# Metering in OVS datapath

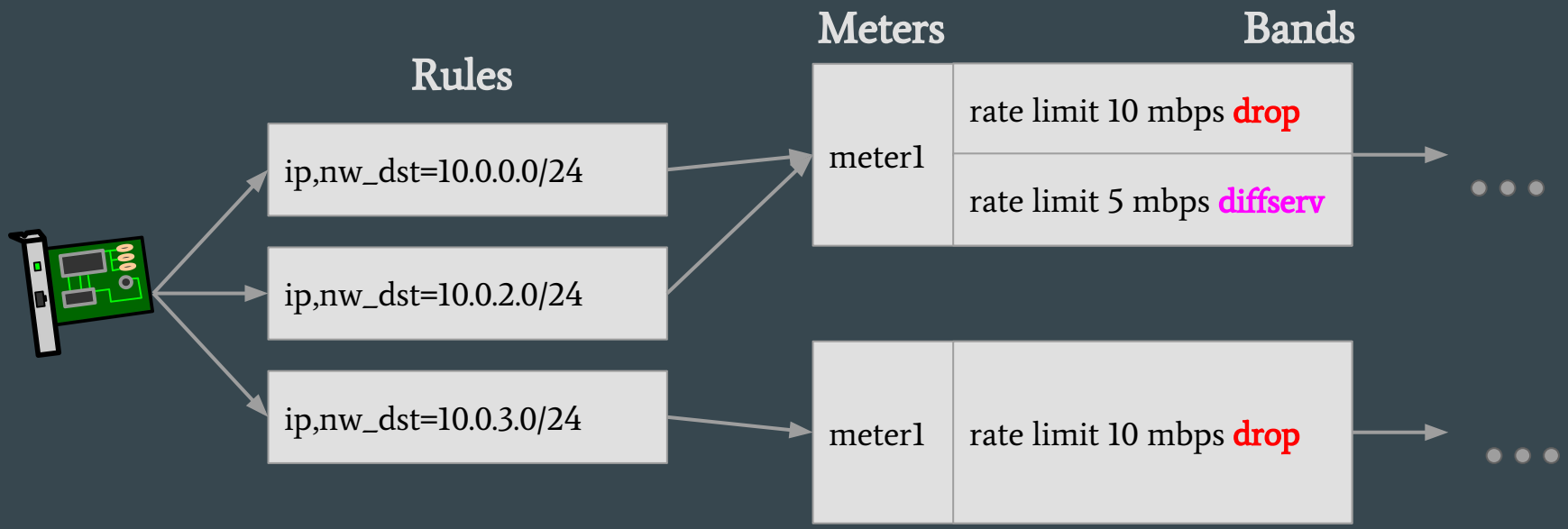
...

Joe Stringer, Andy Zhou
VMware

# Use cases

- Logging through userspace
  - ovs-dpctl add-flow <match> actions=userspace(...)
  - Every packet that hits this flow will send through the nl sockets
- Throttle upcalls
  - Mitigate potential DoS vector for default upcalls
- ISP-style max speed
  - Physical
- OpenFlow 1.3+ meter table
  - Diffserv

# Initial look



# Don’t qdiscs provide this?

- Qdiscs are device-centric
- OVS allows arbitrary ordering of actions, so how to co-ordinate going out to a device and back to continue processing pipeline?
  - Similar to ct()?
- Need configuration/setup/teardown separate from packet-processing pipeline
- Share meter across multiple execution paths (flows in OVS)

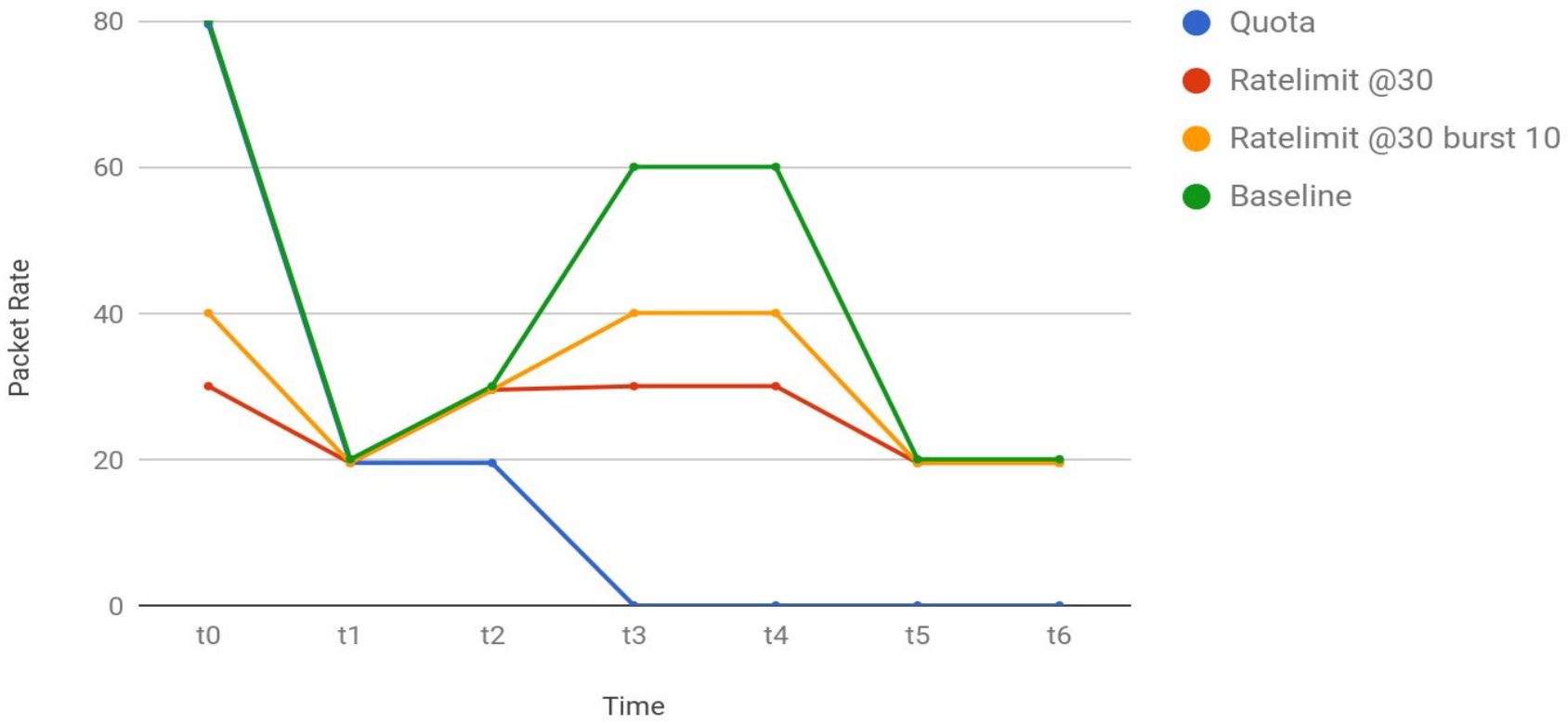
# Floating idea: generic ratelimit module

- # ip ratelimit add foo pkts=100 # per second
- # ovs-dpctl add-flow "in_port(p1),eth(),..." "ratelimit(foo),output(p0)"
- # ovs-dpctl add-flow “in_port(p1),eth(),...” \
  “sample(sample=100.0%,ratelimit(foo),actions(userspace(...)),output(p0))”
- With named ratelimiter, other subsystems could then link in

# Reuse existing infrastructure?

- Nothing provides multi-band metering
- TC filters
  - Ingress/egress filters work fine for rx/tx on device
  - Not structured for referencing from middle of OVS actions list
- Netfilter named objects?
  - Looks similar to the “ip ratelimit” proposal from OVS perspective
  - Would need global NF lookup mechanism

# NFT: Existing packet limiting functions



# Metering vs ratelimiting

- How about implementing bands using multiple ratelimit objects?

# nft add rule filter input icmp limit rate 1 mbytes/second counter drop
# nft add rule filter input icmp limit rate over 0.5 mbytes/second <diffserv opts>
# nft add rule filter input icmp accept

- Rate estimate is performed separately for each rule
- Lock 3 times

# RFC approach

- Extend libnftnl,nftables,linux for new ‘named meter’ object
- Export nft object lookup
  - NFT named objects may be called independent of IPv4/IPv6 family - floating object namespace?
- # ovs-ofctl add-meter <bridge> <meter options>
  - Call NFT kernel netlink API to configure table and meter
  - Keep association between OpenFlow meter # and NFT meter name
- Add OVS datapath action “meter:<nft name>”
  - # ovs-dpctl add-flow dp0 “in_port(0),eth(),eth_type(0x0806),arp()” “meter:foo,1”
  - Look in NFT table for meter with that name, take a reference
    - No meter? Return error.
  - For execution, call object eval()
    - Then decide whether to terminate current OVS processing or continue
- # ovs-ofctl add-flow <bridge> <match> “meter:1,...”

# NFT example: configure

```
# nft add table t1
# nft add meter t1 m1 band rate 3/minute burst 2 packets
# nft add chain t1 output { type filter hook output priority 0\;}
# nft add rule t1 output icmp type echo-request meter name m1 accept
# nft add rule t1 output icmp type echo-request drop
```

# NFT example: list

```
# nft list table t1
table ip t1 {
	meter m1 {
		packets 68 bytes 5712
		rate 3/minute burst 2 packets
			packets 55 bytes 4620
	}
	chain output {
		type filter hook output priority 0; policy accept;
		icmp type echo-request meter name "m1" accept
		icmp type echo-request drop
	}
}
```

# OVS usage example

```
# ovs-ofctl add-br br0
# ovs-ofctl add-port br0 p0 -- add-port br0 p1
# ovs-ofctl add-meter br0 ‘meter=1 pktps burst stats bands=type=drop \
                                    rate=3 burst_size=2’
# ovs-ofctl add-flow br0 ‘priority=10,in_port=1,icmp action=meter:1,2’
# ovs-ofctl add-flow br0 ‘priority=1 action=normal’
# ovs-ofctl dump-meters br0
OFPST_METER_CONFIG reply (OF1.3) (xid=0x2):
meter=1 pktps burst stats bands=
type=drop rate=3 burst_size=2
```

# Discussion

# Context: OpenFlow 1.5 standard, section 5.11

A meter table consists of meter entries, defining per-flow meters. Per-flow meters enable OpenFlow to implement various simple QoS operations, such as rate-limiting, and can be combined with per-port queues (see 5.8) to implement complex QoS frameworks, such as DiffServ.

A meter measures the rate of packets assigned to it and enables controlling the rate of those packets. Meters are attached directly to flow entries (as opposed to queues which are attached to ports). Any flow entry can specify a meter in a list of actions (see 5.8): the meter measures and controls the rate of the aggregate of all flow entries to which it is attached. Multiple meters can be used in the same table, either in an exclusive way (disjoint set of flow entries), or using multiple meter action per flow entry if the switch supports it. Multiple meters can be used on the same set of packets, either by using them in successive flow tables, or using multiple meter action per flow entry if the switch supports it.

| Meter Identifier | Meter Bands | Counters |
|---|---|---|

Each meter entry is identified by its meter identifier and contains:

- **meter identifier**: a 32 bit unsigned integer uniquely identifying the meter
- **meter bands:** an unordered list of meter bands, where each meter band specifies the rate of the band and the way to process the packet
- **counters:** updated when packets are processed by a meter

From *The Open Networking Foundation OpenFlow Switch Specification Version 1.5.0*

# Context: OpenFlow 1.5 standard, section 5.11.1

Each meter may have one or more meter bands. Each band specifies the rate at which the band applies and the way packets should be processed. Packets are processed by a single meter band based on the current measured meter rate. The meter applies the meter band with the highest configured rate that is lower than the current measured rate. If the current rate is lower than any specified meter band rate, no meter band is applied.

| Band | Type | Rate | Burst | Counters | Type specific arguments |
|---|---|---|---|---|---|

Each meter band is identified by its rate and contains:

- **band type:** defines how packet are processed
- **rate:** used by the meter to select the meter band, defines the lowest rate at which the band can apply
- **burst:** defines the granularity of the meter band
- **counters:** updated when packets are processed by a meter band
- **type specific arguments:** some band types have optional arguments

There is no band type *"Required"* by this specification. The controller can query the switch about which of the *"Optional"* meter band types it supports.

- *Optional:* **drop:** drop (discard) the packet. Can be used to define a rate limiter band.
- *Optional:* **dscp remark:** increase the drop precedence of the DSCP field in the IP header of the packet. Can be used to define a simple DiffServ policer.

From *The Open Networking Foundation OpenFlow Switch Specification Version 1.5.0*

# Status

- Proof-of-concept
  - About 0.5KLoC linux changes, 0.5KLoC libnftnl, 0.3KLoC nftables
  - Not yet sent out on list
  - nf_obj_lookup() API to OVS
- WIP
  - DSCP band
  - OVS kernel side patches
  - OVS userspace usage of NFT APIs

# Thanks!

joe@ovn.org