grout

FOSDEM 2025, Brussels

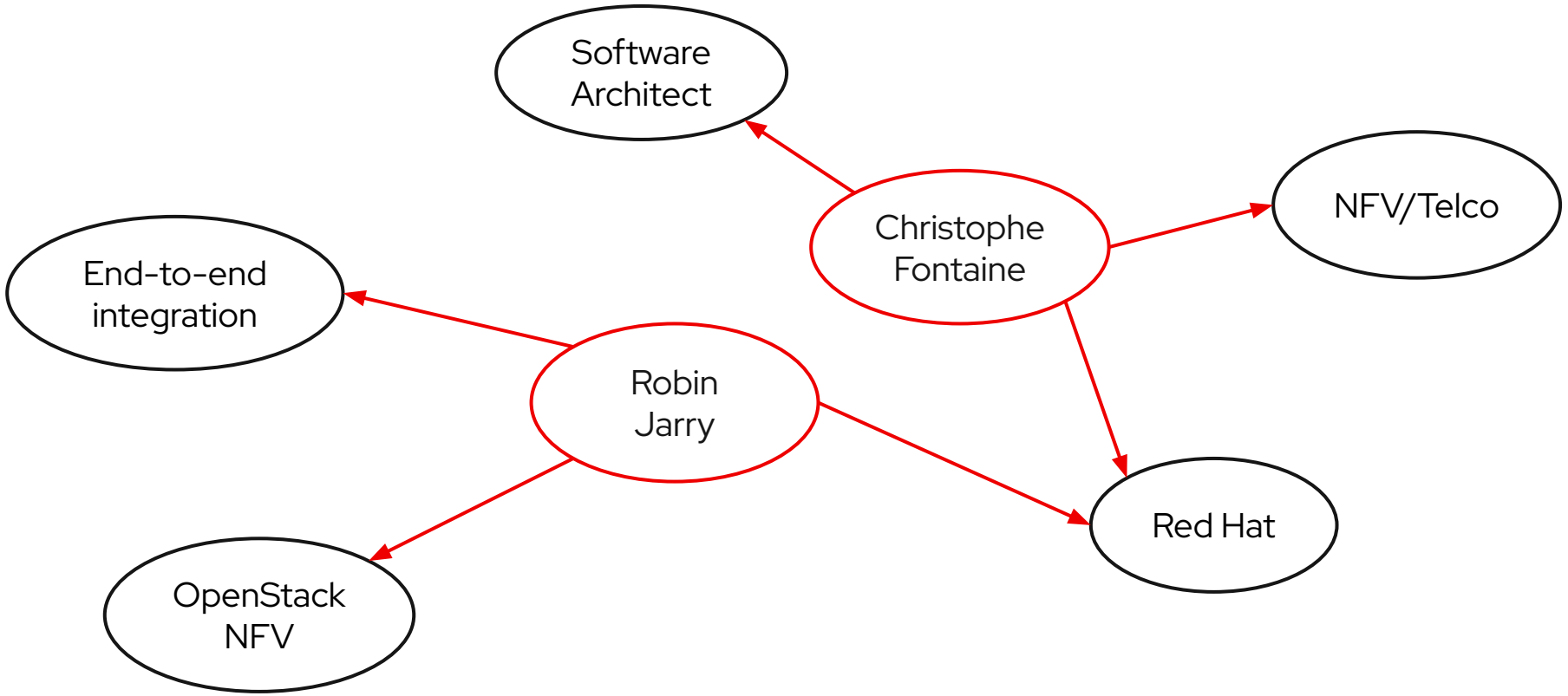
Christophe Fontaine

Robin Jarry

FOSDEM'25

Red Hat

Bonjour


Software Architect
Christophe Fontaine
NFV/Telco
End-to-end integration
Robin Jarry
Red Hat
OpenStack NFV

Agenda

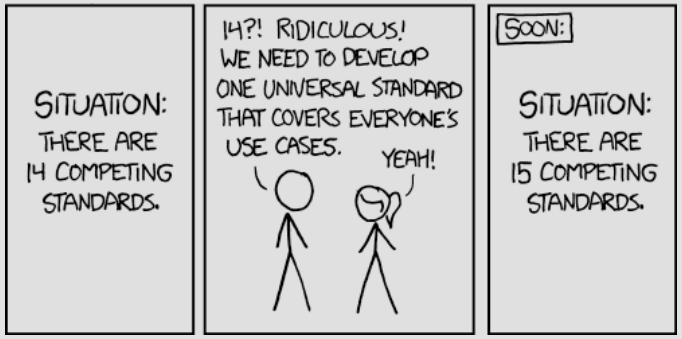
1. Why grout?
2. grout?
3. How grout?
4. What's cooking?
5. Performance

graph
router

Why grout?

- External (telco) platform validation
- Internal end-to-end regression tests
- `testpmd` is great but ...
- Response to 2023 summit question

" How can we make DPDK more visible? "



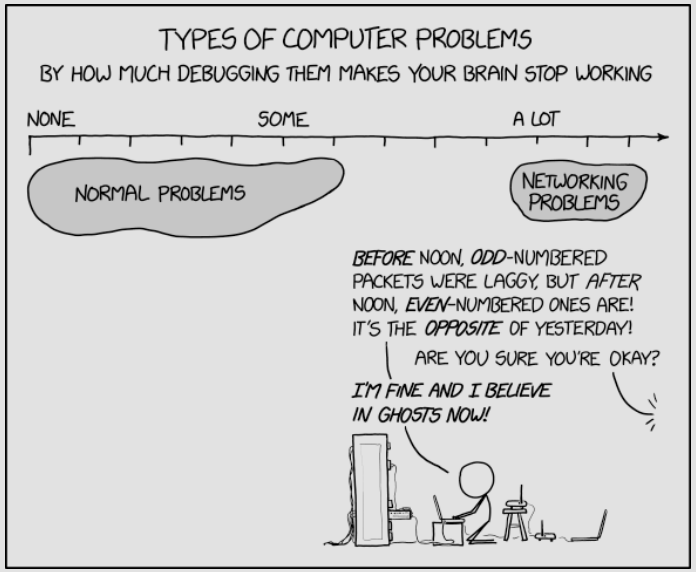
* https://xkcd.com/927/

grout?

- *Vendor independent* application to replicate network functions behavior
- DPDK first, reuse as much as possible
- Runtime configuration via an API
- Extensible platform to incubate new features
- MVP: L3 forwarding in *multiple VRFs*
- 100% Free and Open Source Software

How grout?

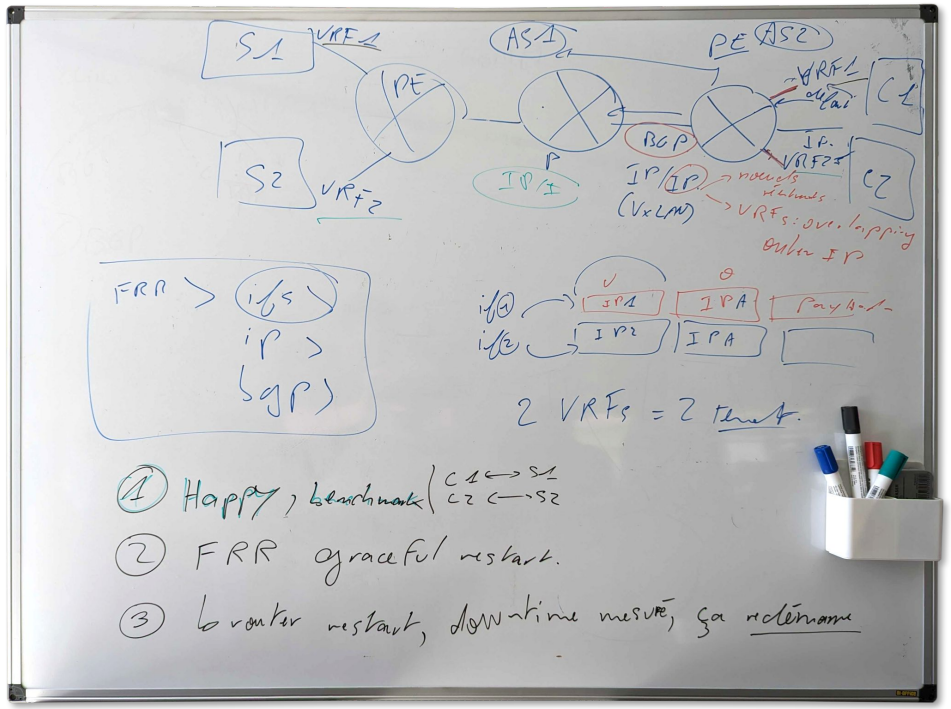
- To the drawing board
- Graph and packet flow
- High level design
- Anatomy of a module



* https://xkcd.com/2259/

To the drawing board

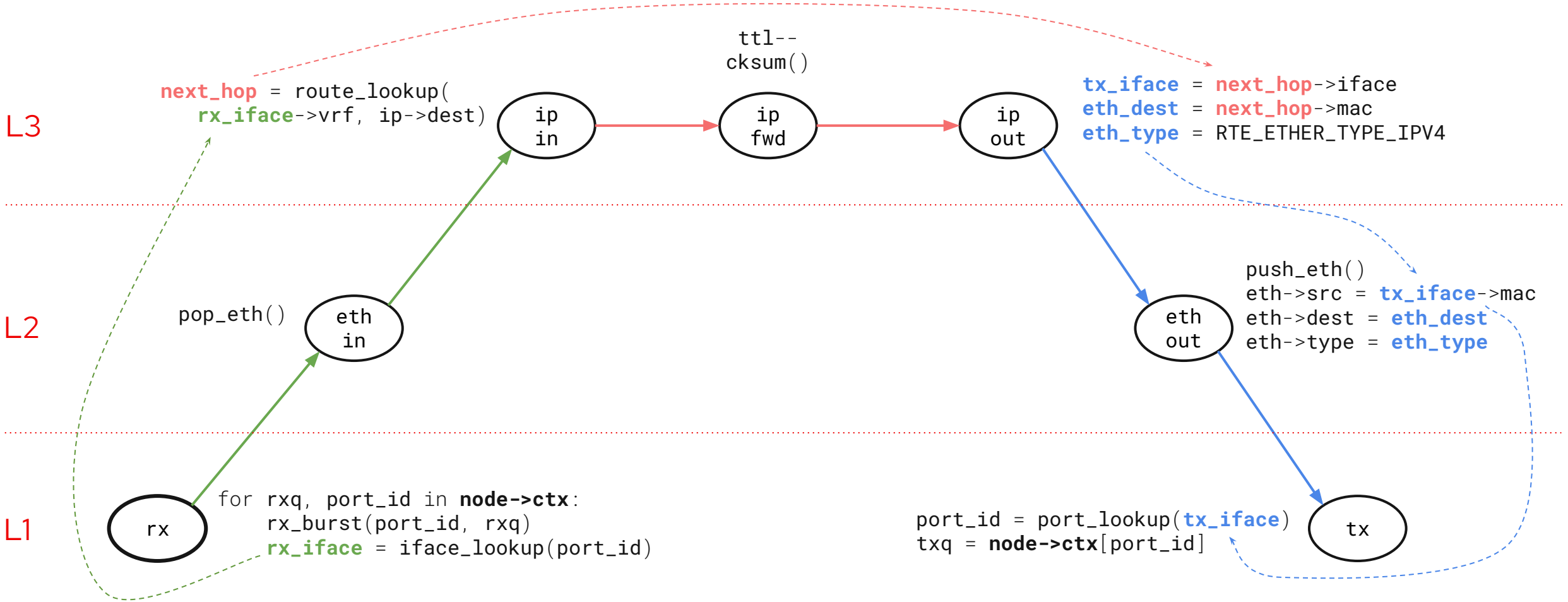
- Leap of faith ⟹ **rte_graph** library [1]
  (inspired from **VPP** node library [2])
- How to write a router from scratch?
- How many RFCs is too many?
- How can we ensure extensibility?
- Enforce OSI model layers for modularity



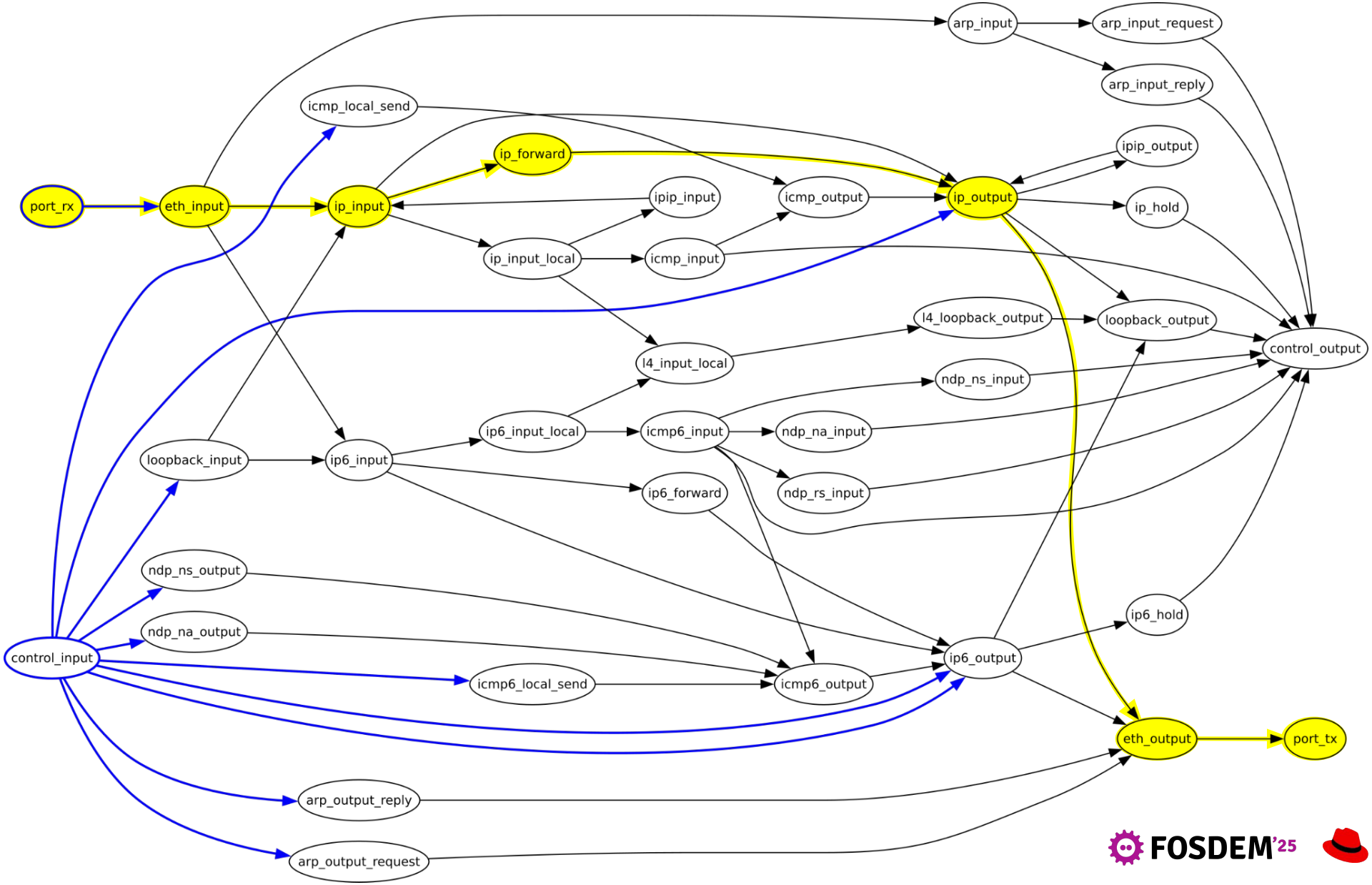
1. DPDK Summit 2023 - Graph Library Design and Implementation Details - Jerin Jacob, Marvell
2. https://git.fd.io/vpp/tree/src/vlib/node.h

Packet Flow Explained

Simplified view of the IPv4 graph



Packet Graph (simplified)

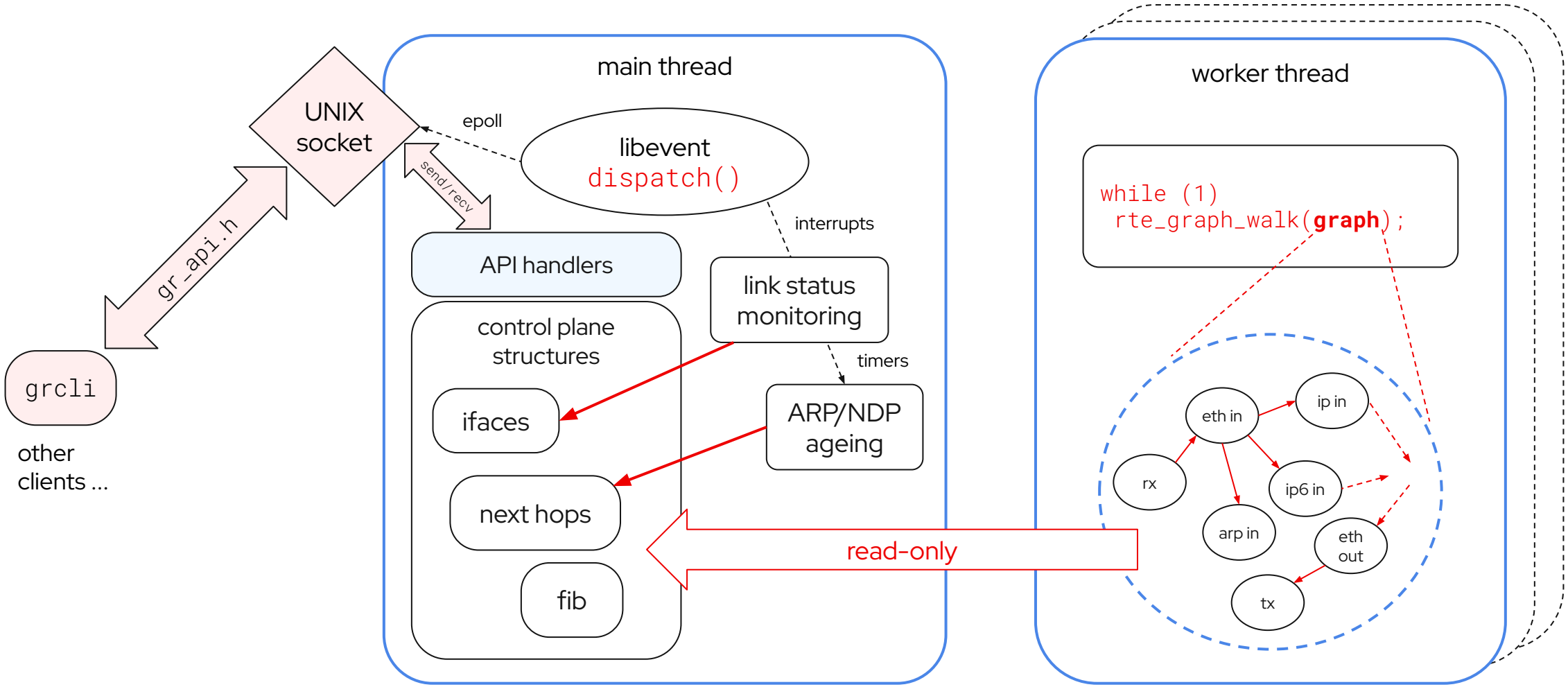


Nodes as statistics

```
grout# show stats software
```

NODE	CALLS	PACKETS	PKTS/CALL	CYCLES/CALL	CYCLES/PKT
ip_input	73455958	3010445229	41.0	2346.8	57.3
eth_input	253809664	3010445245	11.9	554.4	46.7
port_rx	253809664	3010445245	11.9	479.8	40.5
port_tx	73455958	3010445229	41.0	1382.7	33.7
ip_output	73455958	3010445229	41.0	1150.2	28.1
eth_output	73455958	3010445229	41.0	1089.8	26.6
ip_forward	73455958	3010445229	41.0	631.3	15.4
control_input	253809664	1	0.0	35.5	9002614826.0
arp_input_request	4	4	1.0	1827425.0	1827425.0
control_output	1	1	1.0	6874.0	6874.0
arp_input_request_drop	4	4	1.0	1040.5	1040.5
arp_input	4	4	1.0	292.5	292.5

High Level Design

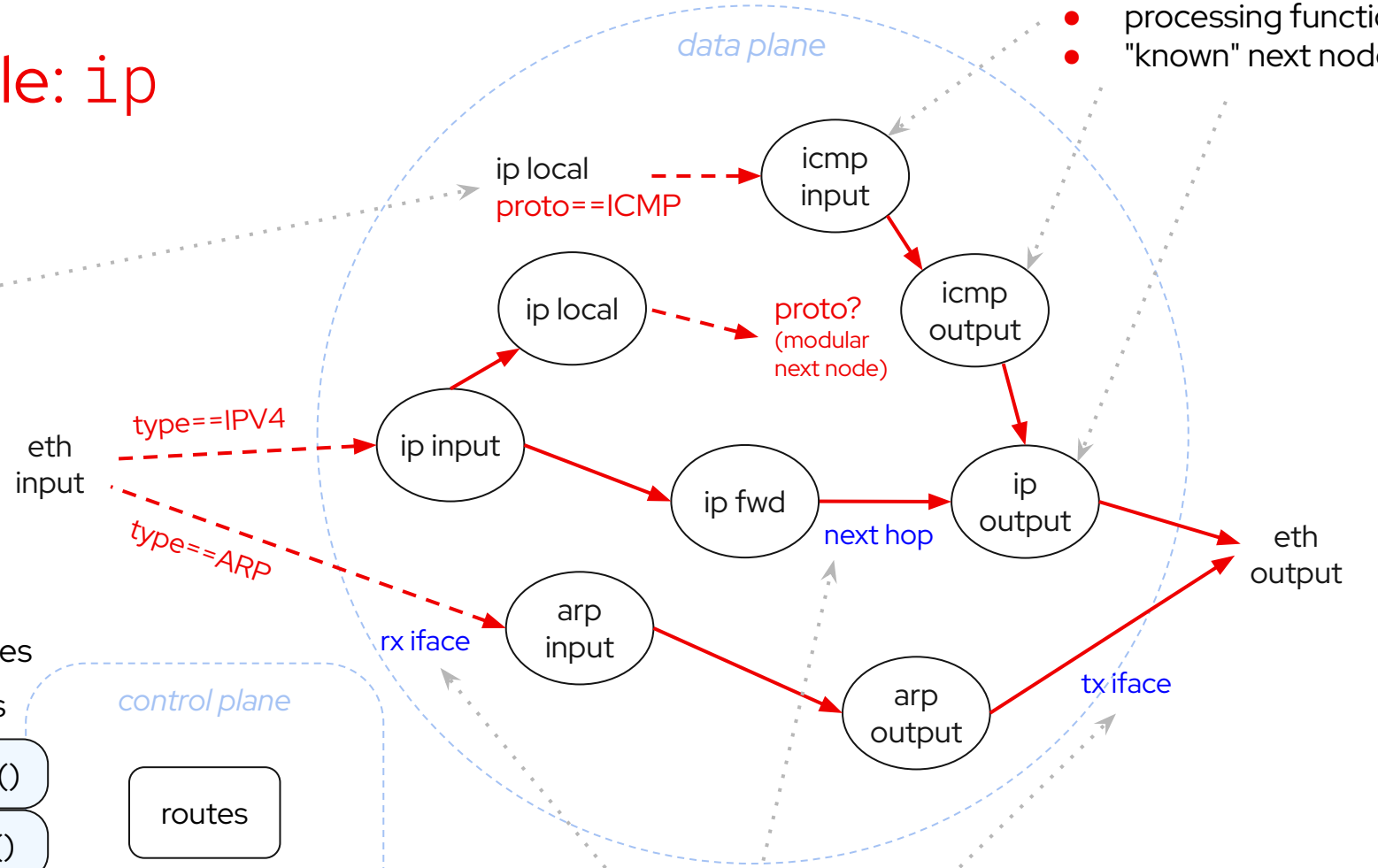


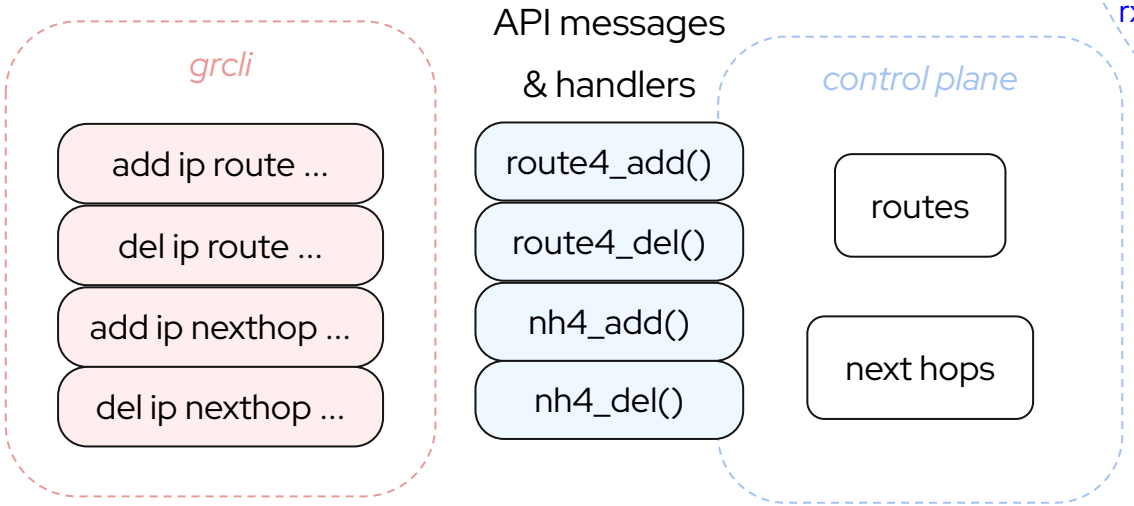
Anatomy of a module: ip

`rte_node` definitions

- processing function
- "known" next nodes

Modular insertion points in the global graph.





Packet context from previous nodes in `mbuf` private data

What's cooking?

- Project status
- Work in progress
- Wishlist

Project Status

September 2024 (v0.2)

- IPv4/IPv6 forwarding
- VLAN sub interfaces
- IPv4 in IPv4 tunnels
- Accepted as a dpdk.org project

February 2025 (v0.6)

- Fixed *a few* bugs 😱
- Loopback interfaces for termination in Linux
- `dnf install grout` on Fedora 42
- Packet tracing and inspection
- Basic ICMP and ICMPv6* stack
- Notifications over the API socket

* ICMPv6 added by a new (non Red Hat) contributor. Thanks!

Packet tracing

```
grout# set trace iface p1
grout# show trace count 1
--------- 11:25:07.994585571 cpu 1 ---------
port_rx: port=1 queue=0
eth_input: ba:d0:ca:ca:00:01 > f0:0d:ac:dc:00:02 type=IP(0x0800) iface=p1
ip_input: 172.16.1.2 > 172.16.1.1 ttl=64 proto=IPIP(4)
ip_input_local:
ipip_input: iface=tun1
ip_input: 10.98.0.2 > 10.99.0.2 ttl=64 proto=ICMP(1)
ip_forward:
ip_output: 10.98.0.2 > 10.99.0.2 ttl=63 proto=ICMP(1)
eth_output: f0:0d:ac:dc:00:01 > ba:d0:ca:ca:00:00 type=IP(0x0800) iface=p0
port_tx: port=0 queue=0
```

Work In Progress

- FRR dataplane plugin
  - Leveraging the grout API
  - Inline BGP & BFD via loopback interface
- ECMP routing
  - Redesign of the next-hop structures
  - Asynchronous rendering of the routing tables

Wishlist

- Graph dispatch mode [1]
- Data path optimization
- Control path scalability (BGP full view injection)
- Automatic CPU/RXQ mapping/rebalancing [2]
- <insert your ideas>

1. https://inbox.dpdk.org/dev/D672A6CZXHPH.XUFCIYI7V0NS@redhat.com/
2. https://docs.openvswitch.org/en/latest/topics/dpdk/pmd/#pmd-automatic-load-balance

Graph dispatch mode

- Control plane operations are currently deferred to a libevent thread (outside the packet graph)
- `rte_graph` supports a pipeline/dispatch mode
- Integrate non-polling threads to the packet graph
- Seamless transfer of "slow path" operations to non-polling threads via `rte_ring`s and `pthread_cond_signal`
  - Consistent programming model
  - "Nodes as statistics" for control plane as well

Performance

- 1 isolated full core (2 threads) Intel Xeon Silver 4316 @ 2.6GHz
- Intel® E810-XXV 2x25G NIC
- `grout` v0.6, DPDK 24.11, `gcc -O2`
- Minimal code optimization effort yet

`testpmd` mac forward (reference)	~54M pkt/s
IPv4 forwarding	~18M pkt/s
IPv4 forwarding + VLAN pop/push	~16M pkt/s
IPv6 forwarding	~15M pkt/s

Wrapping up

- Test it and provide feedback!
- What would you like to do with `grout`?
- All contributions are more than welcome.

Thank you

Questions?

https://github.com/dpdk/grout

grout@dpdk.org

#grout @ dpdkproject.slack.com

FOSDEM'25 Red Hat