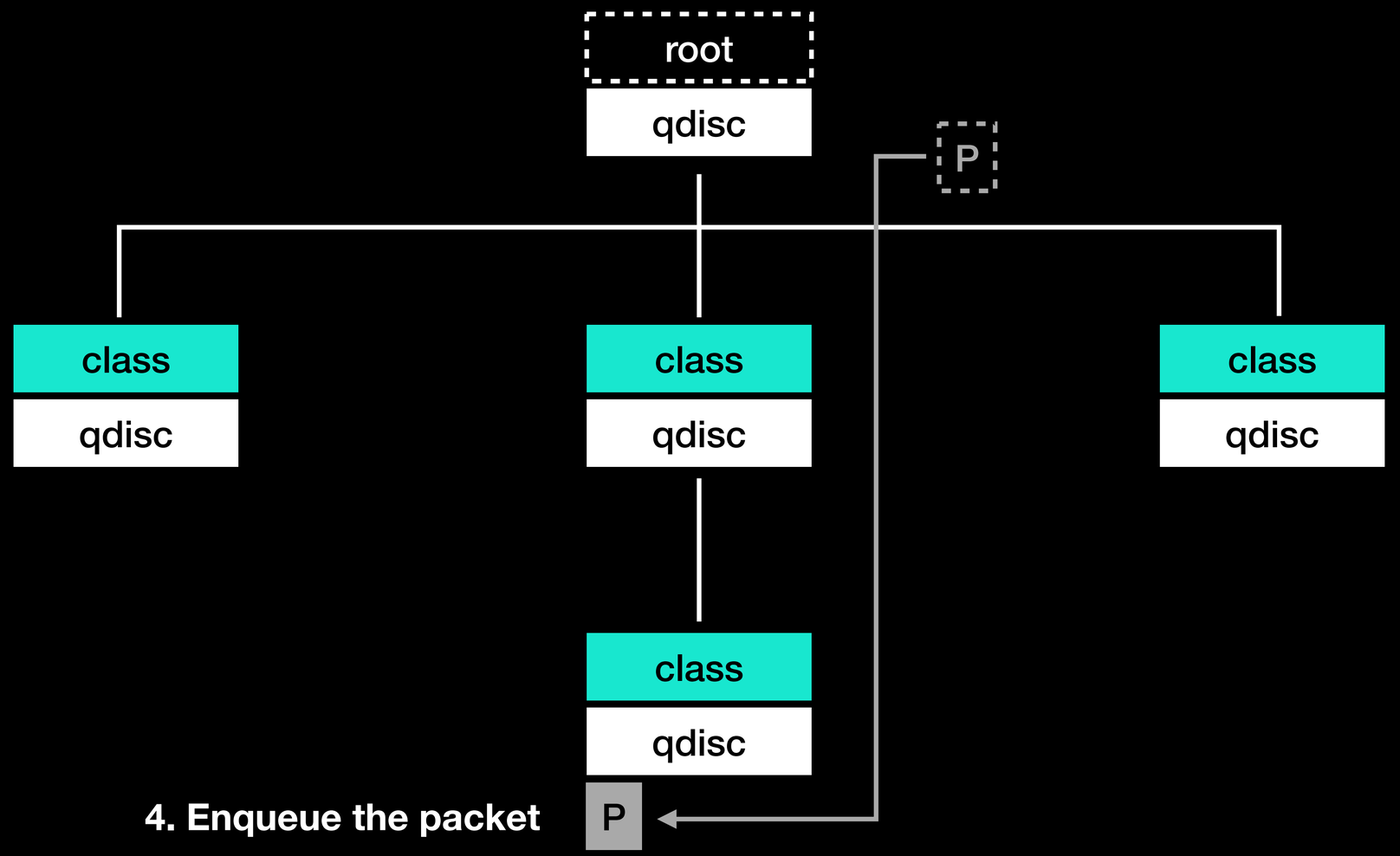
Tree-like Hierarchy




1. Transmit a packet
P
root
qdisc
class
qdisc
class
qdisc
class
qdisc
class
qdisc

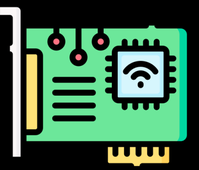

root
qdisc
2. Classify the packet based on filter
P
class
qdisc
class
qdisc
class
qdisc
class
qdisc

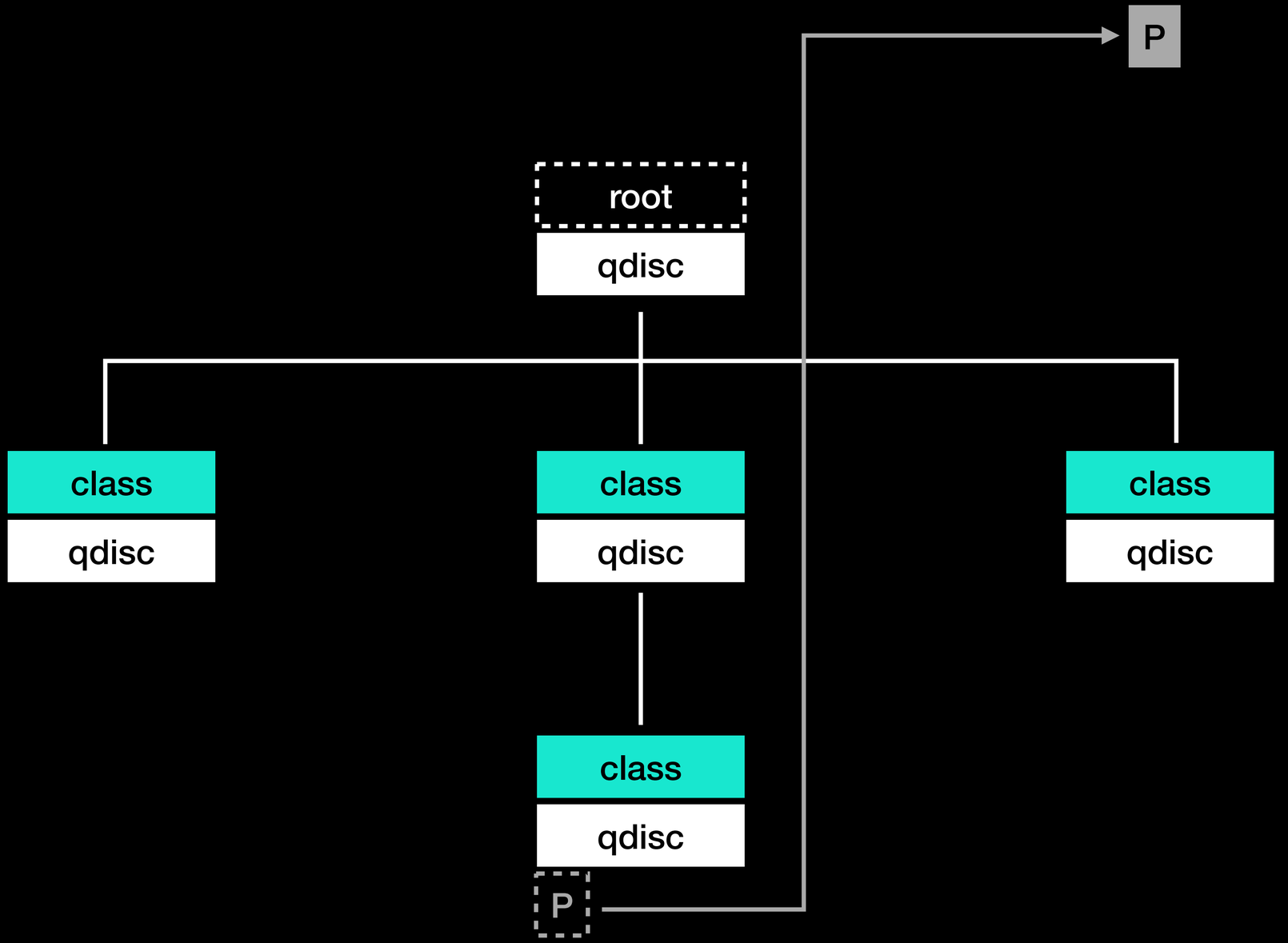

root
qdisc
3. May perform actions such as dropping packets
P
class
qdisc
class
qdisc
class
qdisc
class
qdisc



4. Enqueue the packet

5. Dequeue the packet and send it





$ Bug Types in net/sched

- Earlier bugs: missing checks during parameter changes
  - Some schedulers allow runtime adjustments:
    - Qdisc: e.g., CHOKE via `choke_change()`
    - Class: e.g., DRR via `drr_change_class()`
    - Filter: e.g., FW (Firewall Mark) via `fw_change()`

```
diff --git a/net/sched/sch_multiq.c b/net/sched/sch_multiq.c
index 79e93a19d5fabe..06e03f5cd7ce18 100644
--- a/net/sched/sch_multiq.c
+++ b/net/sched/sch_multiq.c
@@ -185,7 +185,7 @@ static int multiq_tune(struct Qdisc *sch, struct nlattr *opt,

        qopt->bands = qdisc_dev(sch)->real_num_tx_queues;

-       removed = kmalloc(sizeof(*removed) * (q->max_bands - q->bands),
+       removed = kmalloc(sizeof(*removed) * (q->max_bands - qopt->bands),
                          GFP_KERNEL);
        if (!removed)
                return -ENOMEM;
```

Case 1. **CVE-2024-36978**
(net: sched: sch_multiq: fix possible OOB write in multiq_tune())

$ Bug Types in net/sched

- Recent bugs: abusing the scheduling stack to trigger side effects
  - Tree-like qdisc hierarchy → multiple schedulers handle enqueue/dequeue
  - A qdisc has its own state and should be synchronized with its parent/child qdiscs
  - State mismatches may violate the assumptions of net/sched

```
diff --git a/net/sched/sch_cake.c b/net/sched/sch_cake.c
index f2f9b75008bb05..8d8b2db4653c0c 100644
--- a/net/sched/sch_cake.c
+++ b/net/sched/sch_cake.c
@@ -1525,7 +1525,6 @@ static unsigned int cake_drop(struct Qdisc *sch, struct sk_buff **to_free)
        b->backlogs[idx]    -= len;
        b->tin_backlog      -= len;
        sch->qstats.backlog -= len;
-       qdisc_tree_reduce_backlog(sch, 1, len);

        flow->dropped++;
        b->tin_dropped++;
@@ -1536,6 +1535,7 @@ static unsigned int cake_drop(struct Qdisc *sch, struct sk_buff **to_free)

        __qdisc_drop(skb, to_free);
        sch->q.qlen--;
+       qdisc_tree_reduce_backlog(sch, 1, len);

        cake_heapify(q, 0);
```

Case 2. **CVE-2024-53164**
(net: sched: fix ordering of qlen adjustment)

1-day Exploit Attempt:

CVE-2024-56770

$ Motivation

- 1-days are valid in kernelCTF
- Look for public but unpatched issues in upcoming releases:
  1. Linux kernel mailing list
  2. Public kCTF VRP spreadsheet

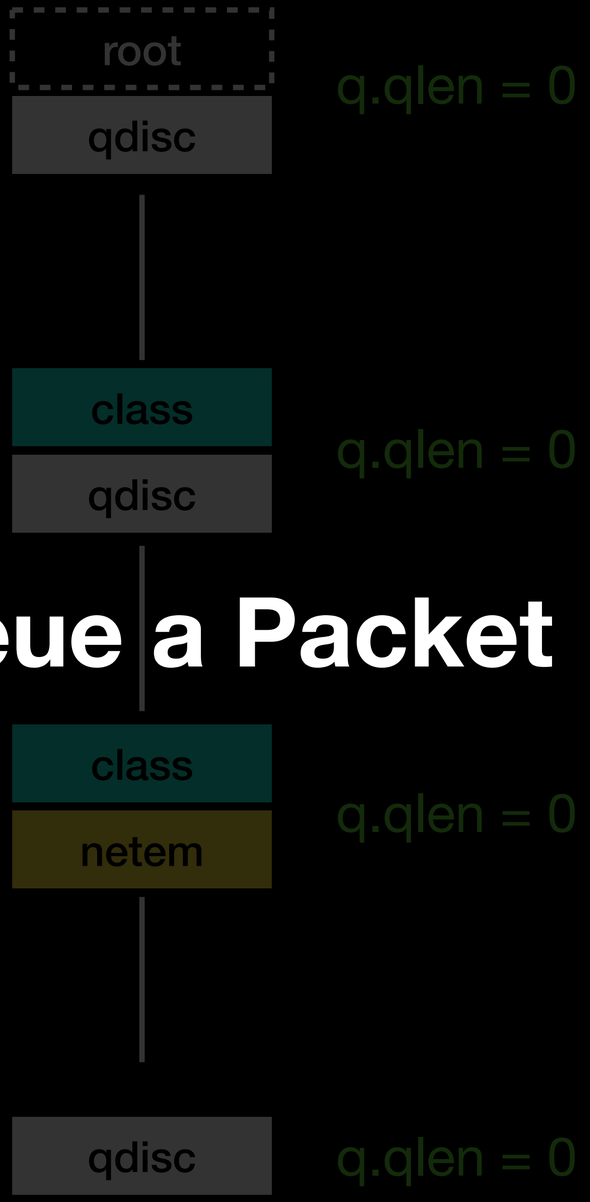
$ Motivation

exp223	2024-12-27T12:01:55.601Z	kernelCTF{v1:lts-6.6.66:1735300904}	1-day	(dupe)		
exp222	2024-12-27T12:01:55.770Z	kernelCTF{v1:lts-6.6.66:1735300822}	0-day	(dupe)		
exp221	2024-12-27T12:01:50.854Z	kernelCTF{v1:lts-6.6.66:1735300879}	0-day	(dupe)		
exp220	2024-12-27T12:00:57.970Z	kernelCTF{v1:lts-6.6.66:1735300833}	1-day	(dupe)		
exp219	2024-12-27T12:00:37.241Z	kernelCTF{v1:lts-6.6.66:1735300805}	0-day	lts-6.6.66		https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=bc50835e83f60f56e9bec2b392fb5544f250fb6f
exp218	2024-12-24T14:33:23.305Z	kernelCTF{v1:mitigation-v3b-6.1.55:1735048115}	1-day			https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/commit/?id=382c27f4ed28f803b1f1473ac2d8db0afc795a1b
exp217	2024-12-22T12:07:27.892Z	kernelCTF{v1:cos-109-17800.372.64:1734867417}	0-day		(dupe)	
exp216	2024-12-21T04:04:55.896Z	kernelCTF{v1:cos-109-17800.372.64:1734753555}	0-day		(dupe)	
exp215	2024-12-19T03:41:54.045Z	kernelCTF{v1:lts-6.6.64:1734577611}	1-day	(dupe)		https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=f8d4bc455047cf3903cd6f85f49978987dbb3027
exp214	2024-12-18T23:04:07.290Z	kernelCTF{v1:mitigation-v4-6.6:1734562384} kernelCTF{v1:cos-109-17800.372.64:1734561991}	1-day		cos-109-17800.372.64	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=f8d4bc455047cf3903cd6f85f49978987dbb3027
exp213	2024-12-13T12:01:47.216Z	kernelCTF{v1:lts-6.6.64:1734091232}	0-day	lts-6.6.64		https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=647cef20e649c576dff271e018d5d15d998b629d

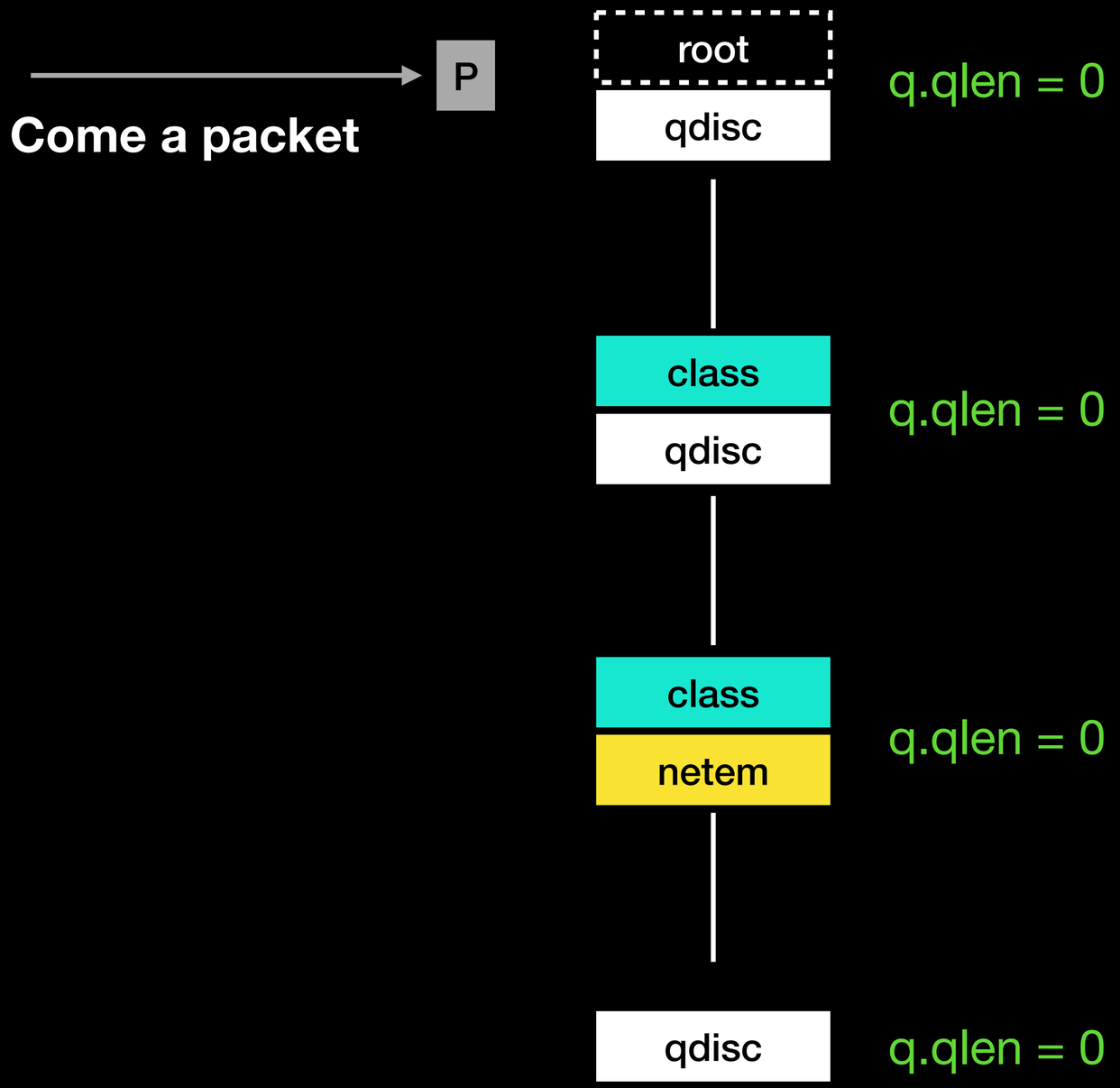
There's a duplicated submission in the lts-6.6.64 with the public patch commit, but the vuln remains unpatched in the next release verision (lts-6.6.66)!

$ CVE-2024-56770 RCA

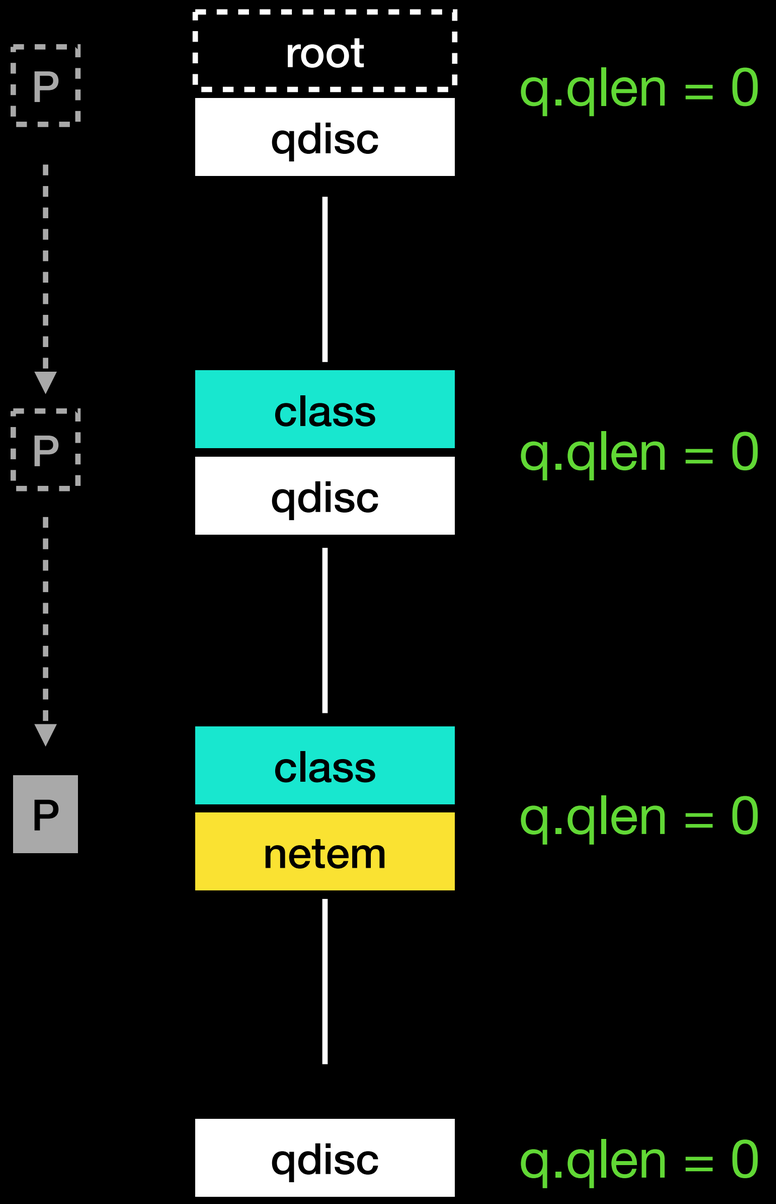
- net/sched: netem: account for backlog updates from child qdisc
  - netem (network emulation) scheduler incorrectly updates packet counts on dequeue
    - Causes packet length mismatch between parent and child qdiscs
    - Matches recent bug patterns
- Duplicated submission confirms the vulnerability is exploitable

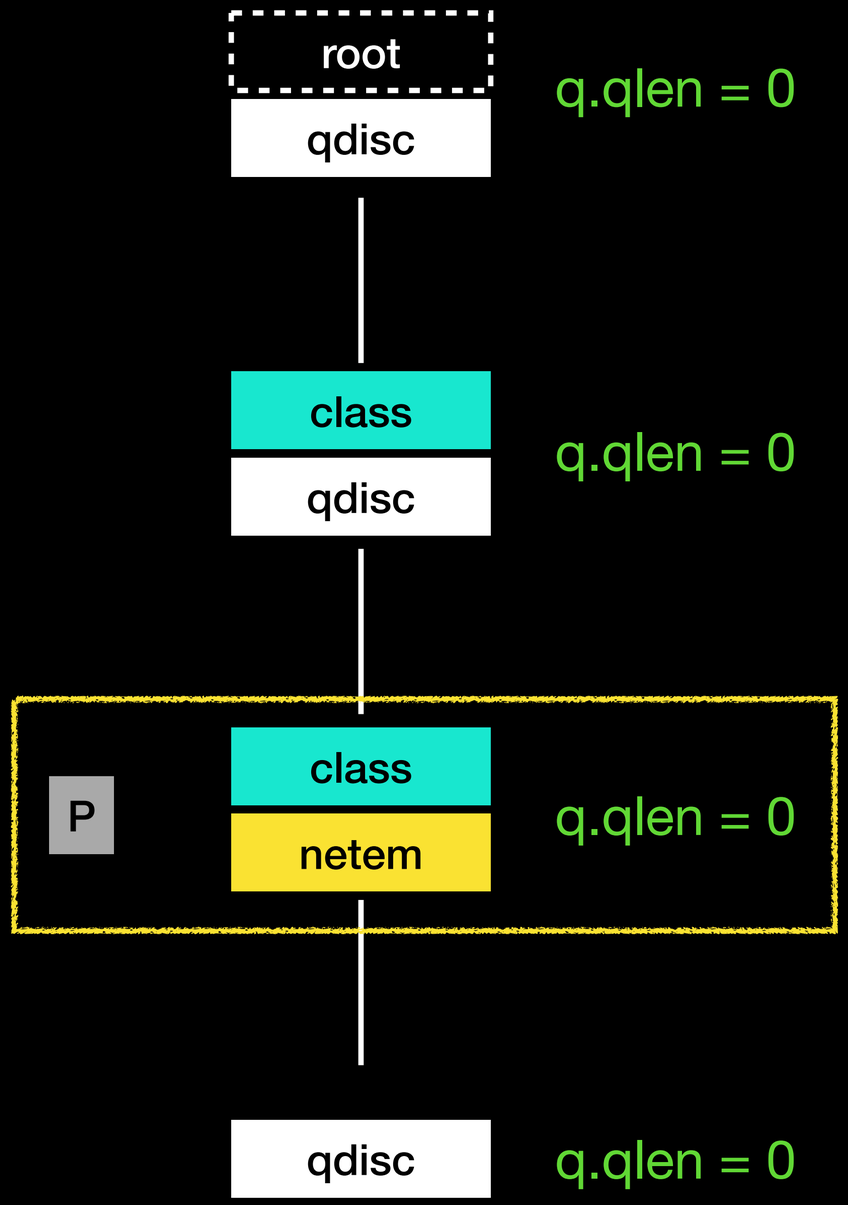


Enqueue a Packet


Come a packet
P
root
qdisc
q.qlen = 0
class
qdisc
q.qlen = 0
class
netem
q.qlen = 0
qdisc
q.qlen = 0

Dispatch to child qdisc

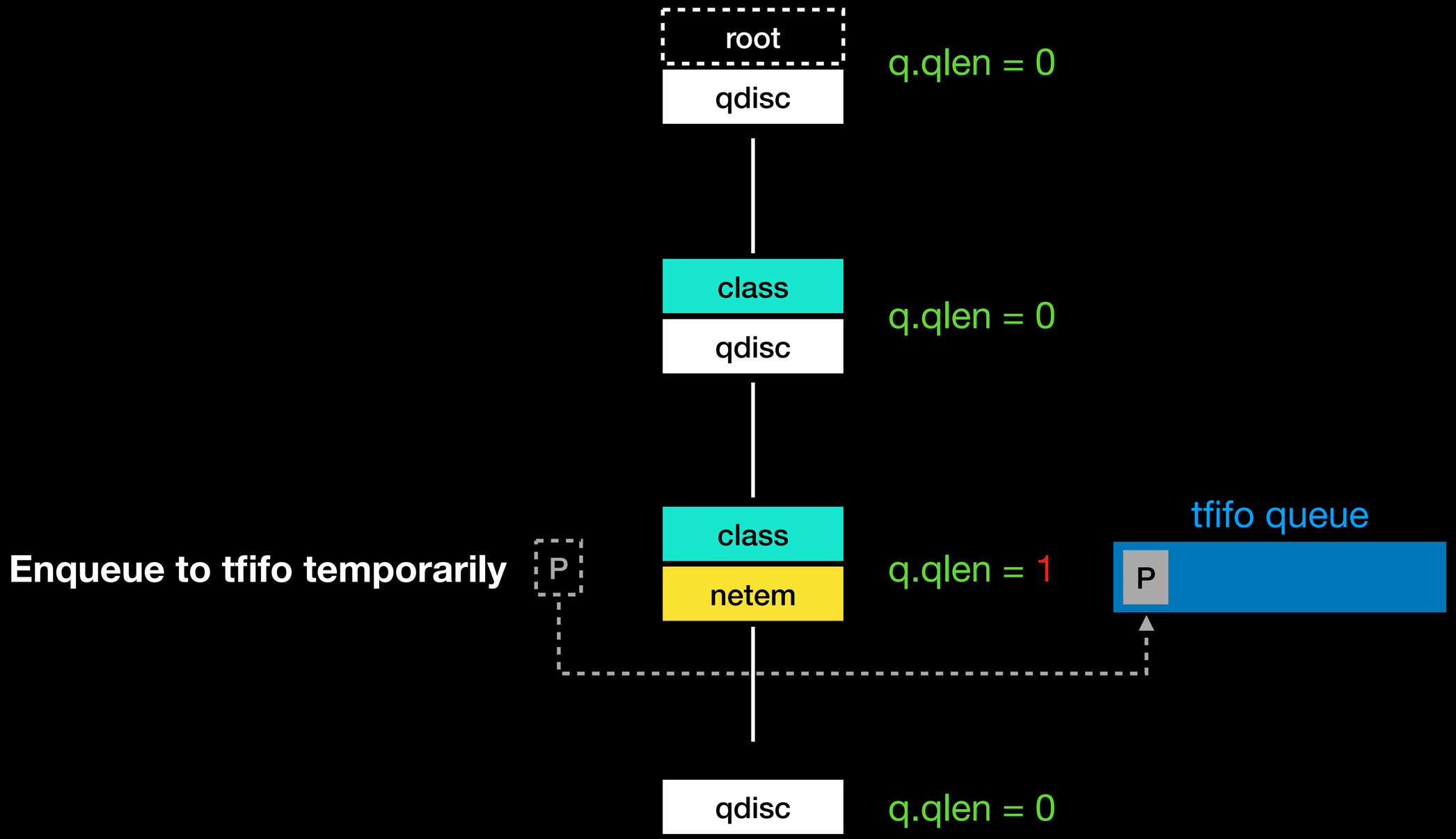


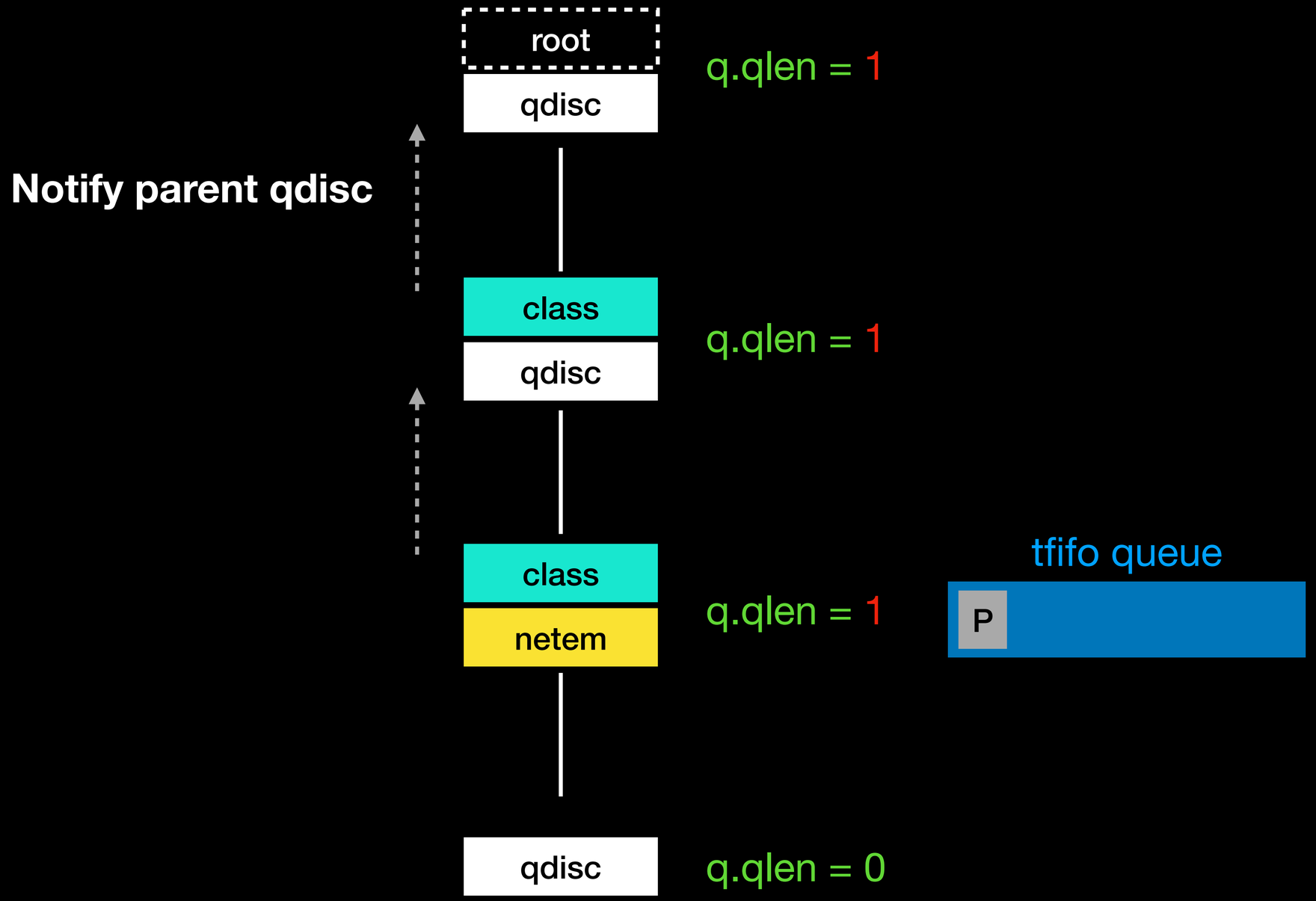

root
qdisc
q.qlen = 0
class
qdisc
q.qlen = 0
P
class
netem
q.qlen = 0
qdisc
q.qlen = 0

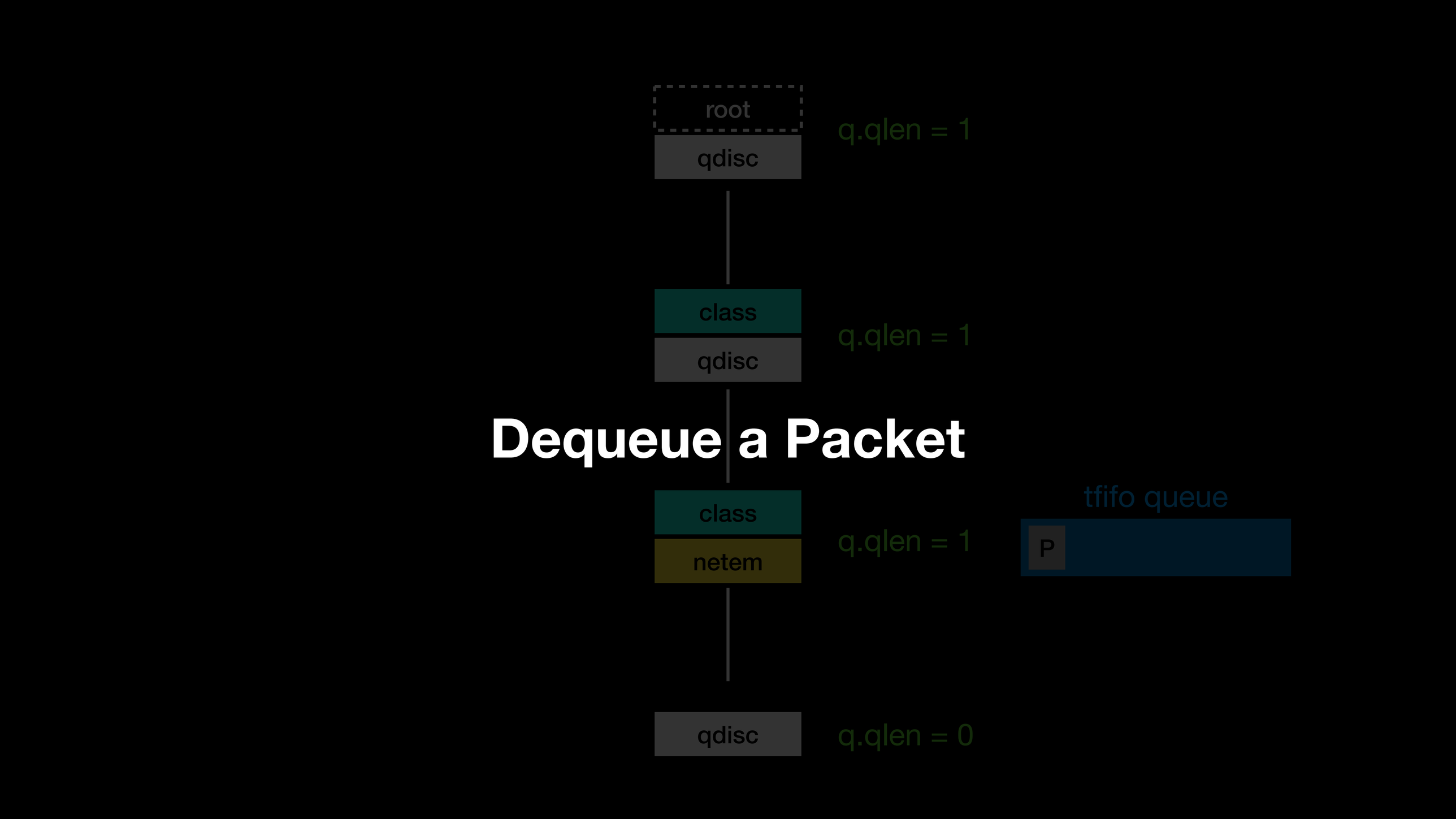
```
if (q->gap == 0 ||          /* not doing reordering */
    q->counter < q->gap - 1 ||  /* inside last reordering gap */
    q->reorder < get_crandom(&q->reorder_cor, &q->prng)) {
    u64 now;
    s64 delay;

    delay = tabledist(q->latency, q->jitter,
                      &q->delay_cor, &q->prng, q->delay_dist);

    now = ktime_get_ns();

    if (q->rate) {…
    }

    cb->time_to_send = now + delay;
    ++q->counter;
    tfifo_enqueue(skb, sch);
```

netem_enqueue()

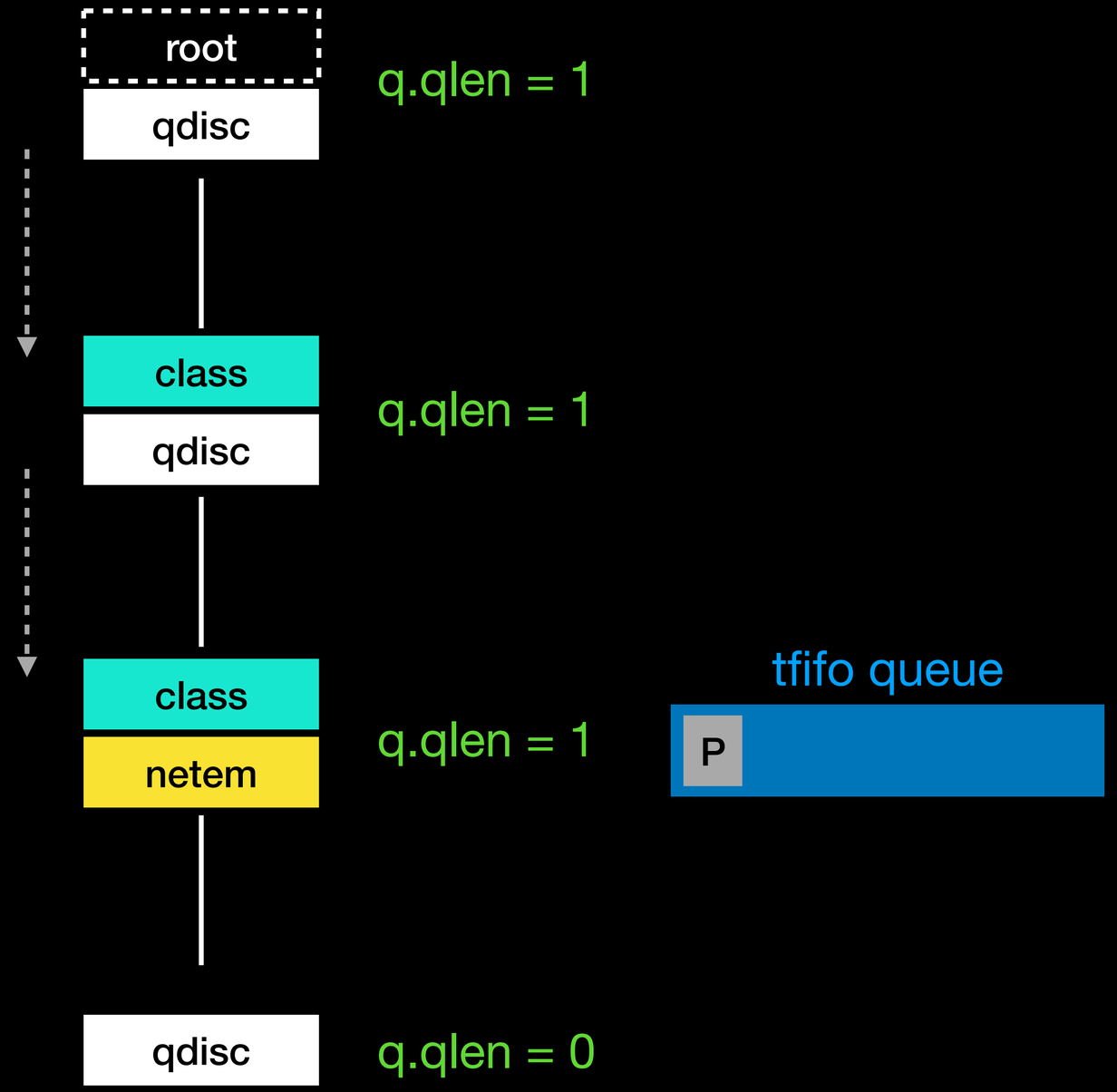
If netem is set to delay transmission,
the packet is first enqueued to the tfifo queue

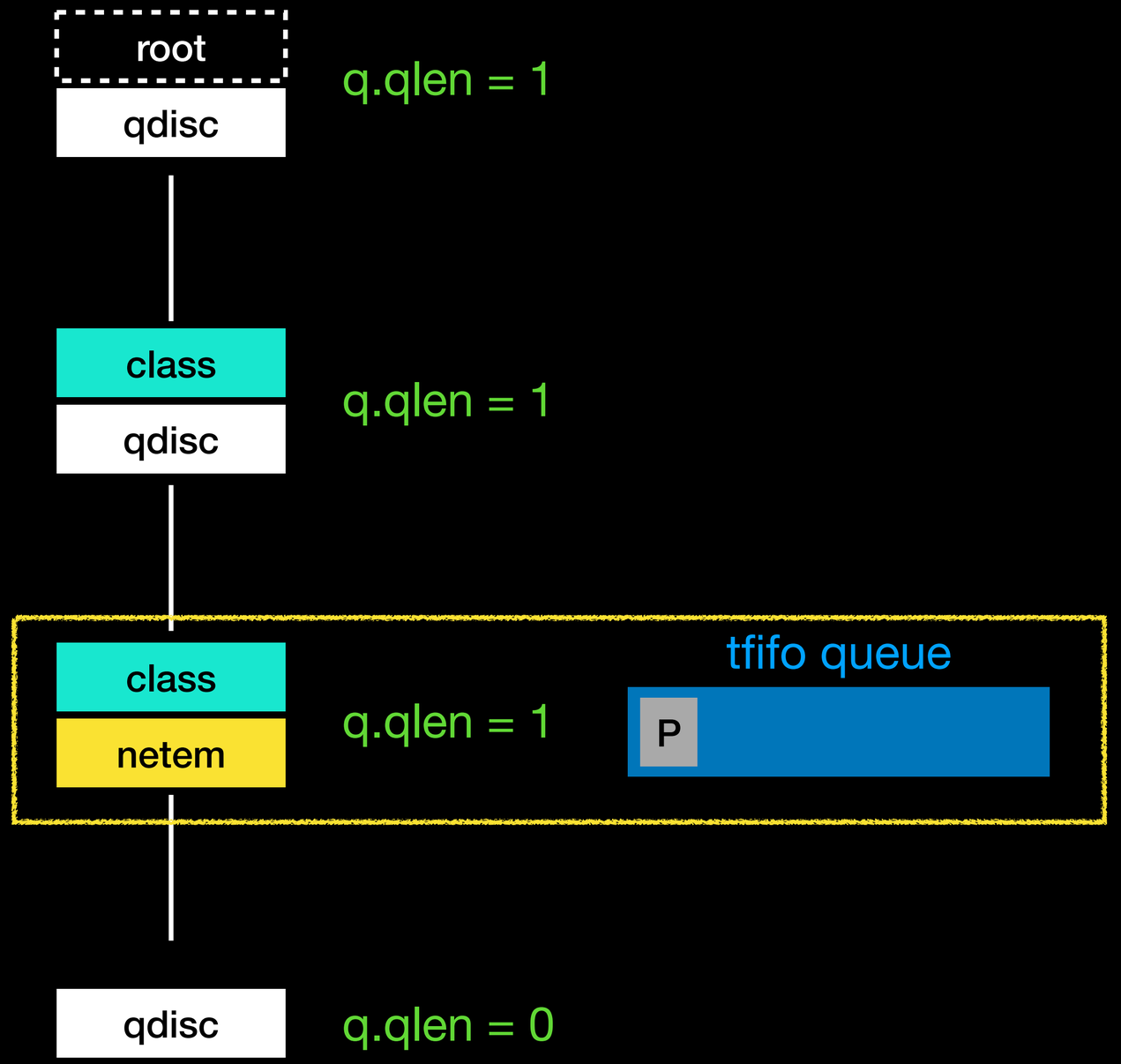

root
qdisc
q.qlen = 0
class
qdisc
q.qlen = 0
class
netem
q.qlen = 1
tfifo queue
P
P
Enqueue to tfifo temporarily
qdisc
q.qlen = 0


root
qdisc
q.qlen = 1
Notify parent qdisc
class
qdisc
q.qlen = 1
class
netem
q.qlen = 1
tfifo queue
P
qdisc
q.qlen = 0


root
qdisc
q.qlen = 1
class
qdisc
q.qlen = 1
Dequeue a Packet
class
netem
q.qlen = 1
tfifo queue
P
qdisc
q.qlen = 0

Try to retrieve a packet from child




root
qdisc
q.qlen = 1
class
qdisc
q.qlen = 1
class
netem
q.qlen = 1
tfifo queue
P
qdisc
q.qlen = 0

```
skb = netem_peek(q);
if (skb) {
        u64 time_to_send;
        u64 now = ktime_get_ns();

        /* if more time remaining? */
        time_to_send = netem_skb_cb(skb)->time_to_send;
        if (q->slot.slot_next && q->slot.slot_next < time_to_send)
                get_slot_next(q, now);

        if (time_to_send <= now && q->slot.slot_next <= now) {
                sch->q.qlen--;
                qdisc_qstats_backlog_dec(sch, skb);
                skb->next = NULL;
                skb->prev = NULL;
                /* skb->dev shares skb->rbnode area,
                 * we need to restore its value.
                 */
                skb->dev = qdisc_dev(sch);

                if (q->slot.slot_next) {
                        q->slot.packets_left--;
                        q->slot.bytes_left -= qdisc_pkt_len(skb);
                        if (q->slot.packets_left <= 0 ||
                            q->slot.bytes_left <= 0)
                                get_slot_next(q, now);
                }

                if (q->qdisc) {
                        unsigned int pkt_len = qdisc_pkt_len(skb);
                        struct sk_buff *to_free = NULL;
                        int err;

                        err = qdisc_enqueue(skb, q->qdisc, &to_free);
                        kfree_skb_list(to_free);
                        if (err != NET_XMIT_SUCCESS) {
                                if (net_xmit_drop_count(err))
                                        qdisc_qstats_drop(sch);
                                qdisc_tree_reduce_backlog(sch, 1, pkt_len);
                        }
                        goto tfifo_dequeue;
                }
                goto deliver;
        }
```

Check whether the delayed packet is ready to be sent

netem_dequeue()

```
skb = netem_peek(q);
if (skb) {
        u64 time_to_send;
        u64 now = ktime_get_ns();

        /* if more time remaining? */
        time_to_send = netem_skb_cb(skb)->time_to_send;
        if (q->slot.slot_next && q->slot.slot_next < time_to_send)
                get_slot_next(q, now);

        if (time_to_send <= now && q->slot.slot_next <= now) {
                netem_erase_head(q, skb);
                sch->q.qlen--;
                qdisc_qstats_backlog_dec(sch, skb);
                skb->next = NULL;
                skb->prev = NULL;
                /* skb->dev shares skb->rbnode area,
                 * we need to restore its value.
                 */
                skb->dev = qdisc_dev(sch);

                if (q->slot.slot_next) {
                        q->slot.packets_left--;
                        q->slot.bytes_left -= qdisc_pkt_len(skb);
                        if (q->slot.packets_left <= 0 ||
                            q->slot.bytes_left <= 0)
                                get_slot_next(q, now);
                }

                if (q->qdisc) {
                        unsigned int pkt_len = qdisc_pkt_len(skb);
                        struct sk_buff *to_free = NULL;
                        int err;

                        err = qdisc_enqueue(skb, q->qdisc, &to_free);
                        kfree_skb_list(to_free);
                        if (err != NET_XMIT_SUCCESS) {
                                if (net_xmit_drop_count(err))
                                        qdisc_qstats_drop(sch);
                                qdisc_tree_reduce_backlog(sch, 1, pkt_len);
                        }
                        goto tfifo_dequeue;
                }
                goto deliver;
        }
```

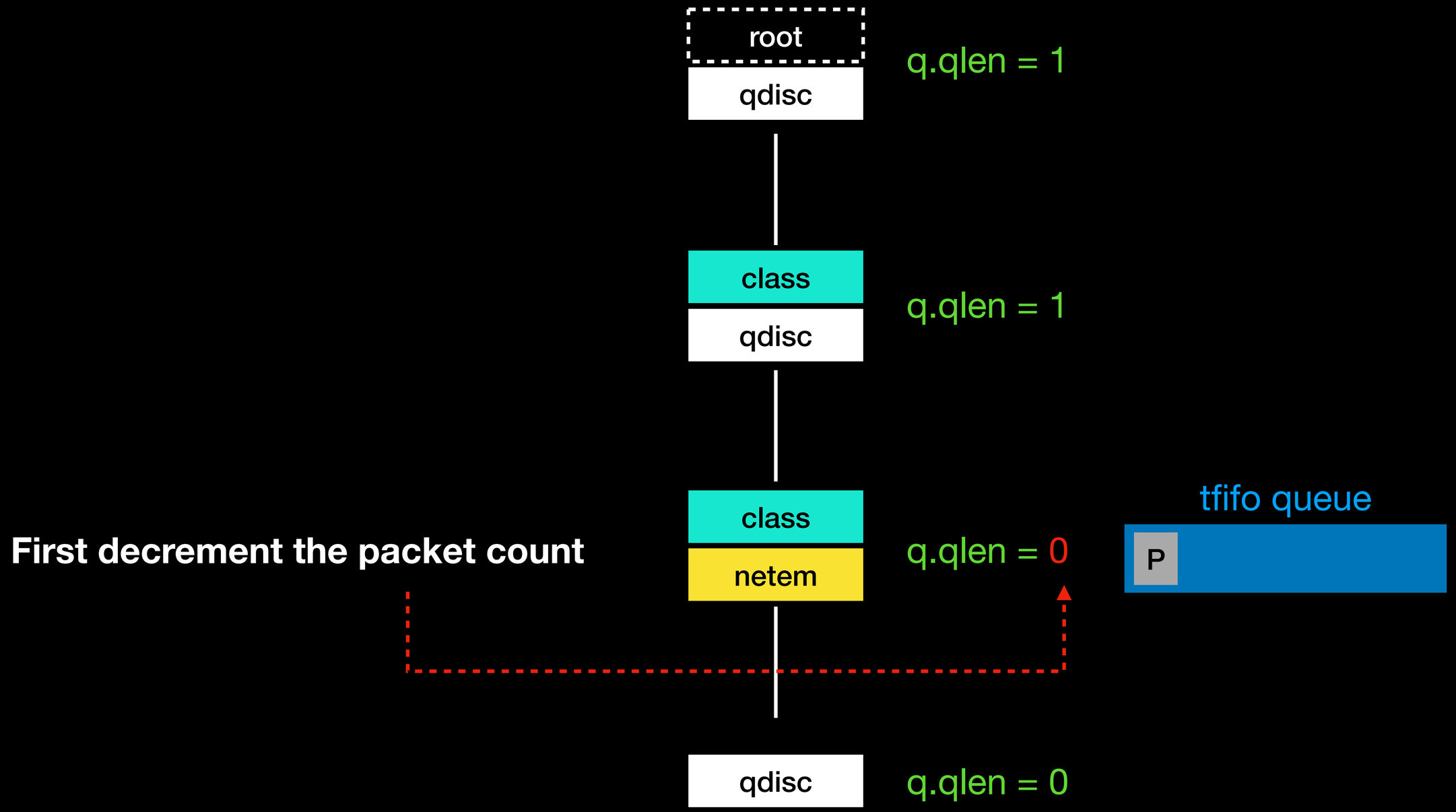
The packet count (q.qlen) is decremented

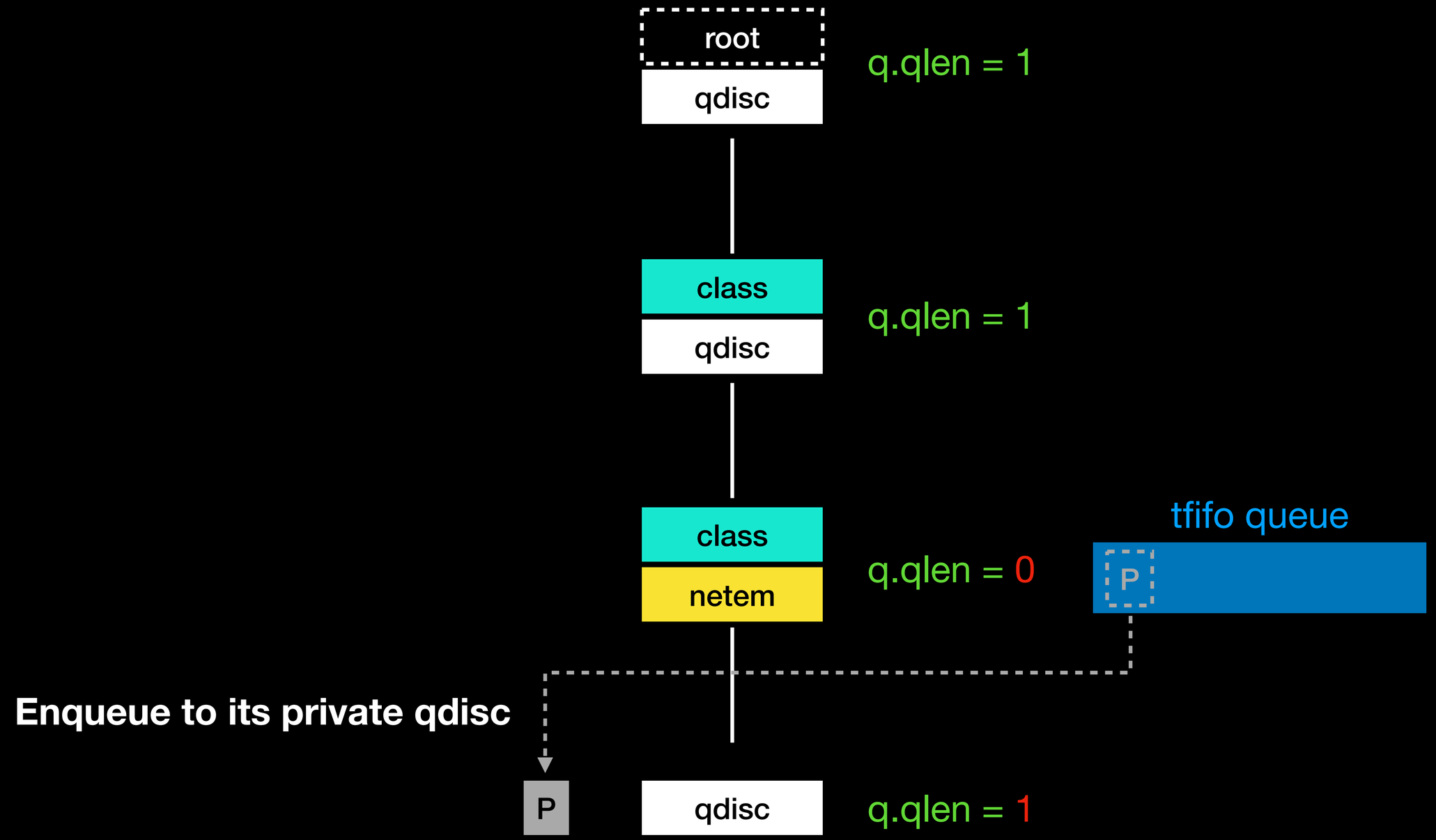
netem_dequeue()

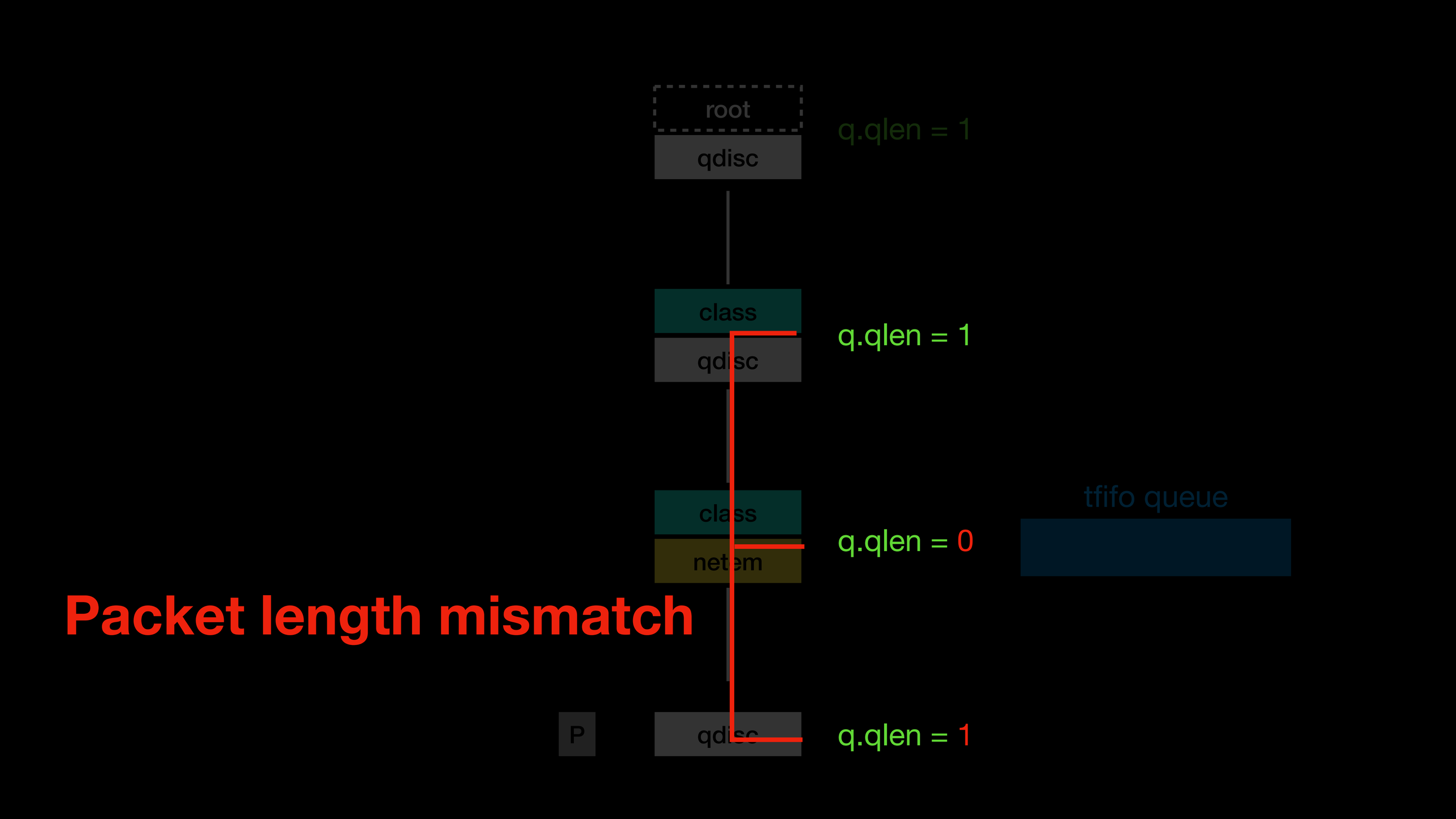
```
skb = netem_peek(q);
if (skb) {
        u64 time_to_send;
        u64 now = ktime_get_ns();

        /* if more time remaining? */
        time_to_send = netem_skb_cb(skb)->time_to_send;
        if (q->slot.slot_next && q->slot.slot_next < time_to_send)
                get_slot_next(q, now);

        if (time_to_send <= now && q->slot.slot_next <= now) {
                netem_erase_head(q, skb);
                sch->q.qlen--;
                qdisc_qstats_backlog_dec(sch, skb);
                skb->next = NULL;
                skb->prev = NULL;
                /* skb->dev shares skb->rbnode area,
                 * we need to restore its value.
                 */
                skb->dev = qdisc_dev(sch);

                if (q->slot.slot_next) {
                        q->slot.packets_left--;
                        q->slot.bytes_left -= qdisc_pkt_len(skb);
                        if (q->slot.packets_left <= 0 ||
                            q->slot.bytes_left <= 0)
                                get_slot_next(q, now);
                }

                if (q->qdisc) {
                        unsigned int pkt_len = qdisc_pkt_len(skb);
                        struct sk_buff *to_free = NULL;
                        int err;

                        err = qdisc_enqueue(skb, q->qdisc, &to_free);
                        kfree_skb_list(to_free);
                        if (err != NET_XMIT_SUCCESS) {
                                if (net_xmit_drop_count(err))
                                        qdisc_qstats_drop(sch);
                                qdisc_tree_reduce_backlog(sch, 1, pkt_len);
                        }
                        goto tfifo_dequeue;
                }
                goto deliver;
        }
```

netem_dequeue()

Enqueue the packet to its private qdisc (q->qdisc), if present


root
qdisc
q.qlen = 1
class
qdisc
q.qlen = 1
class
netem
q.qlen = 0
tfifo queue
P
First decrement the packet count
qdisc
q.qlen = 0

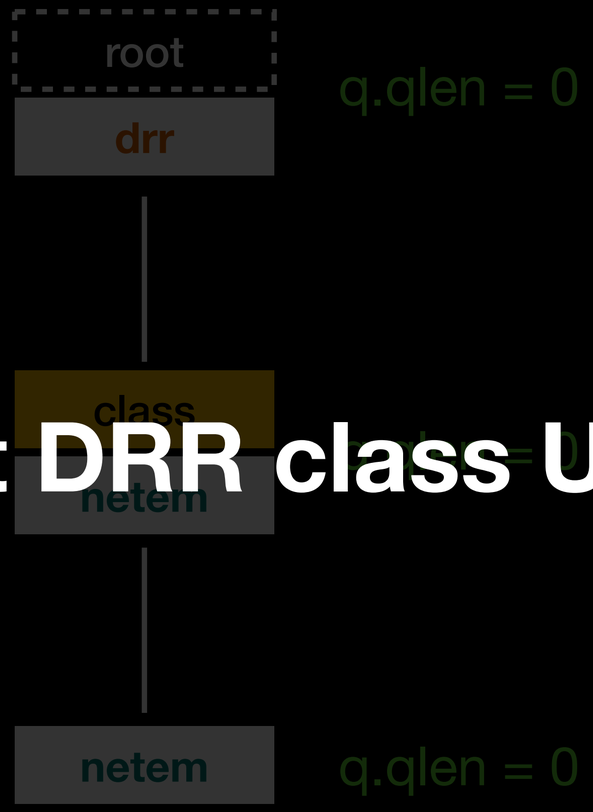

root
qdisc
q.qlen = 1
class
qdisc
q.qlen = 1
class
netem
q.qlen = 0
tfifo queue
P
Enqueue to its private qdisc
P
qdisc
q.qlen = 1



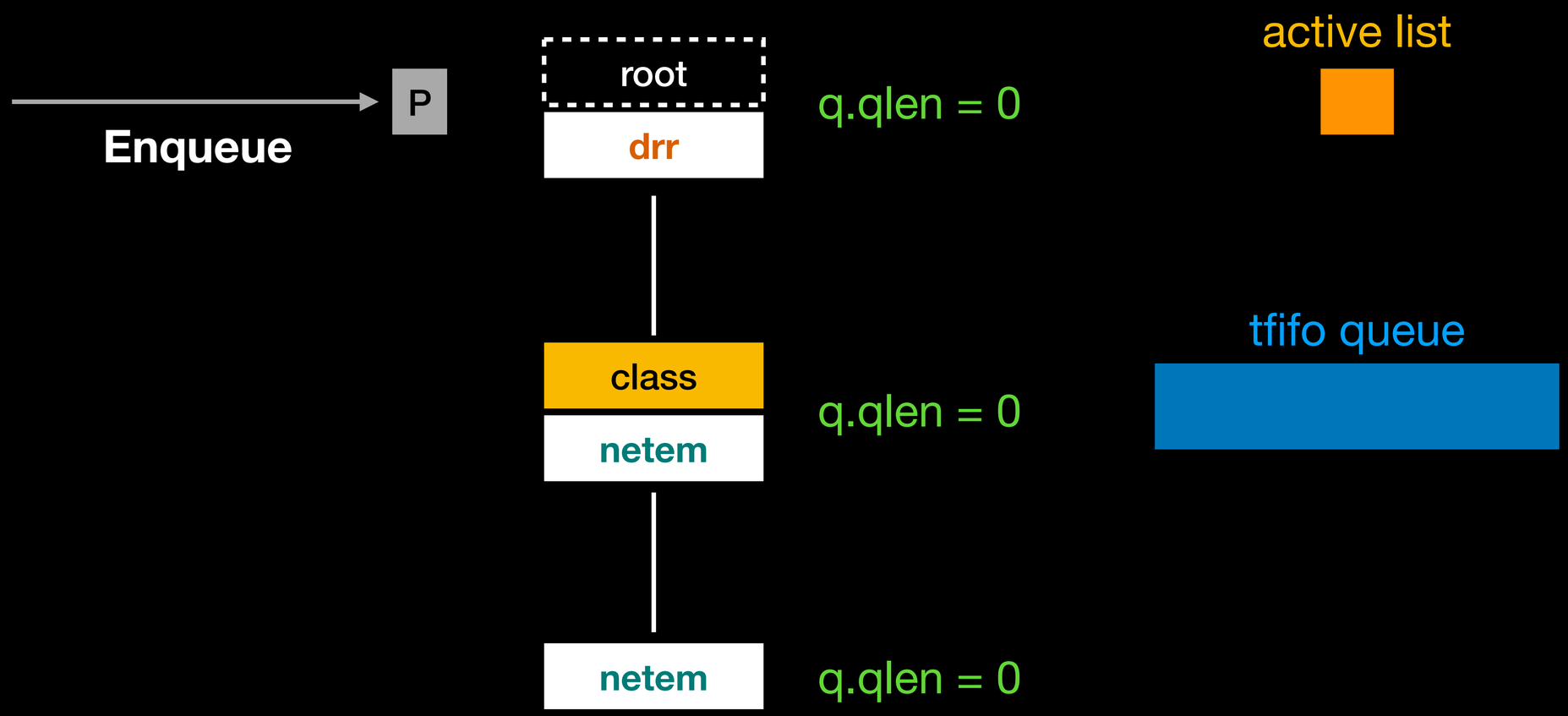
Packet length mismatch

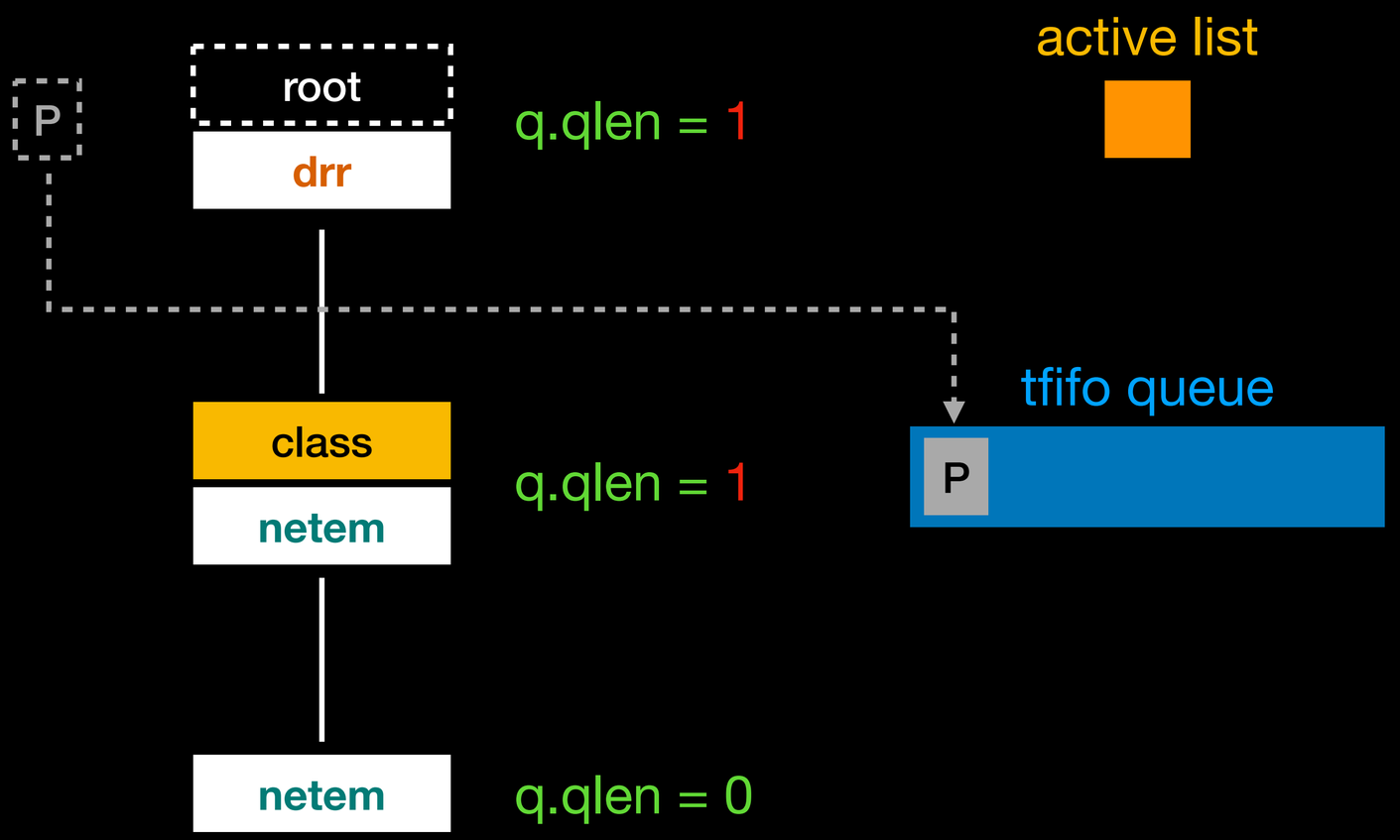
$ The Problem

- Why is packet length mismatch a problem?
  - Some schedulers rely on packet count for state tracking
  - Taking the DRR as an example,
    - If the class destructor detects that $\mathrm{q.qlen}$ is 0, it will skip that portion of the destruction process



Construct DRR class UAF

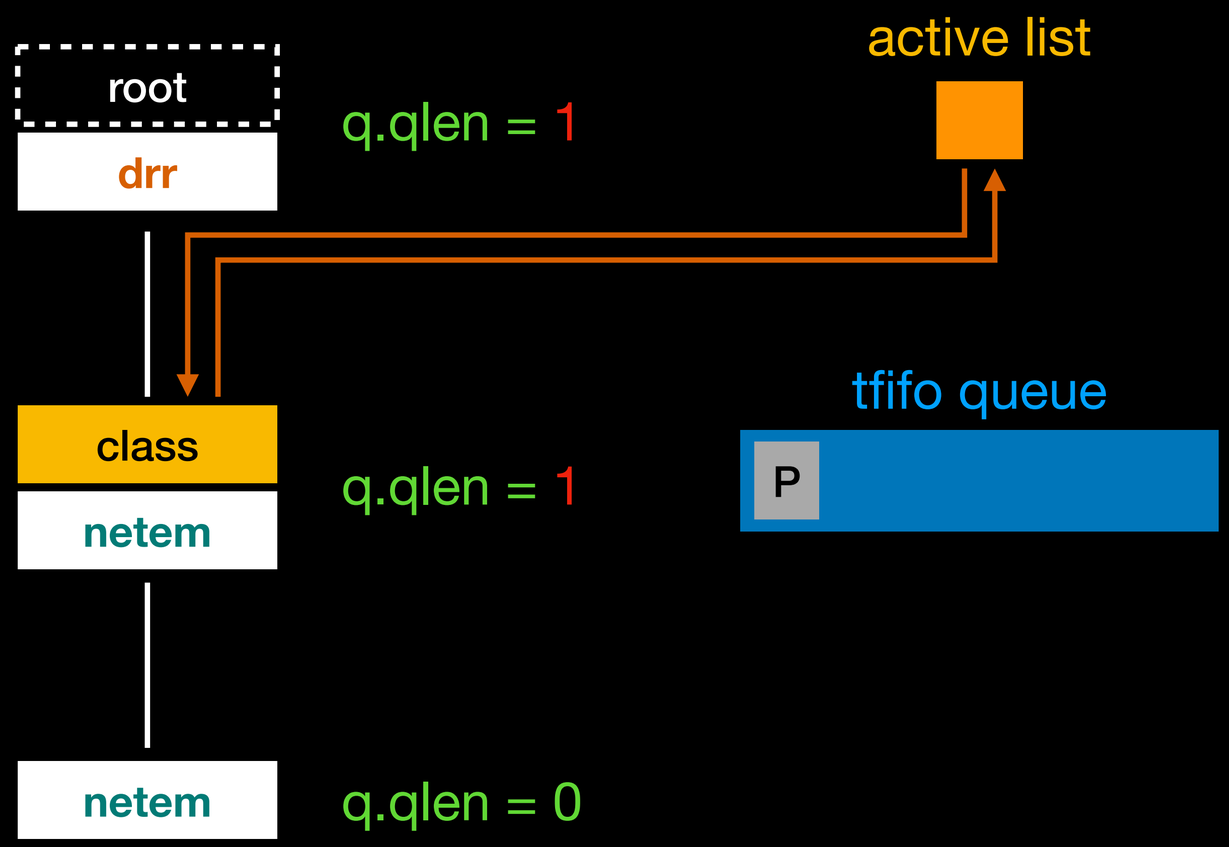

Enqueue
P
root
drr
q.qlen = 0
active list
class
netem
q.qlen = 0
tfifo queue
netem
q.qlen = 0

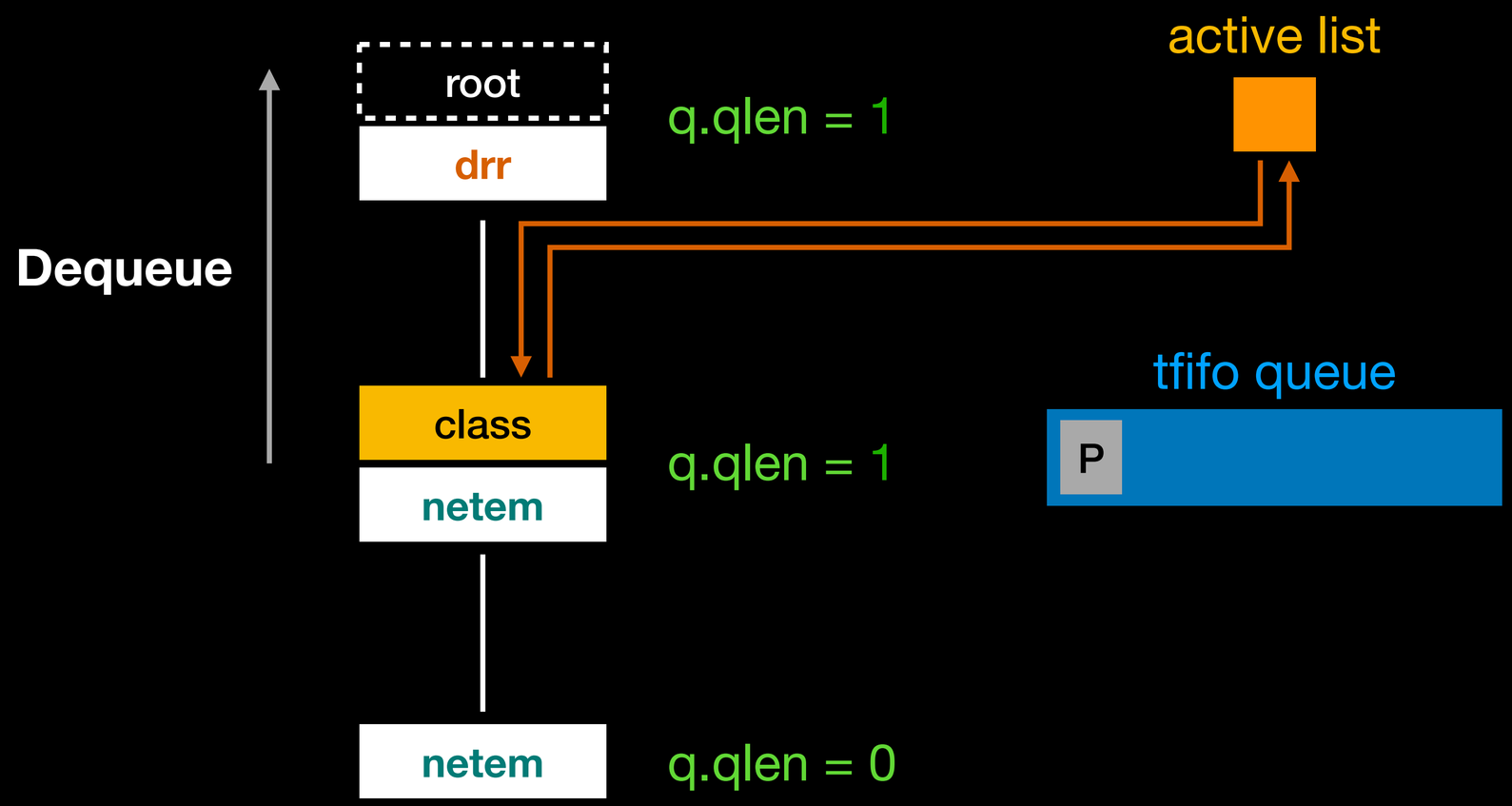

active list
root
P
drr
q.qlen = 1
class
netem
q.qlen = 1
tfifo queue
P
netem
q.qlen = 0

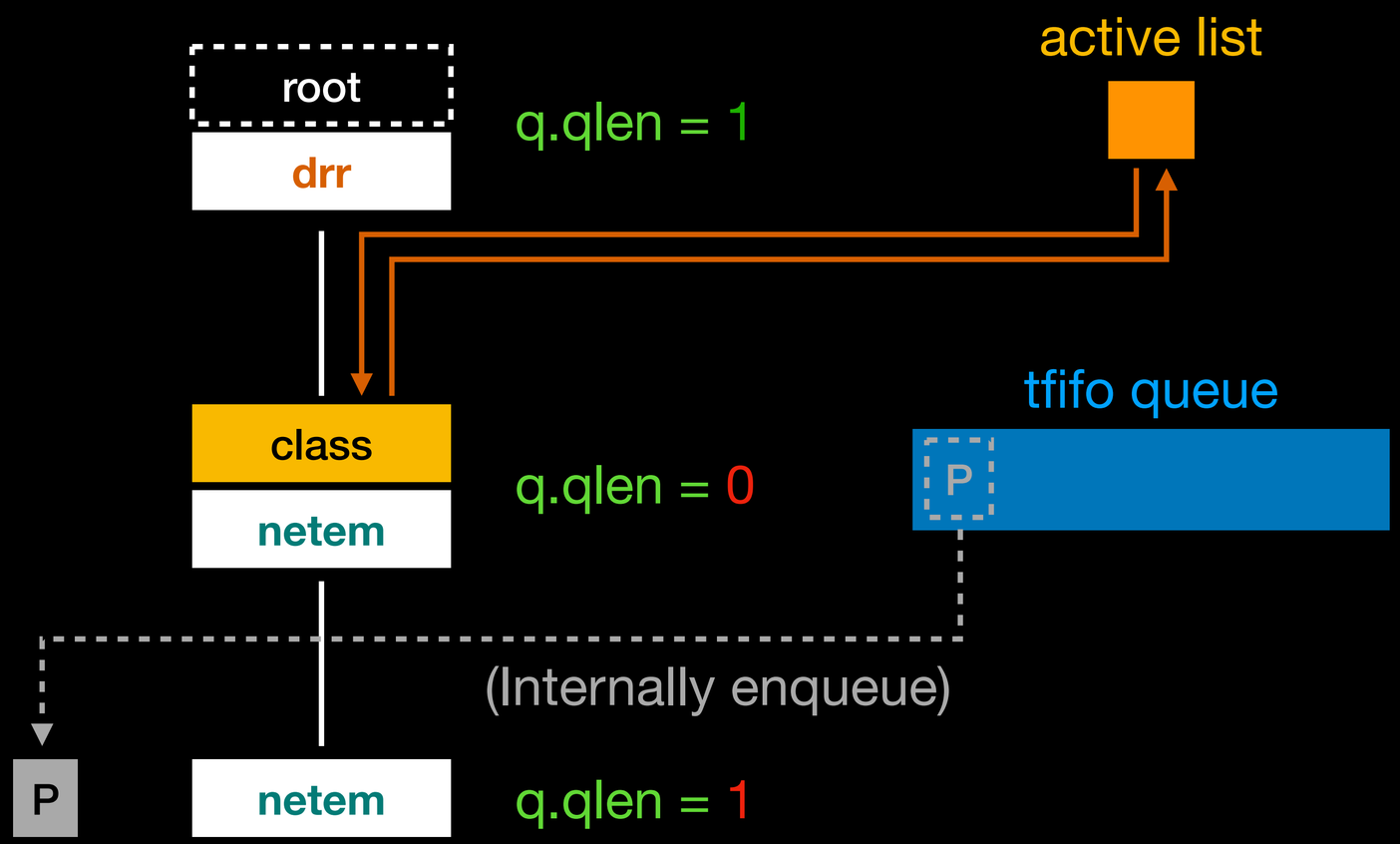
```c
static int drr_enqueue(struct sk_buff *skb, struct Qdisc *sch,
                       struct sk_buff **to_free)
{
    unsigned int len = qdisc_pkt_len(skb);
    struct drr_sched *q = qdisc_priv(sch);
    struct drr_class *cl;
    int err = 0;
    bool first;

    first = !cl->qdisc->q.qlen;
    // [...]

    if (first) {
        list_add_tail(&cl->alist, &q->active);
        cl->deficit = cl->quantum;
    }

    sch->qstats.backlog += len;
    sch->q.qlen++;
    return err;
}
```

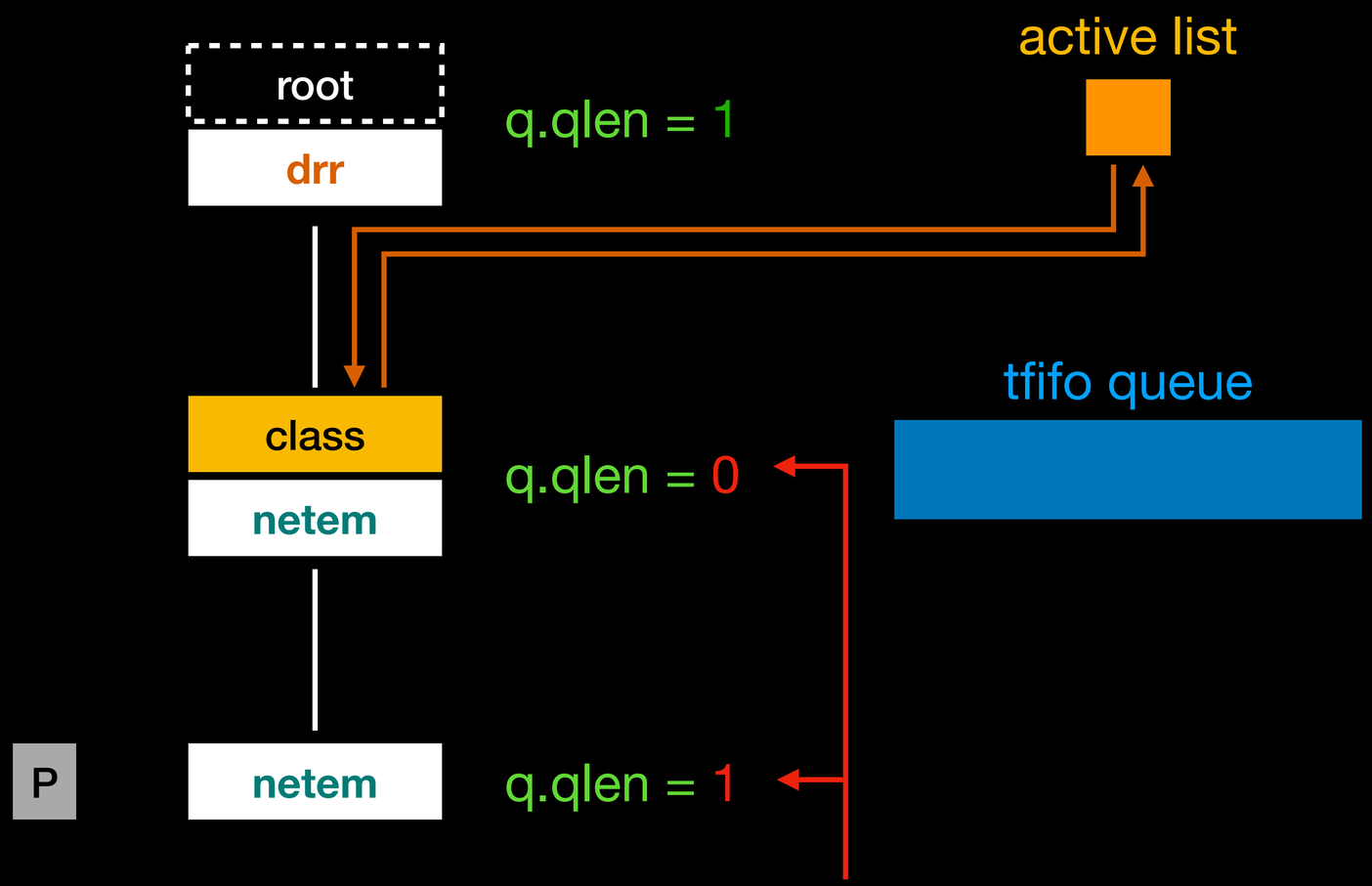
drr_enqueue()

The class is added to the active list when the first packet is enqueued

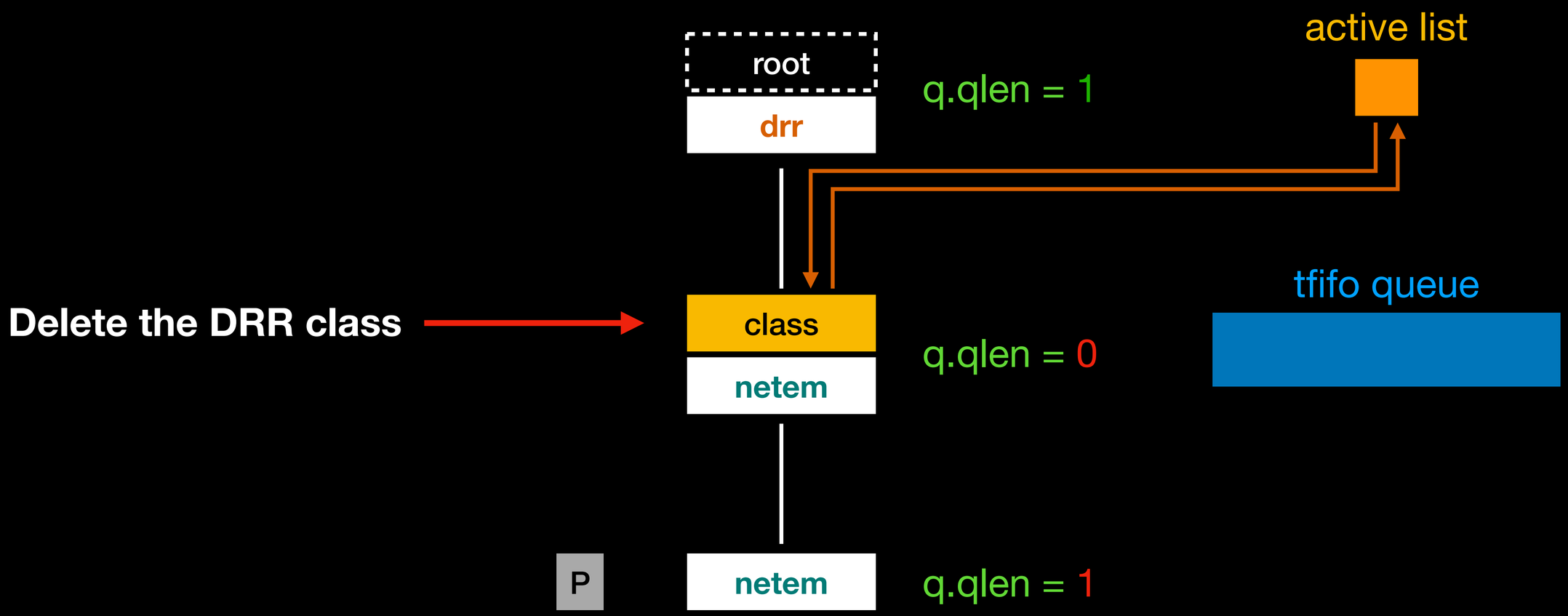

root
drr
q.qlen = 1
active list
class
netem
q.qlen = 1
tfifo queue
P
netem
q.qlen = 0

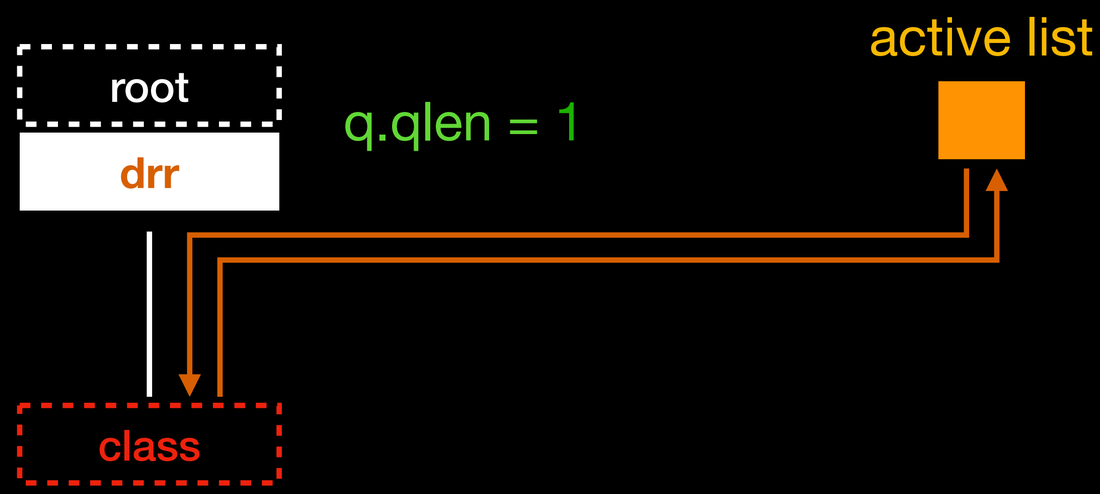

root
drr
q.qlen = 1
active list
Dequeue
class
netem
q.qlen = 1
tfifo queue
P
netem
q.qlen = 0


root
drr
q.qlen = 1
active list
class
netem
q.qlen = 0
tfifo queue
P
(Internally enqueue)
P
netem
q.qlen = 1

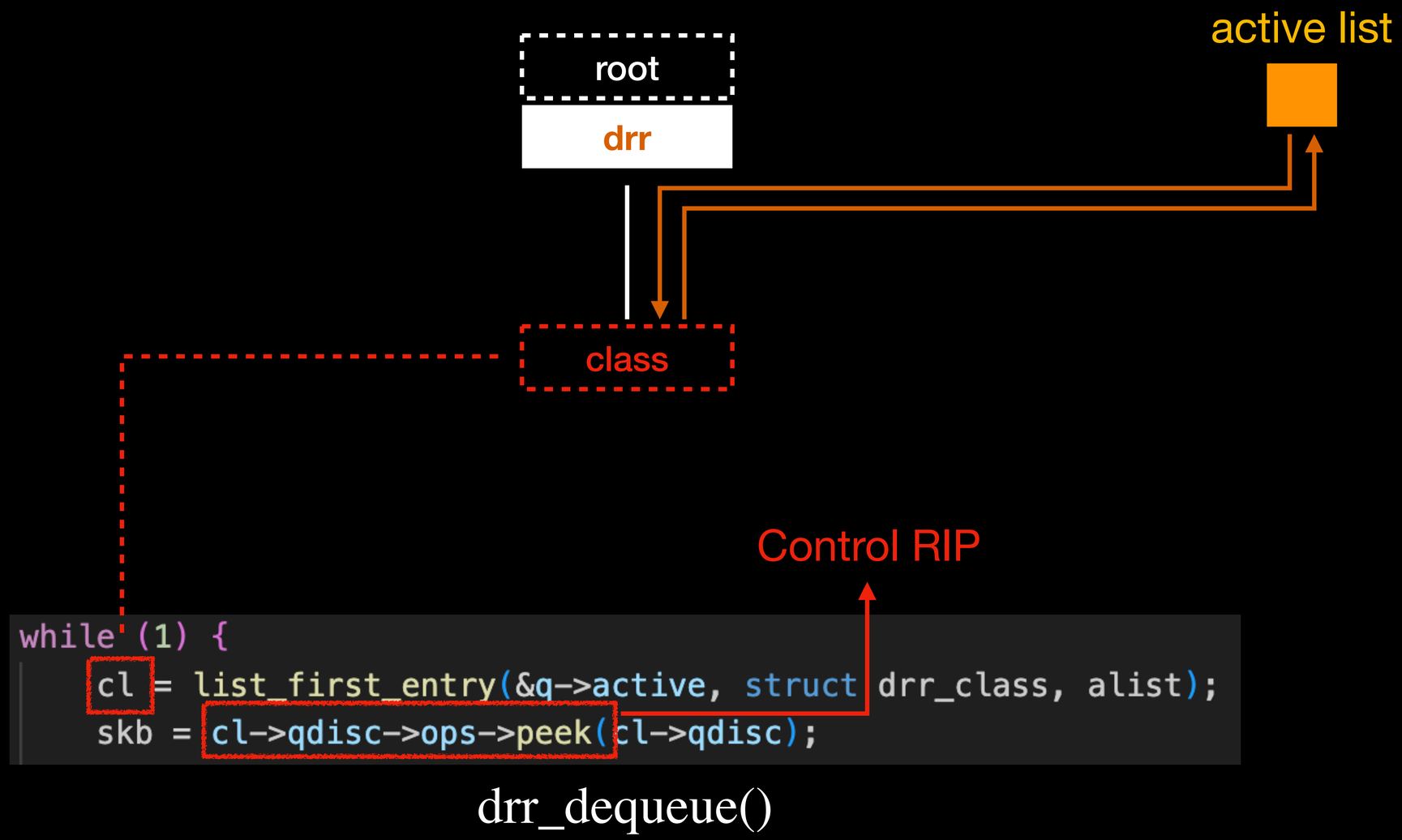


The packet length is mismatched now


active list
root
drr
q.qlen = 1
Delete the DRR class
class
netem
q.qlen = 0
tfifo queue
P
netem
q.qlen = 1



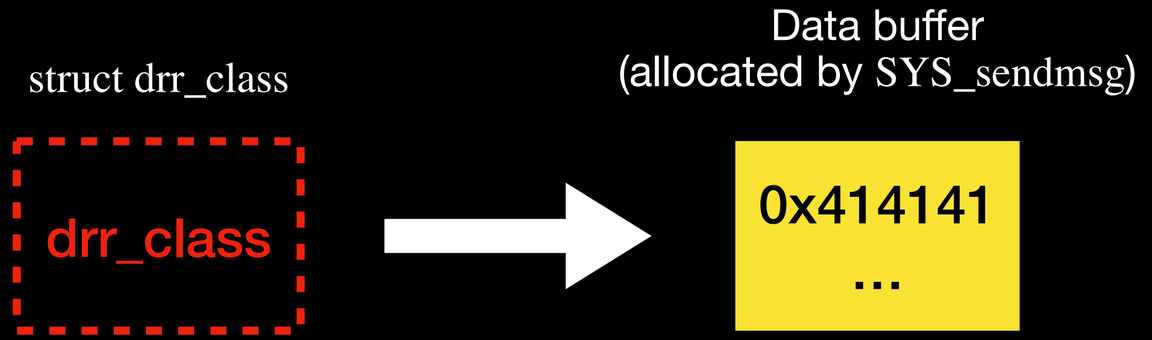
In DRR's destructor, the class object is removed from the active list only if q.qlen != 0


root
drr
active list
class
Control RIP
while (1) {
cl = list_first_entry(&q->active, struct drr_class, alist);
skb = cl->qdisc->ops->peek(cl->qdisc);

drr_dequeue()

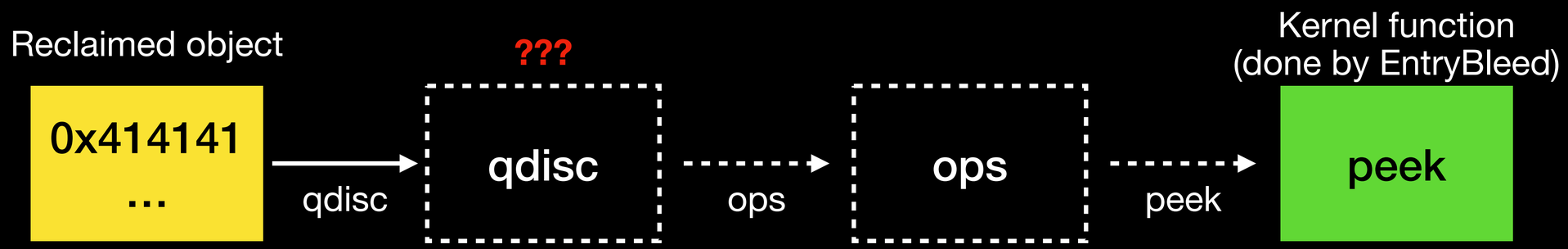
$ Exploitation

- Goal: Gain full control of cl->qdisc->ops->peek function pointer
- Bypass KASLR using EntryBleed
- Use SYS_sendmsg to spray and reclaim the freed drr_class object



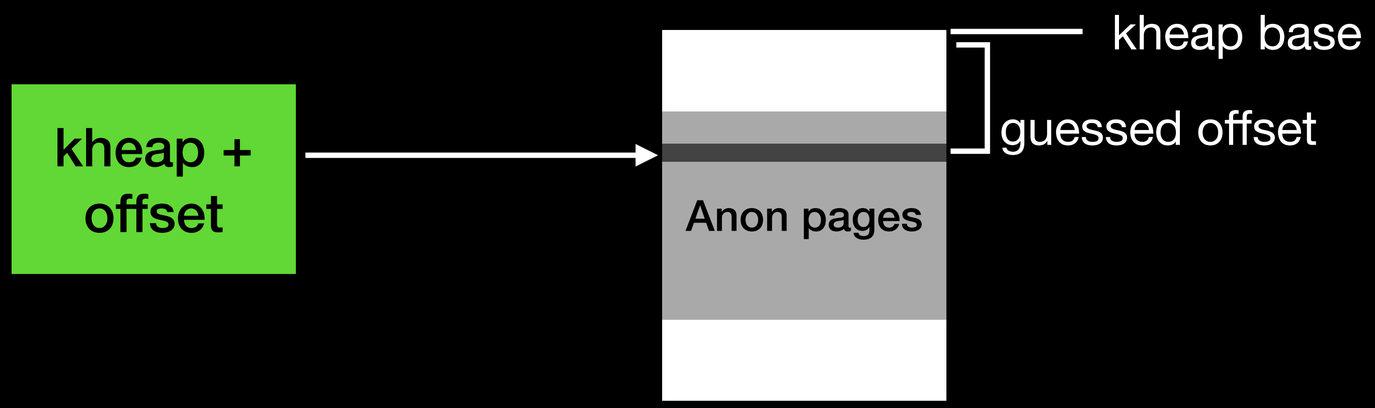
$ Exploitation

- Where should we set `cl->qdisc` ?



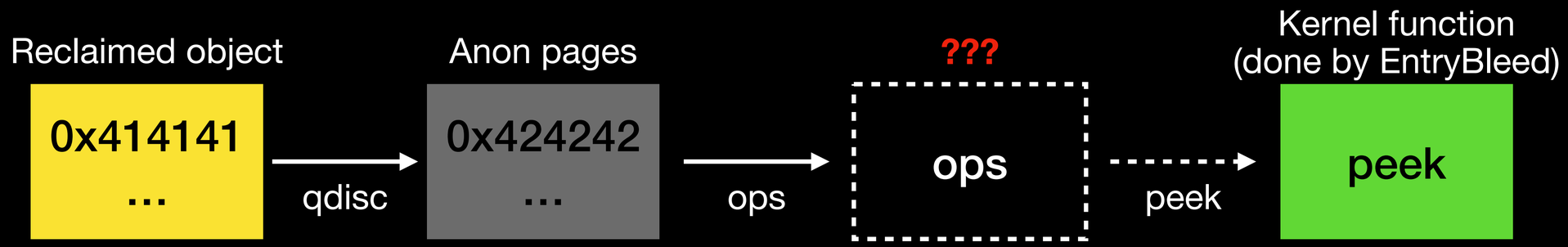
$ Exploitation

- Use EntryBleed again to leak kernel heap base address
- Spray many anonymous pages
- Assign `cl->qdisc` to the kernel heap base address plus a fixed offset



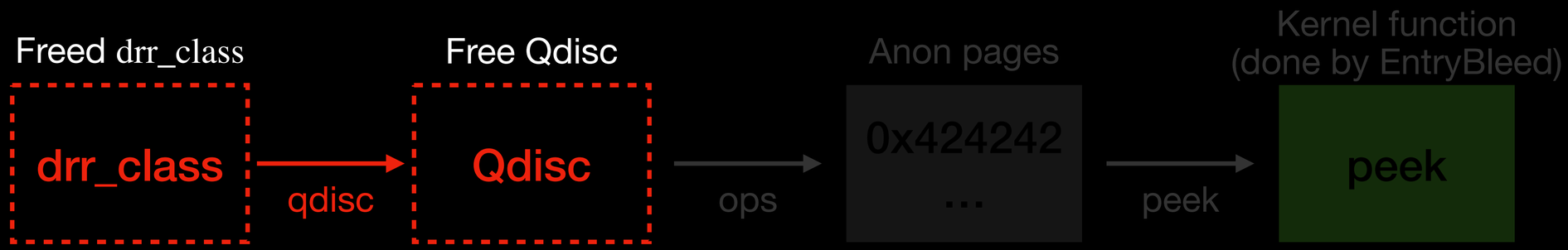
$ Exploitation

- Looks good so far — but what's next?
- Still no reliable way to control cl->qdisc->ops 😔



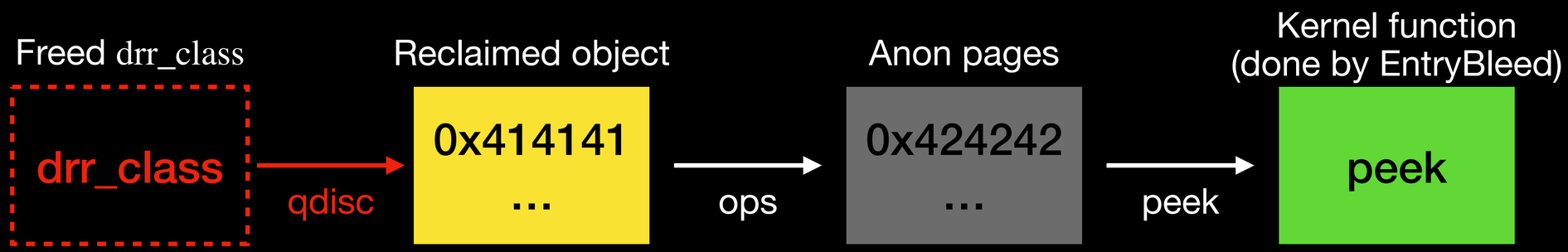
$ Exploitation

- If the freed cl is left untouched, cl->qdisc still points to a freed Qdisc



$ Exploitation

- If the freed cl is left untouched, cl->qdisc still points to a freed Qdisc
- Instead of reclaiming cl, reclaim freed Qdisc to hijack the secondary object



$ Exploitation

- We control the RDI when calling cl->qdisc->ops->peek(cl->qdisc)
- Stack pivoting to RDI and run ROP chains:
  1. Overwrite core_pattern[] with path to our program
  2. Sleep indefinitely
- Trigger a SIGSEGV in user space to read the flag

DEMO
CVE-2024-56770 / LTS 6.6.66

```
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@l
```

```
ts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@l
ts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$ id
id
uid=1000(user) gid=1000(user) groups=1000(user)
user@lts-6:/tmp$ u
```

$ Brief Summary

- Wrote a simple Python script to run exploit
- Submission as exp223 (dupe)
- It failed, but I think it was a nice try!
  - Found a straightforward exploit path
  - Unexpectedly discovered two 0-days!

Unexpected Discovery:

CVE-2025-21703, CVE-2025-21700

```
if (time_to_send <= now && q->slot.slot_next <= now) {
        netem_erase_head(q, skb);
        sch->q.qlen--;
        qdisc_qstats_backlog_dec(sch, skb);
        q->t_len--;
        skb->next = NULL;
        skb->prev = NULL;
        /* skb->dev shares skb->rbnode area,
21 @@ deliver:
                        if (net_xmit_drop_count(err))
                                qdisc_qstats_drop(sch);
                        qdisc_tree_reduce_backlog(sch, 1, pkt_len);
                        sch->qstats.backlog -= pkt_len;
                        sch->q.qlen--;
                }
                goto tfifo_dequeue;
        }
        sch->q.qlen--;
        goto deliver;
}

if (q->qdisc) {
        skb = q->qdisc->ops->dequeue(q->qdisc);
        if (skb)
        if (skb) {
                sch->q.qlen--;
                goto deliver;
        }
}
```

Introduce the field t_len in netem private data to track the length of the tfifo queue

The patch for CVE-2024-56770

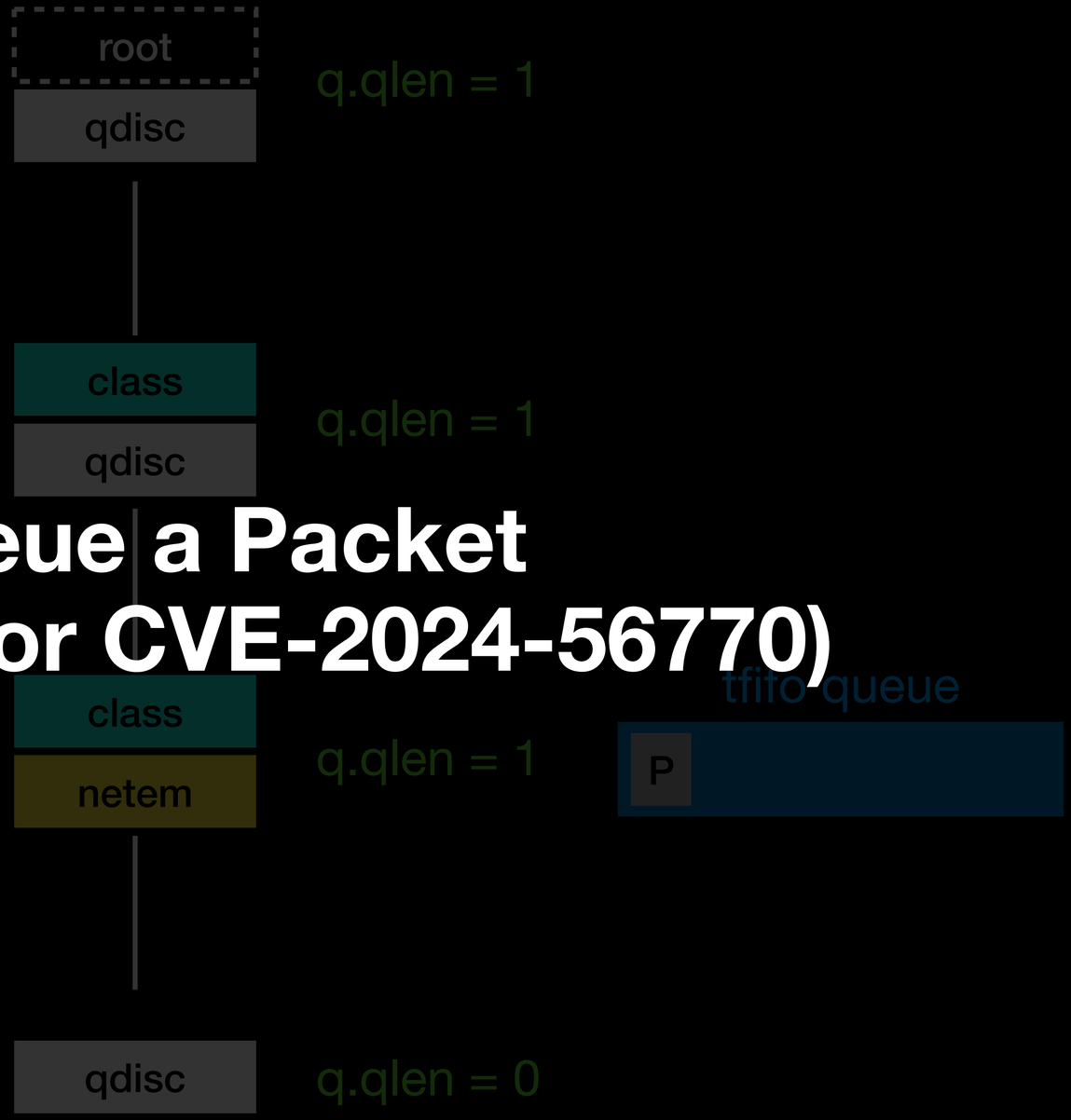
```
if (time_to_send <= now && q->slot.slot_next <= now) {
        netem_erase_head(q, skb);
        sch->q.qlen--;
        qdisc_qstats_backlog_dec(sch, skb);
        q->t_len--;
        skb->next = NULL;
        skb->prev = NULL;
        /* skb->dev shares skb->rbnode area,
21 @@ deliver:
                        if (net_xmit_drop_count(err))
                                qdisc_qstats_drop(sch);
                        qdisc_tree_reduce_backlog(sch, 1, pkt_len);
                        sch->qstats.backlog -= pkt_len;
                        sch->q.qlen--;
                }
                goto tfifo_dequeue;
        }
        sch->q.qlen--;
        goto deliver;
}

if (q->qdisc) {
        skb = q->qdisc->ops->dequeue(q->qdisc);
        if (skb)
        if (skb) {
                sch->q.qlen--;
                goto deliver;
        }
}
```

Only decrease the packet count (qlen) when the private qdisc either: [1] enqueue fails, or [2] dequeue succeeds

The patch for CVE-2024-56770

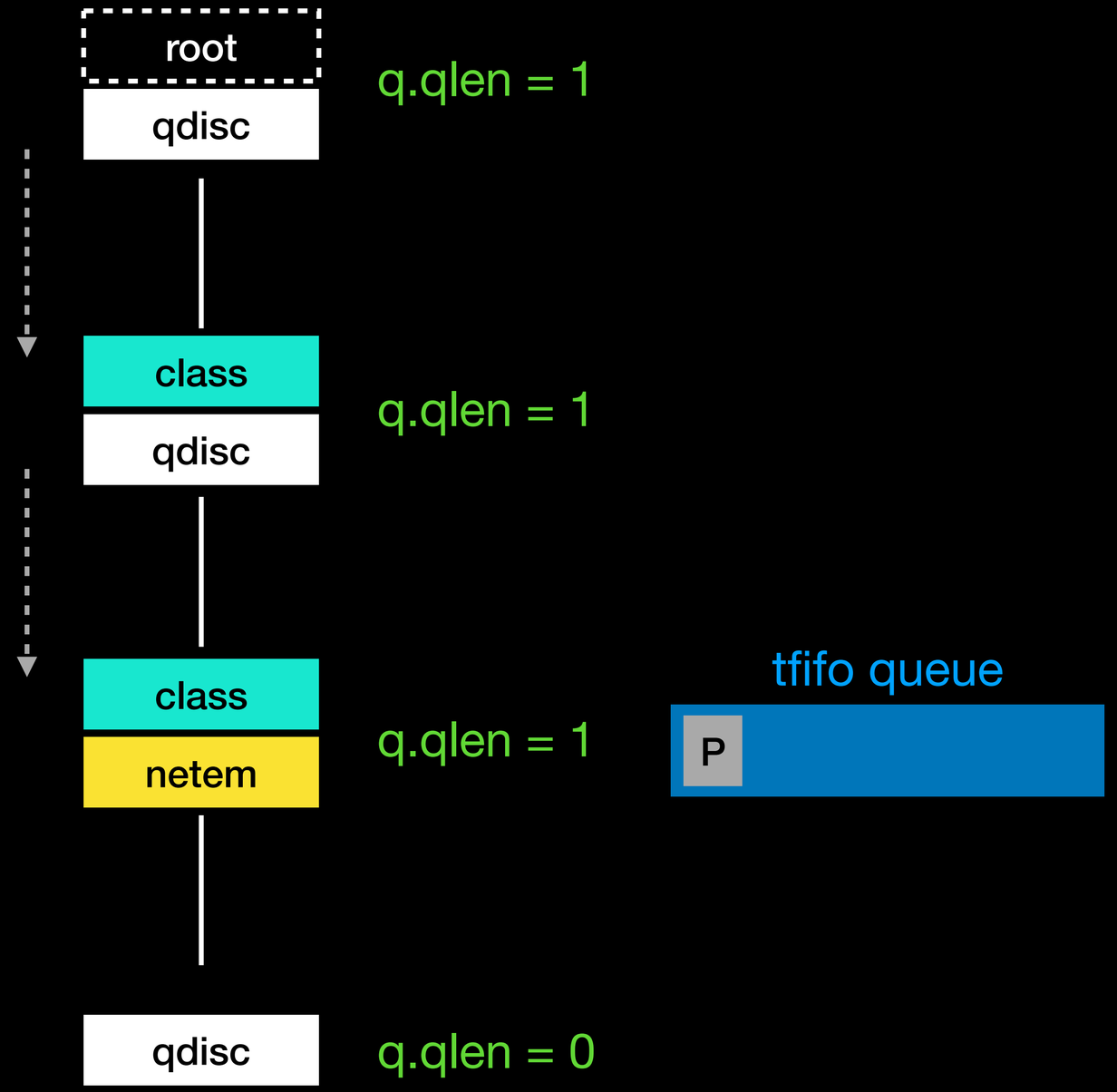
$ CVE-2025-21703 RCA

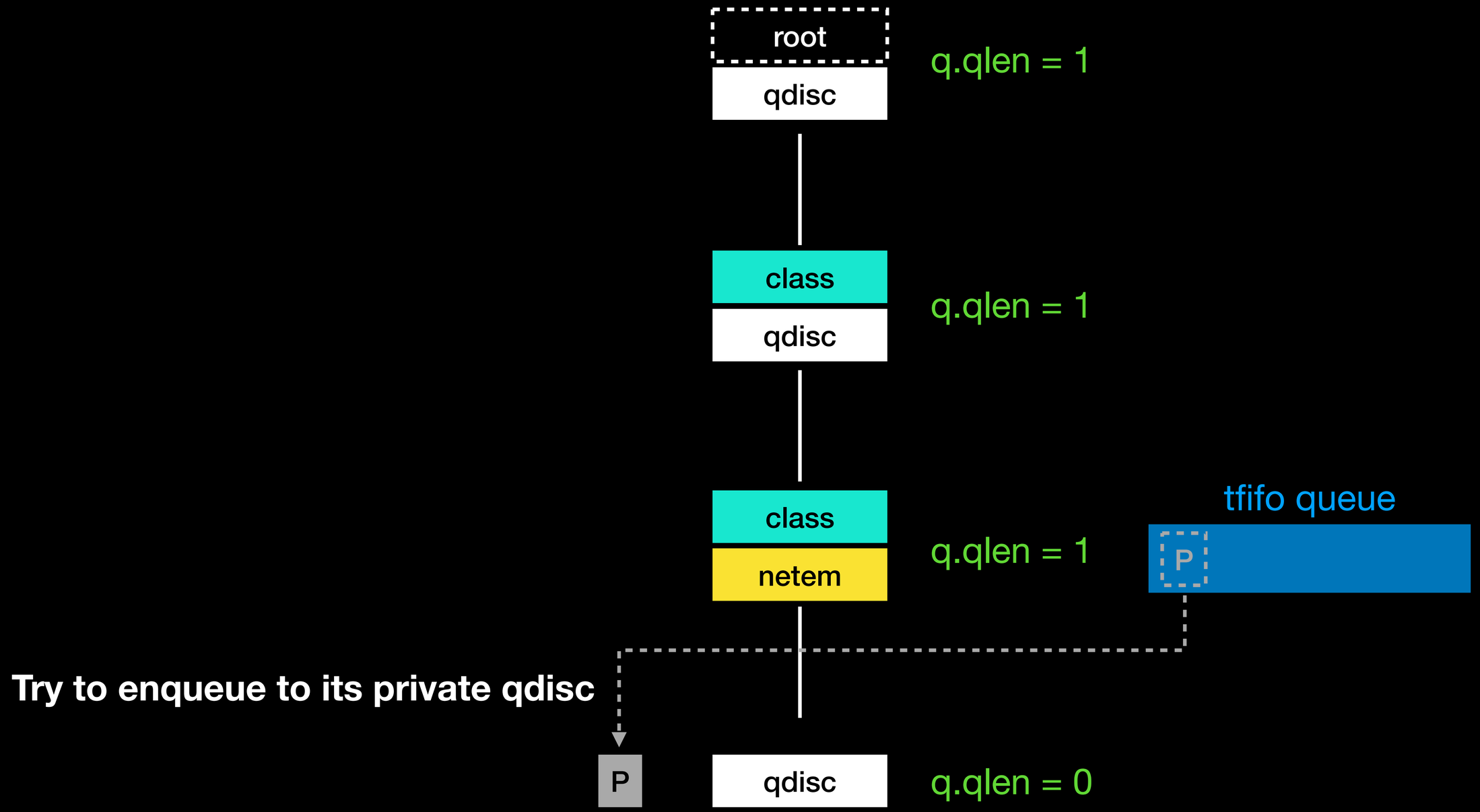
- netem: Update sch->q.qlen before qdisc_tree_reduce_backlog()
  - A side effect of the patch for CVE-2024-56770
  - Notify the parent qdisc before `qlen` is actually updated in the dequeue handler

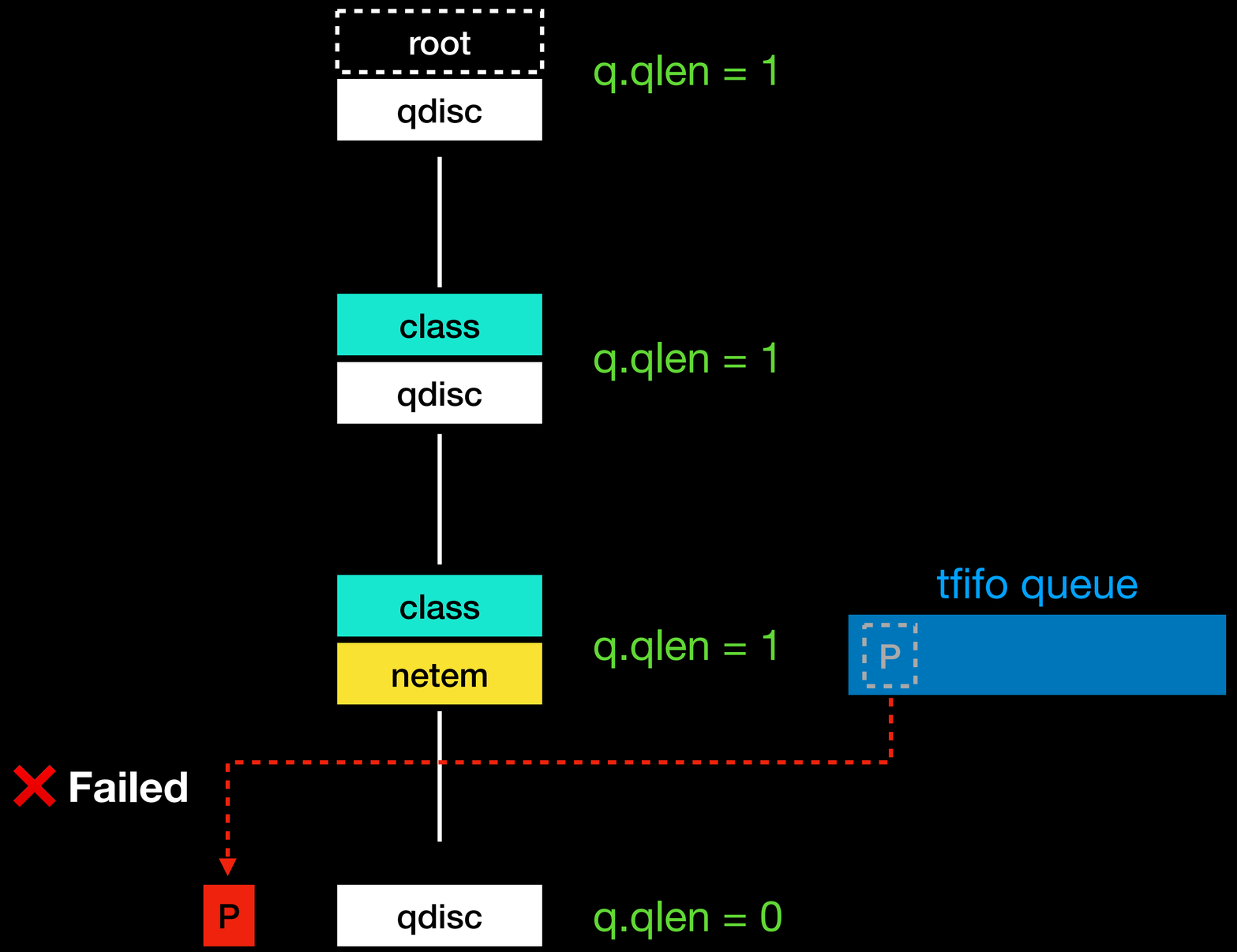


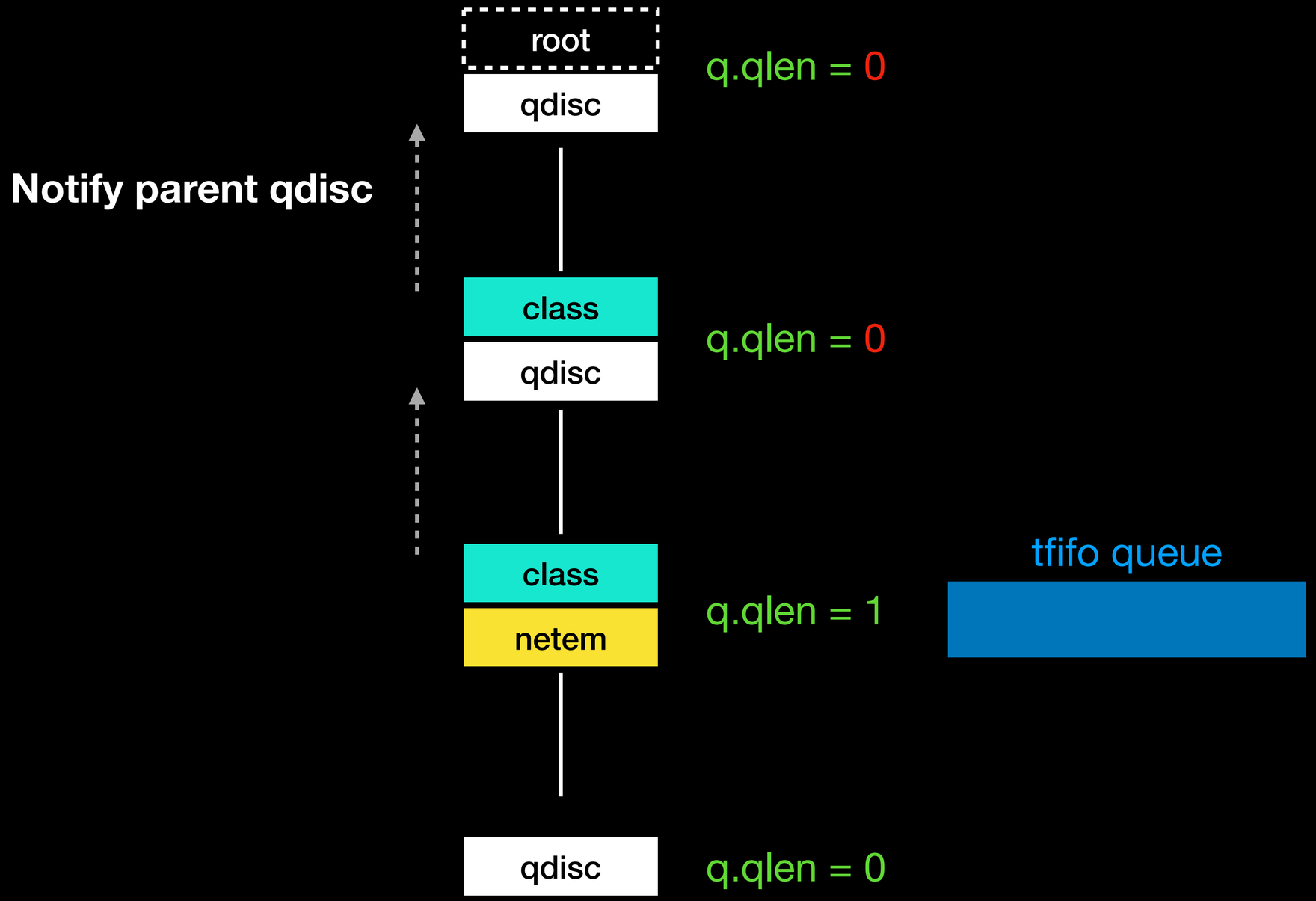
Dequeue a Packet (after patch for CVE-2024-56770)

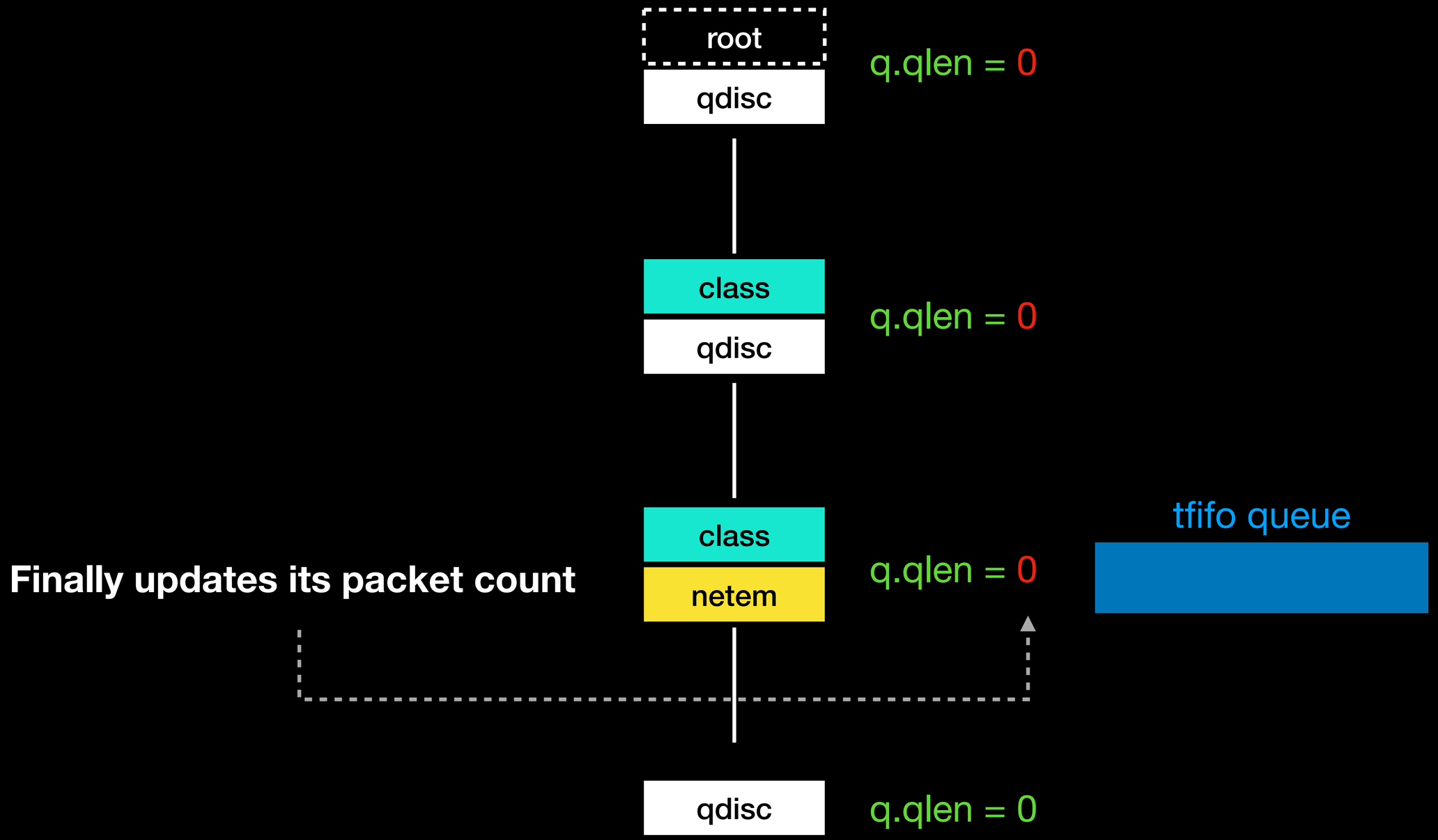
Try to retrieve a packet from child




root
qdisc
q.qlen = 1
class
qdisc
q.qlen = 1
class
netem
q.qlen = 1
tfifo queue
P
Try to enqueue to its private qdisc
P
qdisc
q.qlen = 0


root
qdisc
q.qlen = 1
class
qdisc
q.qlen = 1
class
netem
q.qlen = 1
tfifo queue
P
Failed
P
qdisc
q.qlen = 0


root
qdisc
q.qlen = 0
Notify parent qdisc
class
qdisc
q.qlen = 0
class
netem
q.qlen = 1
tfifo queue
qdisc
q.qlen = 0


root
qdisc
q.qlen = 0
class
qdisc
q.qlen = 0
class
netem
q.qlen = 0
tfifo queue
Finally updates its packet count
qdisc
q.qlen = 0

```c
void qdisc_tree_reduce_backlog(struct Qdisc *sch, int n, int len)
{
    const struct Qdisc_class_ops *cops;
    unsigned long cl;
    u32 parentid;
    bool notify;

    while ((parentid = sch->parent)) {
        if (parentid == TC_H_ROOT)
            break;

        notify = !sch->q.qlen;

        sch = qdisc_lookup_rcu(qdisc_dev(sch), TC_H_MAJ(parentid));
        cops = sch->ops->cl_ops;
        if (notify && cops->qlen_notify) {
            cl = cops->find(sch, parentid);
            cops->qlen_notify(sch, cl);
        }
        sch->q.qlen -= n;
    }
}
```

Simplified qdisc_tree_reduce_backlog()

```c
void qdisc_tree_reduce_backlog(struct Qdisc *sch, int n, int len)
{
    const struct Qdisc_class_ops *cops;
    unsigned long cl;
    u32 parentid;
    bool notify;

    while ((parentid = sch->parent)) {
        if (parentid == TC_H_ROOT)
            break;

        notify = !sch->q.qlen;

        sch = qdisc_lookup_rcu(qdisc_dev(sch), TC_H_MAJ(parentid));
        cops = sch->ops->cl_ops;
        if (notify && cops->qlen_notify) {
            cl = cops->find(sch, parentid);
            cops->qlen_notify(sch, cl);
        }
        sch->q.qlen -= n;
    }
}
```

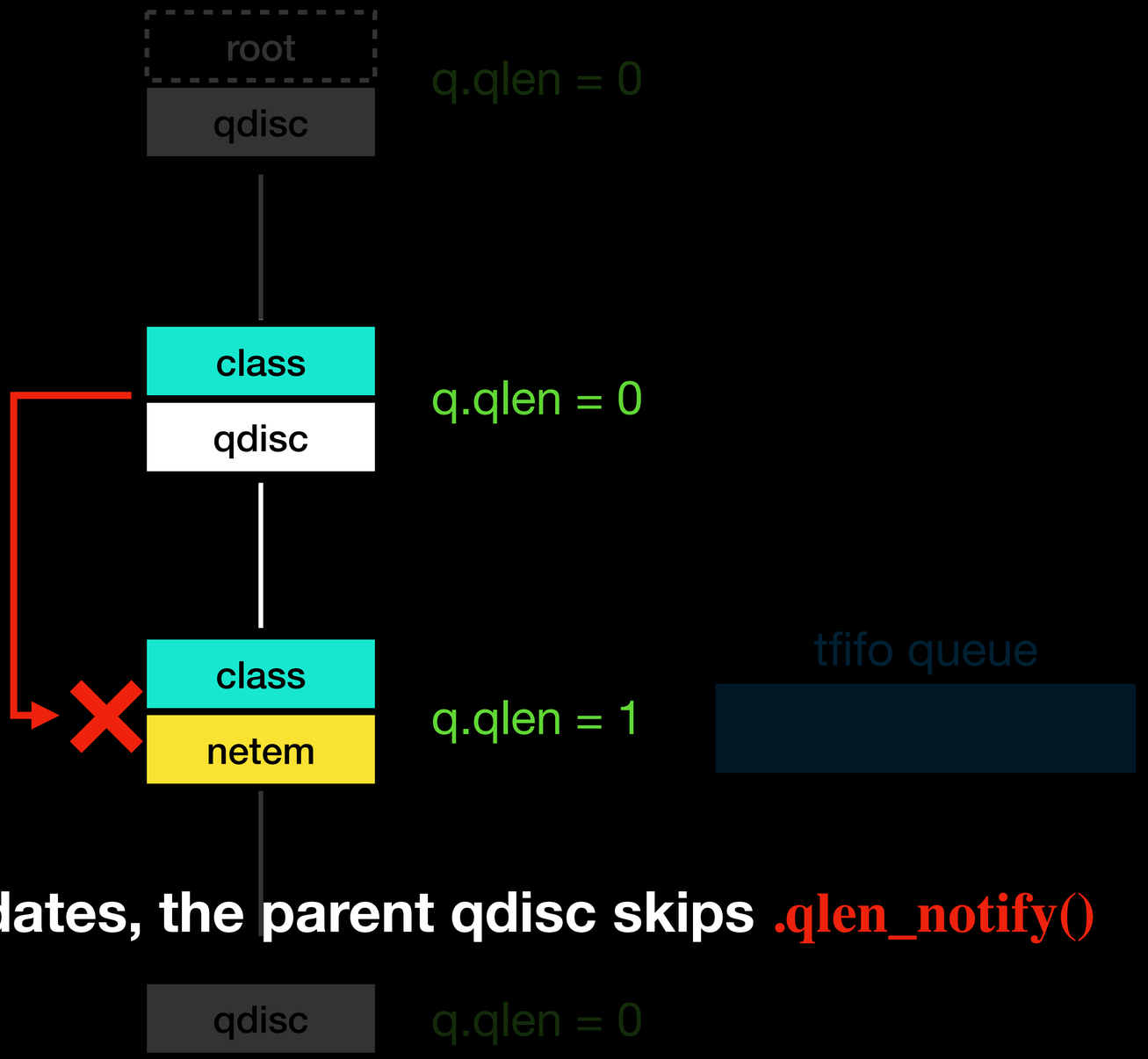
Simplified qdisc_tree_reduce_backlog()

Walk up the qdisc tree,
updating packet count until reaching the root qdisc

```c
void qdisc_tree_reduce_backlog(struct Qdisc *sch, int n, int len)
{
    const struct Qdisc_class_ops *cops;
    unsigned long cl;
    u32 parentid;
    bool notify;

    while ((parentid = sch->parent)) {
        if (parentid == TC_H_ROOT)
            break;

        notify = !sch->q.qlen;

        sch = qdisc_lookup_rcu(qdisc_dev(sch), TC_H_MAJ(parentid));
        cops = sch->ops->cl_ops;
        if (notify && cops->qlen_notify) {
            cl = cops->find(sch, parentid);
            cops->qlen_notify(sch, cl);
        }
        sch->q.qlen -= n;
    }
}
```

When the child's packet count hits zero,
the parent calls .qlen_notify() to update scheduler state

Simplified qdisc_tree_reduce_backlog()



Due to misordered updates, the parent qdisc skips .qlen_notify()

```
if (err != NET_XMIT_SUCCESS) {
        if (net_xmit_drop_count(err))
                qdisc_qstats_drop(sch);
        qdisc_tree_reduce_backlog(sch, 1, pkt_len);
        sch->qstats.backlog -= pkt_len;
        sch->q.qlen--;
        qdisc_tree_reduce_backlog(sch, 1, pkt_len);
}
goto tfifo_dequeue;
```

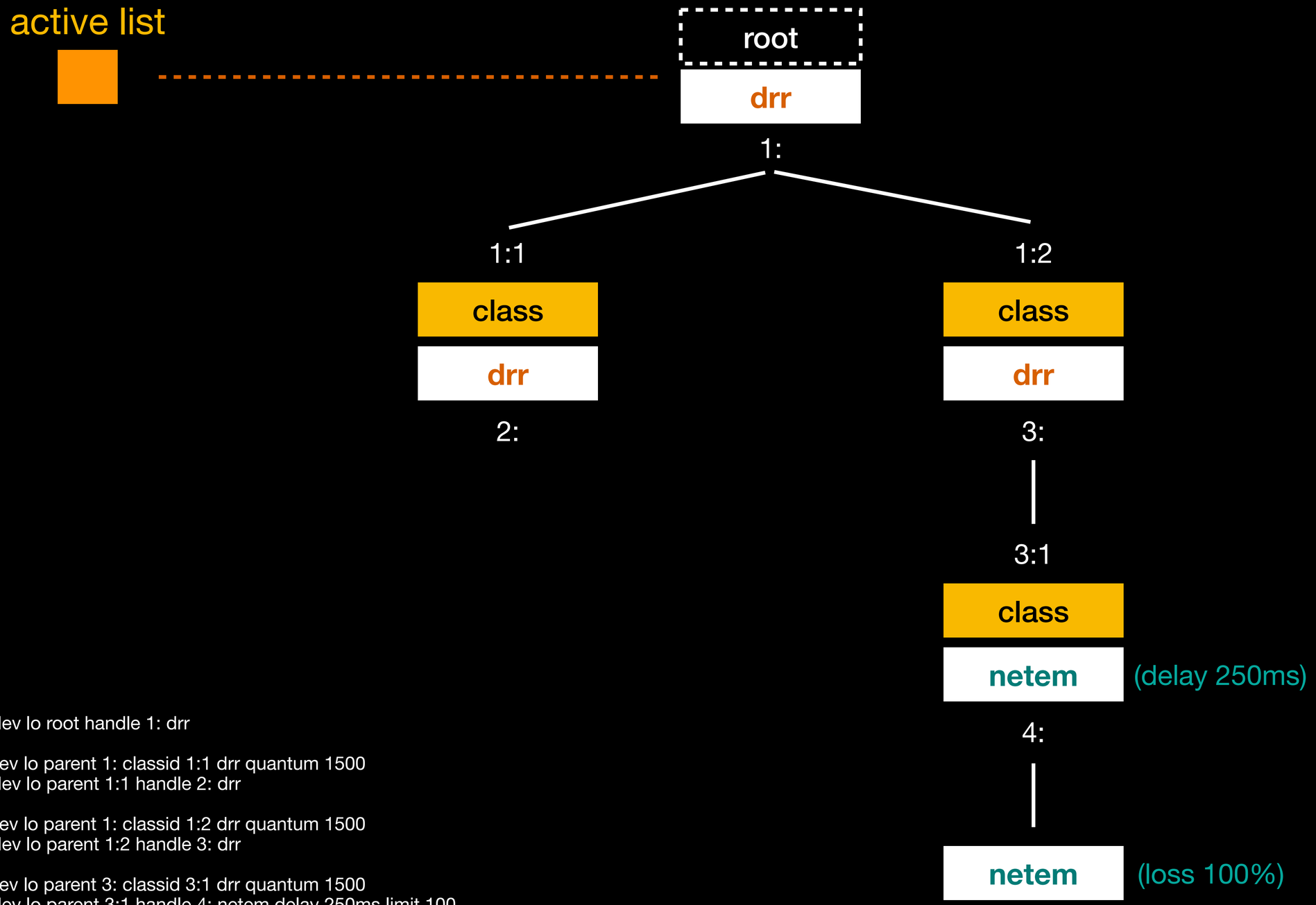
The patch for CVE-2025-21703

$ CVE-2025-21703 RCA

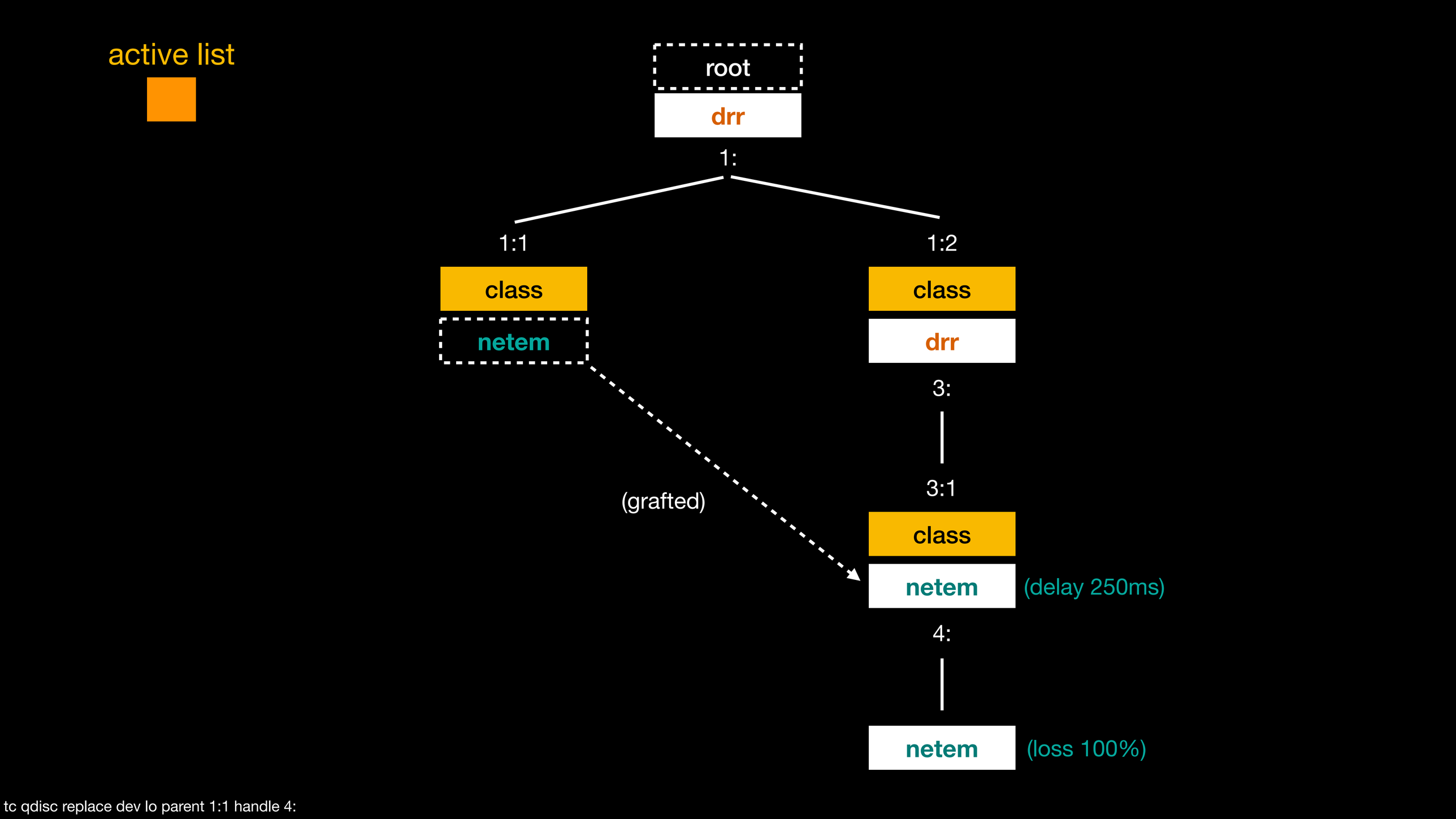
- In DRR, .qlen_notify → drr_qlen_notify() removes the child class from the active list
  - If it is skipped accidentally, a released DRR class object will stay in the active list
- Share the same primitive as CVE-2024-56770 → exploit can be reused!

$ CVE-2025-21700 RCA

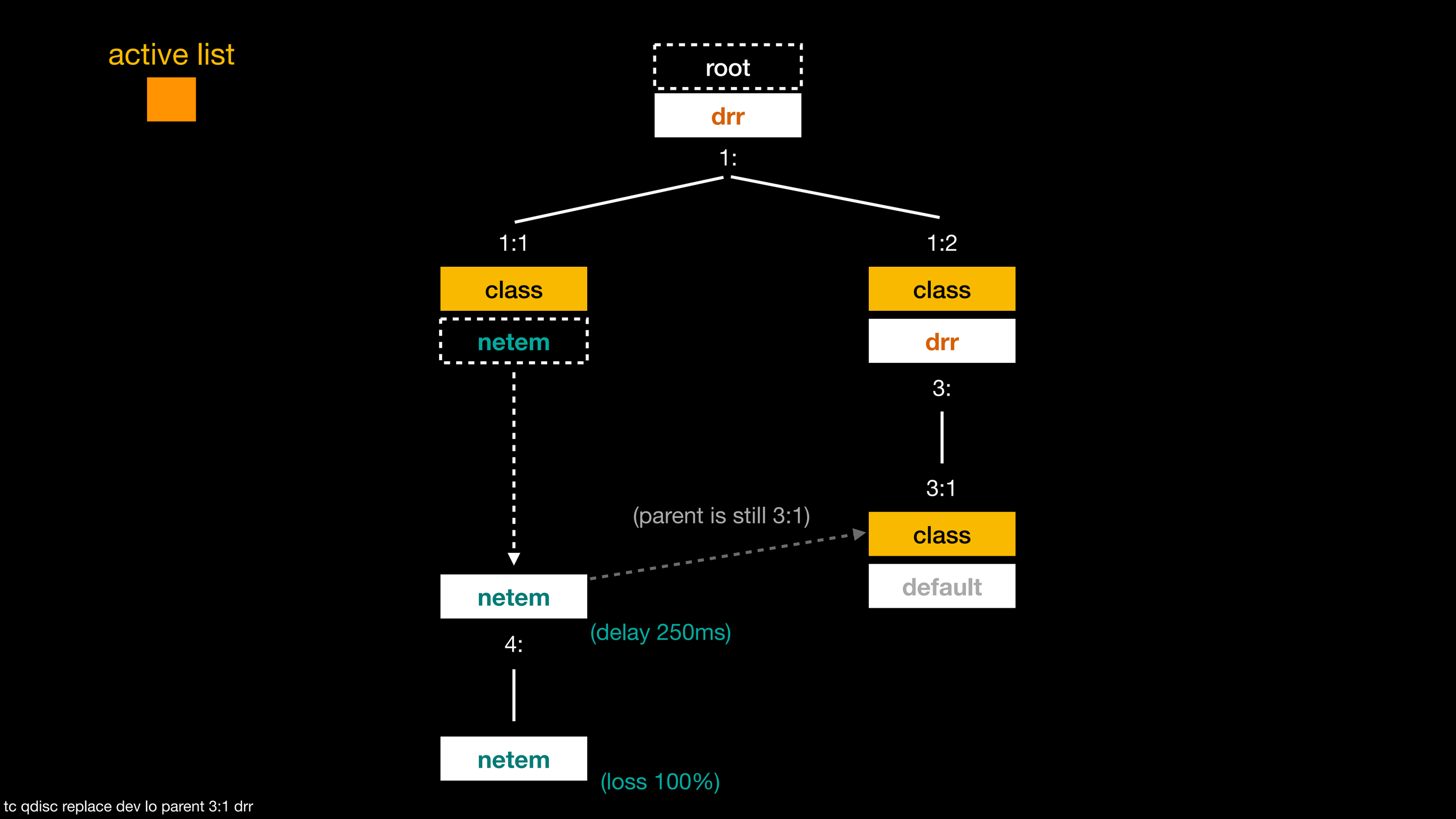
- net: sched: Disallow replacing of child qdisc from one parent to another
  - qdisc_tree_reduce_backlog() finds the parent via sch->parent
  - net/sched allows grafting a qdisc at a handle
    - Grafting won't update the sch->parent
    - qdisc_tree_reduce_backlog() may find the wrong parent qdisc



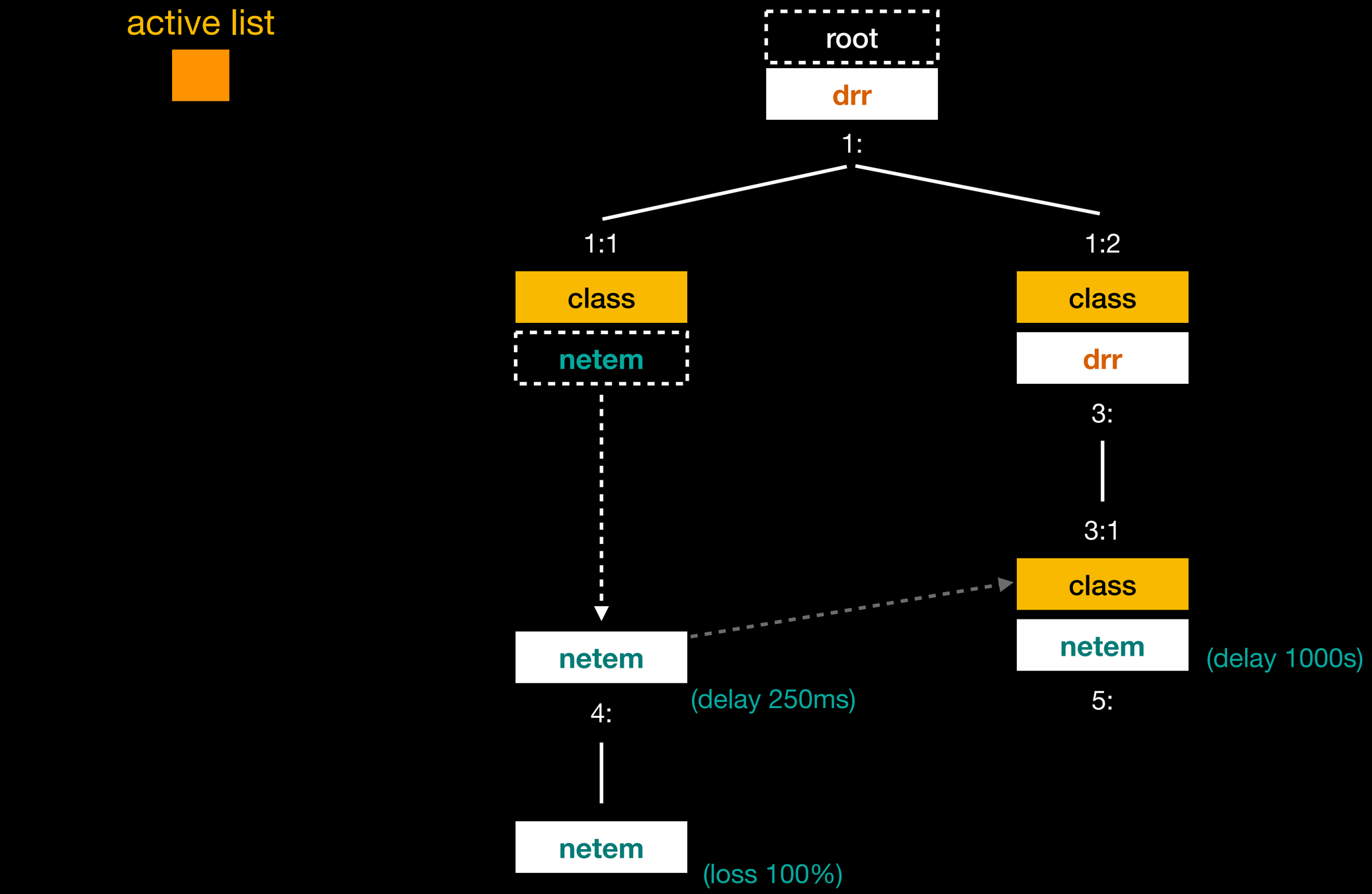

```
tc qdisc add dev lo root handle 1: drr

tc class add dev lo parent 1: classid 1:1 drr quantum 1500
tc qdisc add dev lo parent 1:1 handle 2: drr

tc class add dev lo parent 1: classid 1:2 drr quantum 1500
tc qdisc add dev lo parent 1:2 handle 3: drr

tc class add dev lo parent 3: classid 3:1 drr quantum 1500
tc qdisc add dev lo parent 3:1 handle 4: netem delay 250ms limit 100

tc qdisc add dev lo parent 4: netem delay 1000s loss 100% limit 100
```



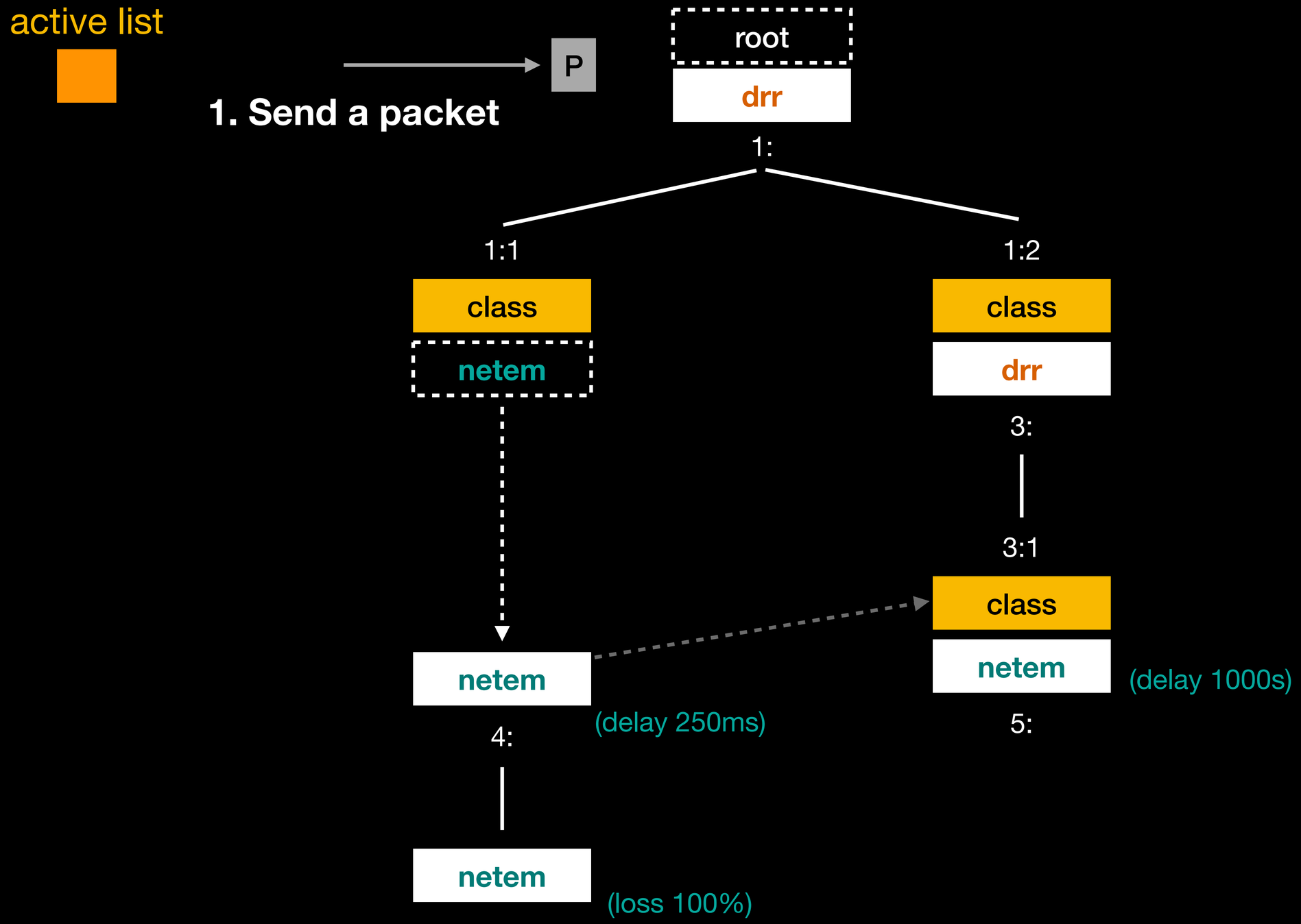
tc qdisc replace dev lo parent 1:1 handle 4:

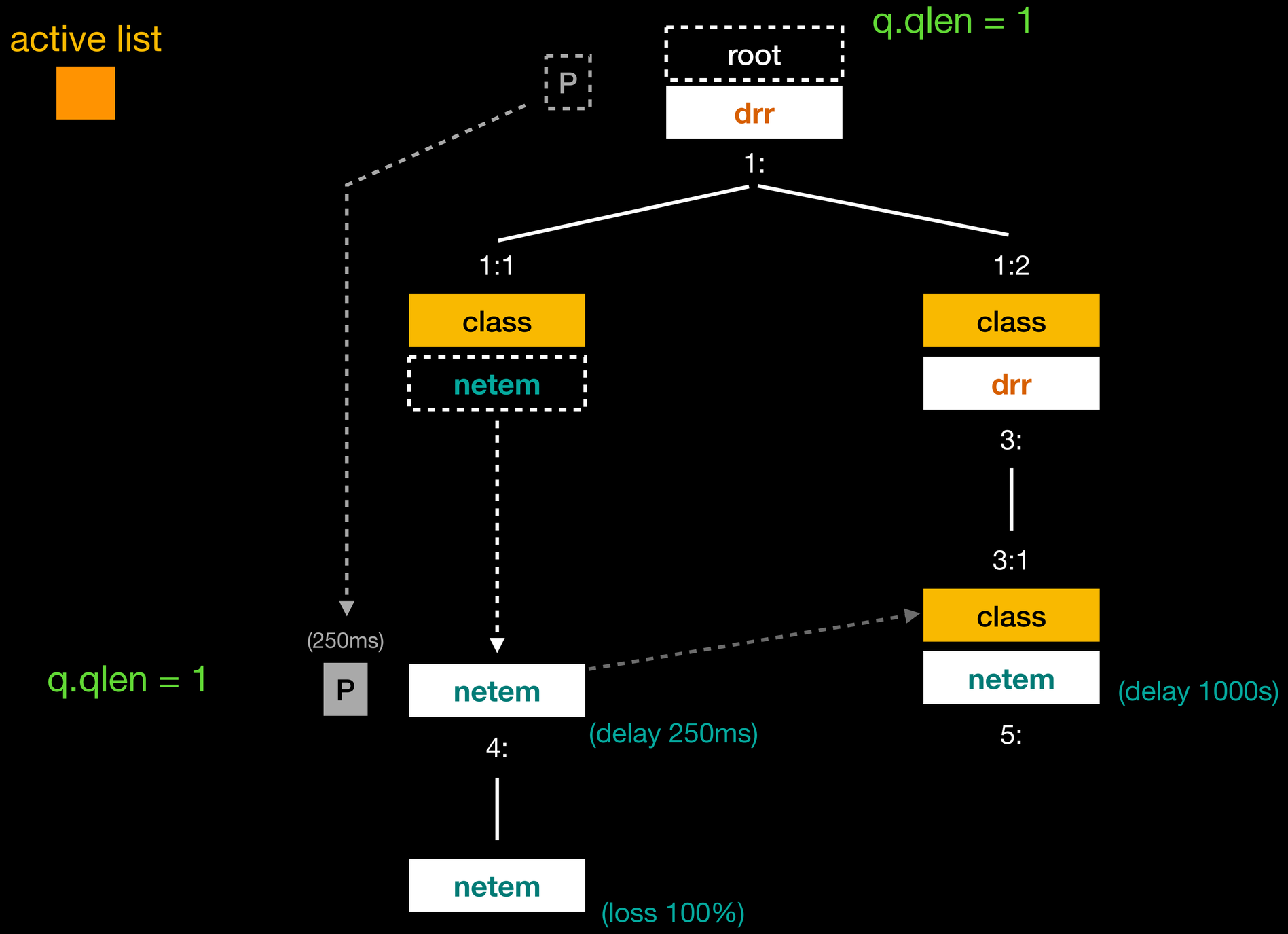


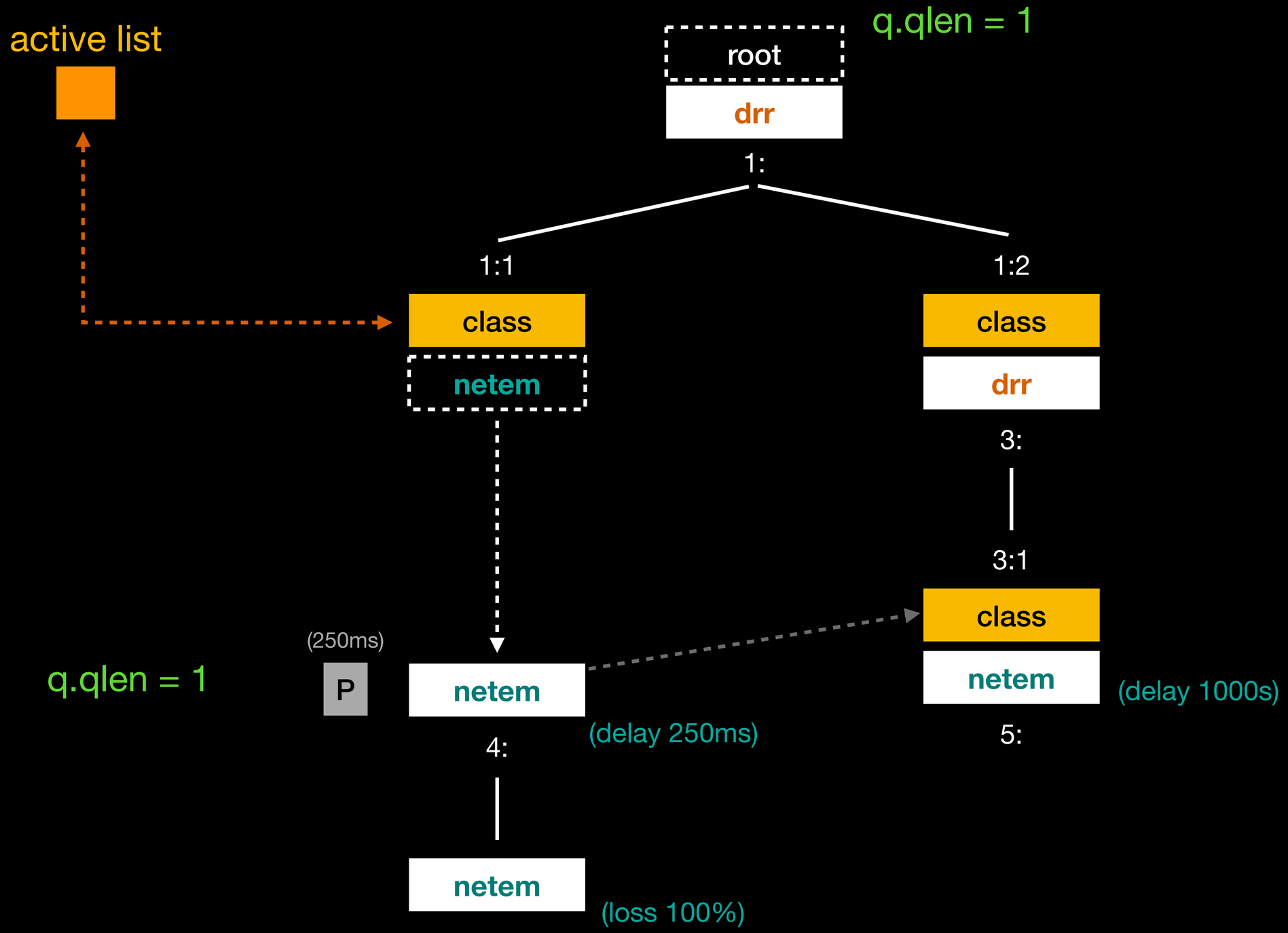
tc qdisc replace dev lo parent 3:1 drr

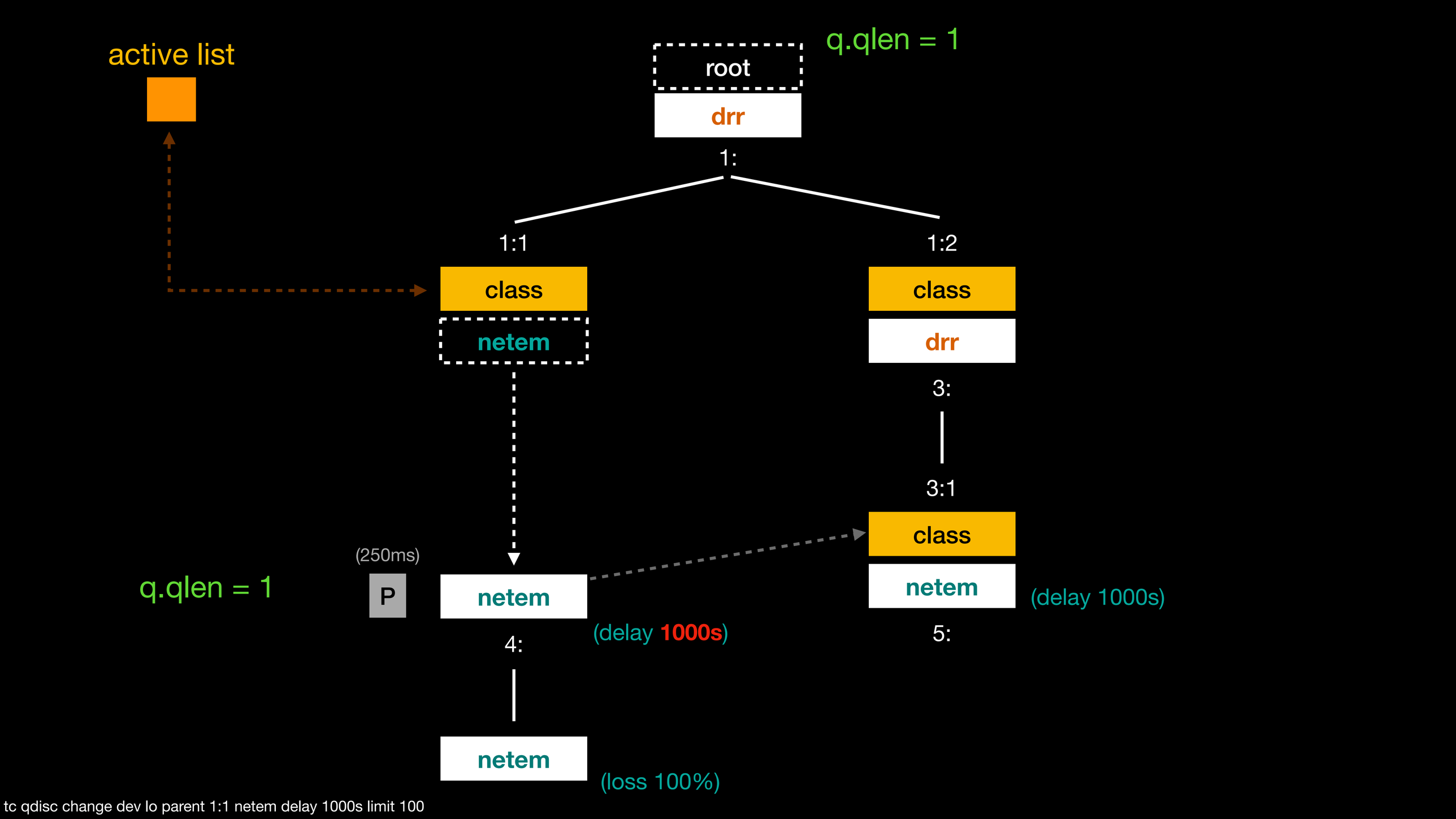


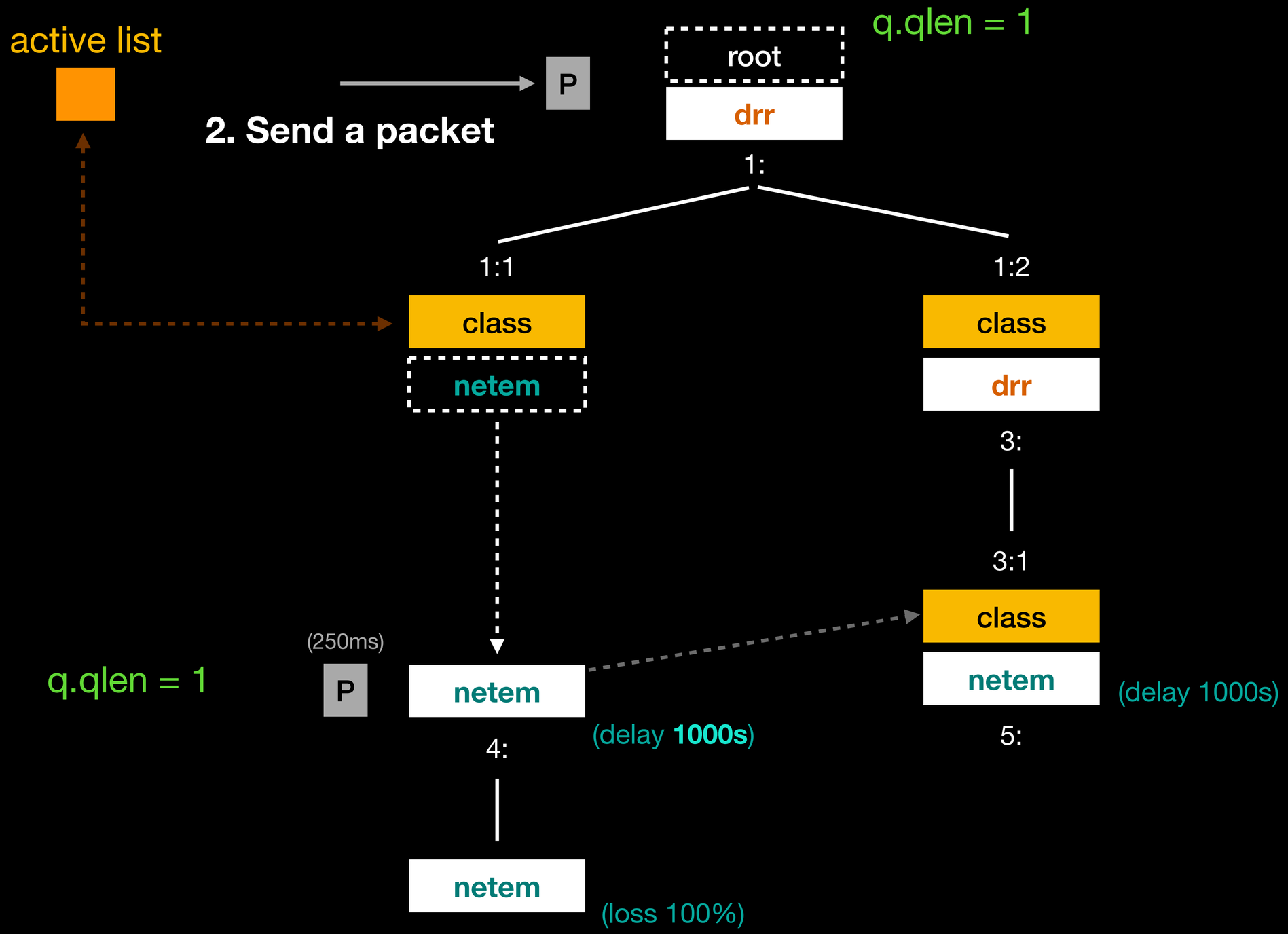
tc qdisc replace dev lo parent 3:1 handle 5: netem delay 1000s limit 100

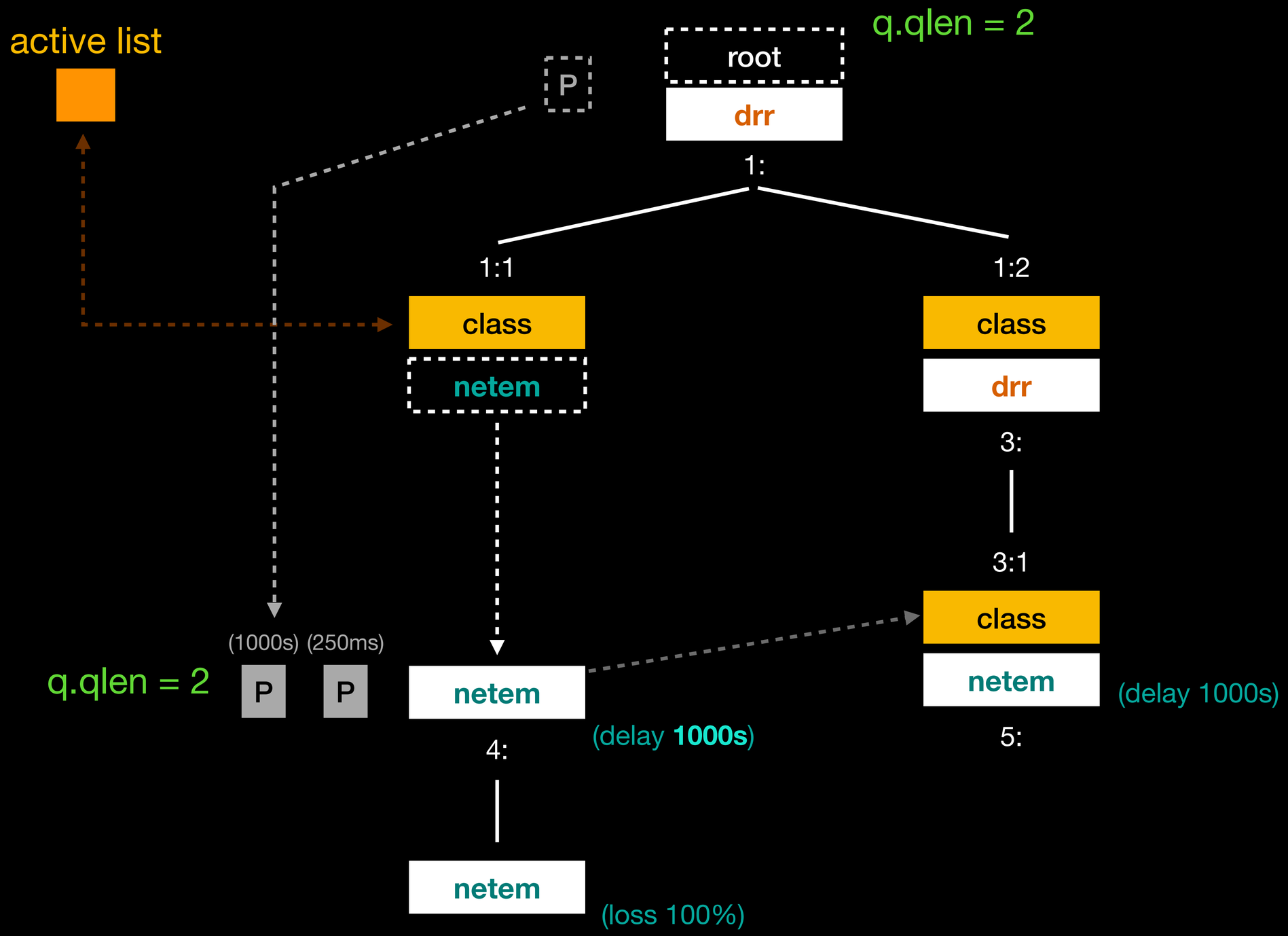

active list
P
1. Send a packet
root
drr
1:
1:1
1:2
class
netem
class
drr
3:
3:1
class
netem
(delay 1000s)
netem
(delay 250ms)
5:
4:
netem
(loss 100%)

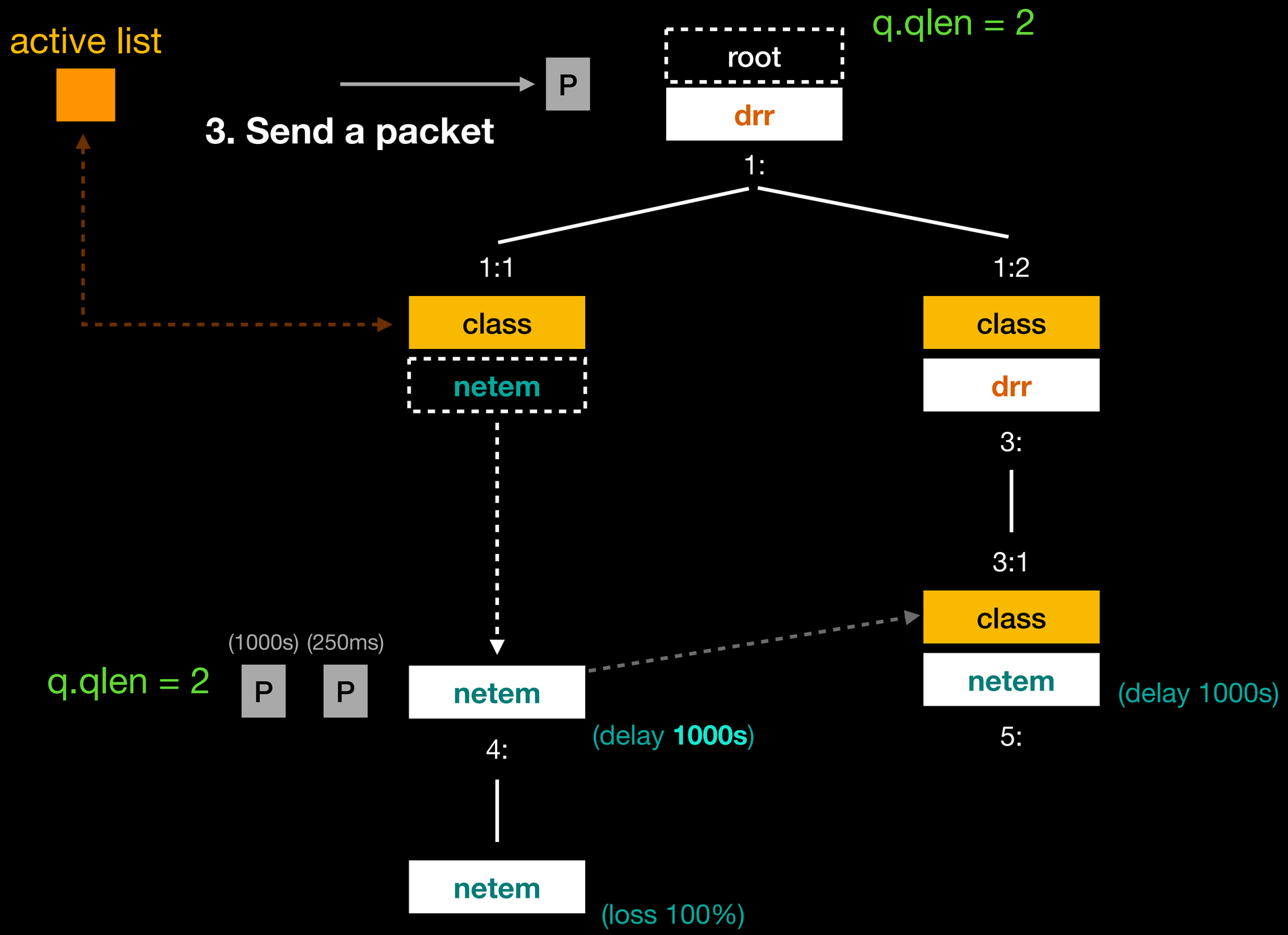

active list
q.qlen = 1
root
drr
P
1:
1:1
1:2
class
netem
class
drr
3:
3:1
class
netem
(delay 1000s)
5:
(250ms)
P
netem
(delay 250ms)
4:
netem
(loss 100%)
q.qlen = 1

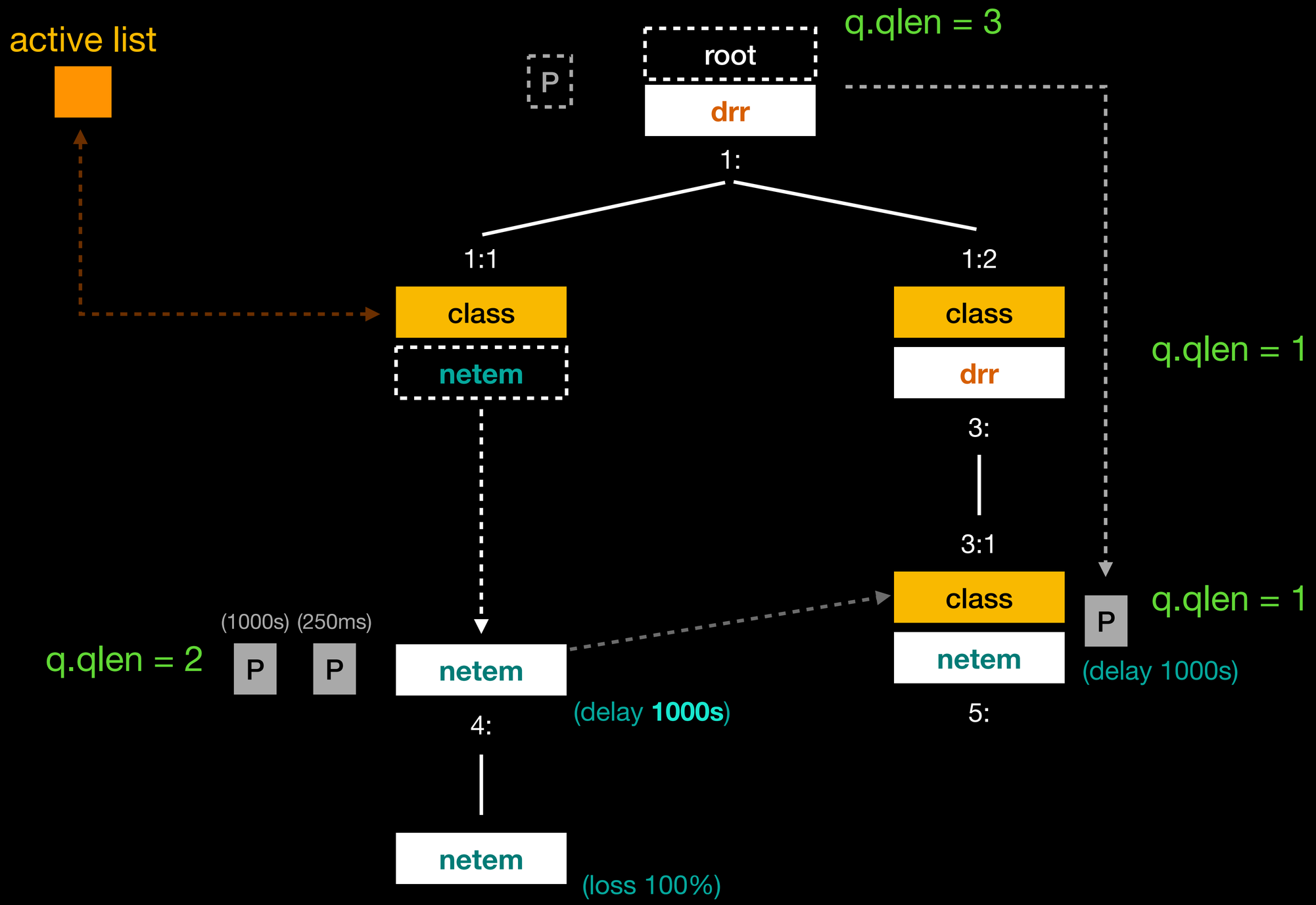

active list
root
q.qlen = 1
drr
1:
1:1
1:2
class
netem
class
drr
3:
3:1
class
netem
(delay 1000s)
5:
(250ms)
P
q.qlen = 1
netem
(delay 250ms)
4:
netem
(loss 100%)

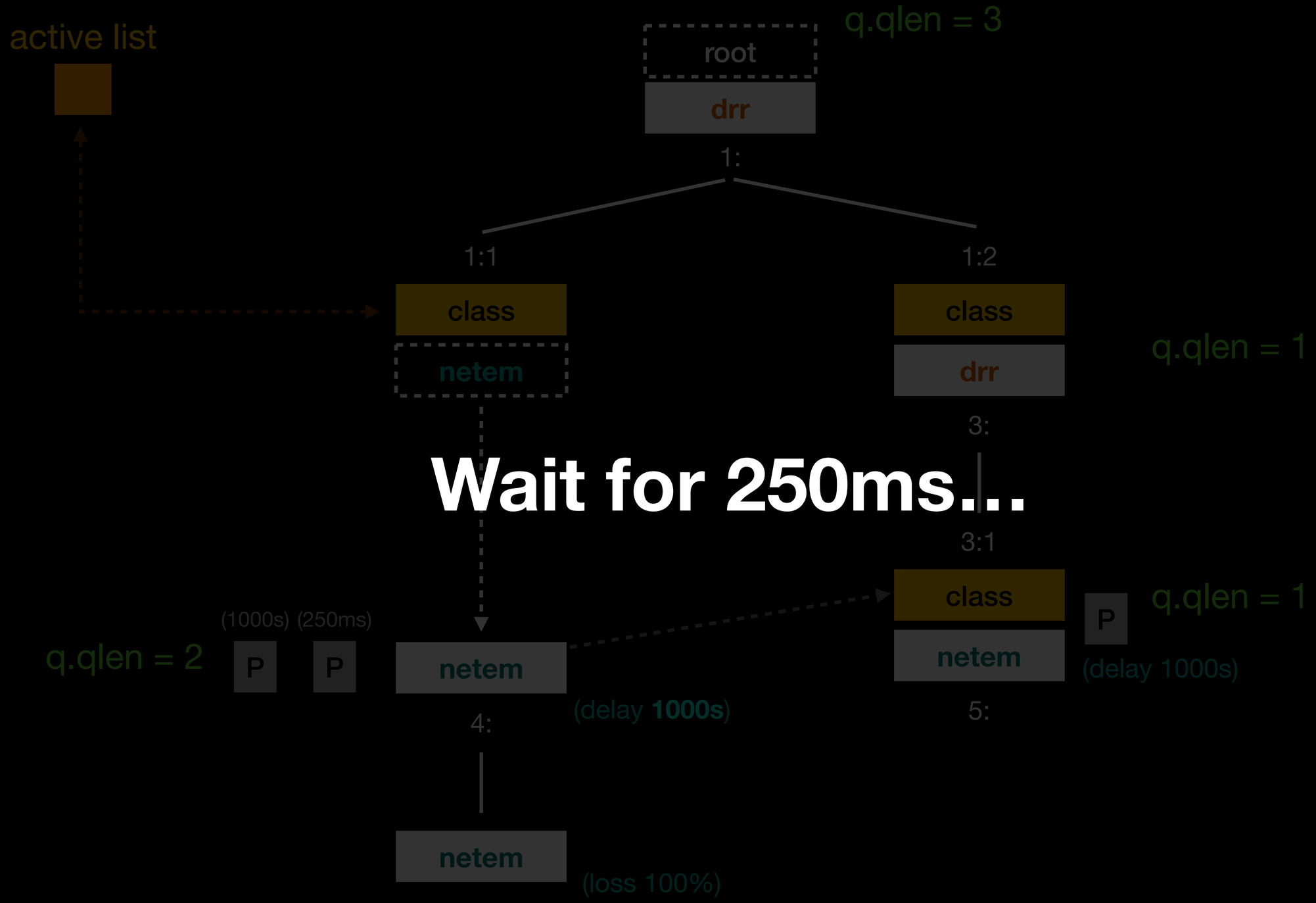

active list
root
drr
q.qlen = 1
1:
1:1
1:2
class
netem
class
drr
3:
3:1
class
netem
(delay 1000s)
5:
(250ms)
P
netem
(delay 1000s)
4:
netem
(loss 100%)
q.qlen = 1
tc qdisc change dev lo parent 1:1 netem delay 1000s limit 100

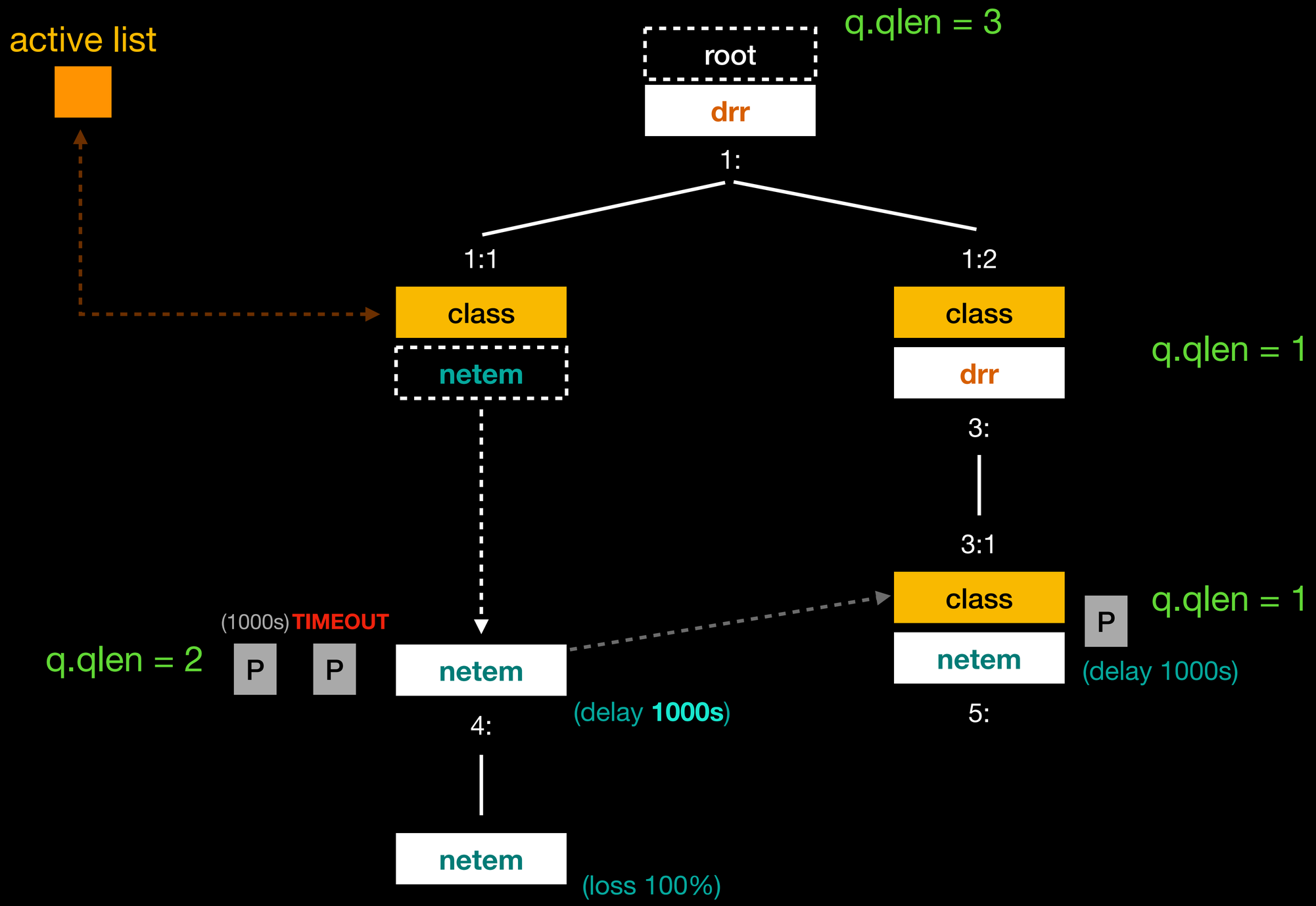

active list
2. Send a packet
P
root
drr
q.qlen = 1
1:
1:1
1:2
class
netem
class
drr
3:
3:1
class
netem
(delay 1000s)
5:
(250ms)
P
netem
(delay 1000s)
4:
netem
(loss 100%)
q.qlen = 1

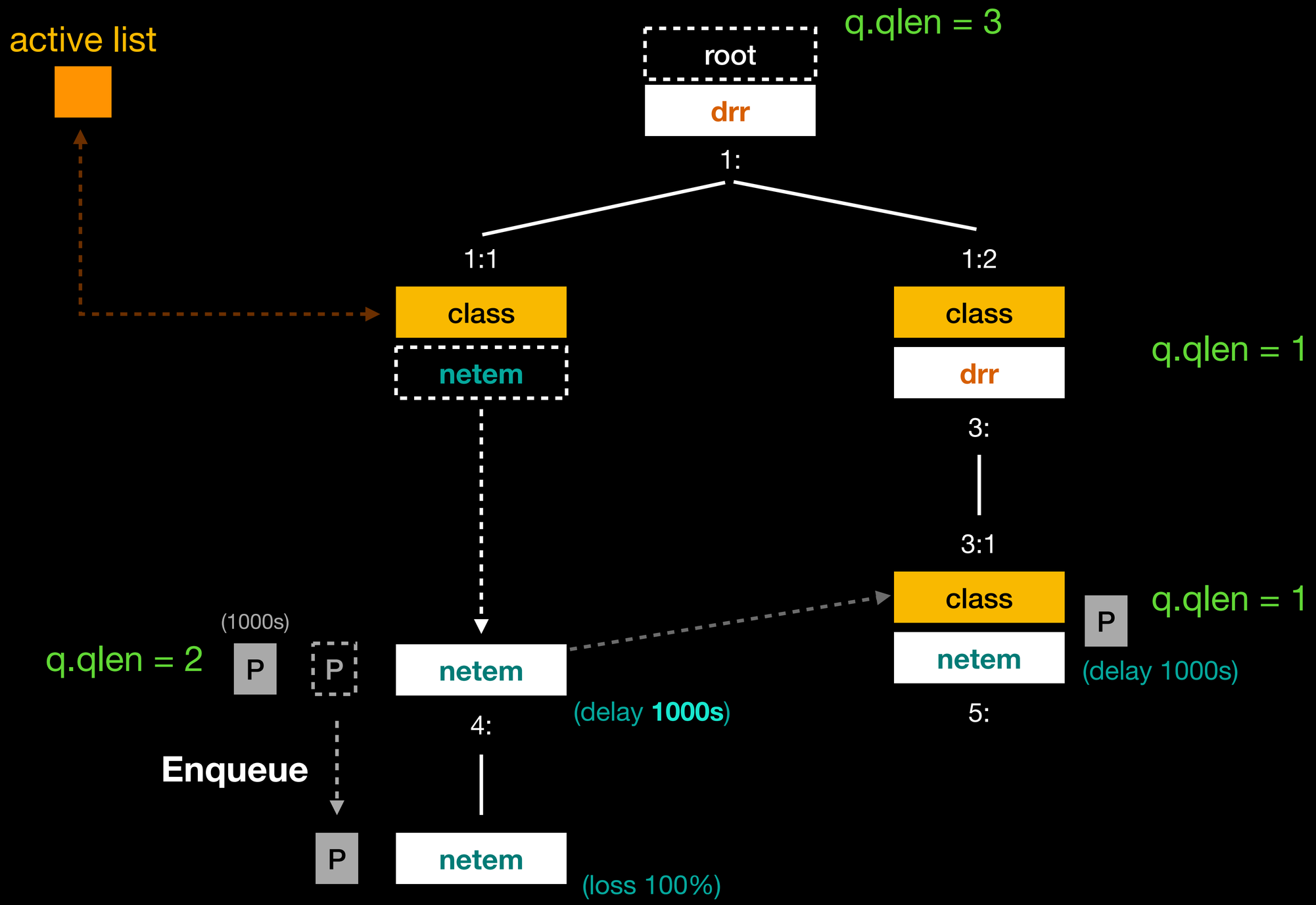

active list
P
root
drr
q.qlen = 2
1:
1:1
1:2
class
netem
class
drr
3:
3:1
class
netem
(delay 1000s)
5:
(1000s)
(250ms)
q.qlen = 2
P
P
netem
(delay 1000s)
4:
netem
(loss 100%)

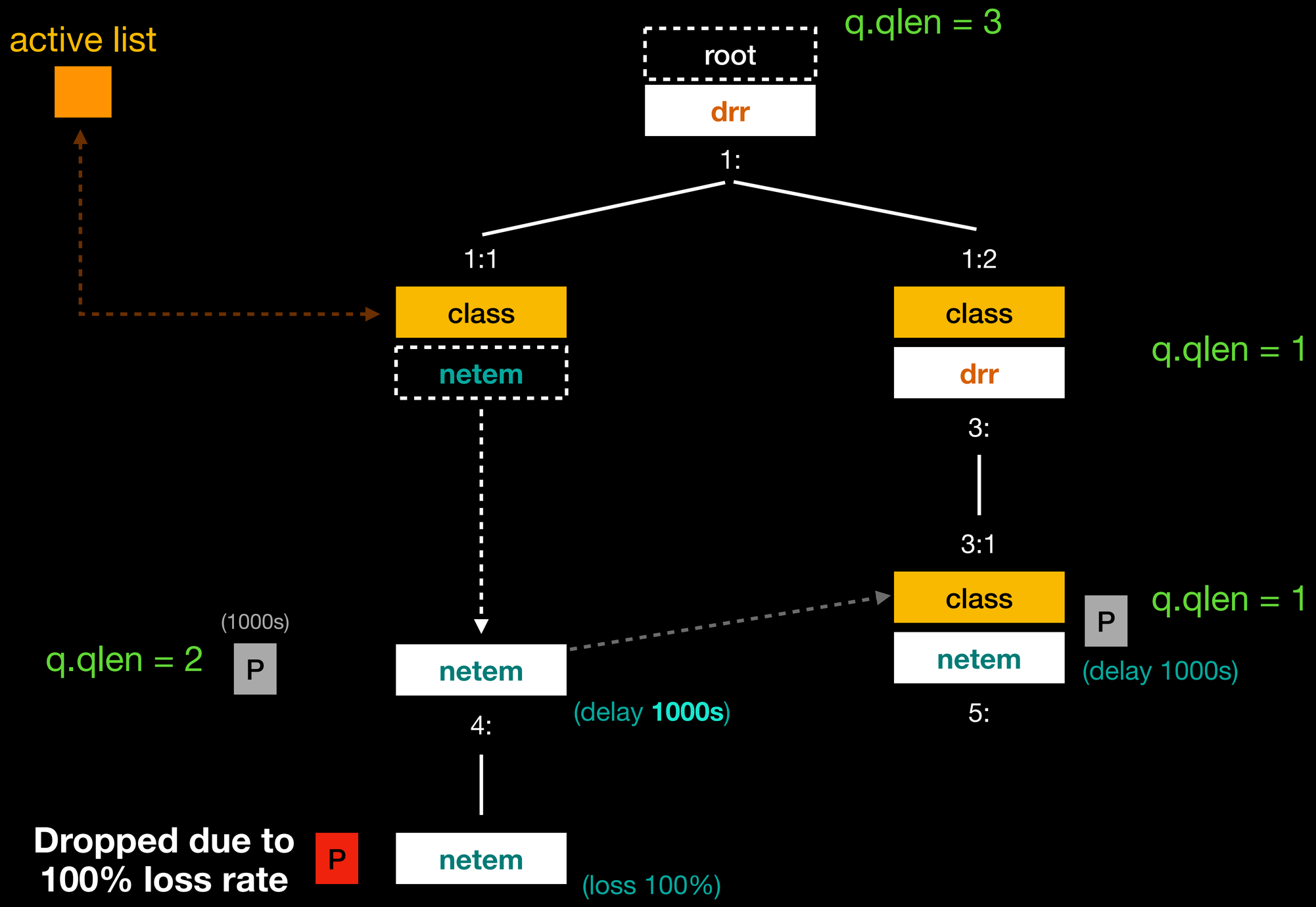

active list
P
3. Send a packet
root
drr
q.qlen = 2
1:
1:1
1:2
class
netem
class
drr
3:
3:1
class
netem
(delay 1000s)
5:
(1000s)
(250ms)
q.qlen = 2
P
P
netem
(delay 1000s)
4:
netem
(loss 100%)

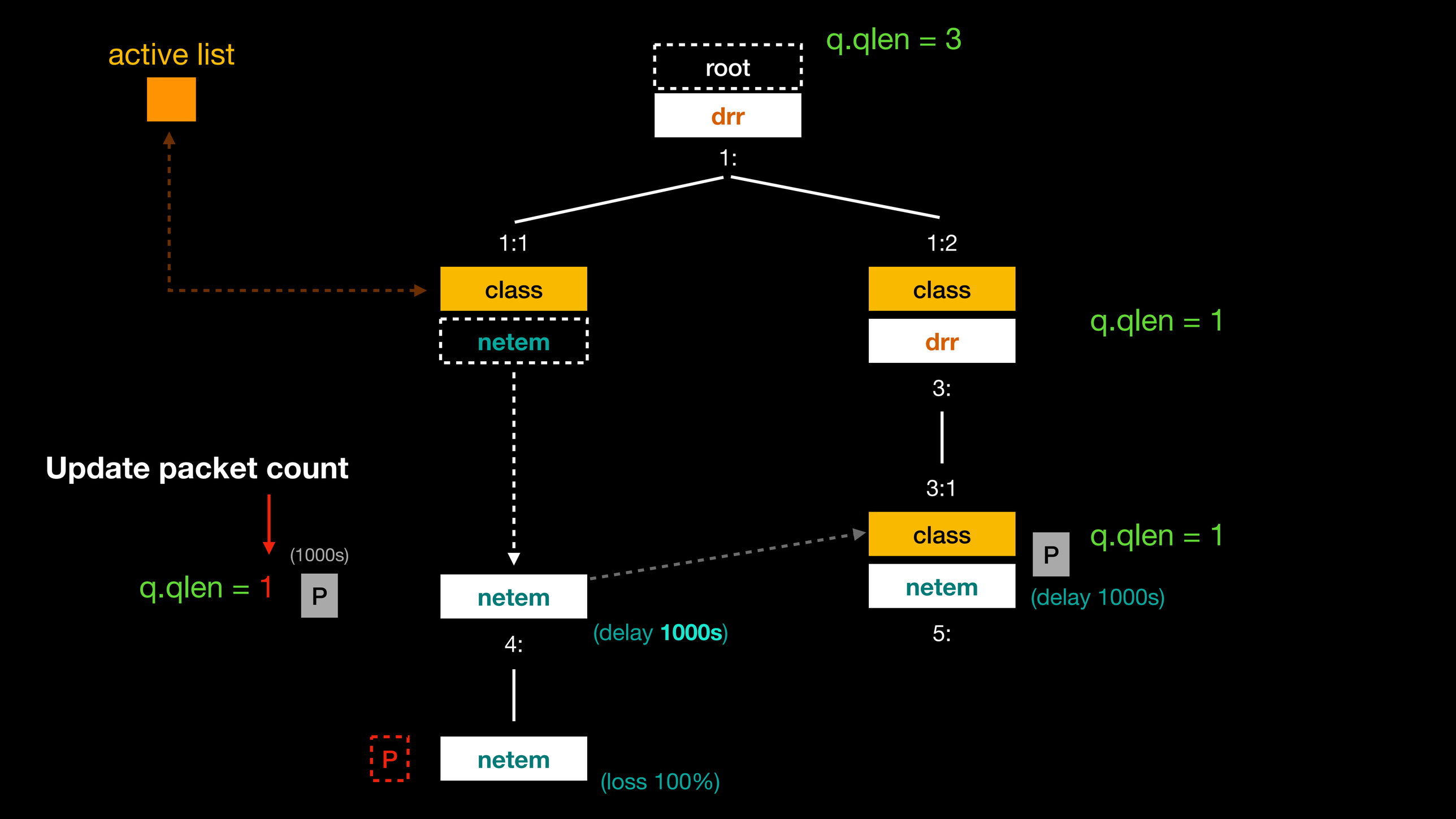

active list
P
root
drr
q.qlen = 3
1:
1:1
1:2
class
netem
class
drr
q.qlen = 1
3:
3:1
class
netem
P
q.qlen = 1
(delay 1000s)
(1000s)
(250ms)
q.qlen = 2
P
P
netem
(delay 1000s)
5:
4:
netem
(loss 100%)

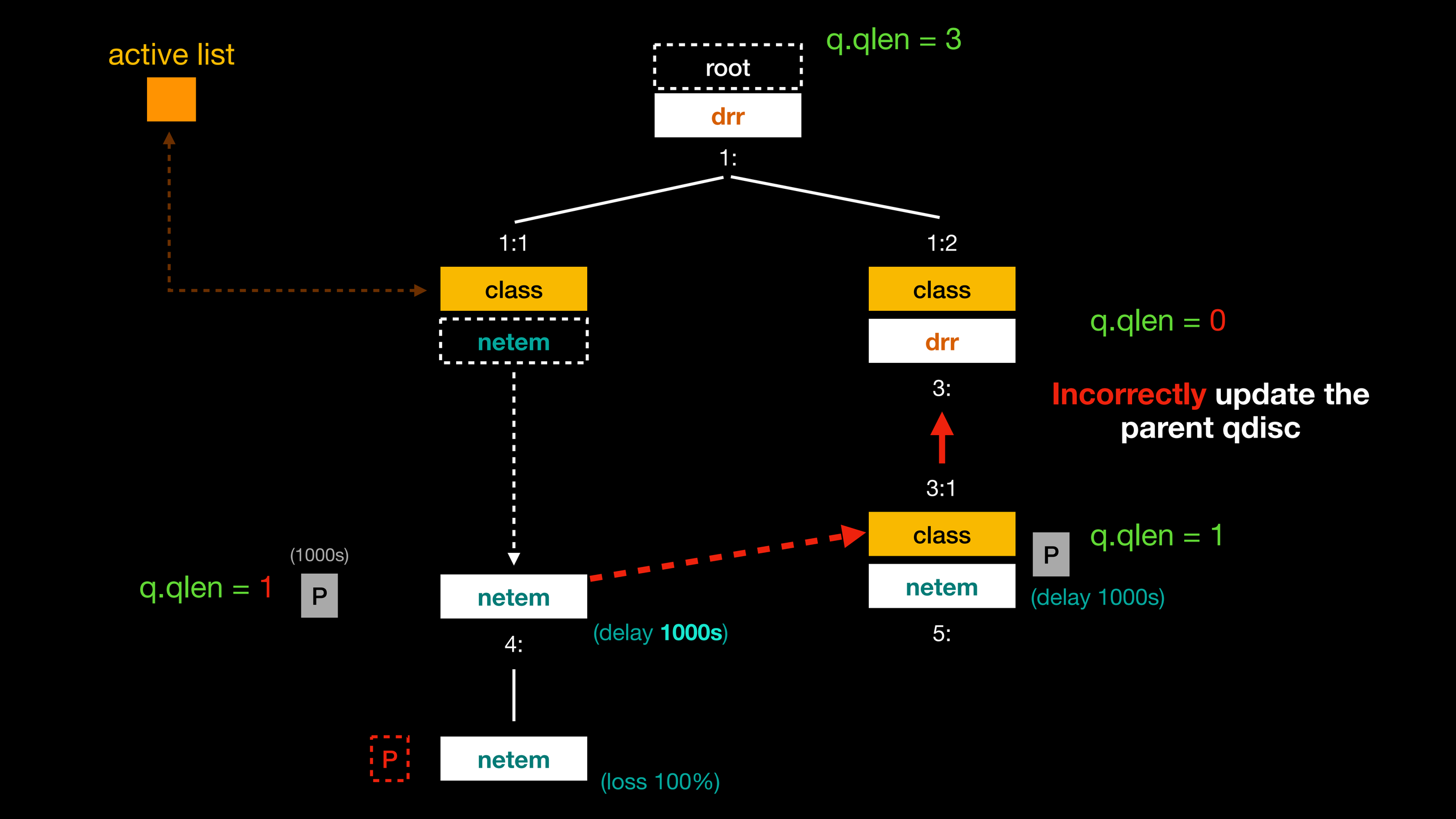

active list
root
q.qlen = 3
drr
1:
1:1
1:2
class
class
netem
drr
q.qlen = 1
3:
Wait for 250ms…
3:1
class
P
q.qlen = 1
(1000s) (250ms)
q.qlen = 2
P
P
netem
netem
(delay 1000s)
(delay 1000s)
4:
5:
netem
(loss 100%)

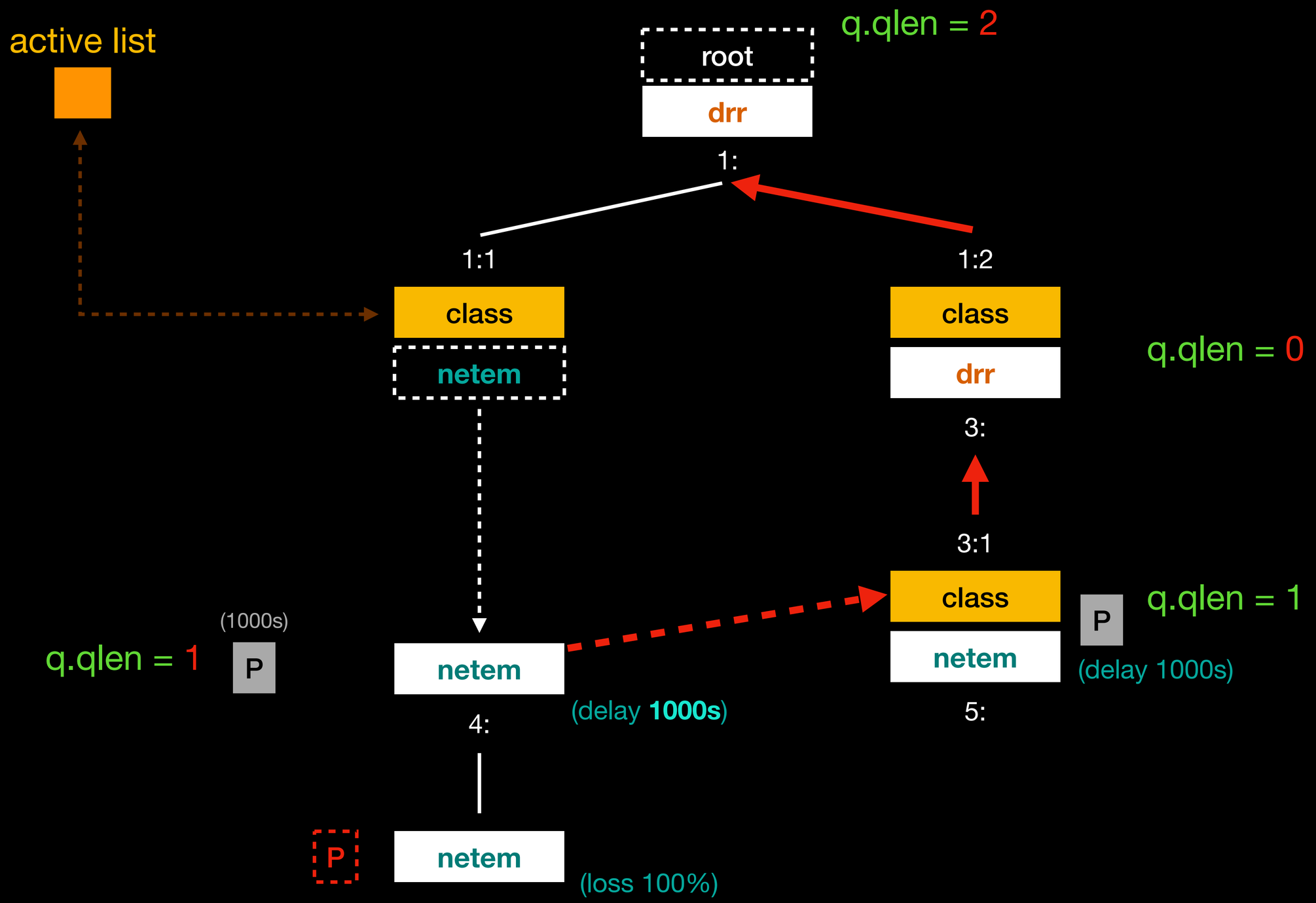

active list
root
drr
q.qlen = 3
1:
1:1
1:2
class
netem
class
drr
q.qlen = 1
3:
3:1
class
netem
P
q.qlen = 1
(delay 1000s)
(1000s) TIMEOUT
q.qlen = 2
P
P
netem
(delay 1000s)
5:
4:
netem
(loss 100%)

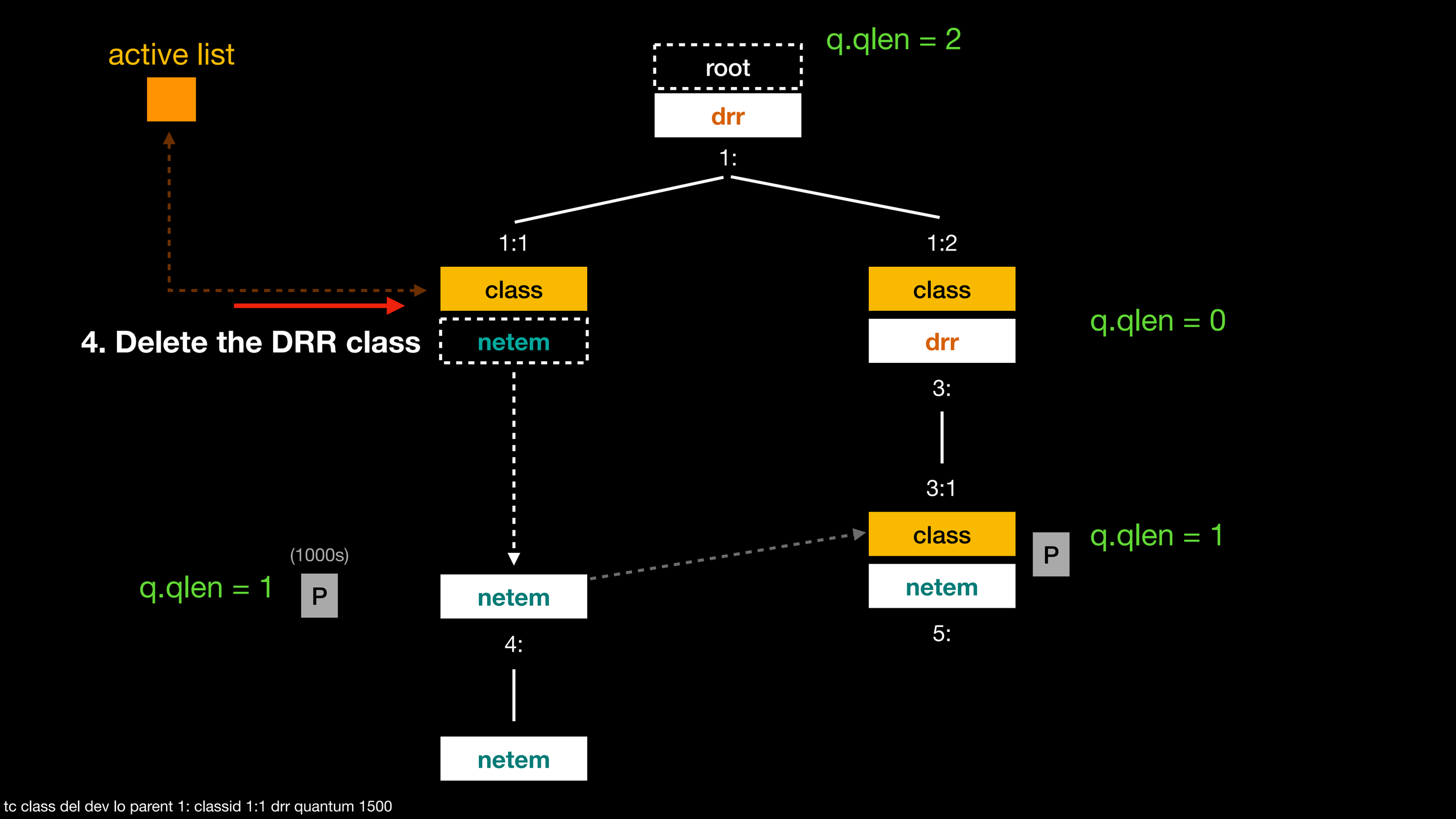

active list
root
drr
q.qlen = 3
1:
1:1
1:2
class
netem
class
drr
q.qlen = 1
3:
3:1
class
netem
P
q.qlen = 1
(delay 1000s)
(1000s)
q.qlen = 2
P
P
netem
(delay 1000s)
5:
4:
Enqueue
P
netem
(loss 100%)


active list
root
drr
q.qlen = 3
1:
1:1
1:2
class
netem
class
drr
q.qlen = 1
3:
3:1
class
netem
P
q.qlen = 1
(delay 1000s)
(1000s)
q.qlen = 2
P
netem
(delay 1000s)
4:
5:
Dropped due to
100% loss rate
P
netem
(loss 100%)

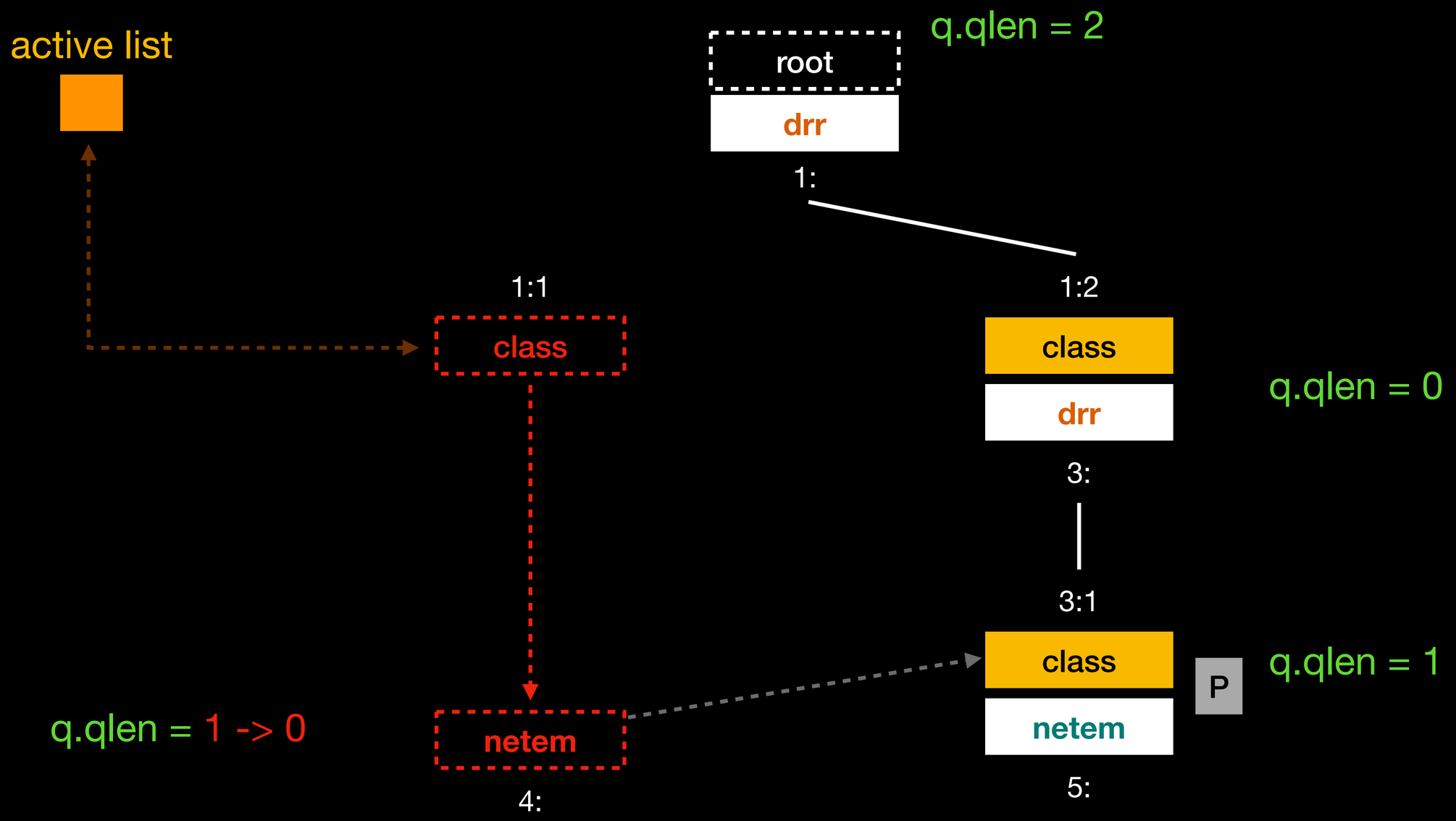

active list
root
drr
q.qlen = 3
1:
1:1
1:2
class
netem
class
drr
q.qlen = 1
3:
3:1
Update packet count
(1000s)
q.qlen = 1
P
netem
(delay 1000s)
class
netem
P
q.qlen = 1
(delay 1000s)
4:
5:
P
netem
(loss 100%)


active list
root
q.qlen = 3
drr
1:
1:1
1:2
class
netem
class
drr
q.qlen = 0
3:
Incorrectly update the parent qdisc
3:1
class
netem
P
q.qlen = 1
(delay 1000s)
(1000s)
q.qlen = 1
P
netem
(delay 1000s)
4:
5:
P
netem
(loss 100%)

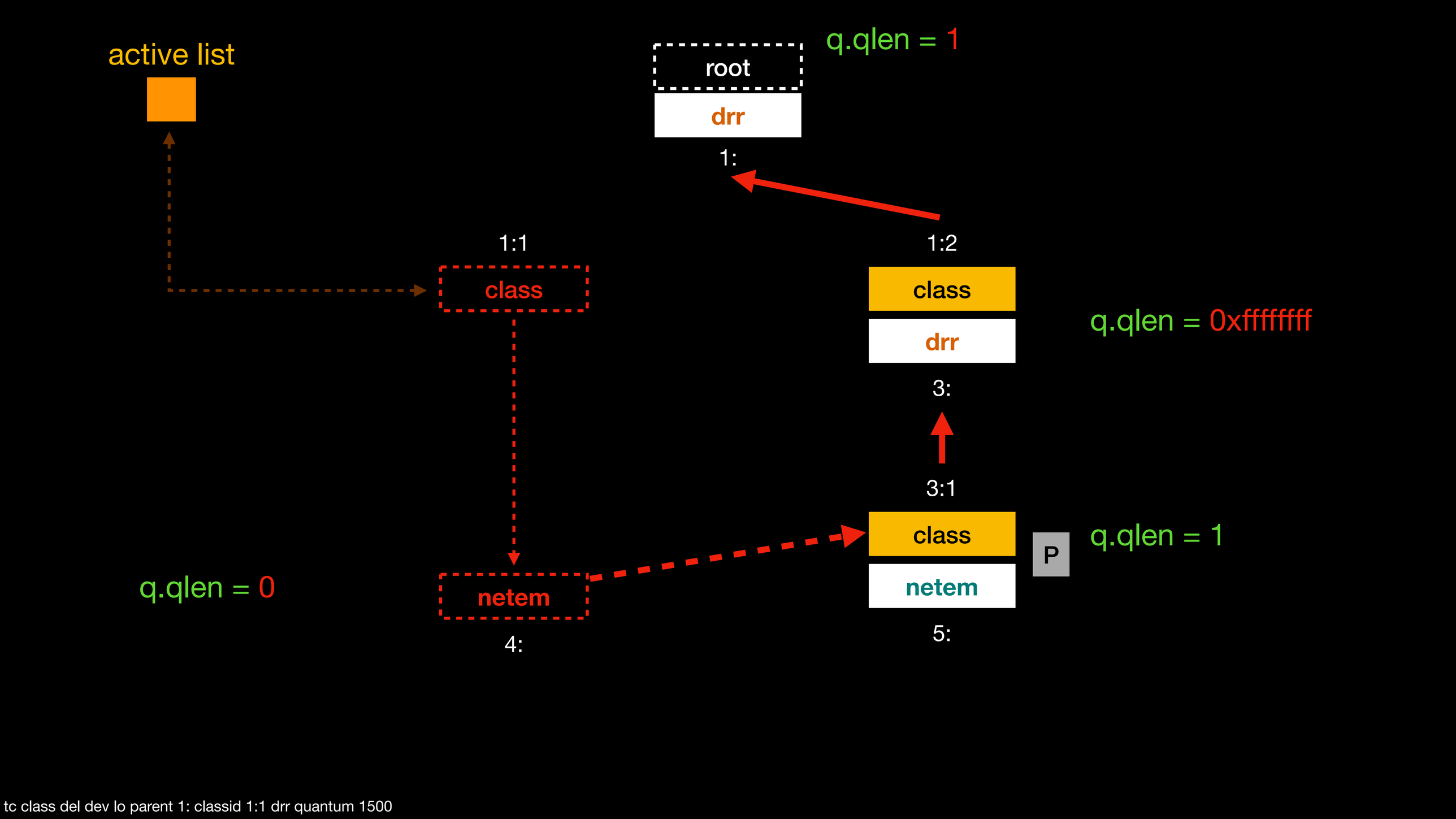

active list
root
drr
q.qlen = 2
1:
1:1
1:2
class
netem
class
drr
q.qlen = 0
3:
3:1
class
netem
P
q.qlen = 1
(delay 1000s)
(1000s)
q.qlen = 1
P
netem
(delay 1000s)
5:
4:
P
netem
(loss 100%)



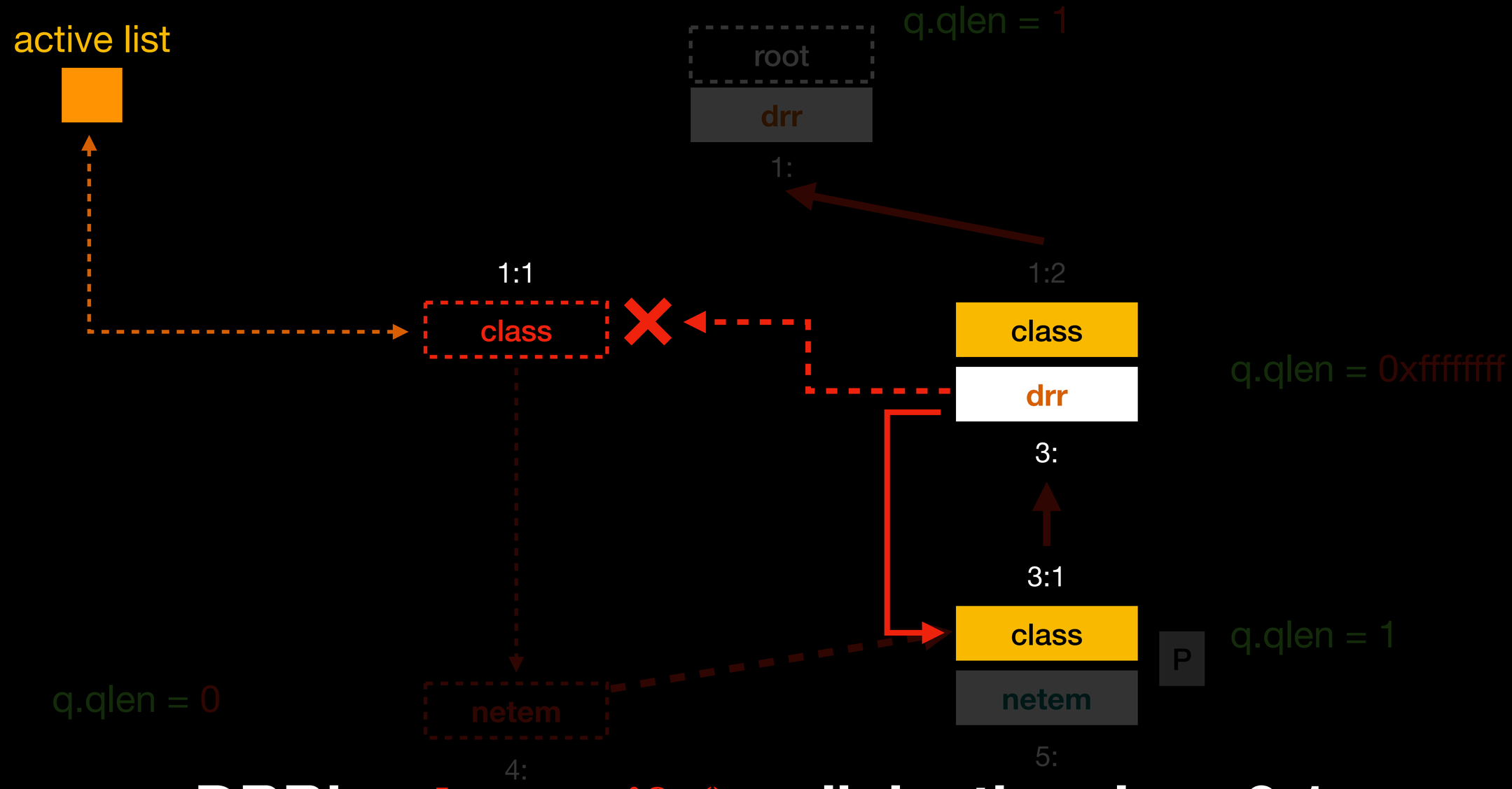
tc class del dev lo parent 1: classid 1:1 drr quantum 1500



tc class del dev lo parent 1: classid 1:1 drr quantum 1500



tc class del dev lo parent 1: classid 1:1 drr quantum 1500



DRR's .qlen_notify() unlinks the class 3:1 rather than the class 1:1

tc class del dev lo parent 1: classid 1:1 drr quantum 1500

$ CVE-2025-21700 RCA

- This vulnerability can be used to construct same DRR class UAF primitive
  - Reuse the exploit again — and again!
- In fact, almost all vulnerabilities causing packet count mismatch can be exploited this way

- Own two 0days, it’s time to race the next kernelCTF release!

$ Exploit Optimization

- Heap spraying is the heaviest bottleneck in our exploitation
  - ~2 GB of anonymous memory needed for successful heap guess
- To speed up, we need to find a user-controlled variable in kernel text

$ Exploit Optimization

- Existing Technique: kernfs_pr_cont_buf[4096]
  - Used by kernfs for formatted printk output
  - Can be controlled via:
    - iptables -A OUTPUT -m cgroup --path <payload>
- Limitations:
  1. Requires unprivileged user namespace
  2. Null bytes not allowed

$ User-Controlled Kernel Variable

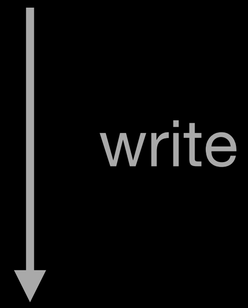
- Pseudo devices: /dev/random and /dev/urandom
  - Supply randomness from kernel entropy pools
- Writable by default
  - Written data will be stored in input_pool.hash.buf[64]
  - Can be used to store ROP gadget!

$ User-Controlled Kernel Variable

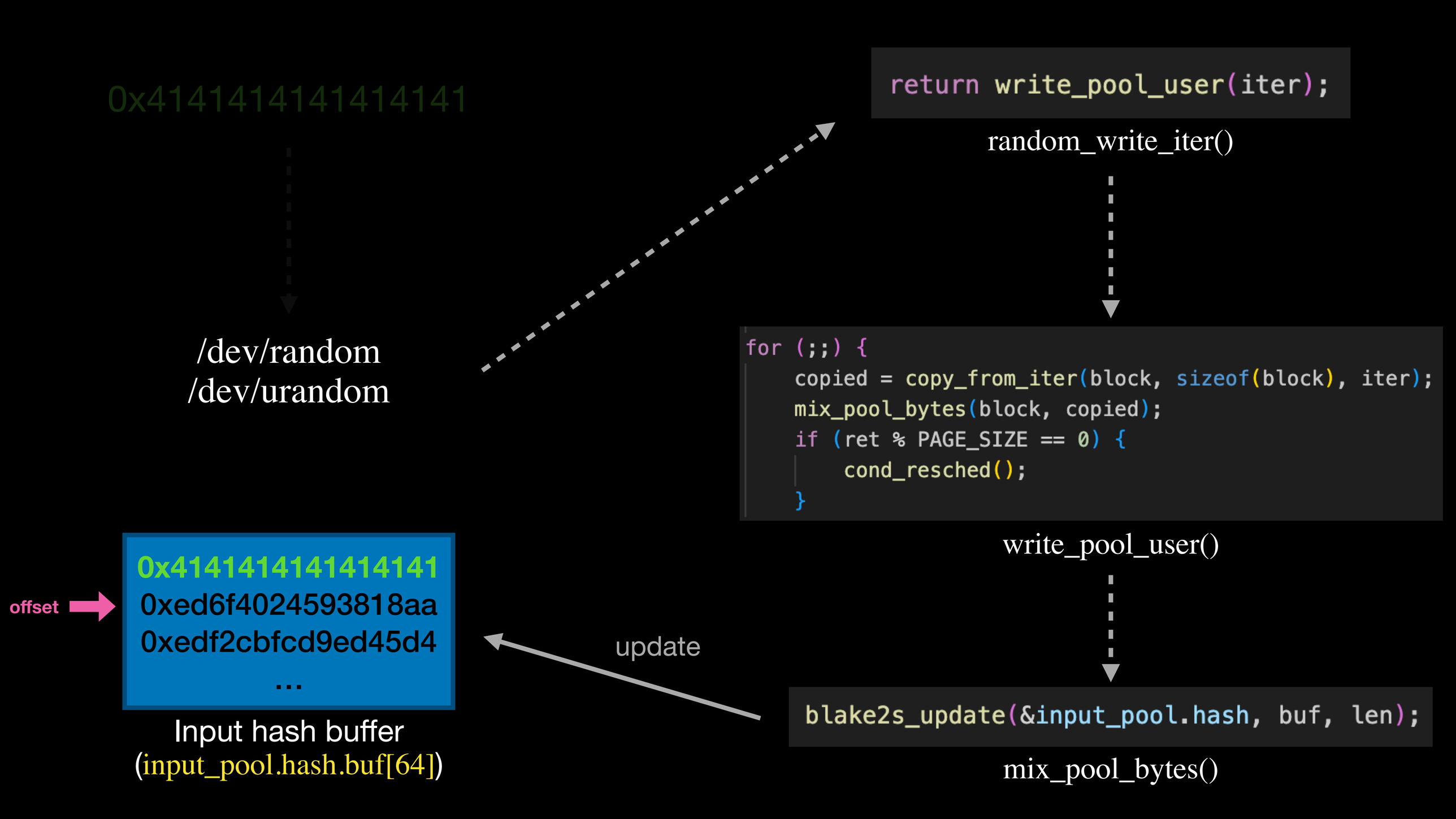
- Advantages:
  - Built-in devices → widely applicable
  - User-accessible → works in low-priv contexts
  - Fully controlled 64 bytes → null bytes allowed

➡ A more general exploit technique

0x4141414141414141



/dev/random
/dev/urandom


0x4141414141414141
/dev/random
/dev/urandom
return write_pool_user(iter);
random_write_iter()
for (;;) {
copied = copy_from_iter(block, sizeof(block), iter);
mix_pool_bytes(block, copied);
if (ret % PAGE_SIZE == 0) {
cond_resched();
}
write_pool_user()
offset
0x4141414141414141
0xed6f4024593818aa
0xedf2cbfcd9ed45d4
...
Input hash buffer
(input_pool.hash.buf[64])
update
blake2s_update(&input_pool.hash, buf, len);
mix_pool_bytes()

$ User-Controlled Kernel Variable

- Nothing is perfect…

  1. Timer overwrites buffer every second
     → May overwrite payload

  2. Kernel may modify buffer internally
     → May overwrite payload too

  3. Offset is unpredictable
     → Can’t craft fake objects or valid pointers reliably

$ User-Controlled Kernel Variable

- Nothing is perfect… but we have workarounds 👍

1. Timer overwrites buffer every second
   → Write payload right before exploit

2. Kernel may modify buffer internally
   → Use multiple threads to keep writing payload

3. Offset is unpredictable
   → Fork a process to reset the buffer state

DEMO
CVE-2025-21703 / LTS 6.6.69

```
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@l




ts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@l


ts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$
user@lts-6:/tmp$ i
```

The End of This kernelCTF Journey

kernelCTF submission form

Please make sure that you read the rules before filling out the form - even if you are familiar with the rules, the submission process may have been changed since the last time you read the page.

pumpkin@devco.re Switch account

* Indicates required question

Email *

Record pumpkin@devco.re as the email to be included with my response

Is this a 0-day or 1-day submission? *

- 0-day, initial stage report (not yet reported to security@kernel.org)
- 0-day, second stage report (after patch commit was merged)
- 1-day, vulnerability is already disclosed, but no patch yet
- 1-day, patch is available
- 1-day, novelty only submission (dupe - vulnerability was already exploited in kernelCTF before)

Flags *

In the format: kernelCTF{[<tag>:]<version>:<environment>:[<capabilities>:]<unix-timestamp>:<signature>}, one flag per line.

Please **do not edit this field after the original submission**. If you pwned another target since then, make a new submission (as eligibility depends on the original submission time, so the flags and the exploit hash cannot be modified after the initial submission).

Check the public spreadsheet and **make sure** that your **LTS/COS slots are not taken** - the 'environment' value(s) from the flag(s) should NOT be listed in the LTS/COS slot columns, otherwise your submission will be a dupe.

Your answer

Exploit hash *

SHA256 hash of the exploit (as a .tar.gz file) which captured the flag(s).

Keep the exploit small, leave out large, publicly available files (vmlinux, bzImage, etc).

If you are submitting multiple exploits for different targets, then put your exploits into separate folders, compress the folders and submit the hash of the compressed .tar.gz file.

Please **do not edit this field after the original submission**. If you pwned another target since then, make a new submission (as eligibility depends on the original submission time, so the flags and the exploit hash cannot be modified after the initial submission).

In hex format: [0-9a-f]{64}

Your answer

Notes

Any additional notes regarding the submission which cannot be added to the other fields.

This will only be read by the kernelCTF team and won't be published publicly.

Your answer

Next Page 1 of 3 Clear form

Never submit passwords through Google Forms.

This form was created inside of Google.com. Privacy & Terms

How to speed up the form submission process 🤔?

docs.google.com

kernelCTF submission form

Please make sure that you read the rules before filling out the form - even if you are familiar with the rules, the submission process may have been changed since the last time you read the page.

pumpkin@devco.re Switch account

* Indicates required question

Email *

Record pumpkin@devco.re as the email to be included with my response

Is this a 0-day or 1-day submission? *

- 0-day, initial stage report (not yet reported to security@kernel.org)
- 0-day, second stage report (after patch commit was merged)
- 1-day, vulnerability is already disclosed, but no patch yet
- 1-day, patch is available
- 1-day, novelty only submission (dupe - vulnerabilit… exploited in kernelCTF before)

Flags *

In the format: kernelCTF{<tag>:}<version>:<environme… <unix-timestamp>:<sig…ature>}, one flag per line.

Please **do not edi… field after the original submiss…** another target since t… make a new submission (as… on the original submission time, so the flags and the e… be modified after the initial submission).

Check the public spreadsheet and **make s…** that your… **not taken** - the 'environment' value(s) from the flag(s) s… listed in the LTS/COS slot columns, otherwise your sub… dupe.

kernelCTF{deprecated:v2:lts-6.6.69:userns:1753261095:6751e7bcbc7ba9579b23c95dc565726d0c7e5890}

Exploit hash *

SHA256 hash of the exploit (as a .tar.gz file) which captured the flag(s).

Keep the exploit small, leave out large, publicly available files (vmlinux, bzImage, etc).

If you are submitting multiple exploits for different targets, then put your exploits into separate folders, compress the folders and submit the hash of the compressed .tar.gz file.

Please **do not edit this field after the original submission**. If you pwned another target since then, make a new submission (as eligibility depends on the original submission time, so the flags and the exploit hash cannot be modified after the initial submission).

In hex format: [0-9a-f]{64}

e22a327df22357210fa4fd393f1ec3·

Notes

Any additional notes regarding the submission which cannot be added to the other fields.

This will only be read by the kernelCTF team and won't be published publicly.

Your answer

Next

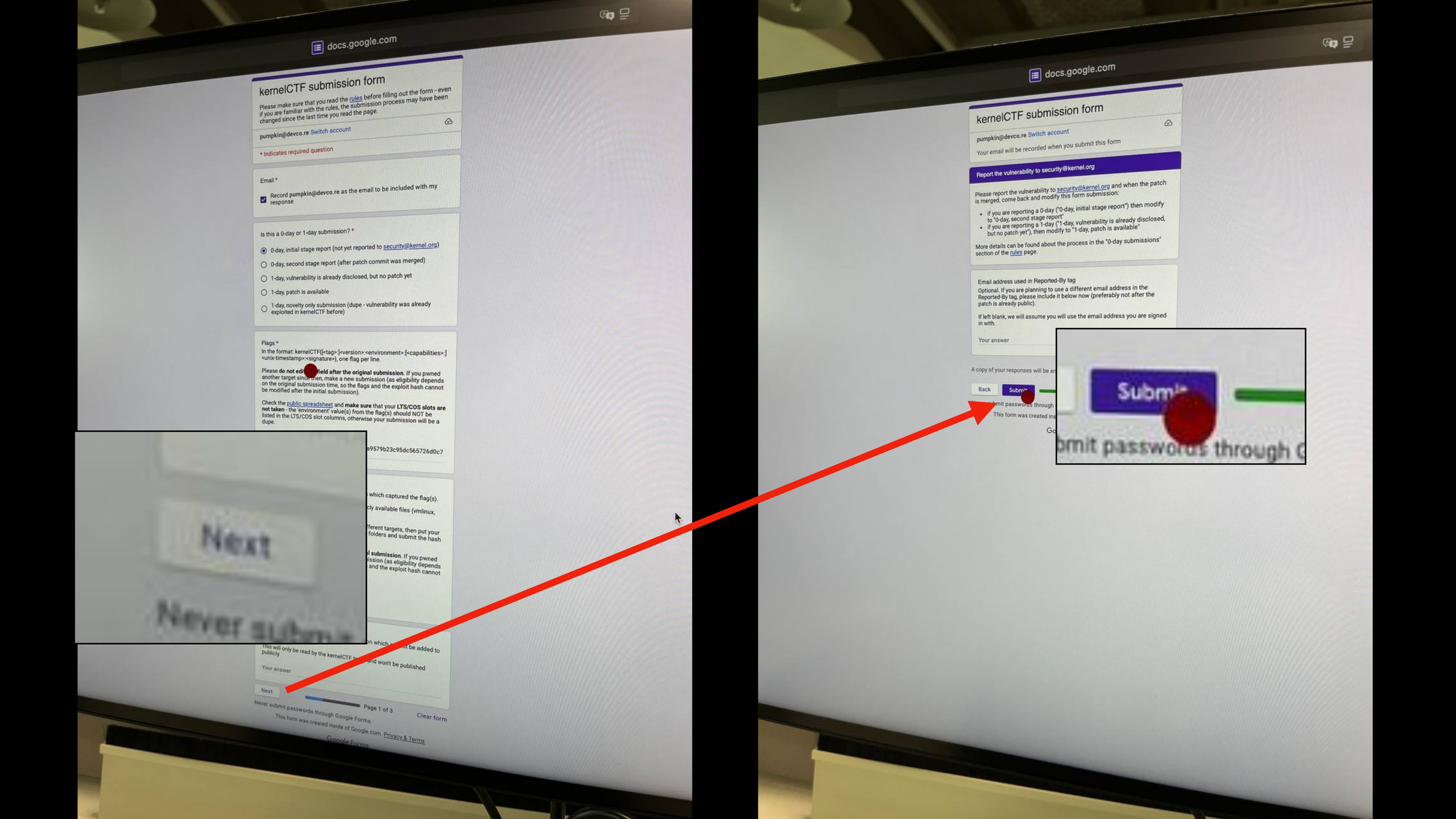
Page 1 of 3

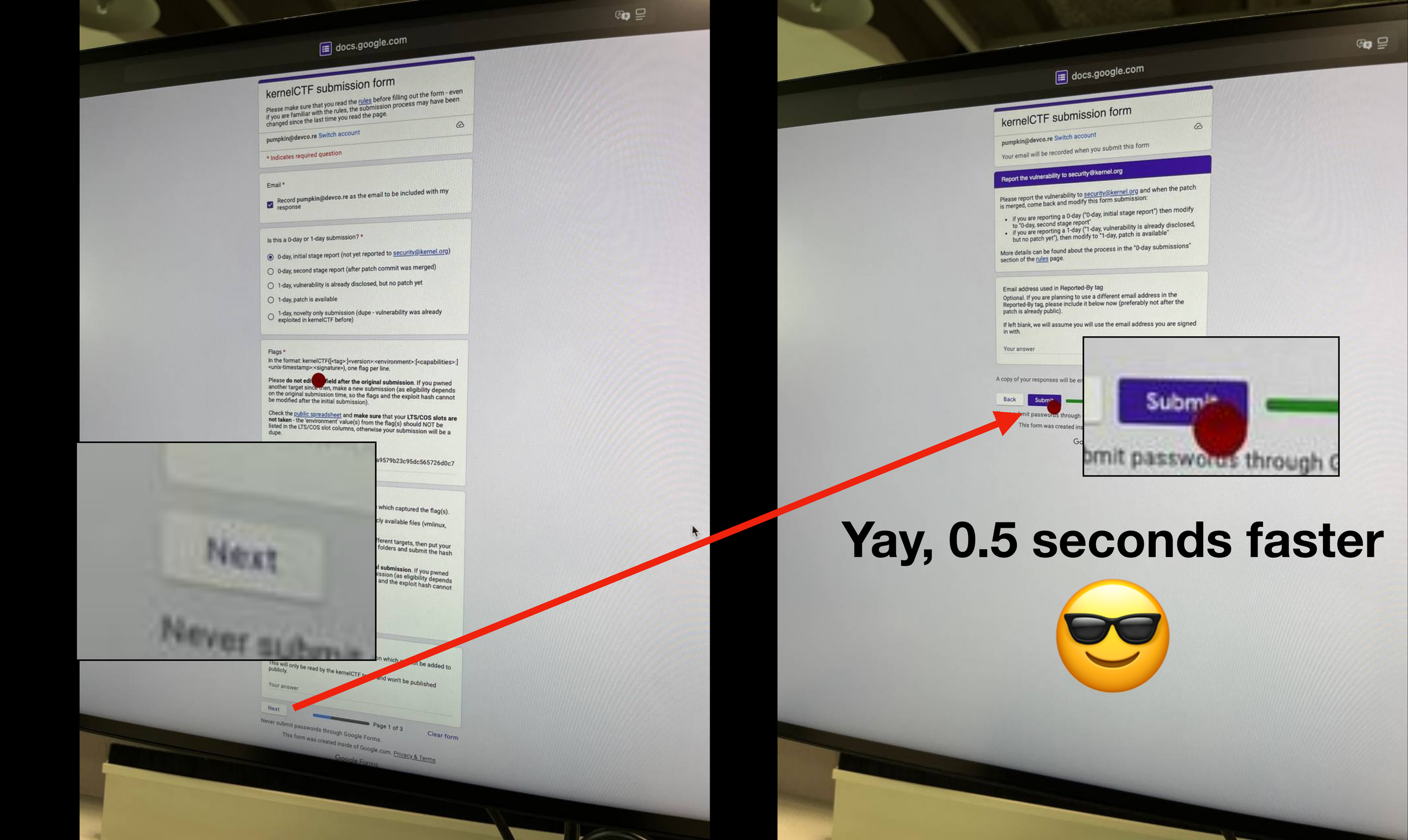
Clear form

Never submit passwords through Google Forms.

This form was created inside of Google.com. Privacy & Terms

Google Forms


docs.google.com
kernelCTF submission form
Please make sure that you read the rules before filling out the form - even if you are familiar with the rules, the submission process may have been changed since the last time you read the page.
pumpkin@devco.re Switch account
* Indicates required question
Email *
Record pumpkin@devco.re as the email to be included with my response
Is this a 0-day or 1-day submission? *
0-day, initial stage report (not yet reported to security@kernel.org)
0-day, second stage report (after patch commit was merged)
1-day, vulnerability is already disclosed, but no patch yet
1-day, patch is available
1-day, novelty only submission (dupe - vulnerability was already exploited in kernelCTF before)
Flags *
In the format: kernelCTF{<tag>:}<version>:<environment>:[<capabilities>:]<unix-timestamp>:<signature>}, one flag per line.
Please do not edit this field after the original submission. If you pwned another target since then, make a new submission (as eligibility depends on the original submission time, so the flags and the exploit hash cannot be modified after the initial submission).
Check the public spreadsheet and make sure that your LTS/COS slots are not taken - the 'environment' value(s) from the flag(s) should NOT be listed in the LTS/COS slot columns, otherwise your submission will be a dupe.
a9579b23c95dc565726d0c7
which captured the flag(s).
cly available files (vmlinux,
ferent targets, then put your folders and submit the hash
l submission. If you pwned ission (as eligibility depends and the exploit hash cannot
Next
Never
This will only be read by the kernelCTF t and won't be published publicly.
Your answer
Next
Page 1 of 3
Clear form
Never submit passwords through Google Forms.
This form was created inside of Google.com. Privacy & Terms
Google Forms
docs.google.com
kernelCTF submission form
pumpkin@devco.re Switch account
Your email will be recorded when you submit this form
Report the vulnerability to security@kernel.org
Please report the vulnerability to security@kernel.org and when the patch is merged, come back and modify this form submission:
if you are reporting a 0-day ("0-day, initial stage report") then modify to "0-day, second stage report"
if you are reporting a 1-day ("1-day, vulnerability is already disclosed, but no patch yet"), then modify to "1-day, patch is available"
More details can be found about the process in the "0-day submissions" section of the rules page.
Email address used in Reported-By tag
Optional. If you are planning to use a different email address in the Reported-By tag, please include it below now (preferably not after the patch is already public).
If left blank, we will assume you will use the email address you are signed in with.
Your answer
A copy of your responses will be er
Back
Submit
Submit
mit passwords through G


docs.google.com
kernelCTF submission form
Please make sure that you read the rules before filling out the form - even if you are familiar with the rules, the submission process may have been changed since the last time you read the page.
pumpkin@devco.re Switch account
* Indicates required question
Email *
Record pumpkin@devco.re as the email to be included with my response
Is this a 0-day or 1-day submission? *
0-day, initial stage report (not yet reported to security@kernel.org)
0-day, second stage report (after patch commit was merged)
1-day, vulnerability is already disclosed, but no patch yet
1-day, patch is available
1-day, novelty only submission (dupe - vulnerability was already exploited in kernelCTF before)
Flags *
In the format: kernelCTF{<tag>:}<version>:<environment>:[<capabilities>:]<unix-timestamp>:<signature>}, one flag per line.
Please do not edit field after the original submission. If you pwned another target since then, make a new submission (as eligibility depends on the original submission time, so the flags and the exploit hash cannot be modified after the initial submission).
Check the public spreadsheet and make sure that your LTS/COS slots are not taken - the 'environment' value(s) from the flag(s) should NOT be listed in the LTS/COS slot columns, otherwise your submission will be a dupe.
a9579b23c95dc565726d0c7
which captured the flag(s).
cly available files (vmlinux,
ferent targets, then put your folders and submit the hash
l submission. If you pwned ission (as eligibility depends and the exploit hash cannot
Next
Never submit
This will only be read by the kernelCTF t and won't be published publicly.
Your answer
Next
Page 1 of 3
Clear form
Never submit passwords through Google Forms.
This form was created inside of Google.com. Privacy & Terms
Google Forms
docs.google.com
kernelCTF submission form
pumpkin@devco.re Switch account
Your email will be recorded when you submit this form
Report the vulnerability to security@kernel.org
Please report the vulnerability to security@kernel.org and when the patch is merged, come back and modify this form submission:
if you are reporting a 0-day ("0-day, initial stage report") then modify to "0-day, second stage report"
if you are reporting a 1-day ("1-day, vulnerability is already disclosed, but no patch yet"), then modify to "1-day, patch is available"
More details can be found about the process in the "0-day submissions" section of the rules page.
Email address used in Reported-By tag
Optional. If you are planning to use a different email address in the Reported-By tag, please include it below now (preferably not after the patch is already public).
If left blank, we will assume you will use the email address you are signed in with.
Your answer
A copy of your responses will be er
Back
Submit
Submit
mit passwords through
This form was created ins
Go
mit passwords through
Yay, 0.5 seconds faster

$ Result

- CVE-2025-21700
  - Already exploited & listed in the sheet while I was writing the exploit
- CVE-2025-21703
  - Race for LTS 6.6.69 slot
    - exp230 (dupe)
    - Missed it by seconds 😭

$ Takeaways

- Reproducing 1-day exploits helps you understand the subsystem design and common bug patterns
  - This applies not only to kernelCTF
- Failures often lead to unexpected personal growth

$ Takeaways

- Reproducing 1-day exploits helps you understand the subsystem design and common bug patterns
  - This applies not only to kernelCTF
- Failures often lead to unexpected personal growth
  - No bounty, just wisdom 😭

DEVCORE

Thanks!

Pumpkin 🎃 (@u1f383)
https://u1f383.github.io/