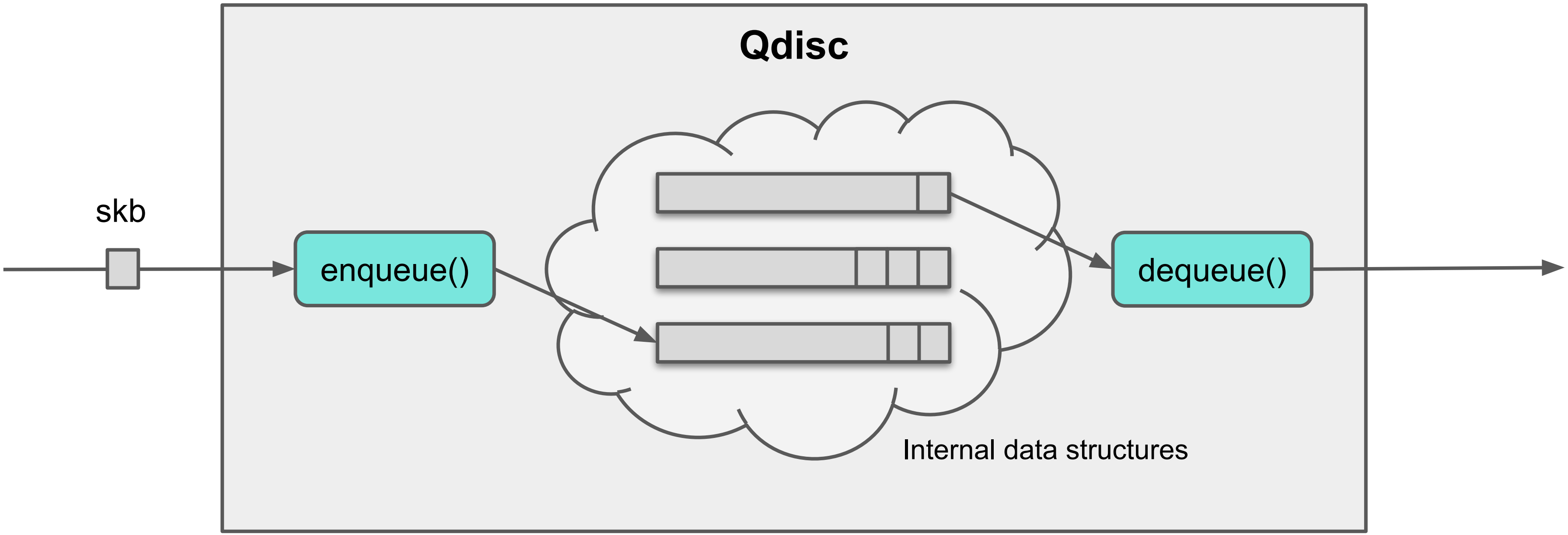
eBPF Qdisc:
A Generic Building Block for Traffic Control

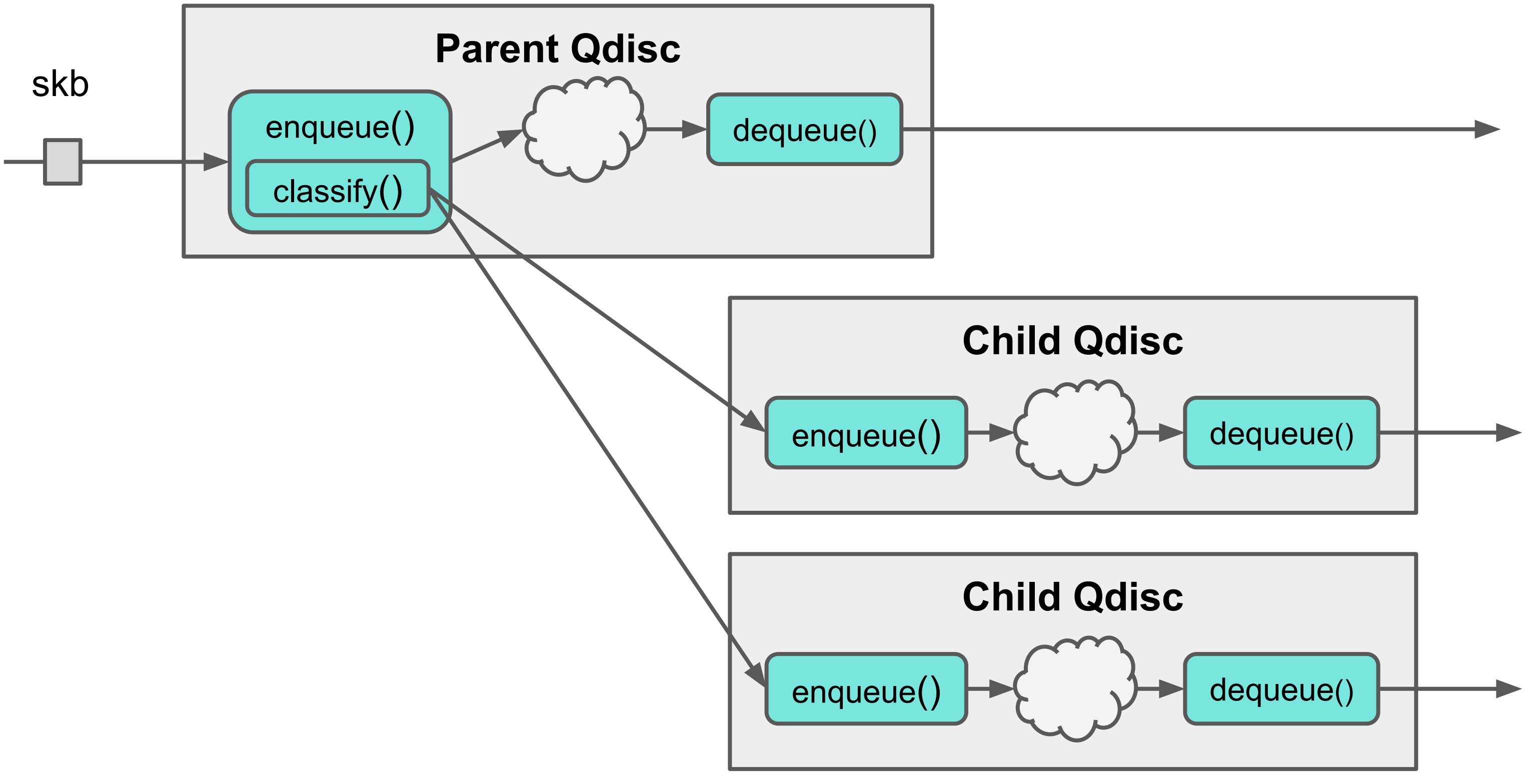
Amery Hung, Cong Wang

System Technologies Engineering, Bytedance

Packet scheduling in the Linux kernel



Classful Qdisc



Motivation

Qdiscs have fixed functionalities

Choosing the right Qdiscs and the combination are hard

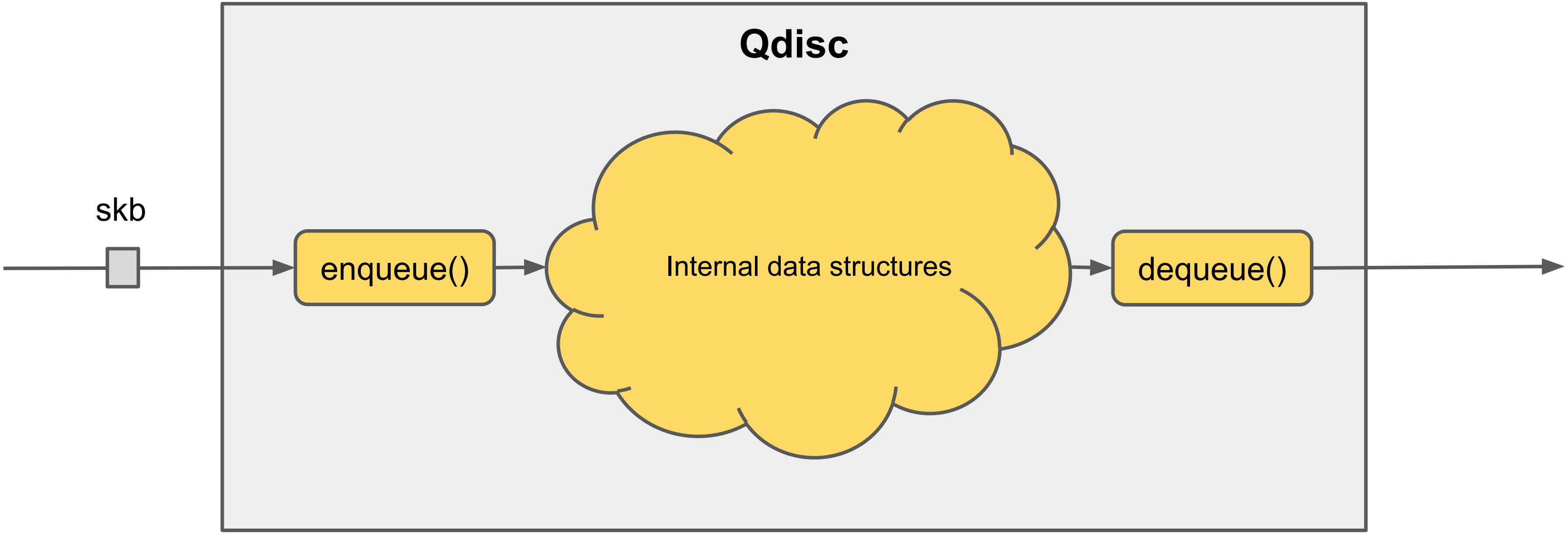
  fq, fq-codel, sfq, cake, choke, htb, hfsc, ...

Programmable packet scheduler designs

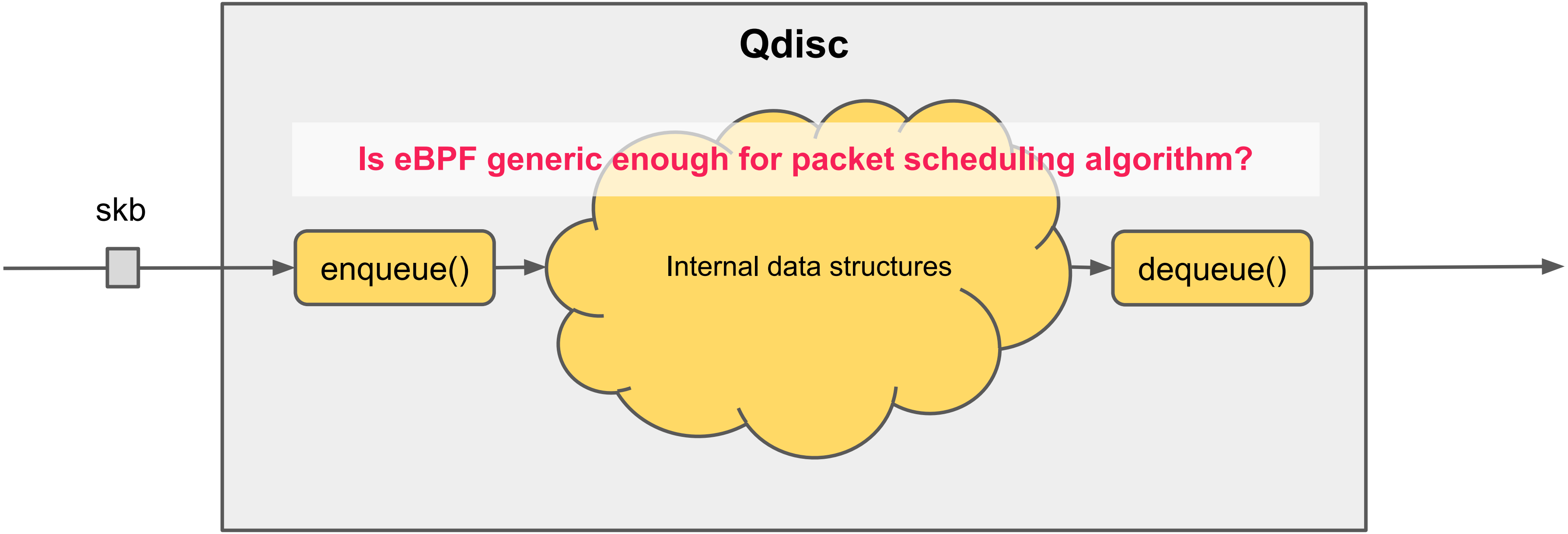
Pushed in first out (PIFO)

Eiffel

Idea: eBPF Qdisc



Idea: eBPF Qdisc



Is eBPF generic enough for packet scheduling algorithm?

New features in eBPF enable more generic programming

- Allocated objects, BPF linked lists
- BPF rbtree
- Shared allocated object ownership

Allocated objects

[PATCH bpf-next v6 00/26] Allocated objects, BPF linked lists - Kumar Kartikeya Dwivedi

```
void prog(void)
{
    struct foo *f;

    f = bpf_obj_new(typeof(*f));
    if (!f)
        return;

    bpf_obj_drop(f);
}
```

BPF linked lists

[PATCH bpf-next v6 00/26] Allocated objects, BPF linked lists - Kumar Kartikeya Dwivedi

```
void prog(void)
{
    struct foo *f;

    f = bpf_obj_new(typeof(*f));
    if (!f)
        return;

    bpf_list_push_front(head, &f->list);
    ...
```

Kfuncs: bpf_list_push/pop_front/back

BPF rbtree

[PATCH v5 bpf-next 0/9] BPF rbtree next-gen datastructure - Dave Marchevsky

```
static bool less(struct bpf_rb_node *a, struct bpf_rb_node *b)
{
    struct foo *foo_a = container_of(a, struct foo, node);
    struct foo *foo_a = container_of(a, struct foo, node);
    return foo_a->val < foo_b->val;
}

void prog(void)
{
    ...
    bpf_rbtree_add(head, &f->node, less);
    ...
```

Kfuncs: bpf_rbtree_add, bpf_rbtree_first, bpf_rbtree_remove

Shared allocated object ownership

[PATCH v2 bpf-next 0/9] Shared ownership for local kptrs - Dave Marchevsky

```
void prog(void)
{
    struct foo *f;

    f = bpf_obj_new(typeof(*f));
    if (!f)
        return;

    bpf_list_push_front(head, &f->list);
    bpf_rbtree_add(head, &f->node, less);
    ...
```

Design

- Programmability
  - Push-In-First-Out has serious limitations
  - Provide a mechanism, not a policy
  - Not enforce or imply any data structure

Design

- Flexibility and usability
  - Easy to use, we are not implementing a kernel module with eBPF
  - Focus on the core parts: enqueue and dequeue
  - It must work well with other TC components
    - Support TC filters and actions attached
    - Fit into any TC hierarchy

Design

struct_ops vs sch_bpf

struct Qdisc_class_ops {
select_queue, graft, leaf, qlen_notify, find,
change, delete, walk, tcf_block, bind_tcf,
unbind_tcf, dump, dump_stats
}

struct Qdisc_ops {
enqueue, dequeue, peek, init, reset,
destroy, change, attach,
change_tx_queue_len,
change_real_num_tx, dump, dump_stats,
ingress_block_set, egress_block_set,
ingress_block_get, egress_block_get
}

enqueue, dequeue

Design

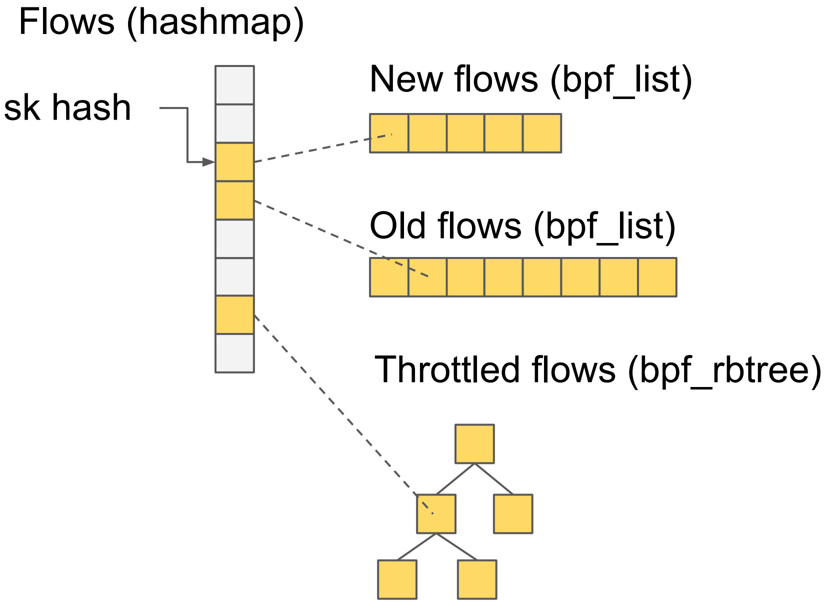
- Manage skb lifetime with kptr
- Reusing tc filters is possible through bpf_skb_tc_classify()
  - Like TC shared filter blocks
  - Some TC filters are complicated, e.g. cls_flower
- ABI compatibility

eBPF call context

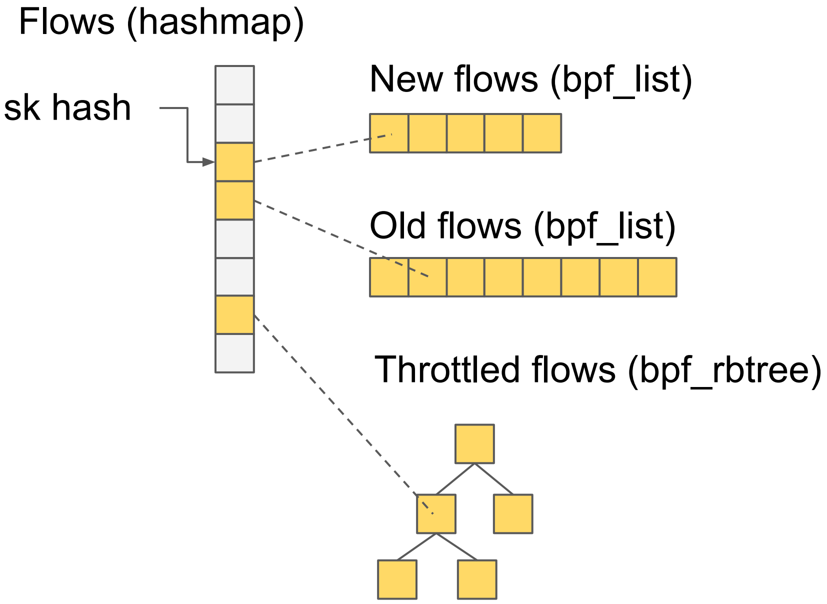
```
struct sch_bpf_ctx {
    struct sk_buff *skb;
    __u32 classid;
    __u64 expire;
    __u64 delta_ns;
}
```

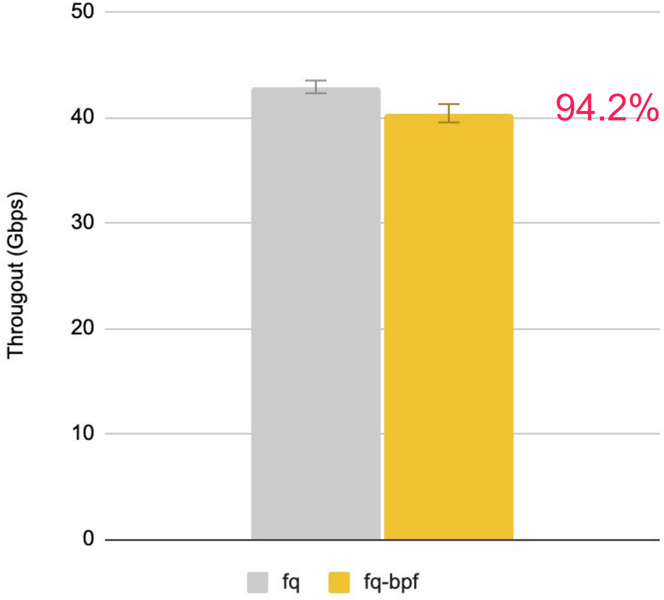
Program	Input	Output
enqueue	ctx->skb ctx->classid	SCH_BPF_THROTTLE: delay, delta_ns SCH_BPF_QUEUED SCH_BPF_DROP SCH_BPF_CN SCH_BPF_PASS: classid
dequeue	ctx->classid	SCH_BPF_THROTTLE: delay, delta_ns SCH_BPF_DEQUEUED: skb SCH_BPF_DROP SCH_BPF_CN SCH_BPF_PASS: classid

We can implement a sophisticated Qidsc using eBPF



We can implement a sophisticated Qidsc using eBPF

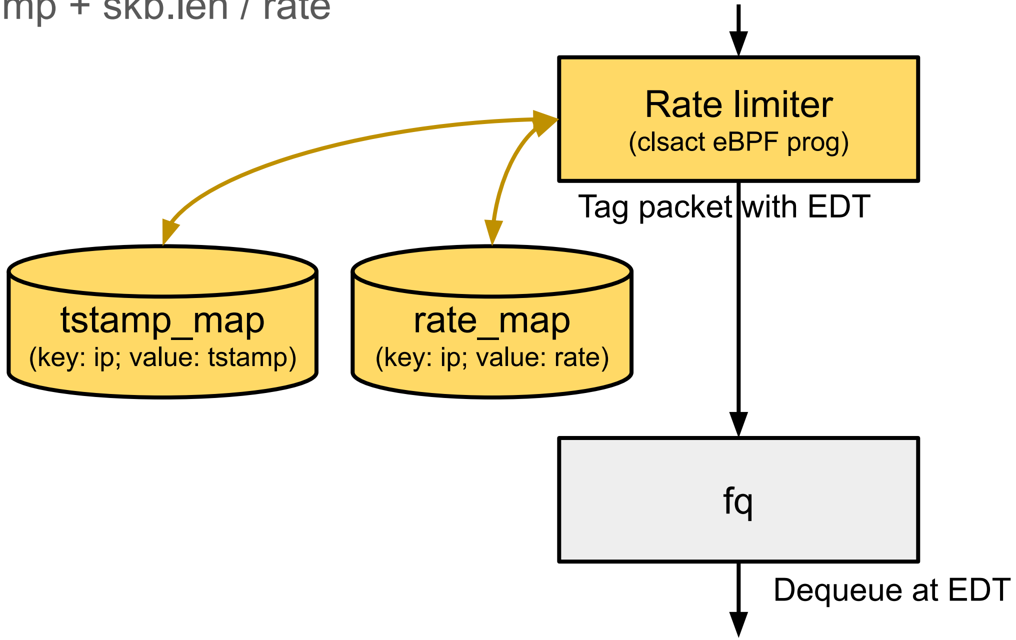




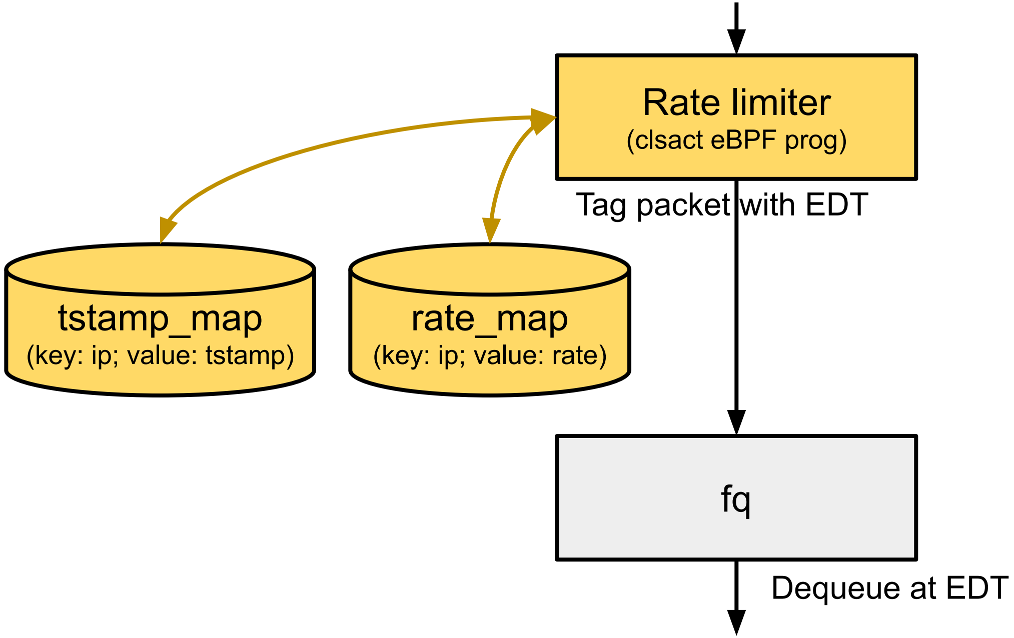
Two Use Cases

Use case 1: EDT + fq

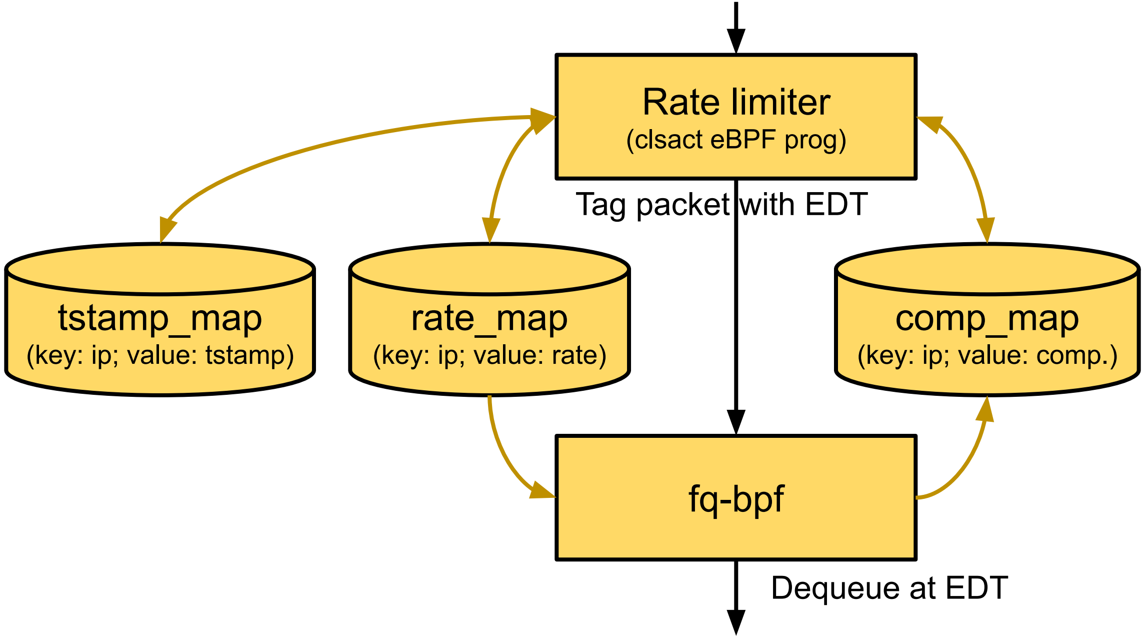
skb.tstamp = tstamp + skb.len / rate



Throughput loss as rate limiter is not aware of skb drops



Easy communication across components

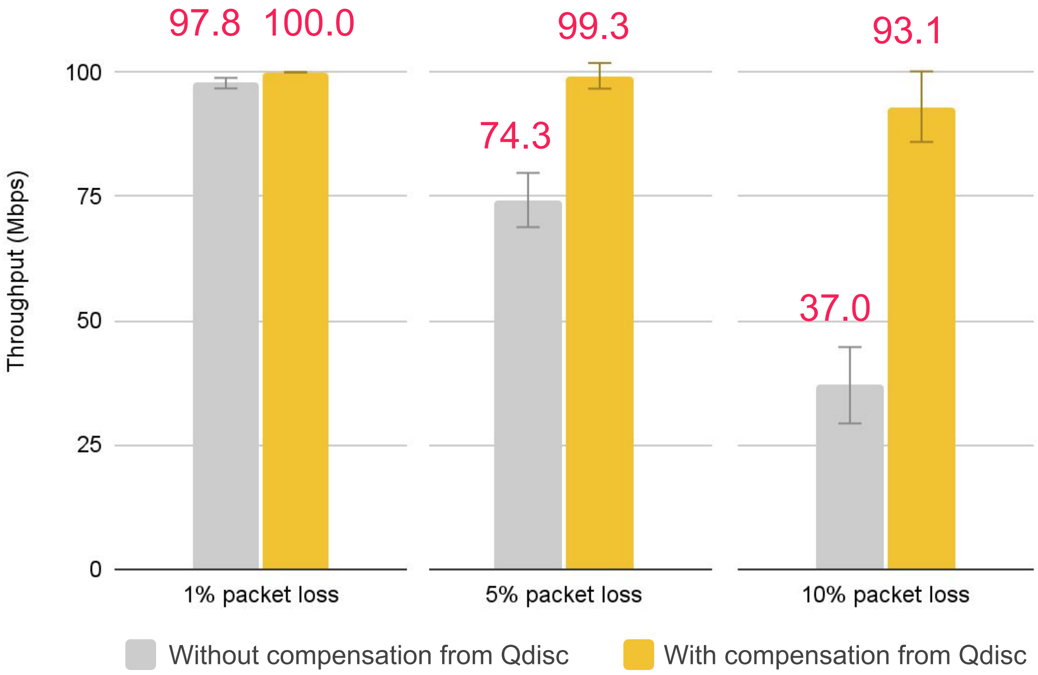


Adjust timestamp in rate limiter based on Qdisc skb-drops

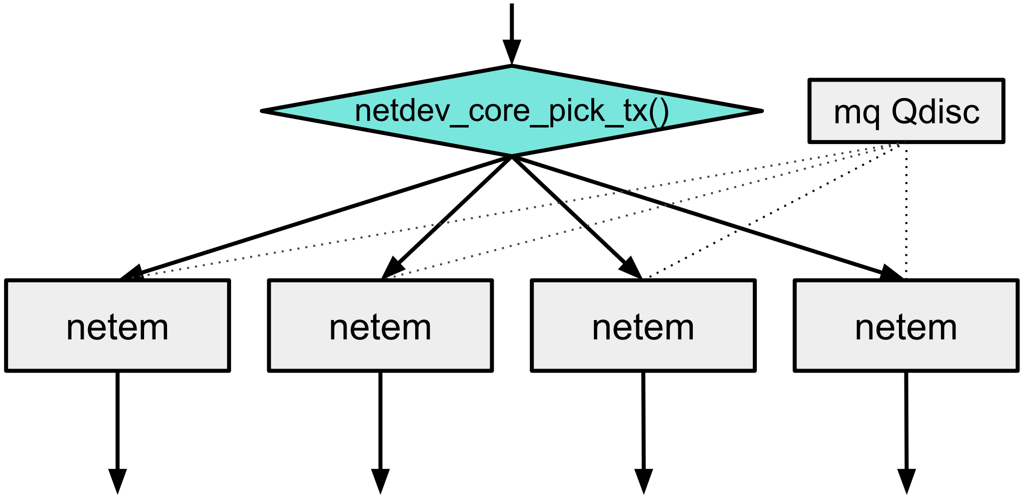
```
comp_ns = __sync_lock_test_and_set(comp, 0);
tstamp_comp = *tstamp - comp_ns;
if (tstamp_comp < now) {
    tstamp_new = tstamp_comp + delay_ns;
    if (tstamp_new < now) {
        __sync_fetch_and_add(comp, now - tstamp_new);
        __sync_lock_test_and_set(tstamp, now);
    } else {
        __sync_fetch_and_add(tstamp, delay_ns - comp_ns);
    }
    skb->tstamp = now;
    return TC_ACT_OK;
}

skb->tstamp = tstamp_comp;
__sync_fetch_and_add(tstamp, delay_ns - comp_ns);
return TC_ACT_OK;
```

Throughput drop reduced

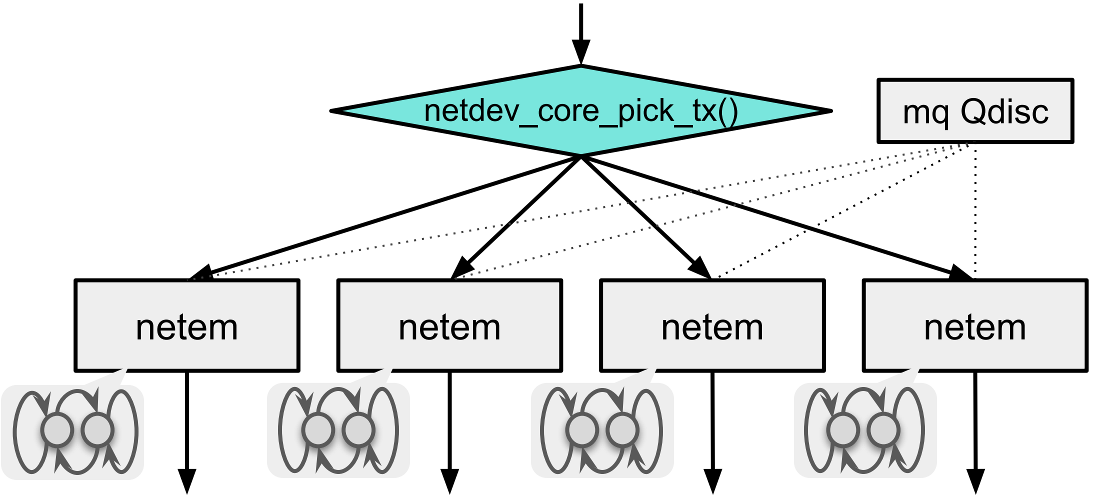
EDT: 100 Mbps



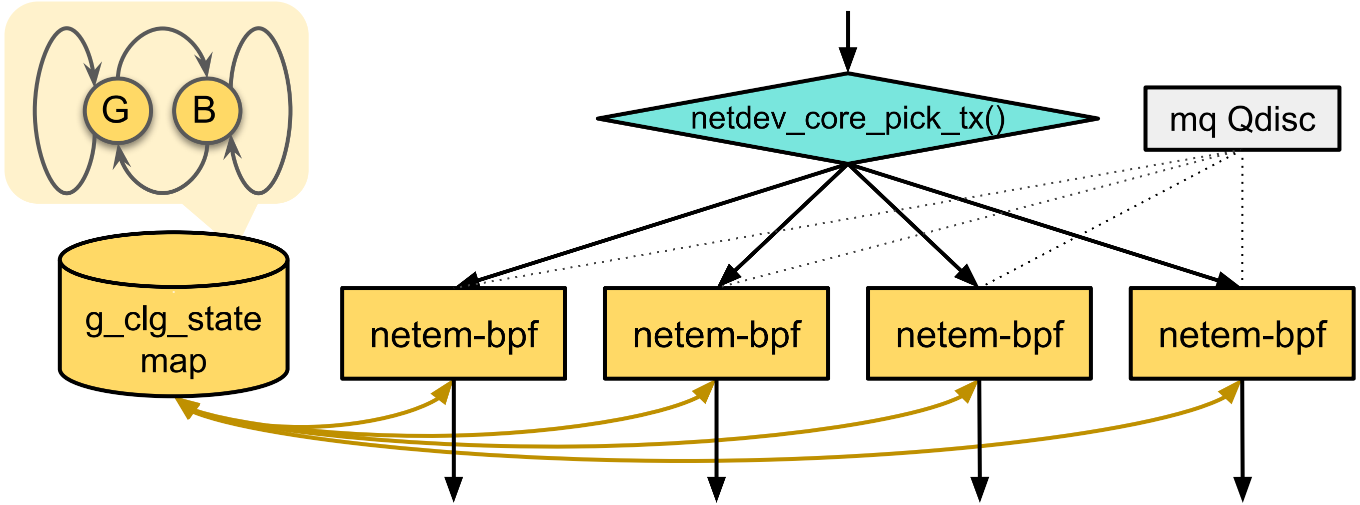
Use case 2: mq + netem



Discrepancy in network behavior caused by independent state machines

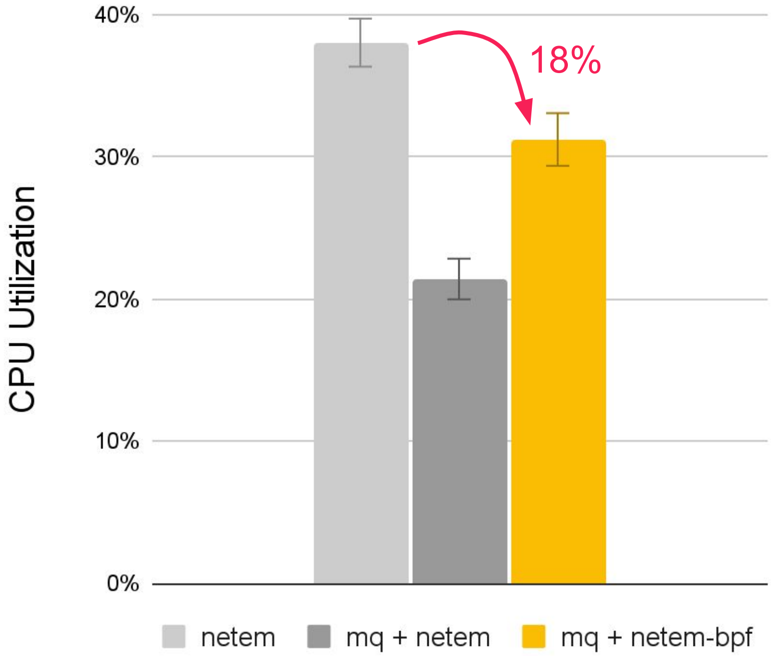


Collaboration between Qdiscs



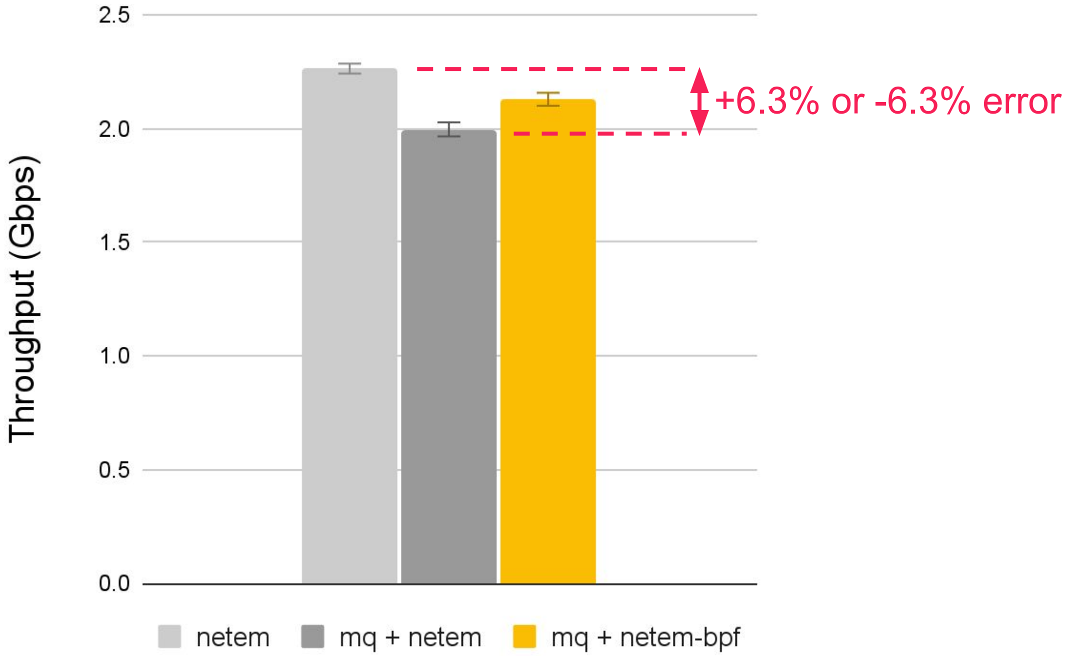
mq reduces contention on Qdisc locks

netem: loss gemodel 5% 95% 70% 0.1%
mq: 4 tx queues (virtio-net)
iperf3: 64 parallel flows



More accurate network behavior

netem: loss gemodel 5% 95% 70% 0.1%
mq: 4 tx queues (virtio-net)
iperf3: 64 parallel flows



Basic statistics

```
$ tc -s -d qdisc

qdisc mq 1: dev ens5 root
 Sent 16492097988 bytes 10895950 pkt (dropped 126999, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc bpf 802: dev ens5 parent 1:2 [Unknown qdisc, optlen=96]
 Sent 2388137376 bytes 1578056 pkt (dropped 19220, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc bpf 804: dev ens5 parent 1:4 [Unknown qdisc, optlen=96]
 Sent 534484773 bytes 353129 pkt (dropped 4661, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc bpf 801: dev ens5 parent 1:1 [Unknown qdisc, optlen=96]
 Sent 12885633183 bytes 8512929 pkt (dropped 97278, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc bpf 803: dev ens5 parent 1:3 [Unknown qdisc, optlen=96]
 Sent 683842656 bytes 451836 pkt (dropped 5840, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Things to be addressed

Storing kptr in local kptr

Egress only right now

Summary

- Recent eBPF developments make eBPF Qdisc possible as a generic and easy-to-use packet scheduler
- Using eBPF Qdisc with mq to avoid contention on the root Qdisc lock while being able to coordinate between queues
- eBPF Qdisc unlocks opportunities for new applications and optimizations

Sample code: https://github.com/ameryhung/ebpf-qdisc-examples