Programming Hardware Tables

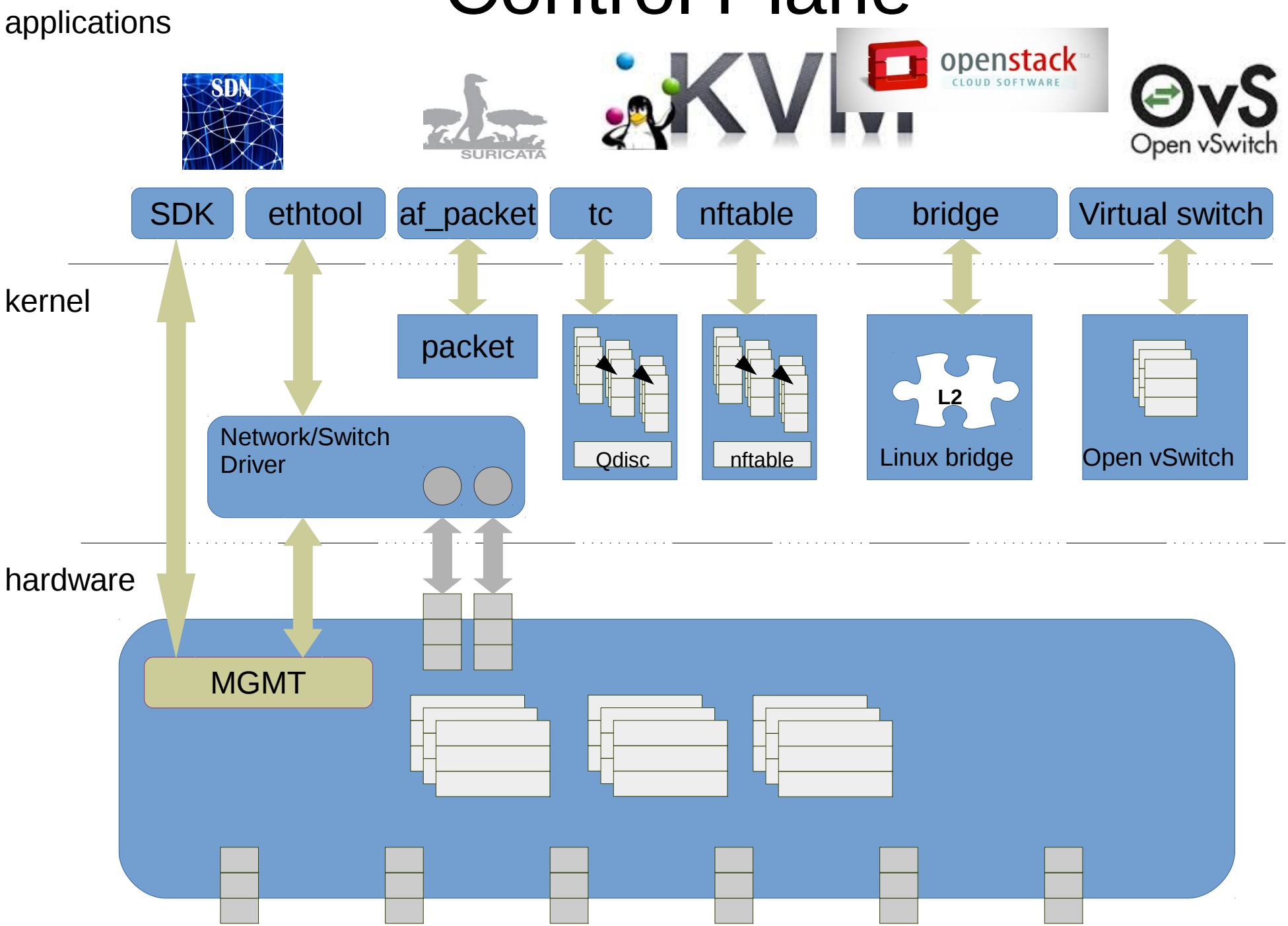
John Fastabend (Intel)

Netdev0.1

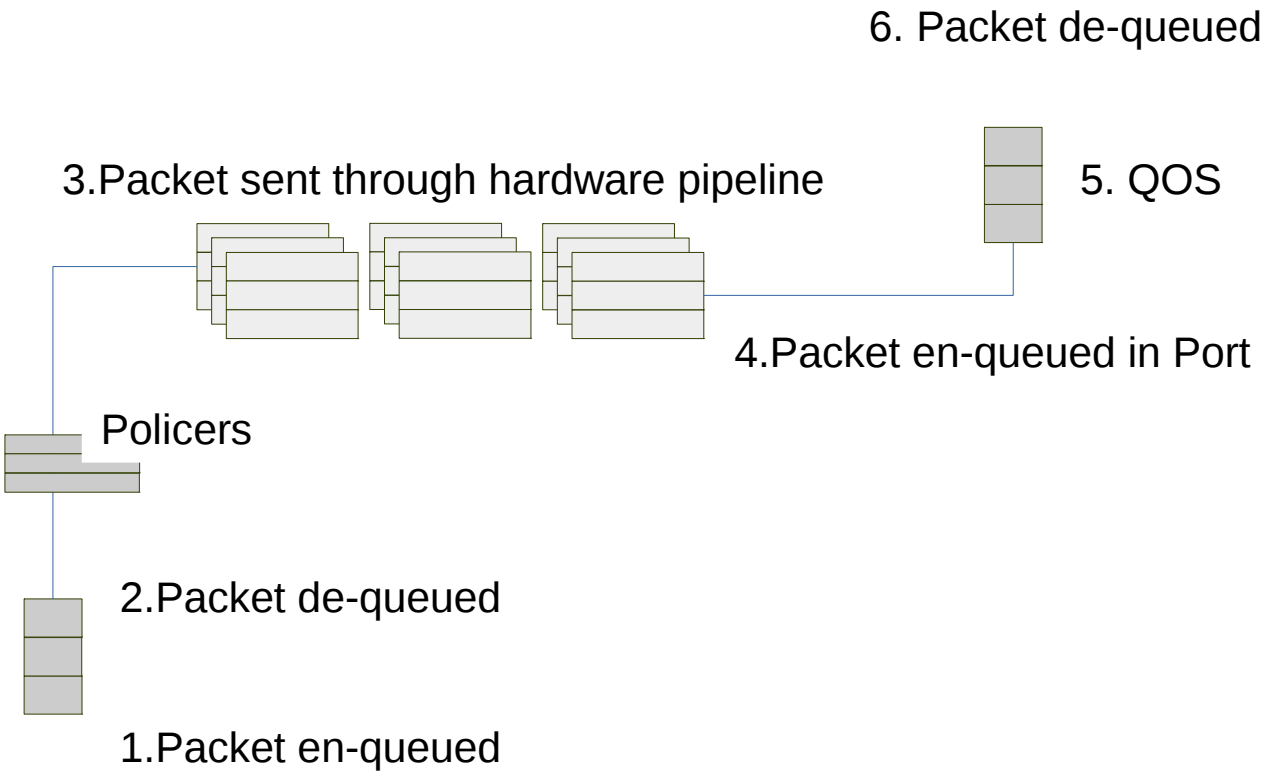
Agenda

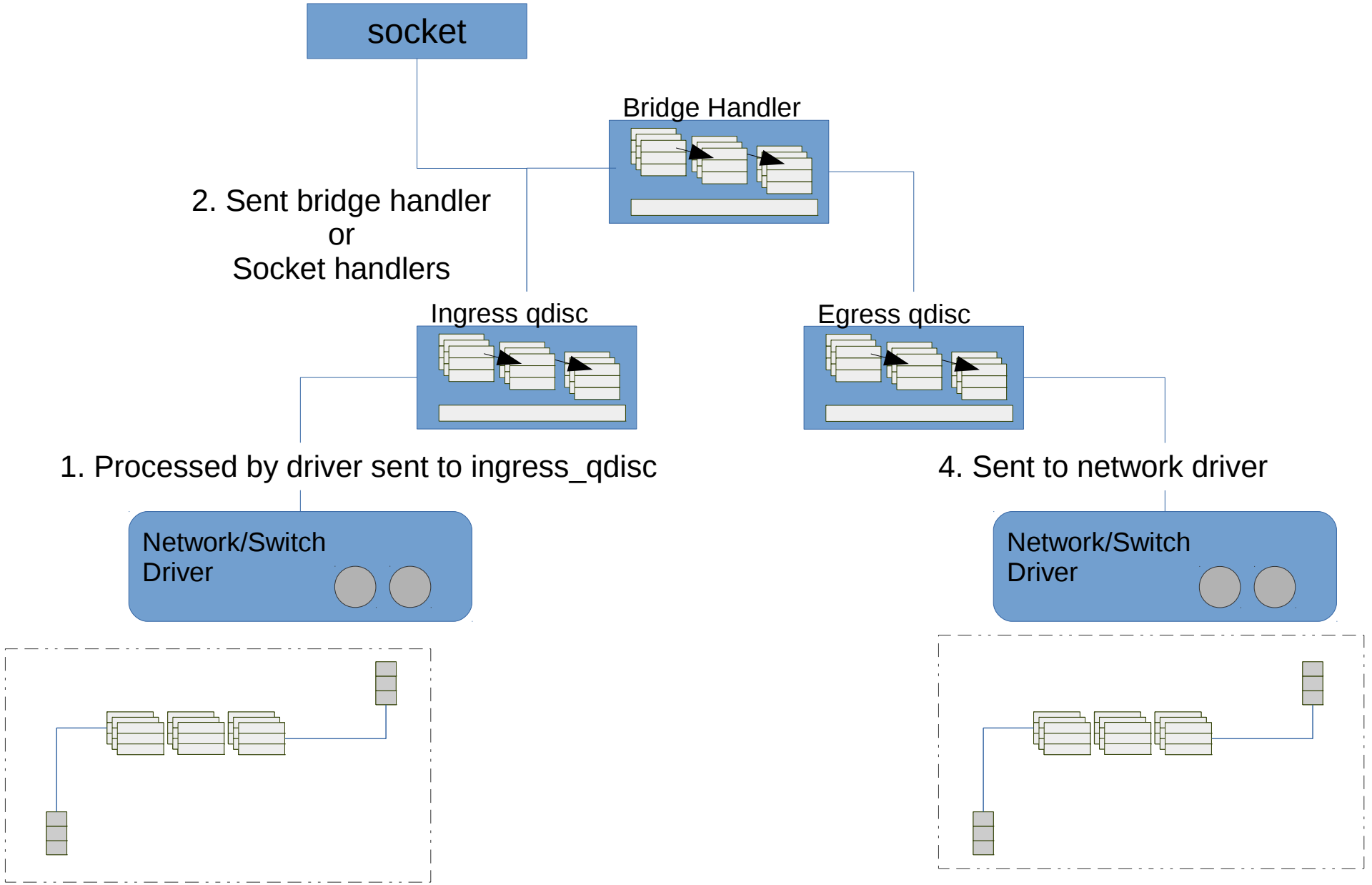
- Status Quo (quick level set)
- Flow API
- Device Independence code
- Future Work

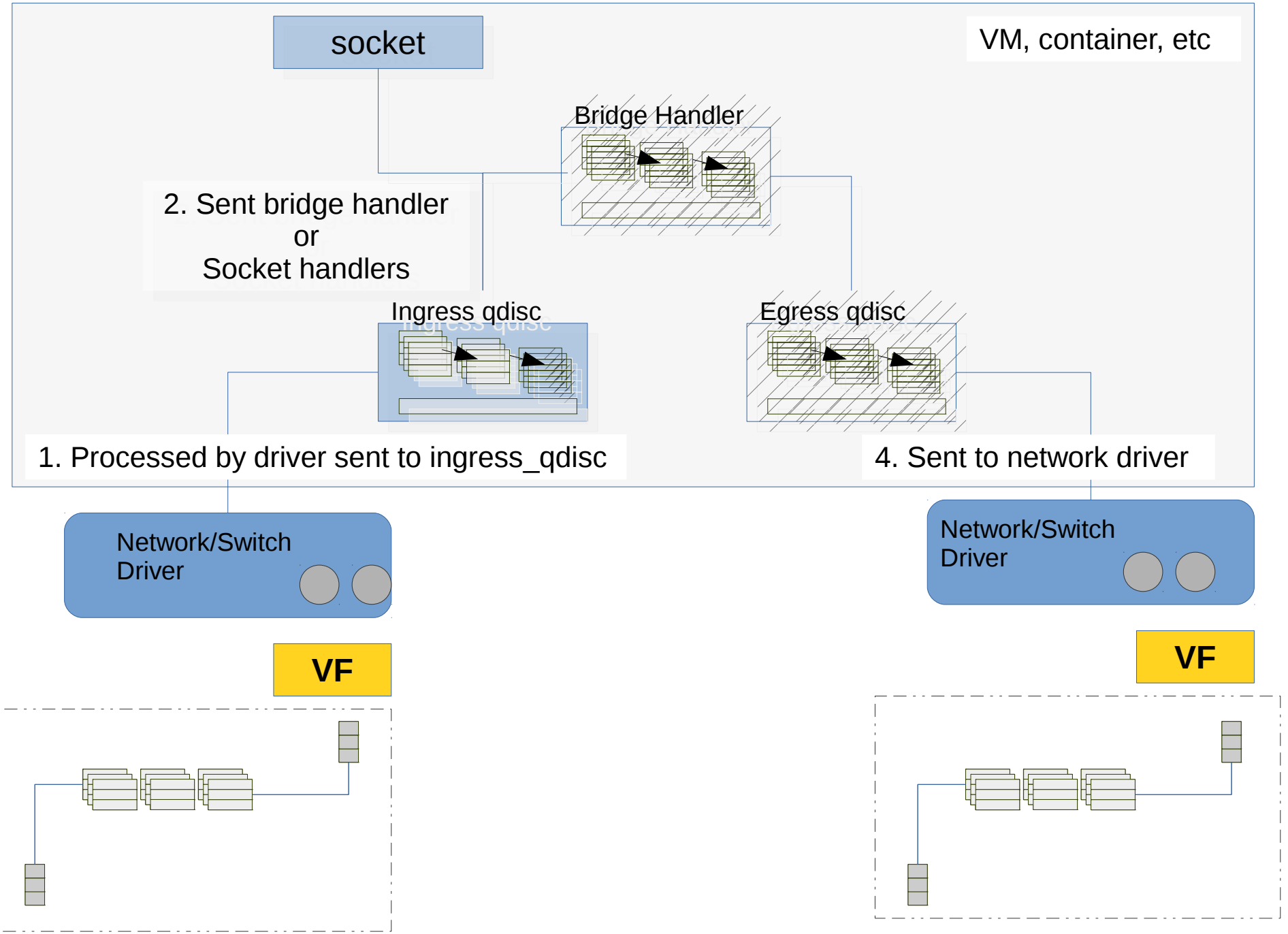
Control Plane



An Abstraction



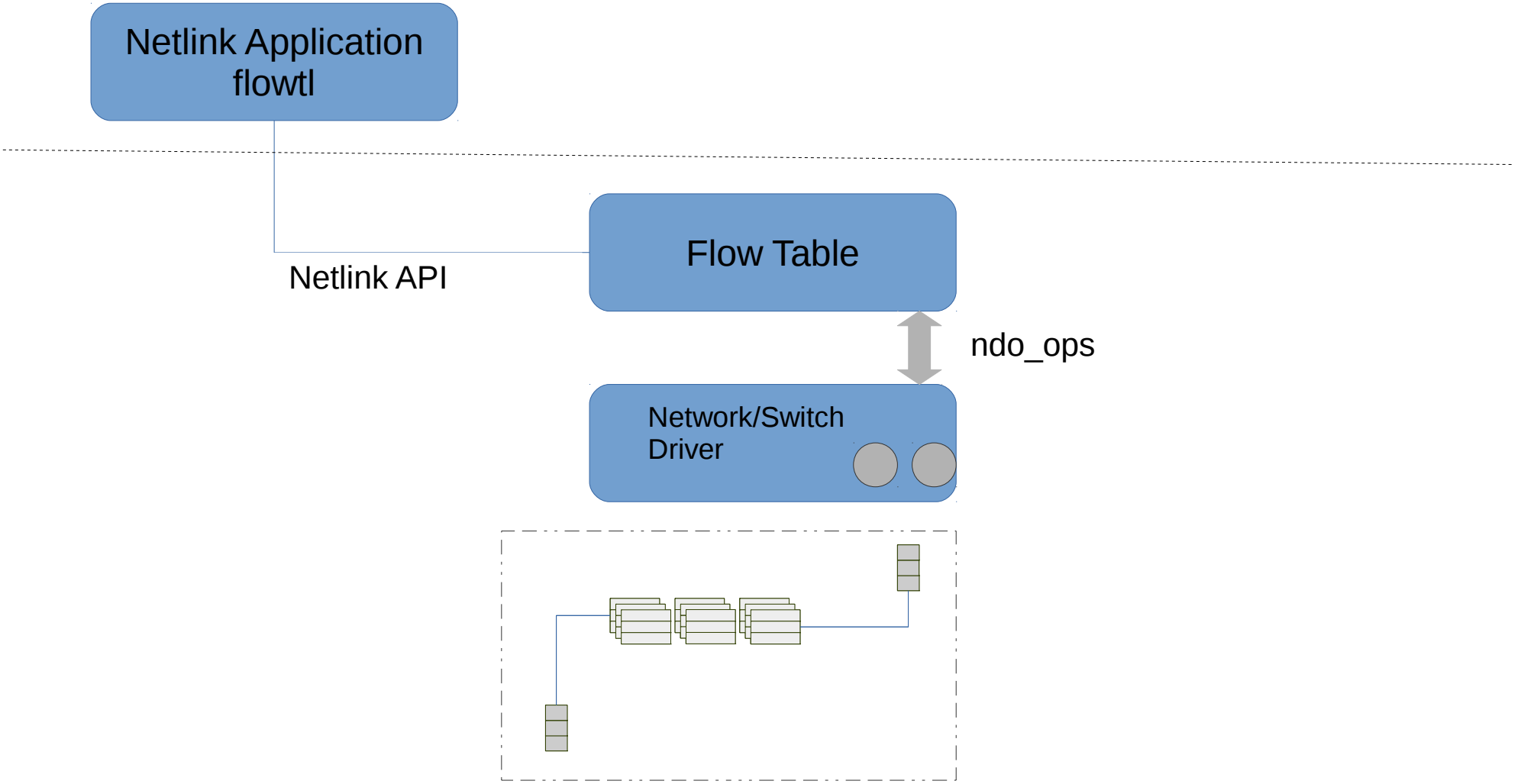

socket
Bridge Handler
2. Sent bridge handler
or
Socket handlers
Ingress qdisc
Egress qdisc
1. Processed by driver sent to ingress_qdisc
4. Sent to network driver
Network/Switch
Driver
Network/Switch
Driver


socket
VM, container, etc
Bridge Handler
2. Sent bridge handler
or
Socket handlers
Ingress qdisc
Egress qdisc
1. Processed by driver sent to ingress_qdisc
4. Sent to network driver
Network/Switch
Driver
Network/Switch
Driver
VF
VF

Flow API

- must be flexible enough to support many different hardware pipelines
- incorporate new packet types and actions easily. Preferably at run time
- policy and optimization are user space driven
- vendor neutral
- extensible

Flow API


Netlink Application
flowtl
Netlink API
Flow Table
ndo_ops
Network/Switch
Driver

NetConf

- Flow API UAPI (netlink)
  - get_headers, get_headers_graph
  - get_actions
  - get_tables
- FlowAPI UAPI (netlink)
  - *set_rule, del_rule*, get_rule
- Nftable (netlink)
  - set_rule, del_rule: nftable chains

NetConf

- Flow API (ndo_ops)
  - get_headers, get_headers_graph
  - get_actions
  - get_tables
- FlowAPI (ndo_ops)
  - set_rule, del_rule, ~~get_rule~~

NetConf

- Flow API (core)
  - Packing/Unpacking Netlink
  - Provides standard structures for drivers
  - Publish structures for In-kernel consumers
  - minimal ndo_op set
  - Validation
  - get_rules()
- Does Not
  - Provide mapping strategies between device models
  - Pipeline optimizations

Flow API: Creating a device model

- *netlink: net_flow_cmd_get_headers*

#./flow -i eth0 get_headers

ethernet { src_mac:48 dst_mac:48 ethertype:16 }

vlan { pcp:3 cfi:1 vid:12 ethertype:16 }

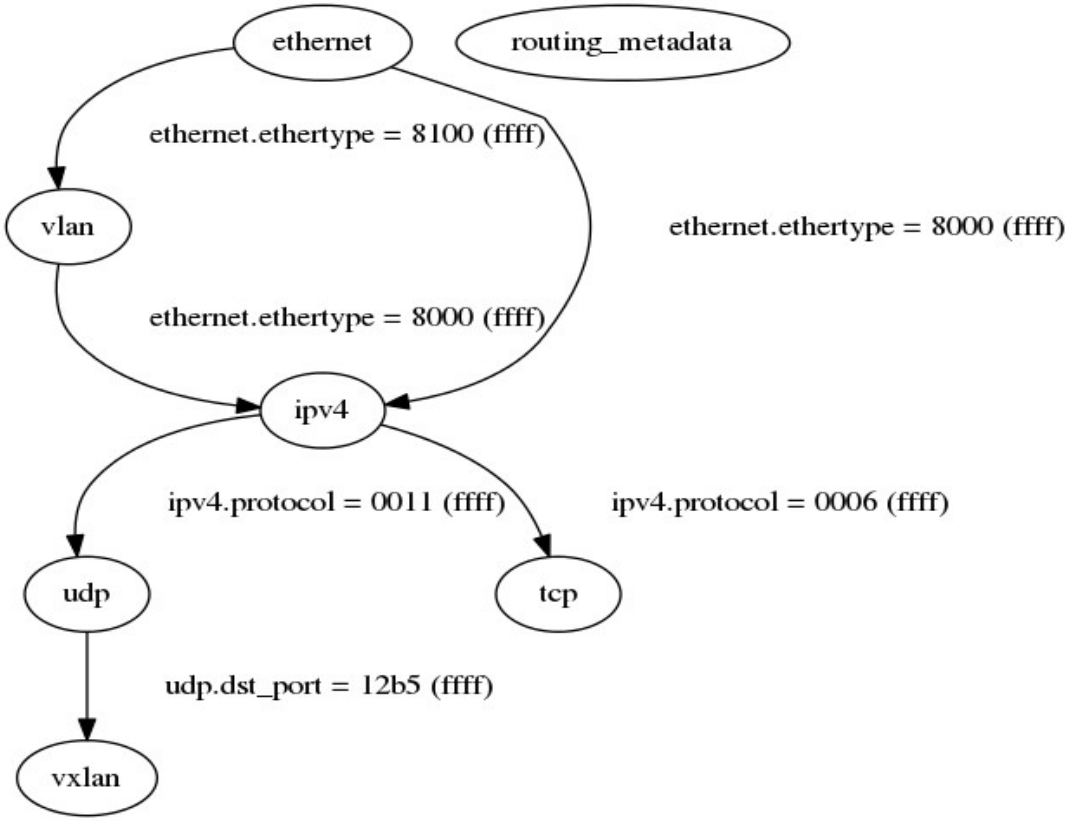
[...]

vxlan { vxlan_header:32 vni:24 reserved:8 }

Flow API: Creating a device model

- *netlink: net_flow_cmd_get_headers*

  ./flow -i eth0 -g get_header_graph



Flow API: Actions

- netlink: net_flow_cmd_get_actions

  ./flow -i eth0 get_actions

  6: dec_ttl (void)

  7: set_dst_mac (u48 mac)

  8: push_vlan (u16 vlan)

  9: drop(void)

  ./flow -i eth1 get_actions_primitives *route**

  set_field (DMAC_FIELD, DMAC)

  push_header(VLAN_HEADER, VLAN)

  dec_field(IPV4_TTL)

Flow API: Tables

- netlink: net_flow_cmd_get_tables

  ./flow -i eth0 get_tables

tcam: 1 src 1 apply 1 size 4096

matches:

field: ethernet [dst_mac (mask) src_mac (mask) ethertype (mask)]

field: vlan [pcp (mask) cfi (mask) vid (mask) ethertype (mask)]

field: ipv4 [dscp (mask) ecn (mask) ttl (mask) protocol (mask) dst_ip (lpm) src_ip (lpm)]

field: tcp [src-port (mask) dst-port (mask)]

field: udp [src-port (mask) dst-port (mask)]

actions:

1: set_egress_port (u32 egress_port)

3: set_dst_mac (u48 mac_address)

4: set_src_mac (u48 mac_address)

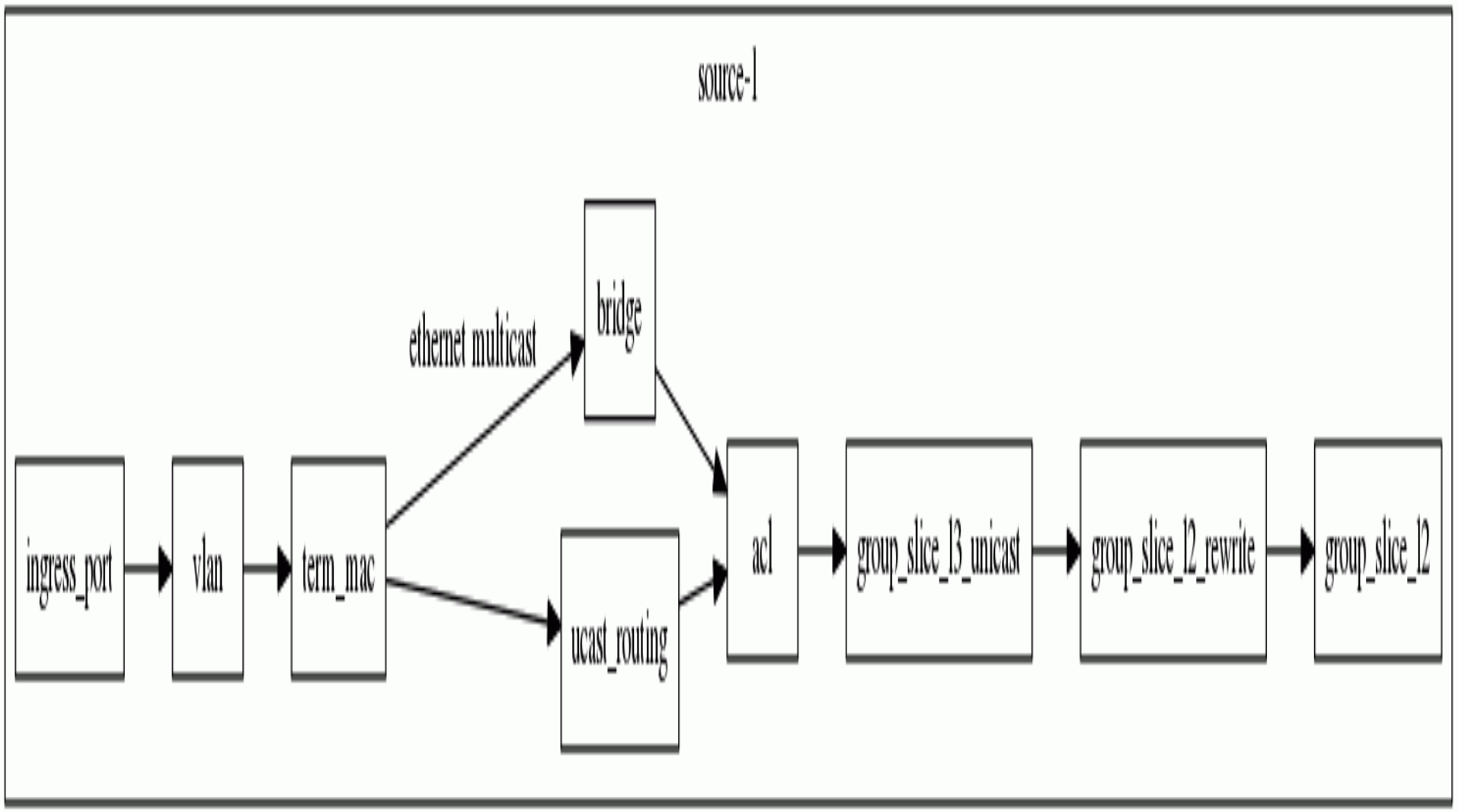
5: trap()

Flow API: Table Graph

- netlink: net_flow_cmd_get_table_graph

  ./flow -i eth0 get_graph

Flow API: Table Graph



Flow API: set_flow

- netlink: net_flow_cmd_set_rule

  ./flow -i eth0 set_rule

```
#flow  -i eth1 set_flow prio 1 handle 4 table 3 \
    match ethernet.ethertype 0x0800 0xffff \
    match in_lport.in_lport 1 0xffffffff          \
    match vlan.vid 10 0xffff                      \
    action copy_to_cpu


table : 3  uid : 4  prio : 1
 ethernet.ethertype = 0800 (ffff)
 metadata_t.in_lport = 00000001 (ffffffff)
 vlan.vid = 000a (ffff)
  2: copy_to_cpu (  )
```

Flow API: set_flow

./nft list table hw_table <- table container for hardware chains

```
table hw_table {
	chain acl {
		ip protocol icmp ip daddr 1.2.3.4 counter packets 5 bytes 420 drop
		[...]
	}
	chain next {
		[...]
	}
```

./nft add rule hw_table acl {statement}

Current Work

- header/header_graph/actions: normalize, intersection, disjoint
- map routines between pipelines
- example applications
- handlers for standard protocols L3 (speculative)
- generate test code (pcap/tcpreplay)

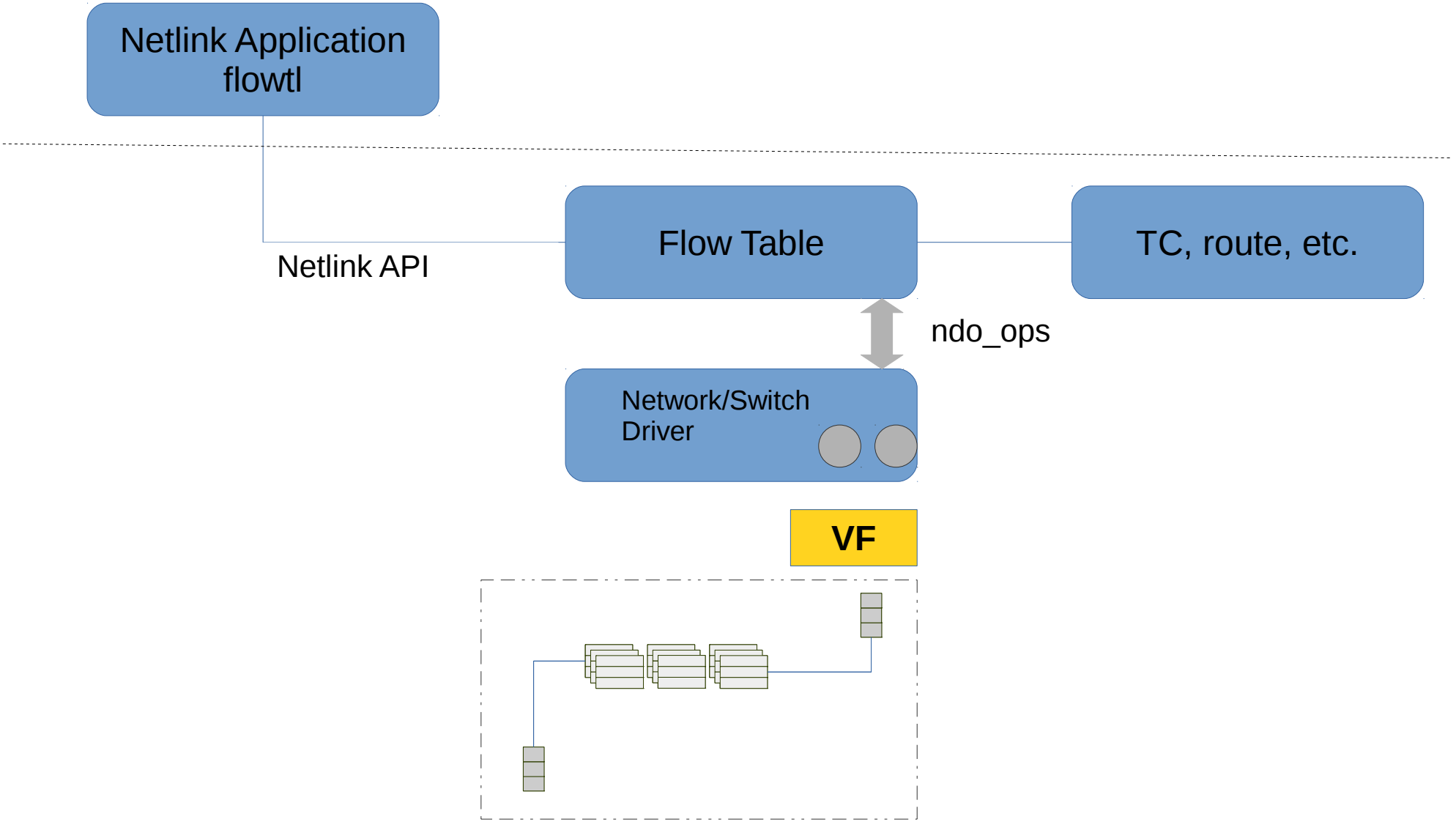
Questions

https://github.com/jrfastab/iprotue2-flow-tool

https://github.com/jrfastab/rocker-net-next

http://jrfastab.github.io/jekyll/update/2014/12/21/flow-api.html

Flow API


Netlink Application
flowtl
Netlink API
Flow Table
TC, route, etc.
ndo_ops
Network/Switch
Driver
VF