Rethinking Software Network Data Planes in the Era of Microservices

Sebastiano Miano

PhD Final Defense

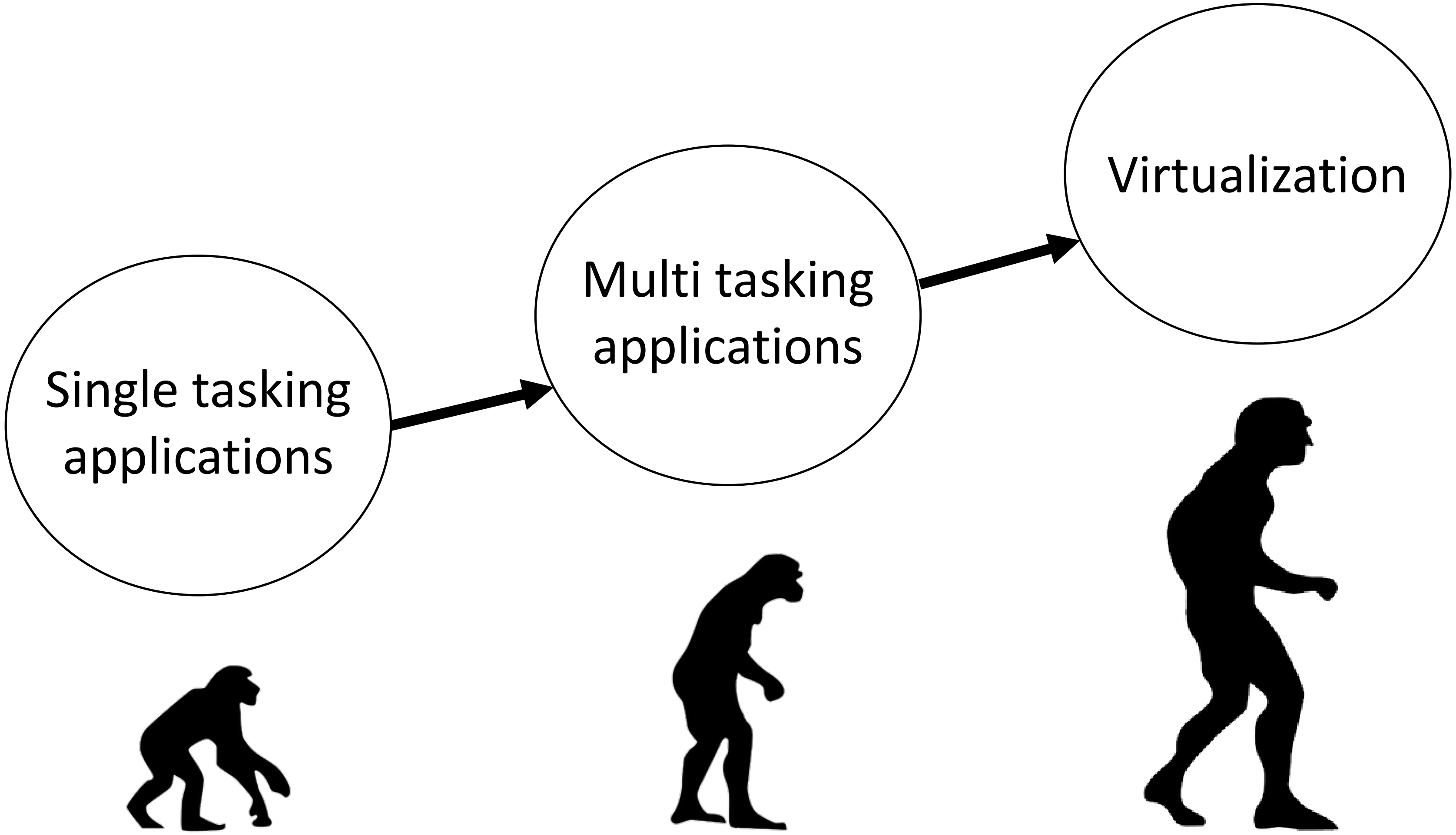
Politecnico di Torino

July 13th 2020

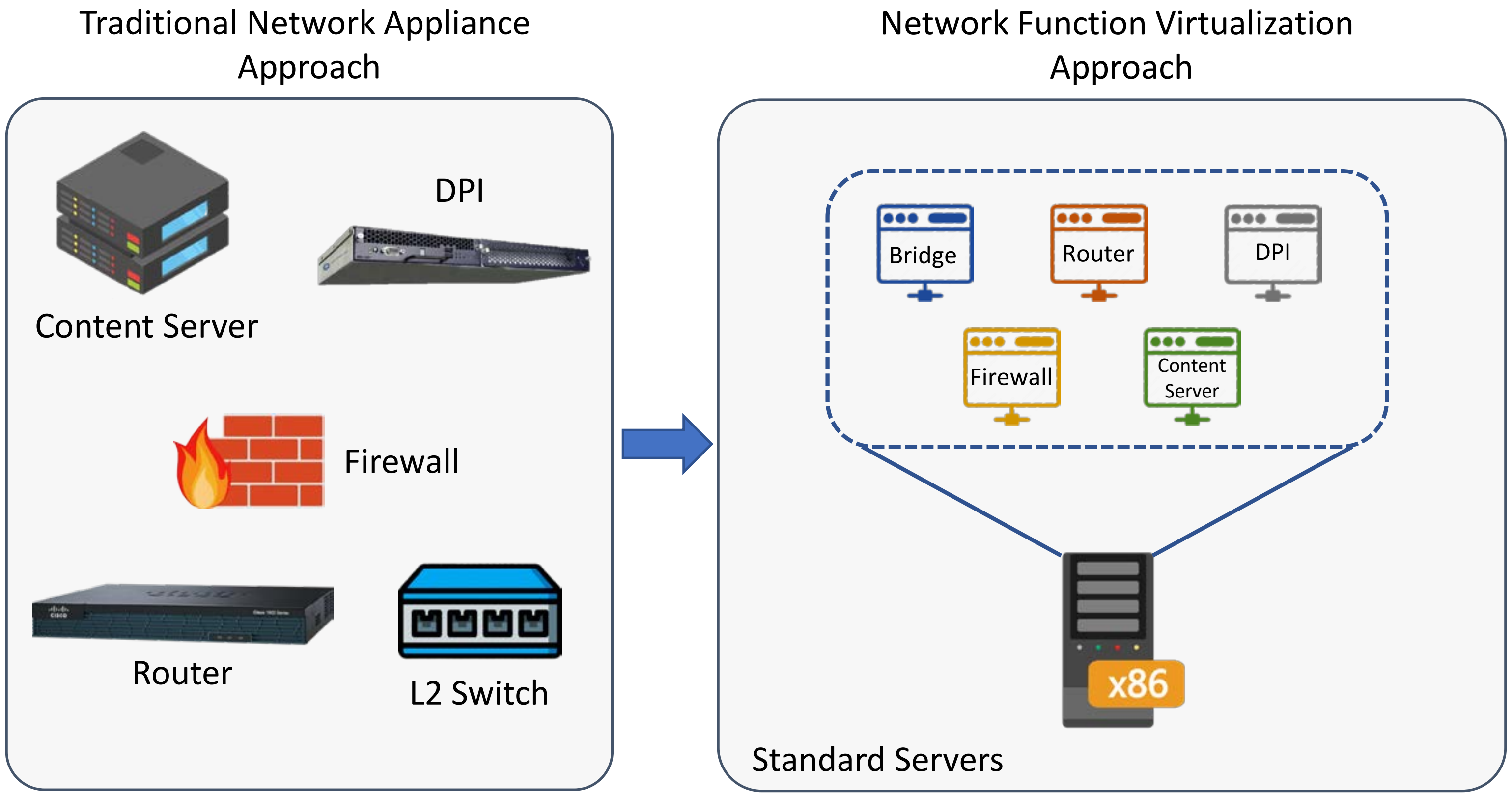
PhD Advisor

Fulvio Risso

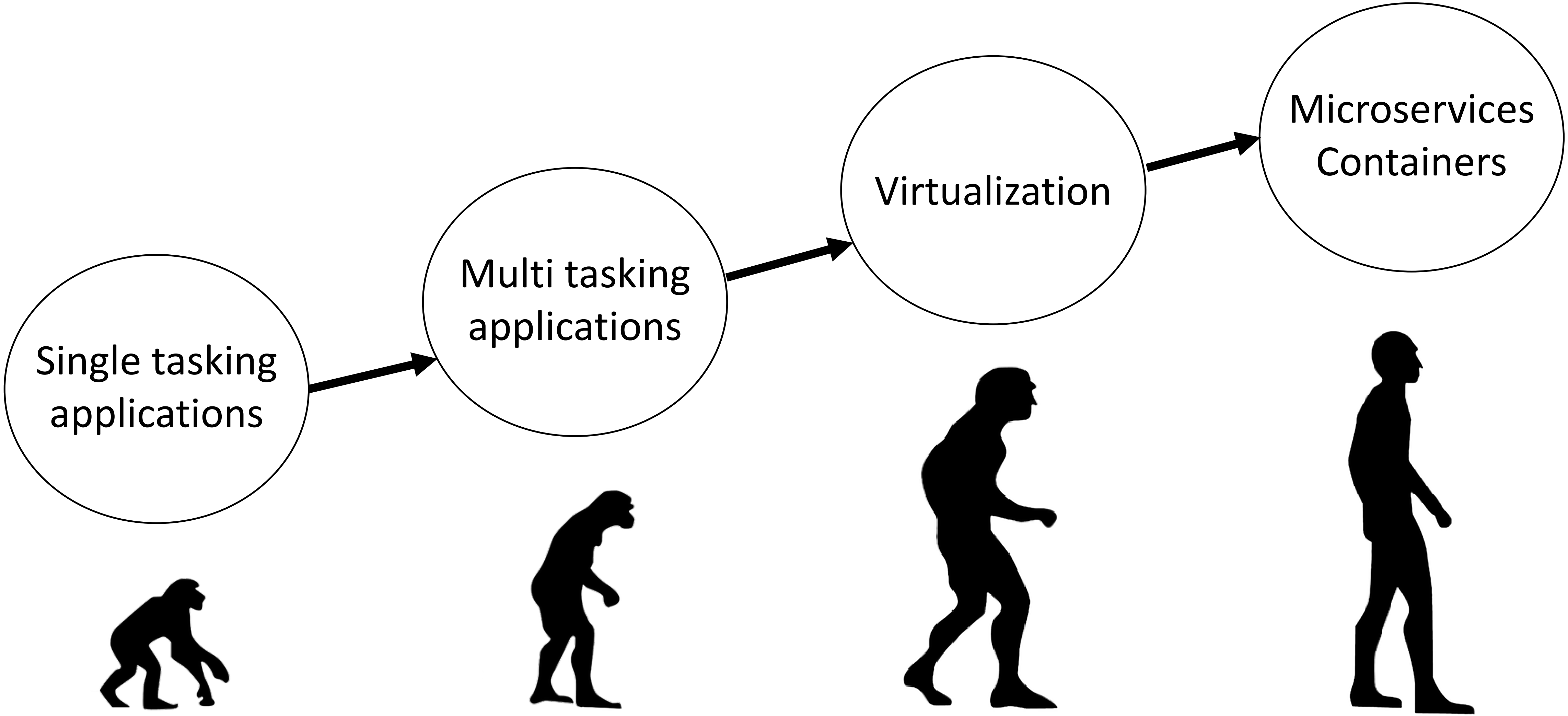
Evolution of end-host applications



Network Function Virtualization



Evolution of end-host applications



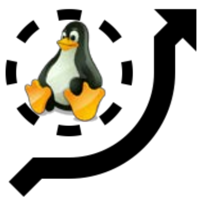
Microservice Era – New requirements for NFs

- Low overhead
  - 5G, IoT have limited resources available to be shared among the applications
- Run on commodity Linux servers
  - Run the NF as part or all of the large fleet of currently deployed servers
- Allow low-disruption maintenance
  - Agile service development
  - Re-build and re-deploy a small part of the entire service without disruption
- Coexist with other services on a given server
  - No need for dedicated servers that only run a single application

[1] Katran Facebook Load Balancer: https://engineering.fb.com/open-source/open-sourcing-katran-a-scalable-network-load-balancer/

[2] Cisco, Cloud-native Network Functions - https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/1128_TECHAD_CKN_PDF.pdf

User vs Kernel Space Networking



✓ High throughput and latency

X High resource consumption

X Custom kernel modules/drivers

X Difficult integration with "native" applications

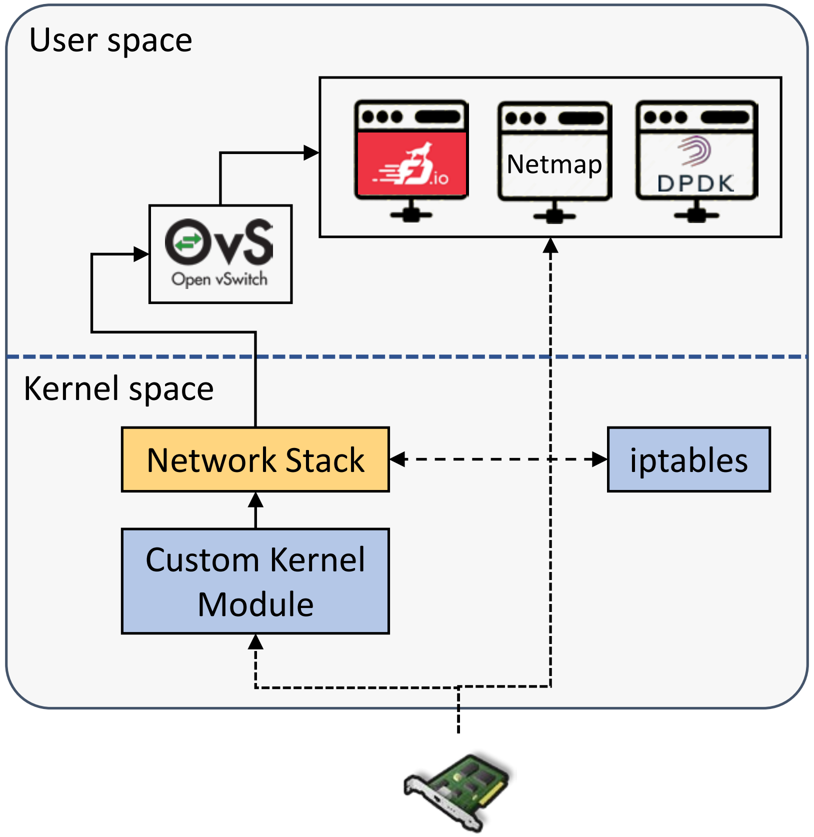
X Not yet mature TCP/IP stack

X Performance

✓ **Stable** development process

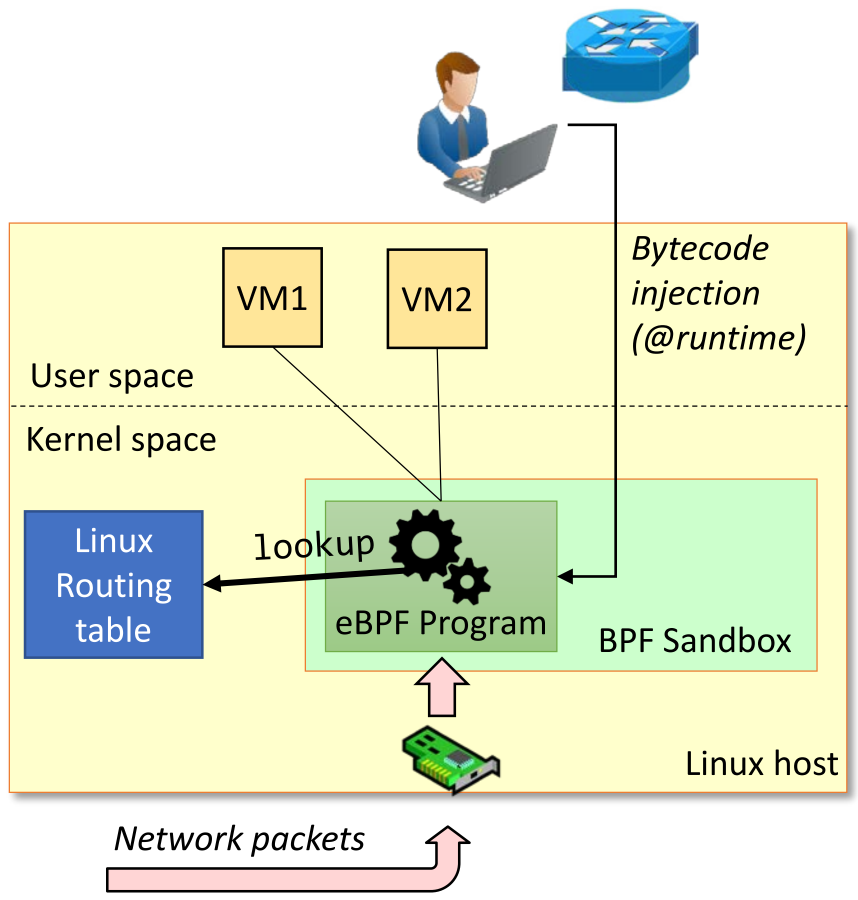
X **Upstreaming** code is hard

X Application awareness



Alternatives? The extended BPF (eBPF)

- Good candidate to enable **customized**, **fast**, **flexible**, **dynamic** network processing
    - So far, mainly used for tracing/monitoring
- Flexible and efficient virtual machine-like construct in Linux kernel
    - Dynamic injection
        - **Low-disruption maintenance**
        - JIT deployment process
    - Integration with the kernel subsystem
        - **Coexist with other services on a given server**
    - **Safety**
    - **Performance**
        - **XDP** (eXpress Data Path)



Thesis Goal

- Explore the possibility to use a new technology (eBPF) to build network functions suitable for the «cloud-native» environment
    1. Comprehensive and in-depth analysis of how eBPF can be used to write complex network applications
    2. Overcome eBPF limitation with a general framework for in-kernel NFs
    3. Validate the framework with the design and implementation of real-world applications based on such paradigm

- Looking ahead
    4. Dynamic optimization of generic software data planes

Creating Complex Network Functions with eBPF

Advantages and Main Limitations

Build NFs with eBPF: Limitations

- All that glitters is not gold
  - Limitations mainly given by the eBPF controlled environment
  - Many of them have been (partially) «solved» over the years

- Several limitations specific for NF processing

Build NFs with eBPF: Limitations

- Limited program size
  - 4096 maximum BPF instructions (130K simulated) for normal users
    - Kernel v5.1+ supports 1M **simulated** instructions for root
    - Large programs can easily hit this limit [1]
- Unbounded loops
  - Pragma unroll to expand the loop
    - Kernel v5.3+ supports **bounded** loops

- How to handle ***exceptional*** cases?

[1] Alexei Starovoitov, bpf: improve verifier scalability. https://lwn.net/Articles/784571/

Build NFs with eBPF: Limitations

- eBPF programs are event-driven
  - Executed after a packet is received

1. Generate a packet as a consequence of an external event (e.g., a timeout)
   - Routing protocols, Spanning Tree Protocol (STP)
2. Putting packets on hold
   - Cannot stop the execution of a program
   - E.g., a router that has to retain a packet while discovering the MAC address of the next hop

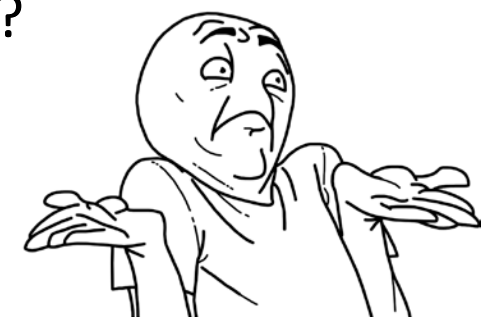
Build NFs with eBPF: Lessons Learned

- A lot of advanced features make eBPF a very good candidate for running data plane applications, however…
- …creating complex network applications require functionality that cannot be achieved given the eBPF restricted sandbox
  - Although many limitations are being “solved” by the eBPF community
- We need a solution that allows NF developer to:
  - Overcome eBPF limitations
  - Simplify the interaction with the eBPF environment
  - Provide abstraction typical of NFV that are not available in the general eBPF subsystem

Polycube: a Framework for eBPF-based Network Functions

Polycube: Goals and Challenges

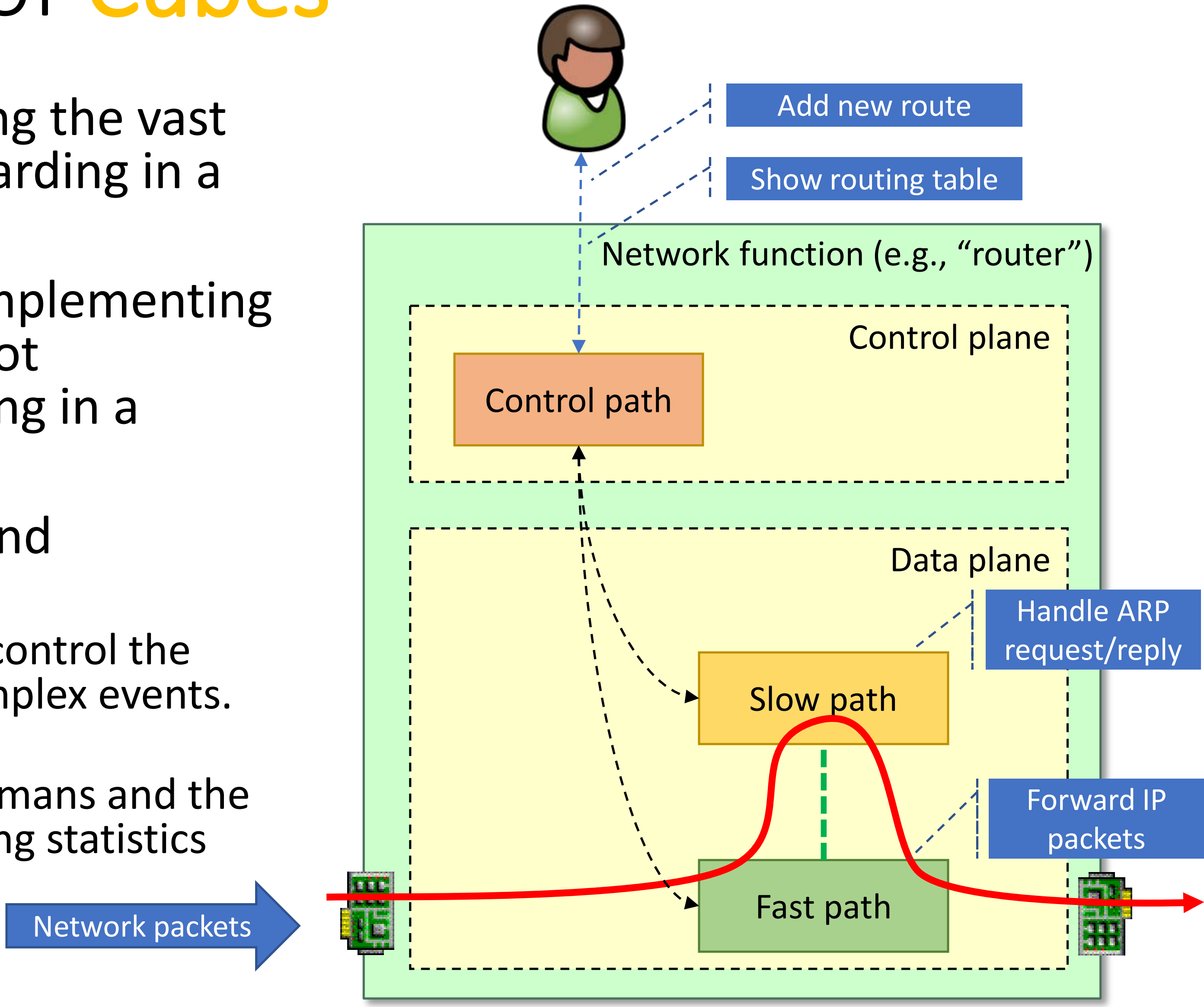
- Main goal: Enable NFV to the world of in-kernel packet processing
  - Myriad of userspace NFV framework (e.g., Netbricks, BESS, OpenBox)
  - In-kernel NFV?



- *Goal 1:* Common structure and abstractions of in-kernel NFs
  - **Virtual ports** from with traffic is received and sent out
  - Support for **Control** & **Data** Plane

Polycube: Structure of Cubes

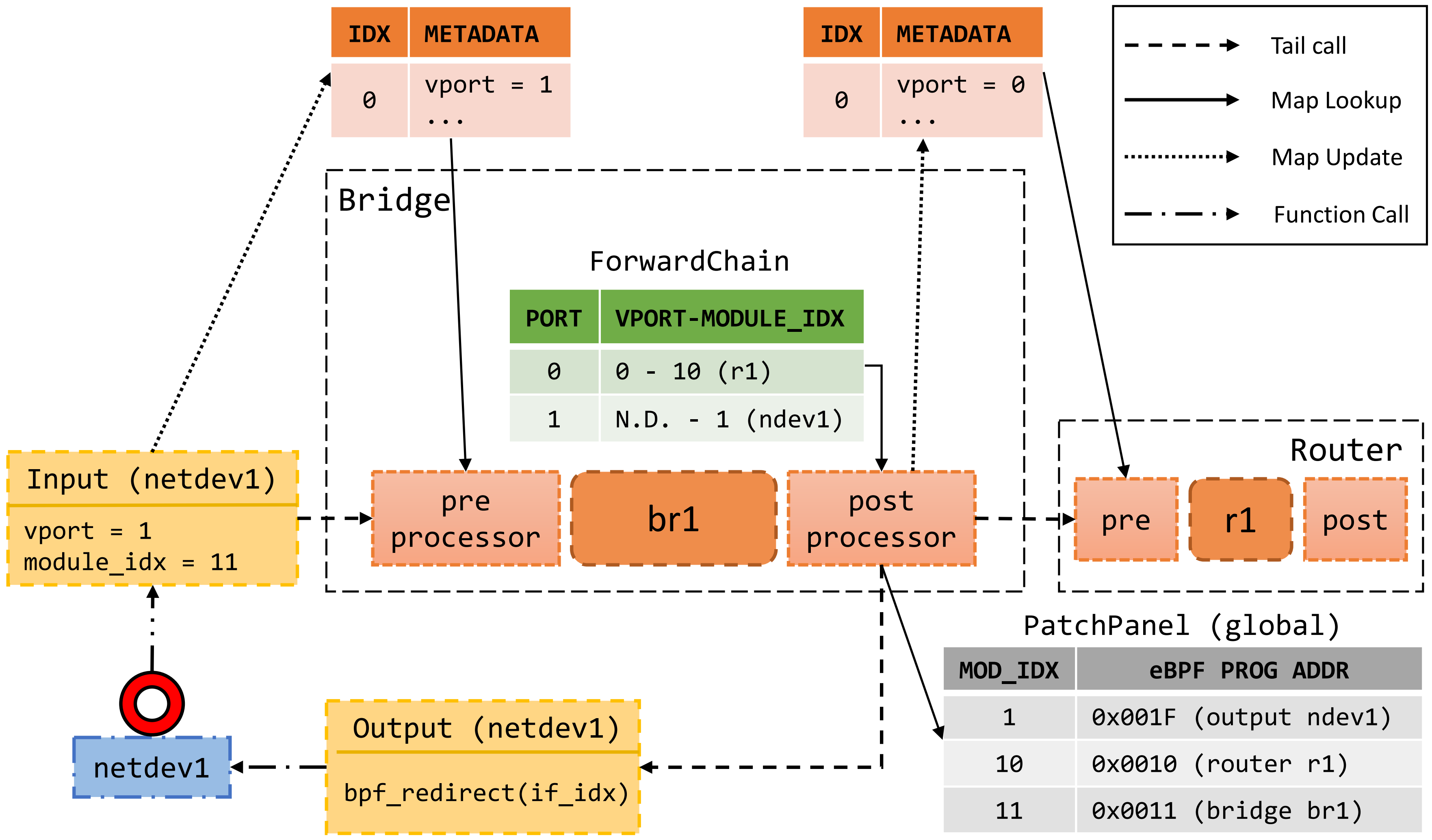
- **Fast path**: running in kernel, handling the vast majority of packets. E.g., plain forwarding in a router.
- **Slow path**: running in user space, implementing complex data plane tasks that are not supported by eBPF. E.g., ARP handling in a router.
- **Control path**: implements **control** and **management** tasks
  - Control: out-of-band tasks needed to control the dataplane and to react to possible complex events. E.g., Routing Protocols, Spanning Tree
  - Management: interaction between humans and the software, e.g., for configuration, reading statistics



Polycube: Goals and Challenges

- **Main goal**: Bring the power and innovation promised by NFV to the world of in-kernel packet processing
- *Goal 1:* Common structure and abstractions of in-kernel NFs
  - **Virtual ports** from with traffic is received and sent out
  - **Control** & **Data** Plane separation
- *Goal 2:* Programmable and extensible service chaining
  - Multiple **in-kernel** NF chains

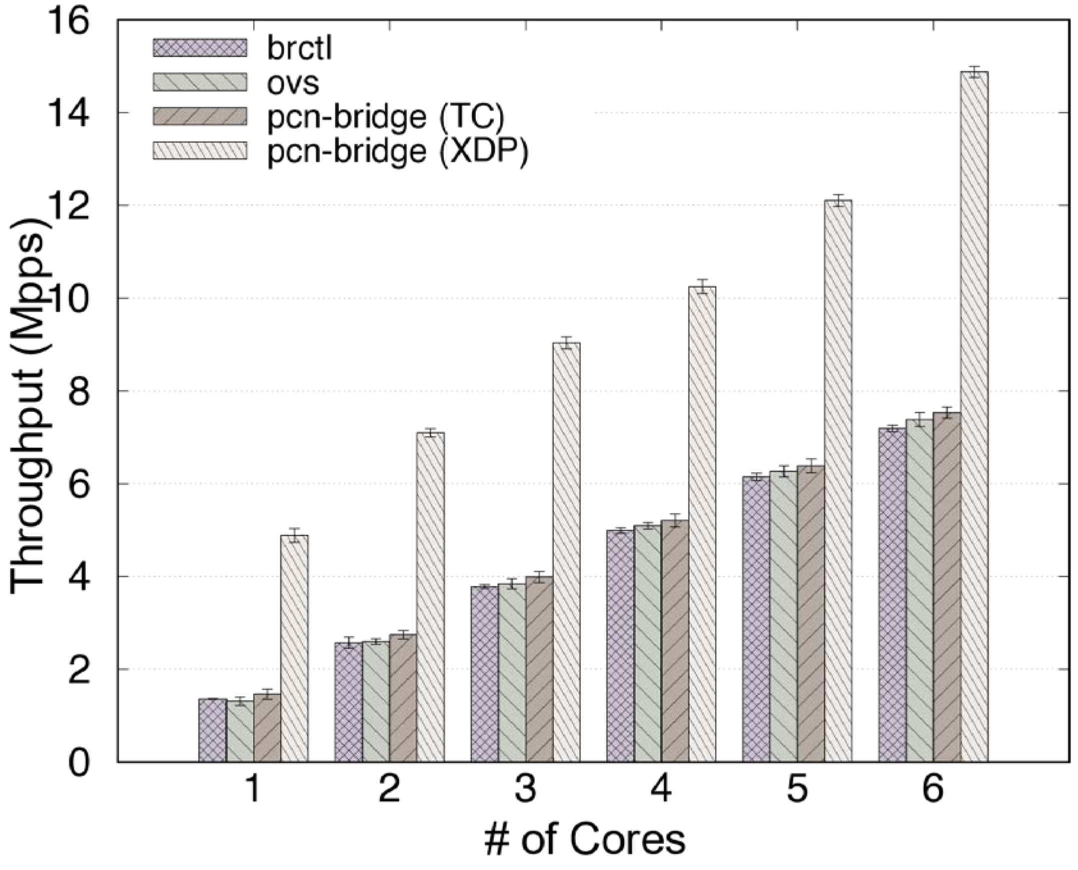
Polycube: Service Function Chaining

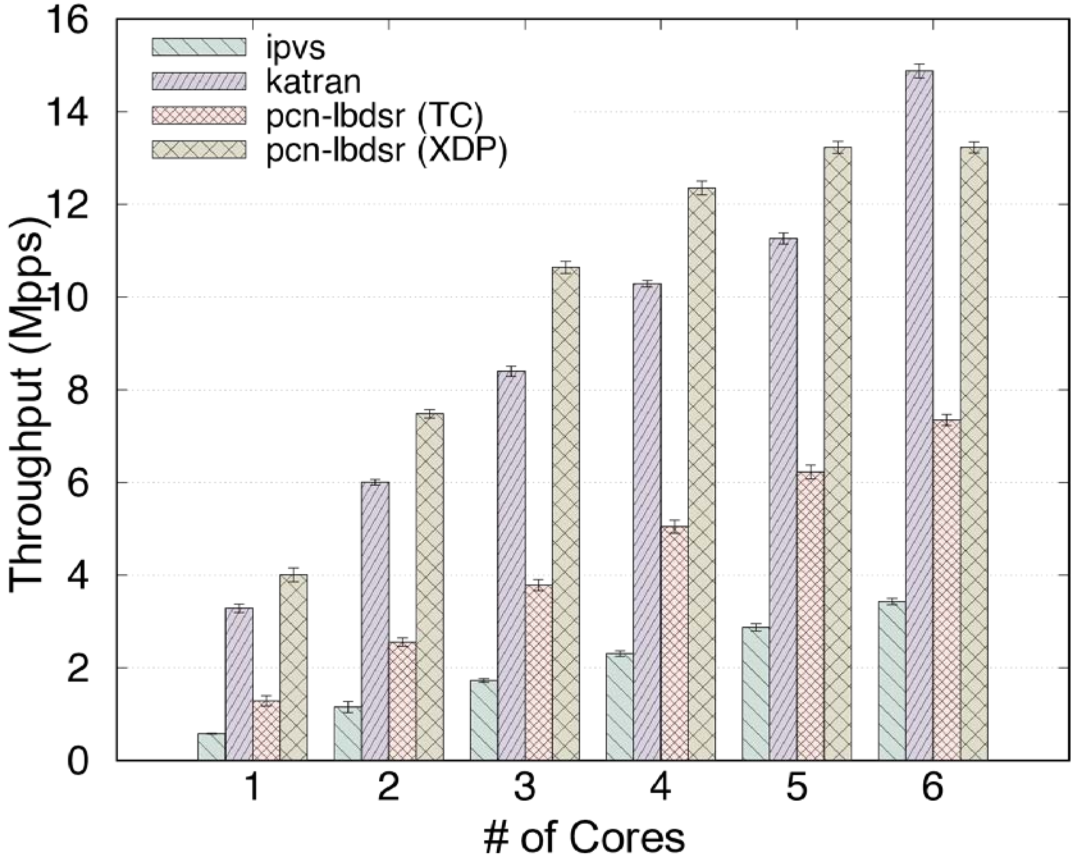


Polycube: Goals and Challenges

- **Main goal**: Bring the power and innovation promised by NFV to the world of in-kernel packet processing
- *Goal 1:* Common structure and abstractions of in-kernel NFs
  - **Virtual ports** from with traffic is received and sent out
  - **Control** & **Data** Plane separation
- *Goal 2:* Programmable and extensible service chaining
  - Multiple **in-kernel** NF chains
- *Goal 3:* Simple NF management and execution
  - Dynamic loading and update of existing programs
- *Goal 4:* Simplify development of control and management plane
  - Automatic control plane and REST API generation

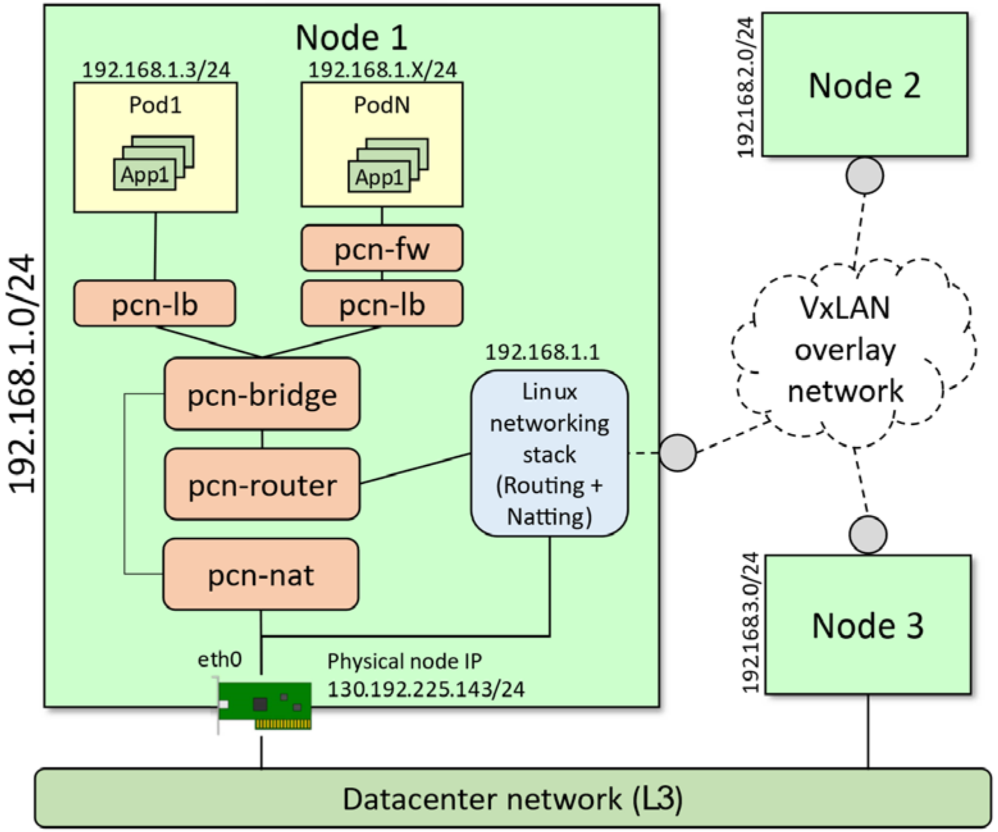
Evaluation of Polycube NFs



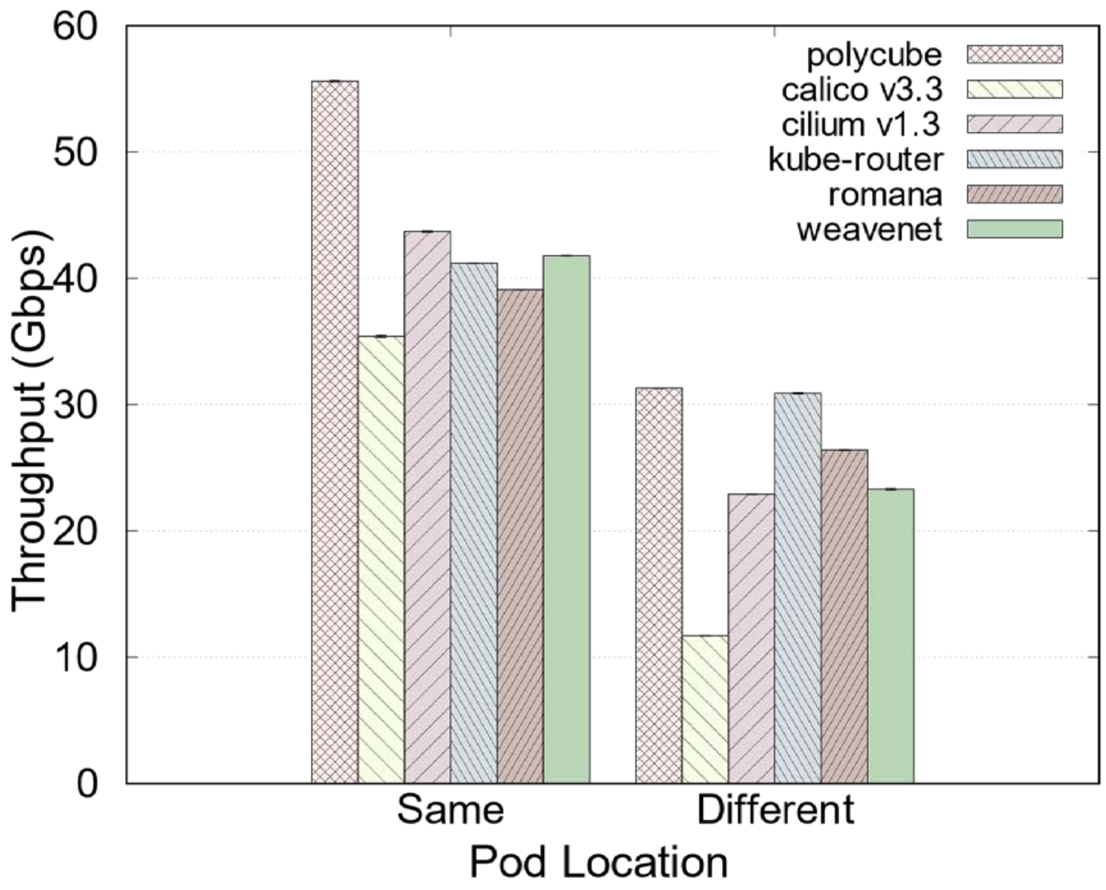


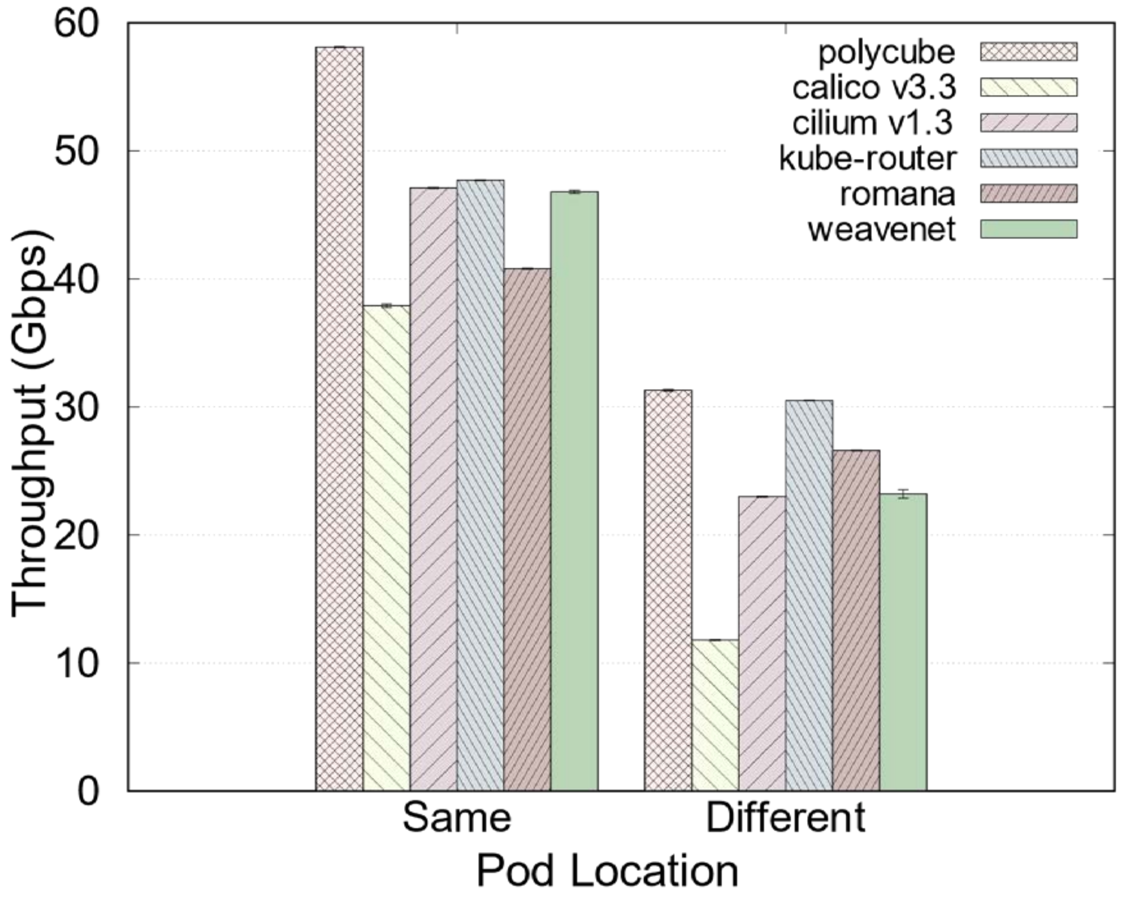
K8s Network Plugin Use Case

- Demonstrate the capability of creating complex network applications with Polycube
- Several independent Polycube services chained together
  - Some custom services to implement k8s specific functionality
- Full interaction with the Linux networking stack e.g., for tunneling and security purposes



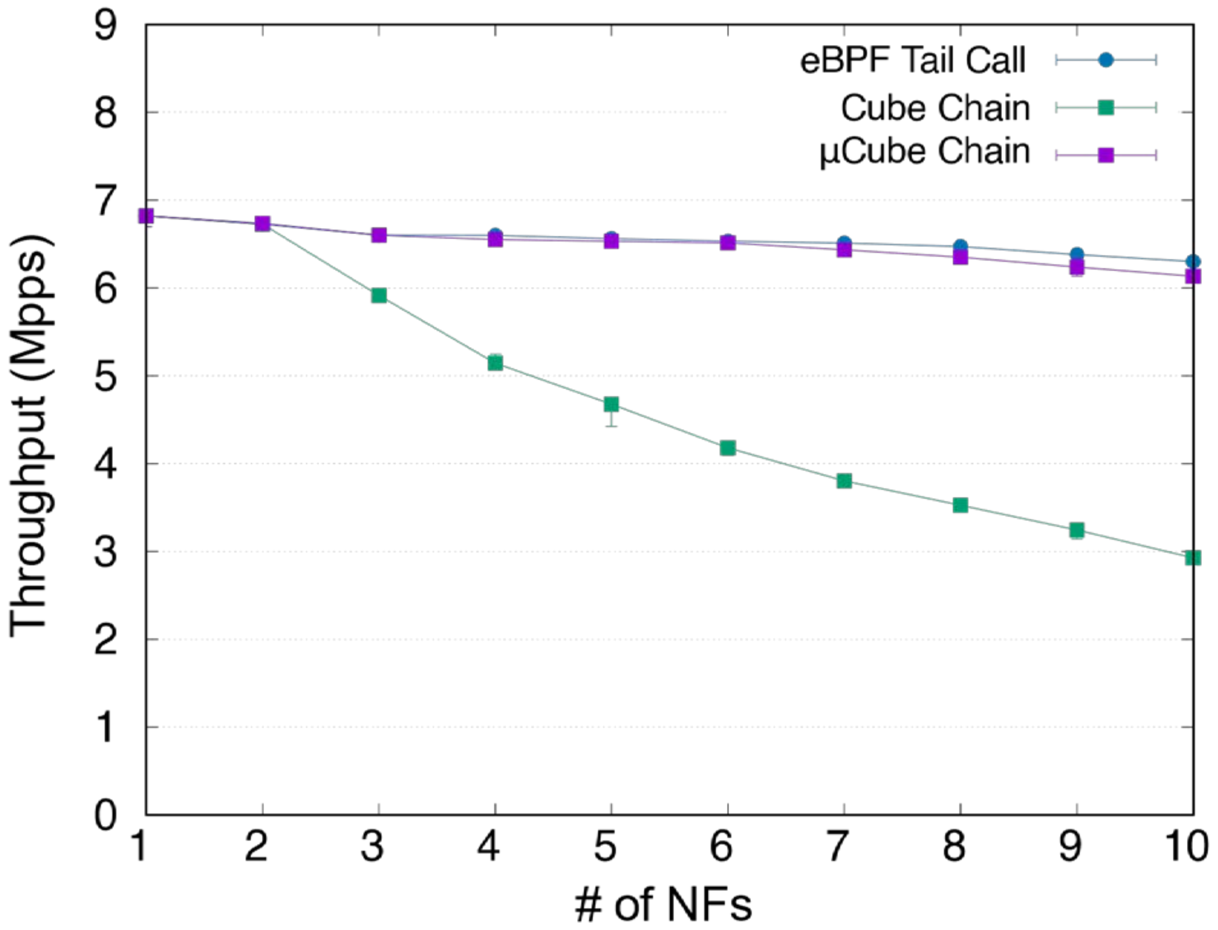
Polycube: K8s Plugin Performance





Polycube: Framework Overheads

Application	Through.	LoC (FP)	LoC (S/CP)
xdp_redirect	6.97Mpps	64	176
pcn_simplefwd (XDP)	6.86Mpps	53	56
tc_redirect	1.60Mpps	17	0
pcn_simplefwd (TC)	1.55Mpps	17	0



Polycube: Concluding Remarks

- First in-kernel NF framework based on eBPF
- Advantages:
  - Cubes can be dynamically injected/updated and optimized
  - Extend eBPF programming model with NF specific abstractions and solutions
  - Low overhead compared to “vanilla” eBPF
  - Performance
    - Better performance compared to in-kernel alternatives
  - Possibility to create complex in-kernel topologies by combining different services together
    - K8s Network Plugin
    - Bpf-iptables (next slides)

Bpf-iptables: Accelerating Linux Security with eBPF iptables

Best CCR Paper SIGCOMM 2020

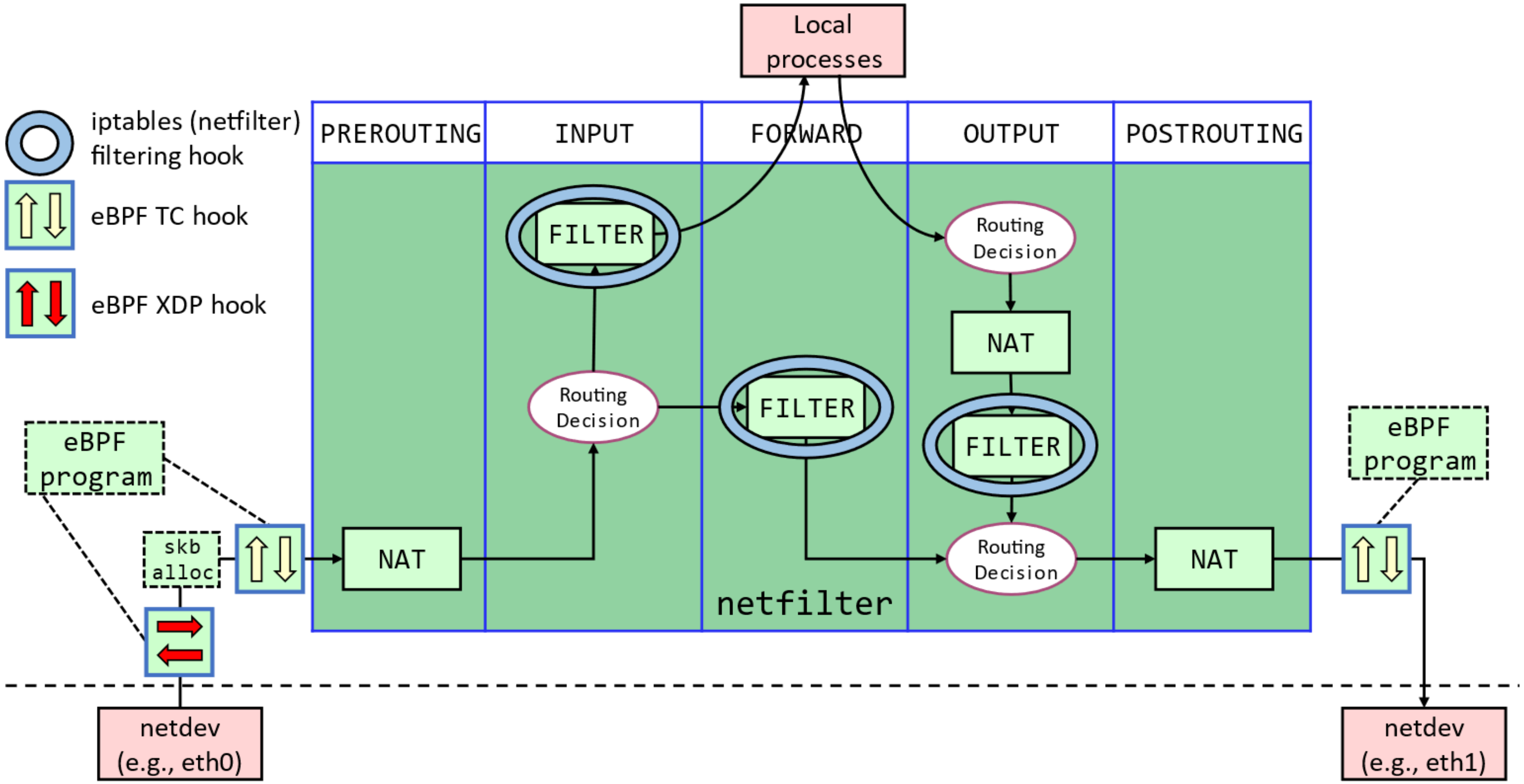
Bpf-iptables: Goal & Challenges

- Main goal: validate the architecture and show the advantages of the eBPF-based NF model

- Improve the `iptables` kernel implementation
  - One of the most used software todays
    - Speeding up its performance would improve consequentely all the other applications that rely on it

Goal #1: Preserve filtering semantic

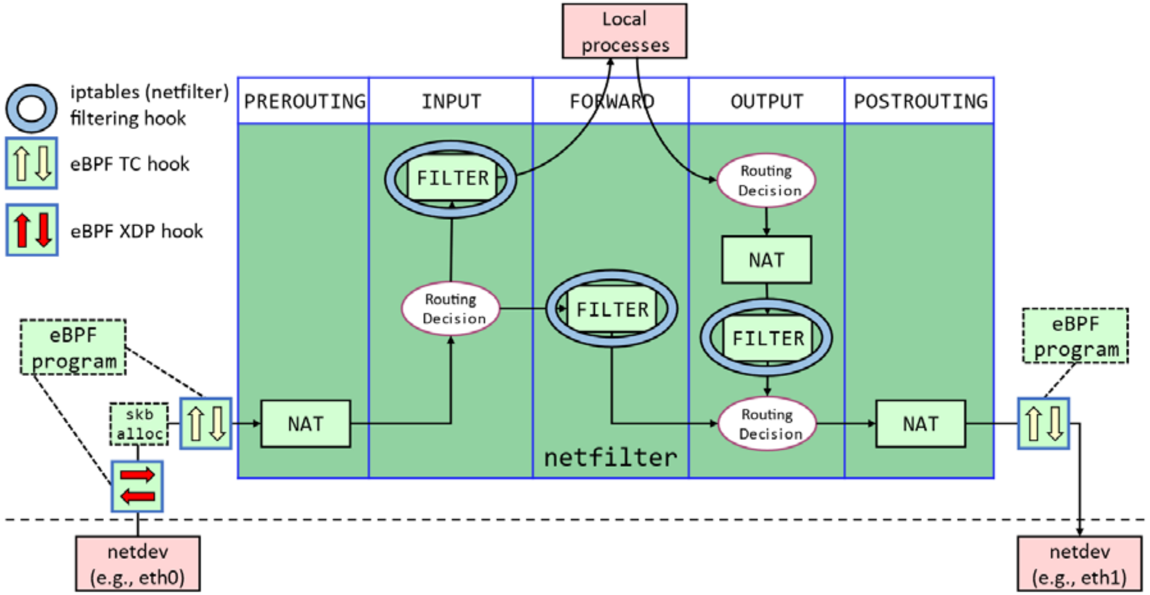
- Small of subtle differences could create serious security problems



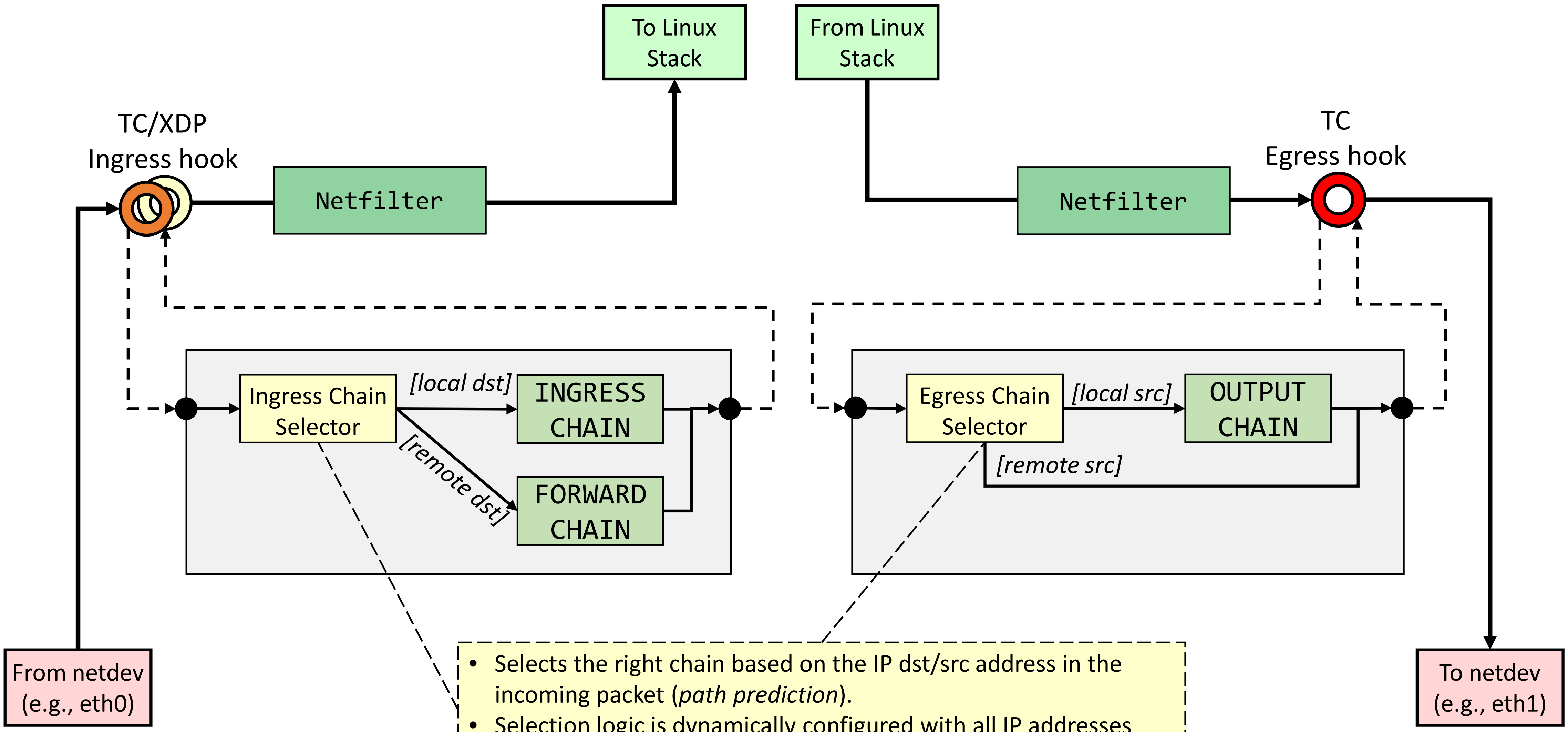
Challenge #1: Preserve filtering semantic

- **Two** (TC_INGRESS/XDP_INGRESS and TC_EGRESS) hooks, which must **emulate three** chains
  - No hooks available in eBPF to easily intercept locally terminated/generated traffic
  - The eBPF code has to process only the packets that would hit each specific chain (INPUT, FORWARD, OUTPUT)
    - We have to "guess" very early which
    - chain will be hit
  - XDP **alone** is not enough

  (no support for egress traffic)



Solution #1: Preserve filtering semantic



- Selects the right chain based on the IP dst/src address in the incoming packet (*path prediction*).
- Selection logic is dynamically configured with all IP addresses visible from the root namespace (intercepts NETLINK messages).

Goal #2: Improve iptables classification

- Linear search algorithm used by `iptables`
  - Poor performance when lot of rules are used (e.g., k8s)

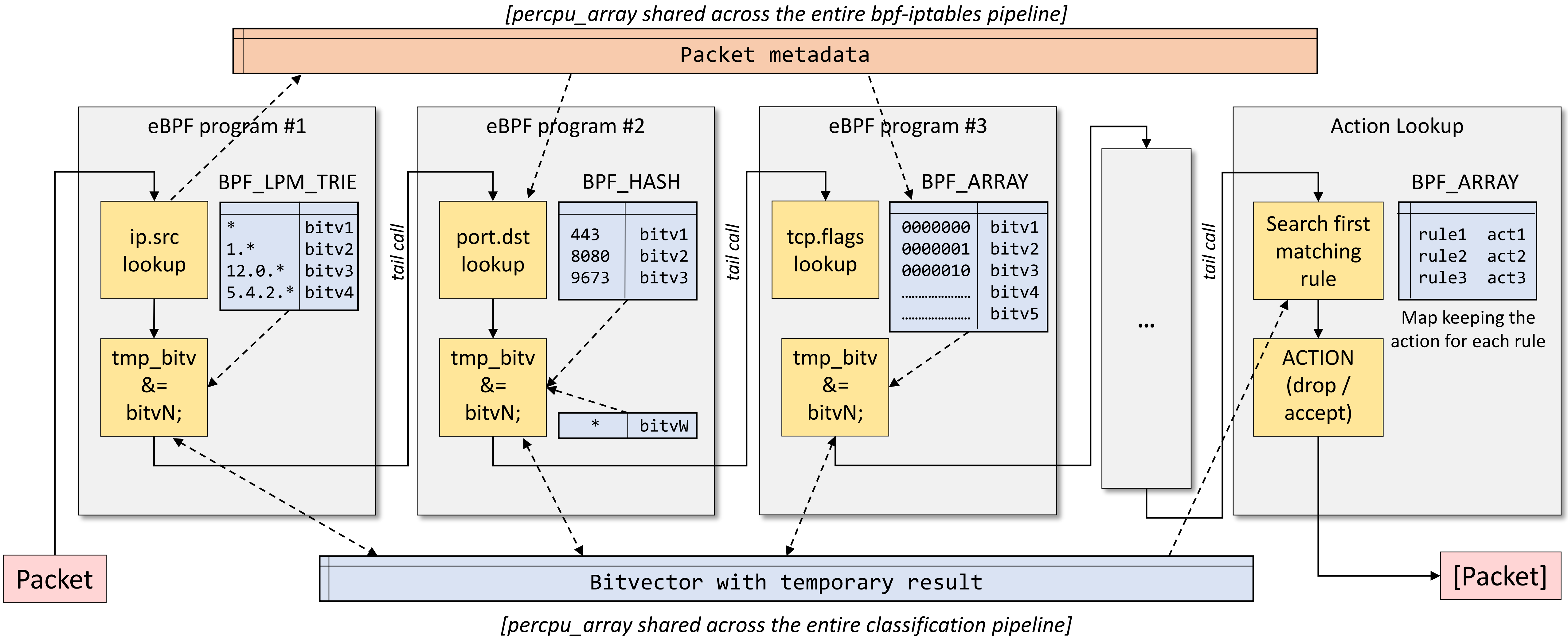
Challenge #2: Deal with eBPF limitations

- eBPF constraints for the selection of the algorithm:
  - Feasible with available eBPF data structures (maps)
  - Possibility to rearrange the code in multiple eBPF programs to overcome the limitation of the maximum number of instructions per program

- Linear Bit Vector Search [1] (LBVS)

[1] T. Lakshman and D.Stidialis. High speed policy-based packet forwarding using efficient multi-dimensional range matching. In Proc. ACM Sigcomm'98, Sept. 1998

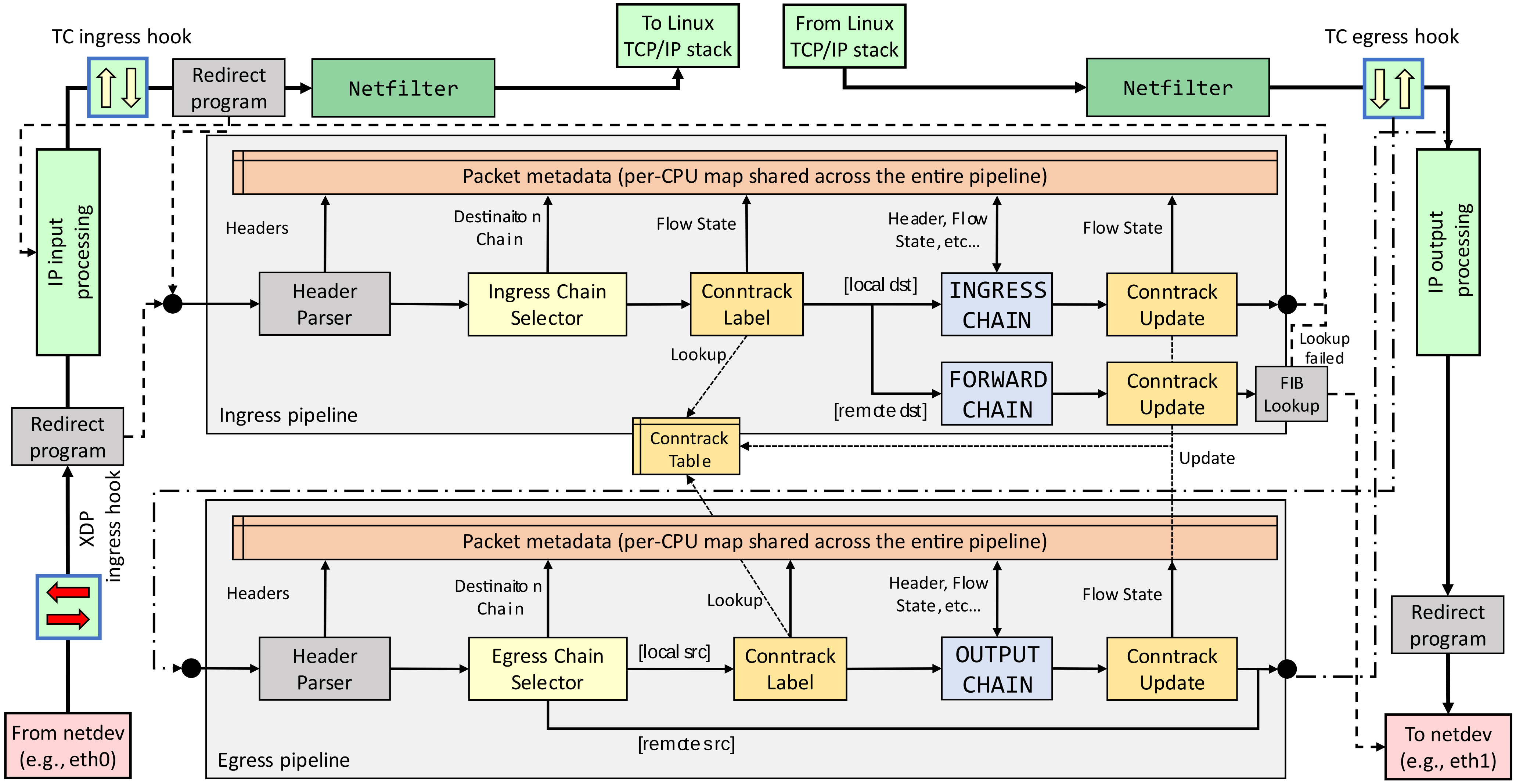
Bpf-iptables: Classification Pipeline



Goal #3: Support for stateful filters

- Netfilter tracks the state of TCP/UDP/ICMP connections and stores them into the Linux *conntrack* table
- ebpf-iptables is executed before Netfilter
  - Packets are classified/dropped before the Linux conntrack is hit
- To support stateful filters, ebpf-iptables has to implement its own connection tracking module

Bpf-iptables: Overall Architecture

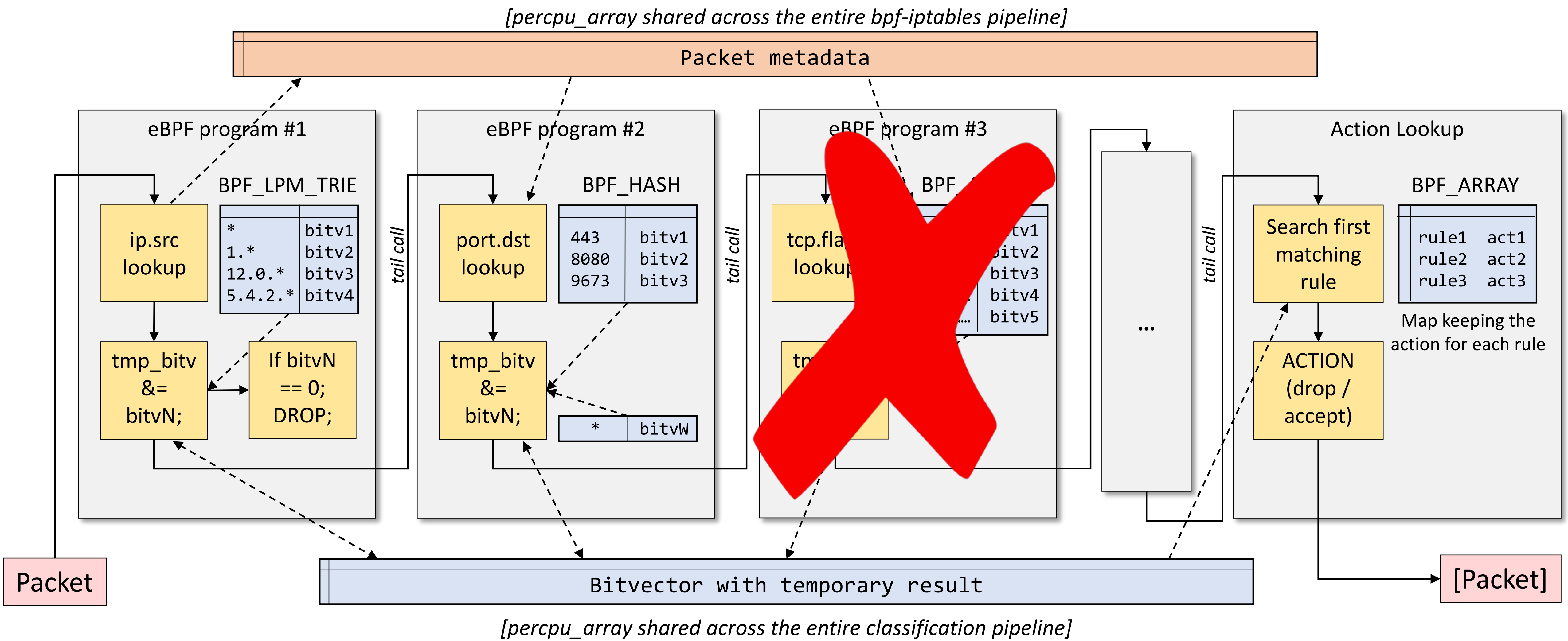


Bpf-iptables: Control/Data Plane Optimizations

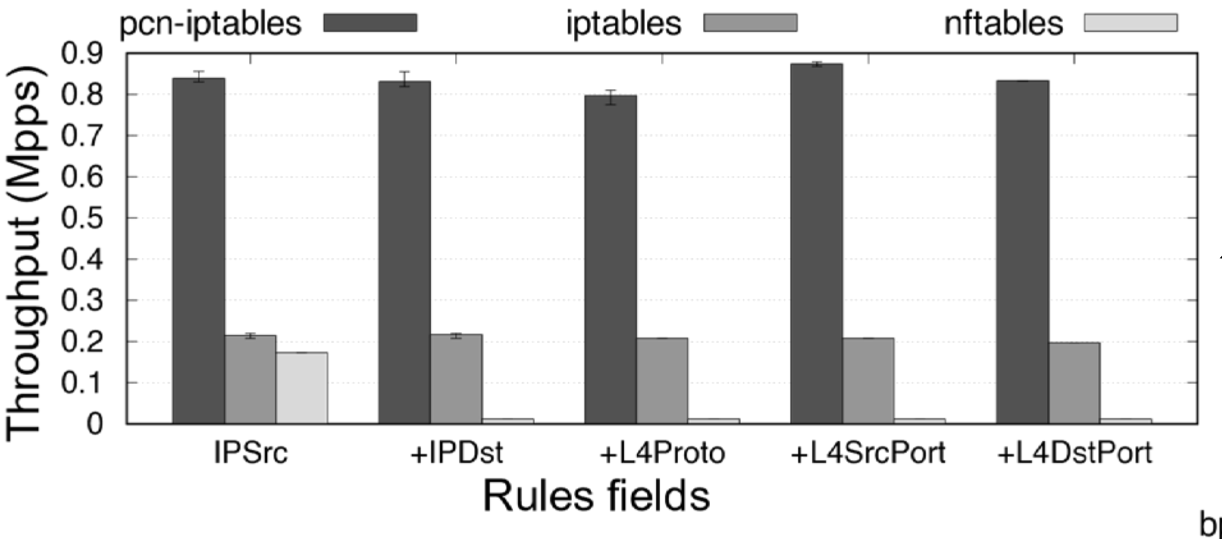
- Lot of optimizations can be applied by knowing the structure and type of rules installed
  - Thanks to the dynamic injection feature of eBPF
  - Thanks to the dynamic pipeline substitution of Polycube

- Optimizations:
  - **Early pipeline break**
  - **Minimum processing pipeline**
  - HOmogeneous RUleset analySis (HORUS)
  - Many others (e.g., accept established, use Linux FIB)

Bpf-iptables: Optimizations

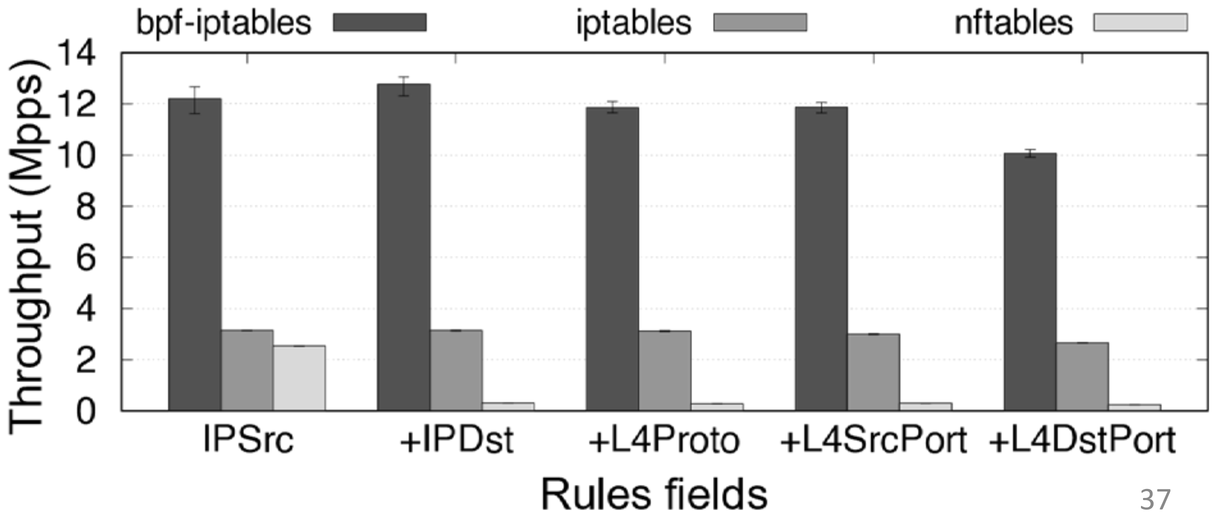


Evaluation: Rule Complexity



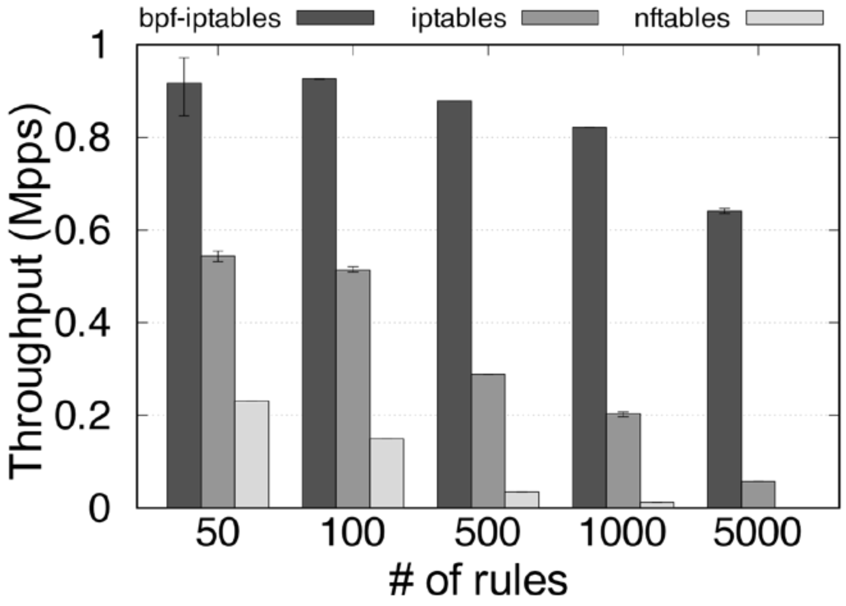
UDP Throughput
(Single core)

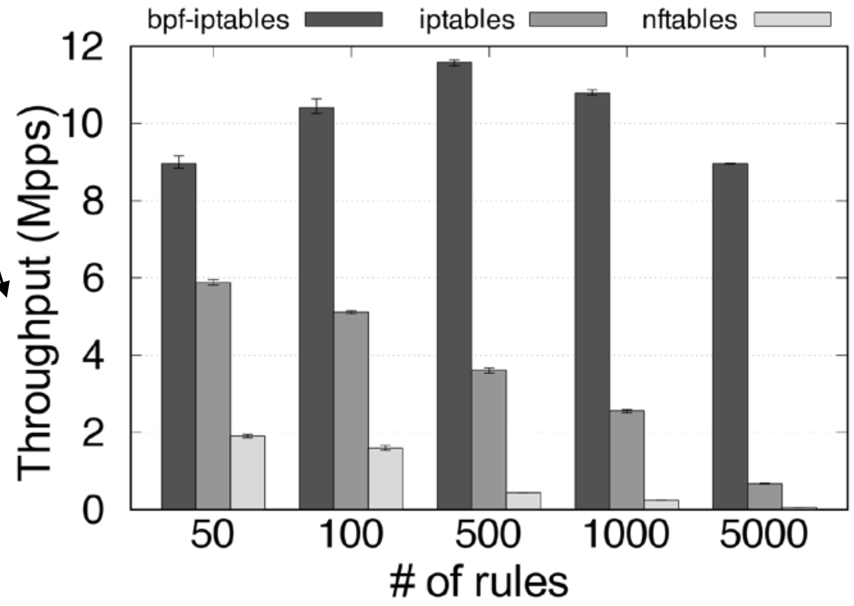
UDP Throughput
(Multi 14 core)



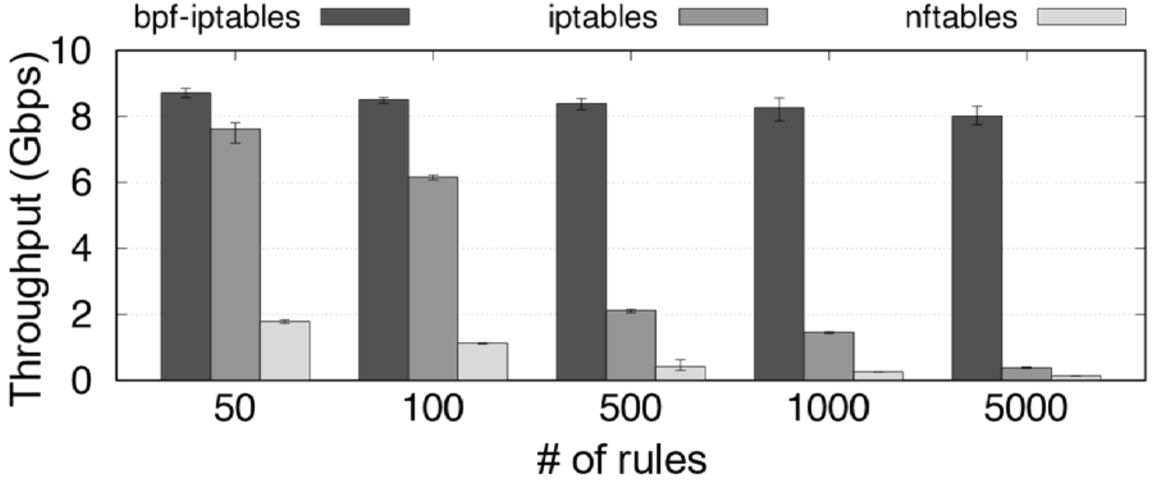
Evaluation: Scalability

UDP Throughput





TCP Throughput



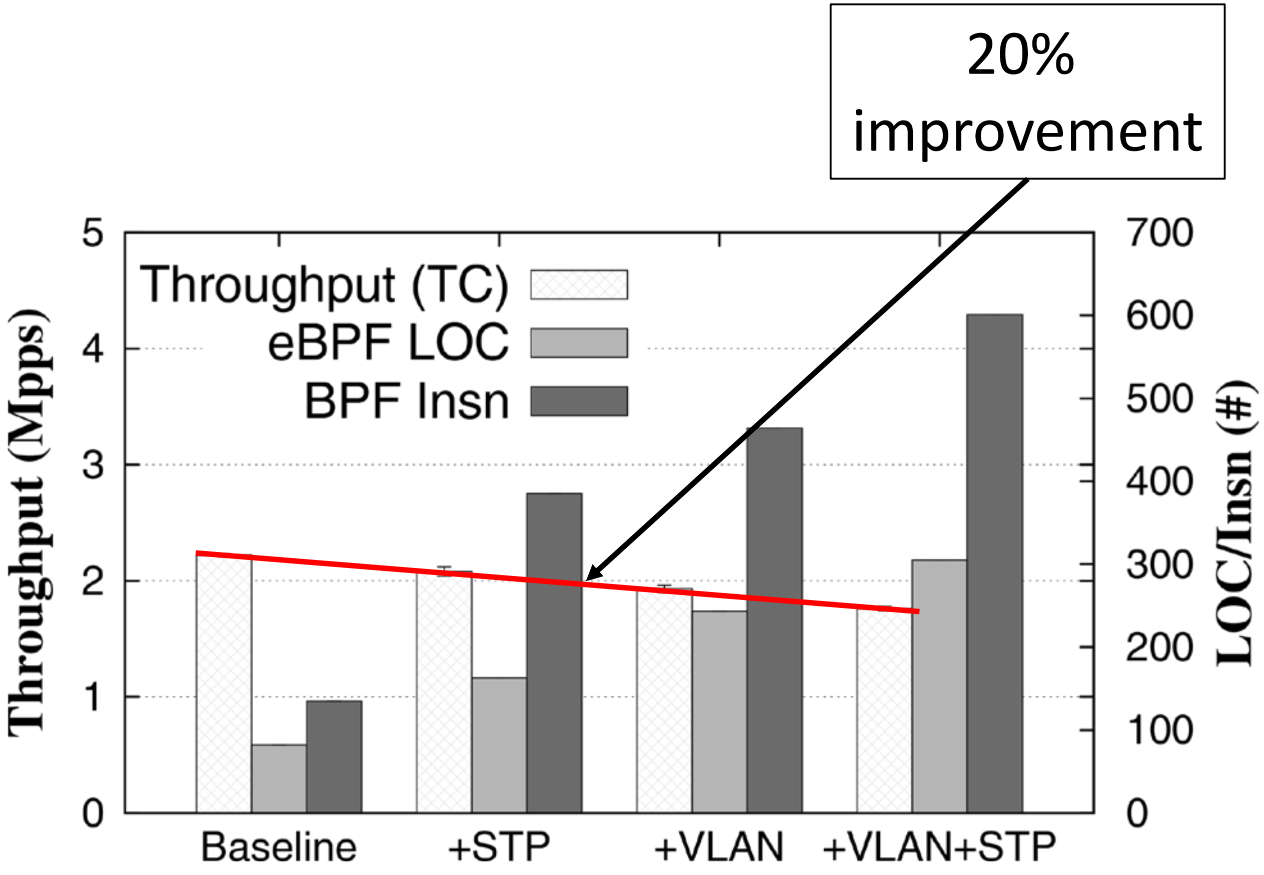
Bpf-iptables: Conclusions

- Bpf-iptables demonstrated huge advantages over existing in-kernel solutions, in particular when a high number of rules are used
  - No custom kernel required (min v4.14+, 4.18+ for some optimizations)

- Performance improvement thanks to control plane optimizations
- **Can we apply those optimizations automatically?**

Looking Ahead:
Automatic Optimizations of Software Data Planes

Observation #1: Performance depend on runtime configuration

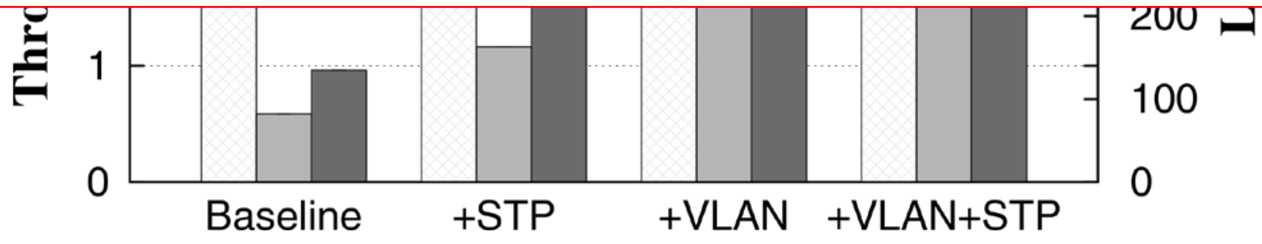


Observation #1: Performance depend on runtime configuration

20% improvement

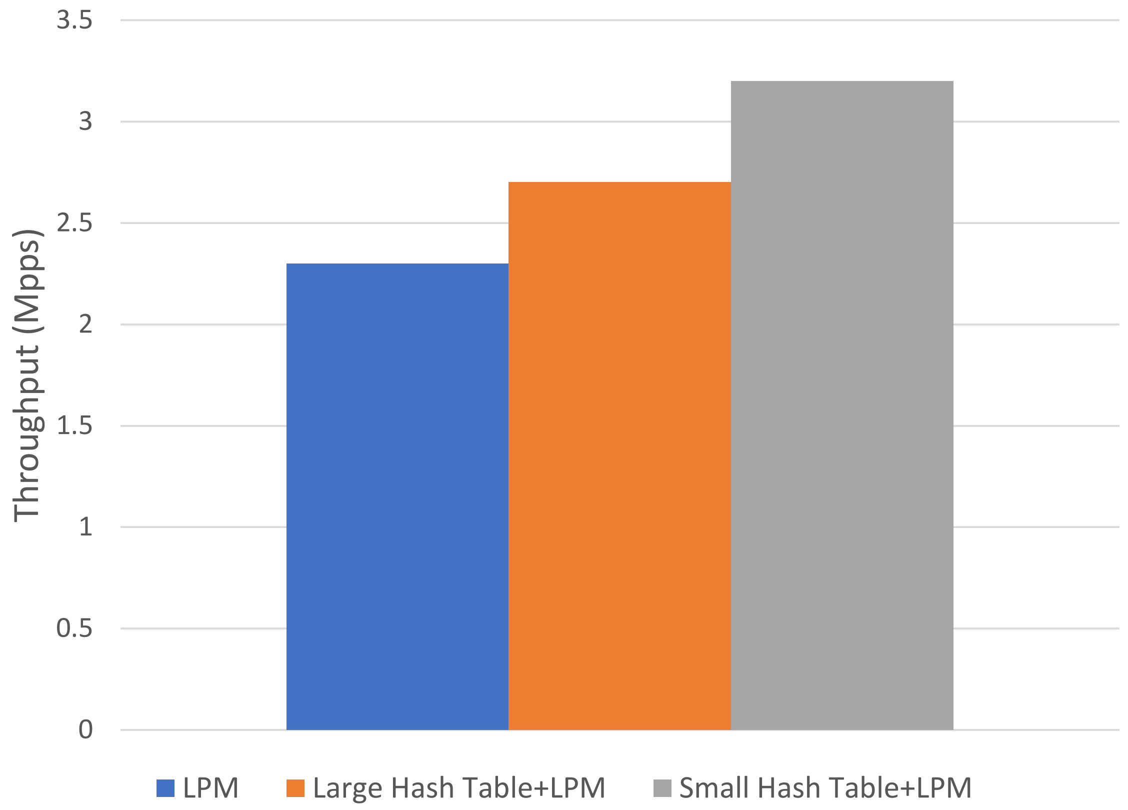
This is a call for dead code elimination:

Prunes instructions unreachable within the runtime configuration



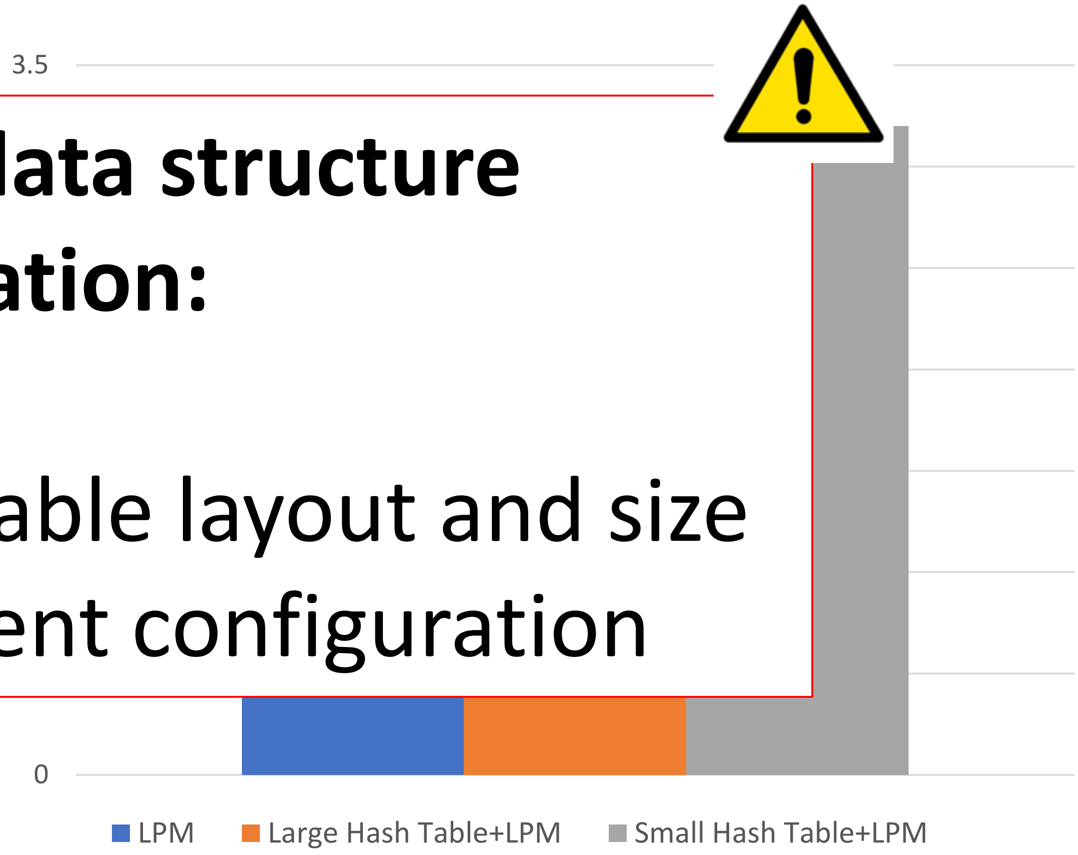
Observation #2: Performance depend on runtime table content

- At any point, many NF features may go unused.
- May run with empty tables
- Non-appropriate data structures
  - Layout
  - Size
  - Algorithm



Observation #2: Performance depend on runtime table content

- At any point, many NF features may go unused.
- May run with empty tables
- Non-a
  - Lay
  - Size
  - Alg



This is a call for data structure specialization:

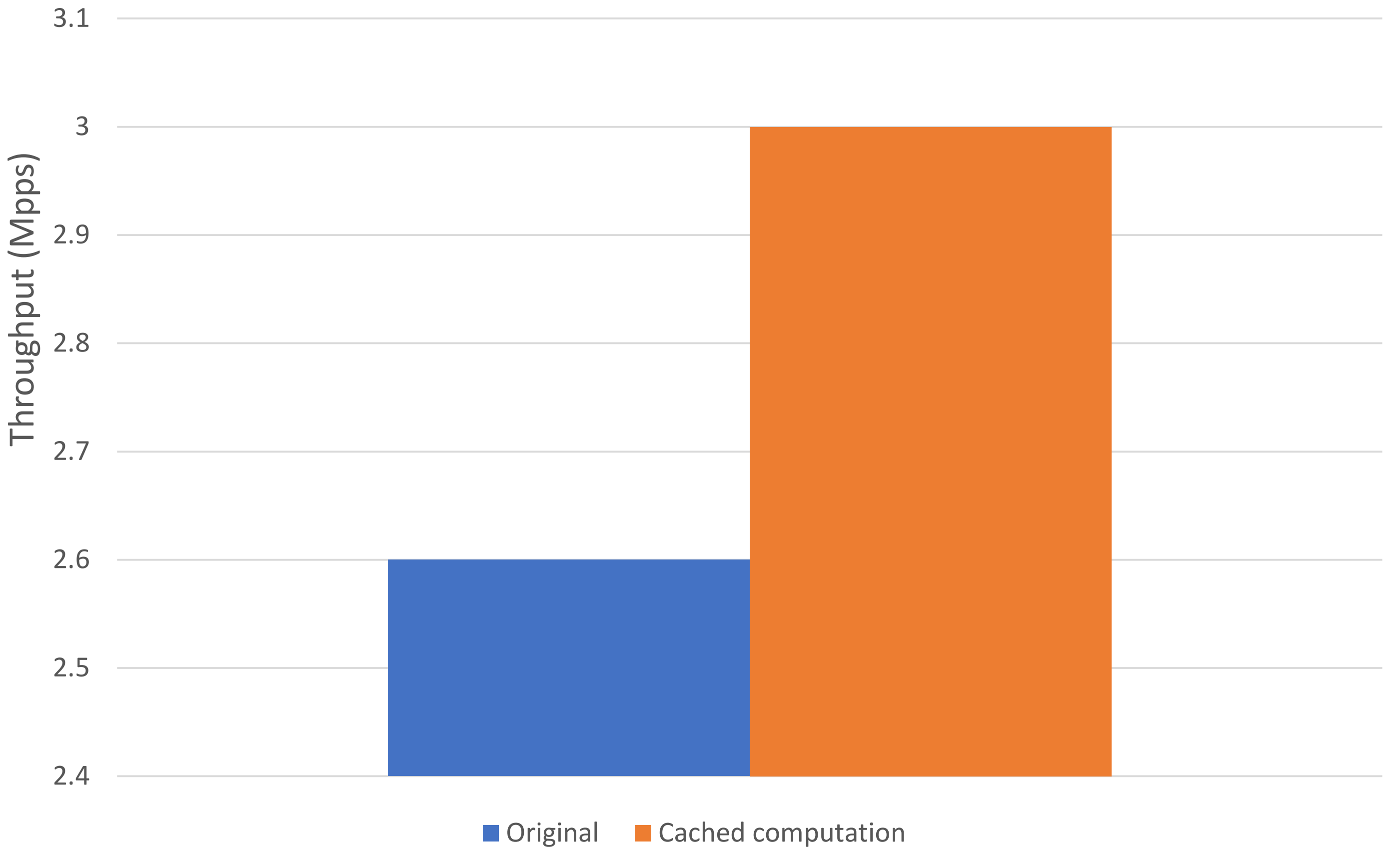
Change match/action table layout and size to better fit the current configuration

Observation #3: Performance depend on run-time traffic patterns

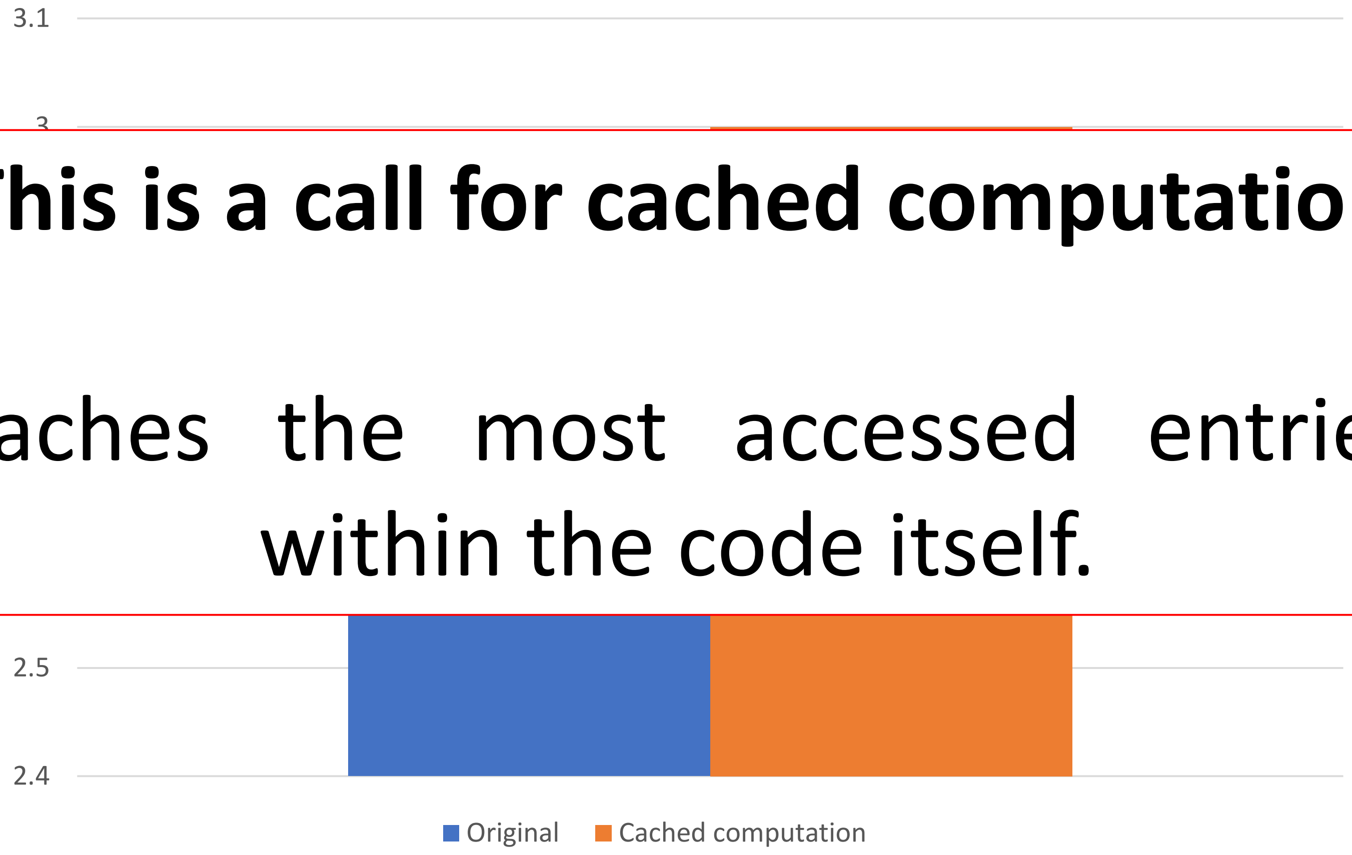
- What if we have 100k flows but only the 1% contributes to 99% of the traffic?

```
if !top-5-flows:
    goto  [Access match/action table]
else:
    goto  [Here the result!]
```

Observation #3: Performance depend on run-time traffic patterns



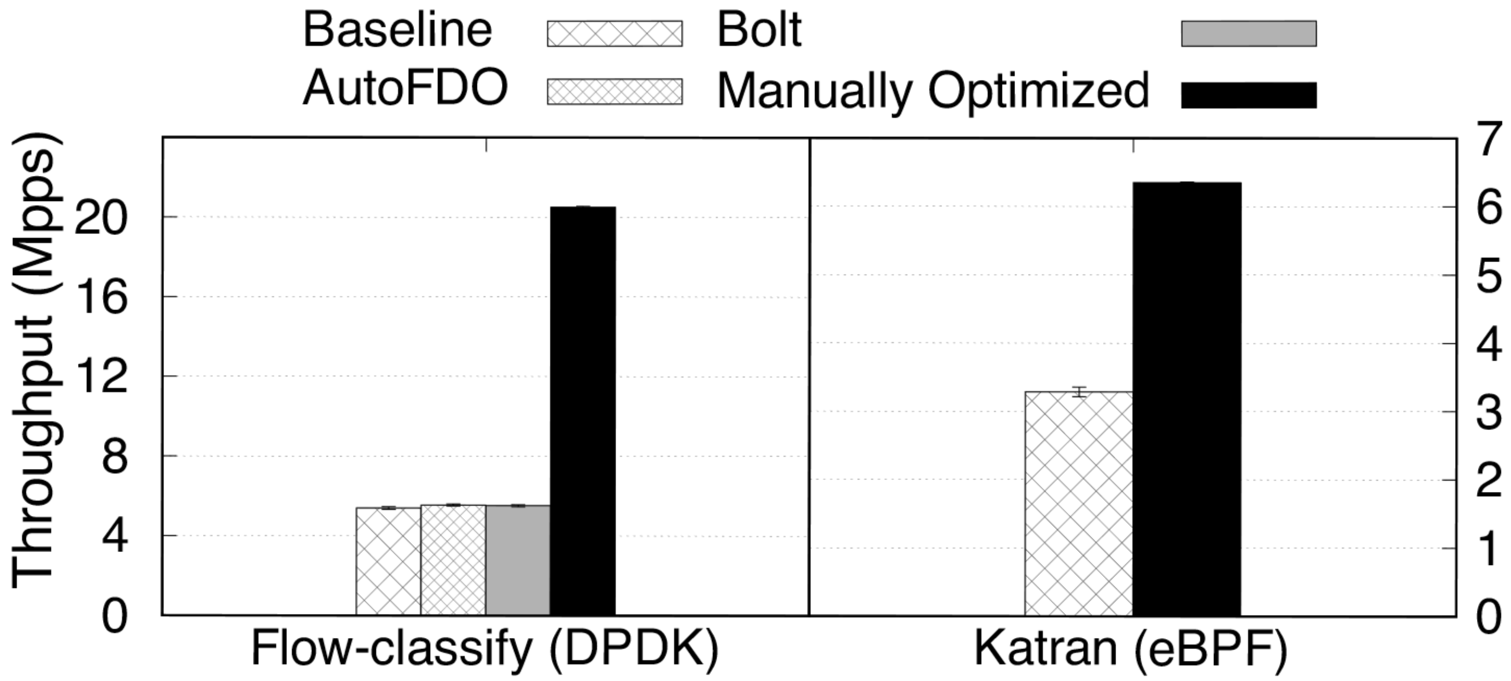
Observation #3: Performance depend on run-time traffic patterns



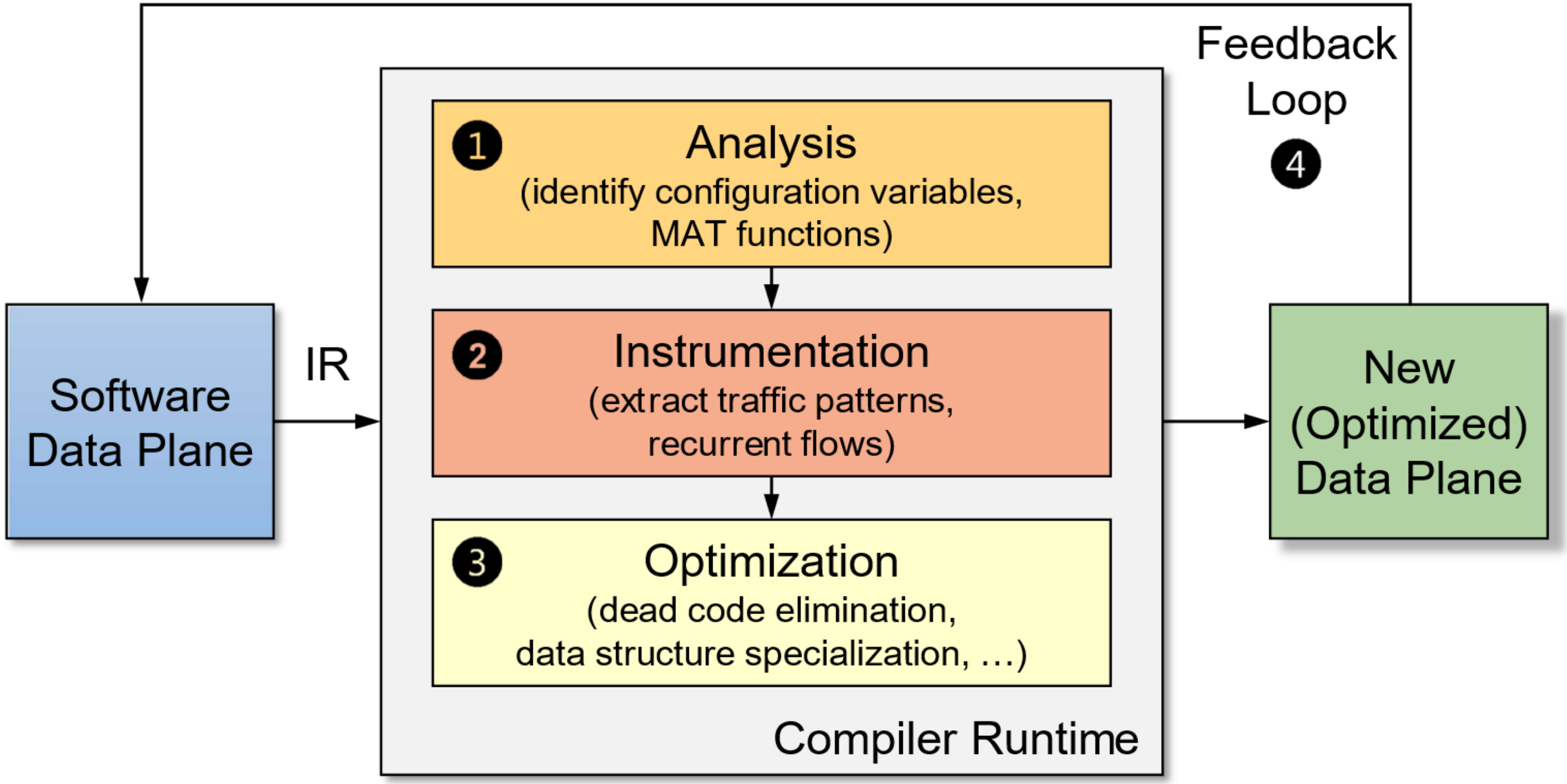
This is a call for cached computation

Caches the most accessed entries within the code itself.

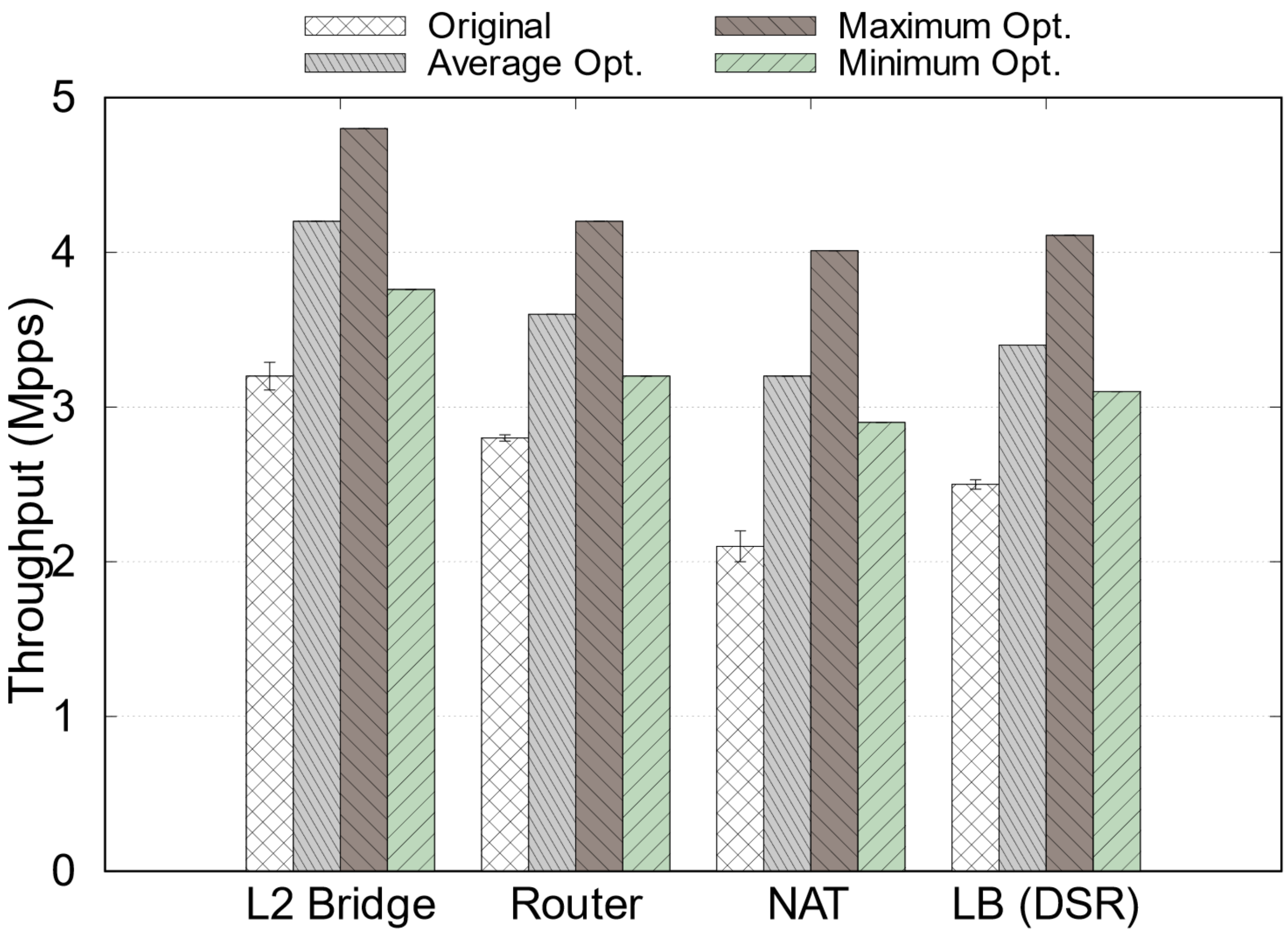
Any help from the literature?



High-level Architecture



Preliminary Performance Evaluation



Conclusions

- So far, validated on eBPF-based (in-kernel) NFs
- Framework is general enough to be applied to DPDK-based NFs
  - Optimizations at the IR level
  - More fine-grained control of parameters (e.g., batching, pre-fetching) not possible with eBPF
- Better understanding of the different configuration parameters and how the impact in the overall performance
  - E.g., some opts may change «original» performance patterns
  - Machine learning to decide right set of optimizations
  - Model for NF performance prediction (e.g., Bolt [1])

[1] Iyer, Rishabh, et al. "Performance contracts for software network functions." *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 2019.

Concluding Remarks

Concluding Remarks

- We have explored the challenges and limitations of a new paradigm to build in-kernel packet processing applications with eBPF
  - **Polycube**: a framework that simplify the development and deployment of in-kernel network services
    - Micro-service approach applied to NFs
    - Cloud-native friendly
  - **BPF-iptables**: demonstrate the power of the eBPF/Polycube environment and programming model to enhance the performance of *iptables* (one of the most used software todays)
  - **Kecleon**: enables the possibility to automatically re-compile a NF (without any user intervention) to better fits the surrounding runtime conditions

Publications

- Journals
  - Miano, Sebastiano; Bertrone, Matteo; Risso, Fulvio; Vasquez Bernal, Mauricio; Lu, Yunsong; Pi, J.
    Securing Linux with a Faster and Scalable Iptables
    In: *ACM SIGCOMM Computer Communication Review, Volume 49, Issue 3 (Best CCR Paper SIGCOMM 2020)*
  - Miano, Sebastiano; Doriguzzi-Corin, Roberto; Risso, Fulvio; Siracusa, Domenico; Sommese, Raffaele
    Introducing SmartNICs in Server-Based Data Plane Processing: The DDoS Mitigation Use Case
    In: *IEEE Access, Volume 7*
  - Miano, Sebastiano; Risso, Fulvio
    Transforming a Traditional Home Gateway into a Hardware-accelerated SDN Switch
    In: *International Journal of Electrical and Computer Engineering (IJECE)*
- Conferences
  - Miano, Sebastiano; Risso, Fulvio
    A Micro-service Approach for Cloud-Native Network Services
    In: *ACM Symposium on SDN Research (SOSR) Demo, Santa Clara (CA)*
  - Miano, Sebastiano; Bertrone, Matteo; Risso, Fulvio; Vasquez Bernal, Mauricio; Lu, Y.; Pi, J.; Shaikh, A.
    A Service-Agnostic Software Framework for Fast and Efficient In-Kernel Network Services
    In: *15th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*
  - Miano, Sebastiano; Bertrone, Matteo; Risso, Fulvio; Tumulo, Massimo; Vasquez Bernal, Mauricio.
    Creating Complex Network Services with eBPF: Experience and Lessons Learned
    In: *19th IEEE International Conference on High Performance Switching and Routing (HPSR), Bucharest (RO)*
  - Bertrone, Matteo; Miano, Sebastiano; Risso, Fulvio; Tumulo, Massimo.
    Accelerating Linux with eBPF Iptables
    In: *ACM SIGCOMM 2018 Conference Posters and Demos, Budapest (H)*
  - Bertrone, Matteo; Miano, Sebastiano; Pi, Jianwen; Risso, Fulvio; Tumolo, Massimo.
    Toward an eBPF-based clone of iptables
    In: *Netdev 0x12, The Technical Conference on Linux Networking, Montréal (Canada)*
  - Miano, Sebastiano; Risso, Fulvio; Woesner, Hagen.
    Partial offloading of OpenFlow rules on a traditional hardware switch ASIC
    In: *3rd IEEE Conference on Network Softwarization (NetSoft), Bologna (IT)*
  - Bonafiglia, Roberto; Miano, Sebastiano; Nuccio, Sergio; Risso, Fulvio; Sapio, Amedeo.
    Enabling NFV Services on Resource-Constrained CPEs
    In: *5th IEEE Conference on Cloud Networking (CloudNet), Pisa (IT)*