Evaluation of virtualization and traffic filtering methods for container networks

Łukasz Makowski
makowski@uva.nl

Cees de Laat
delaat@uva.nl

Paola Grosso
pgrosso@uva.nl

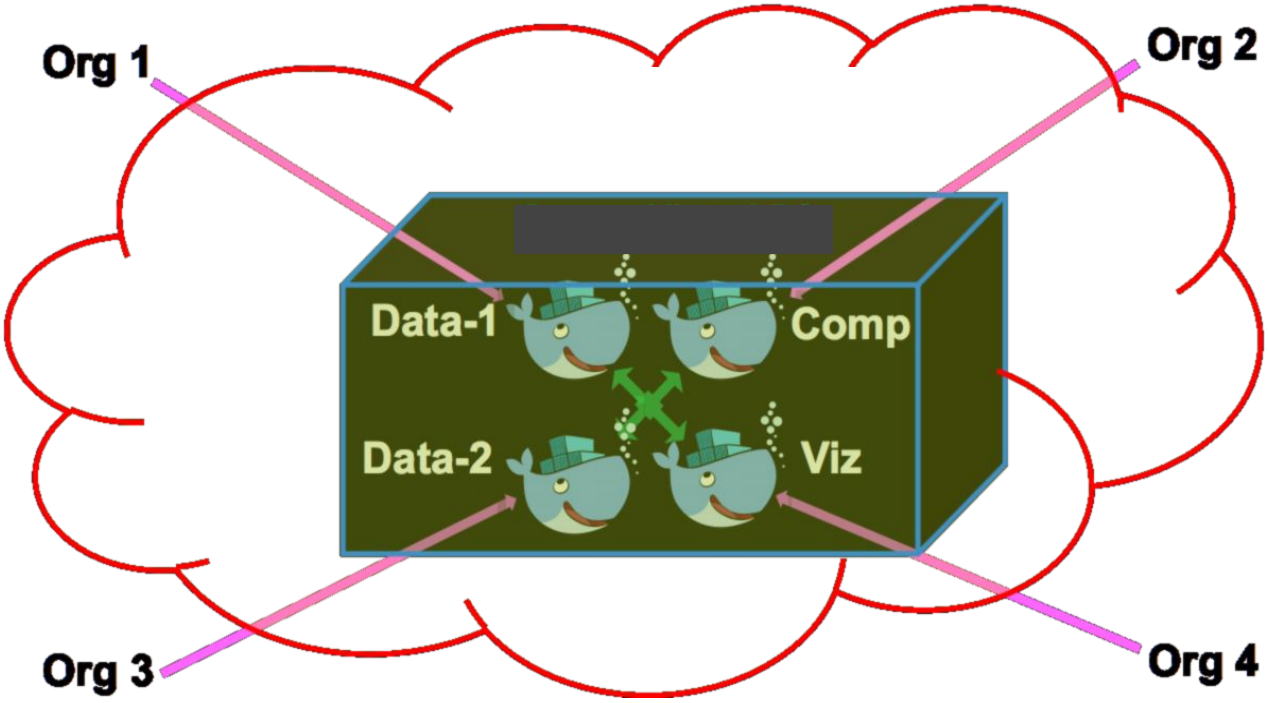
UNIVERSITEIT VAN AMSTERDAM

System and Network Engineering

SURF NET

Our goal: Improving on scientific workloads

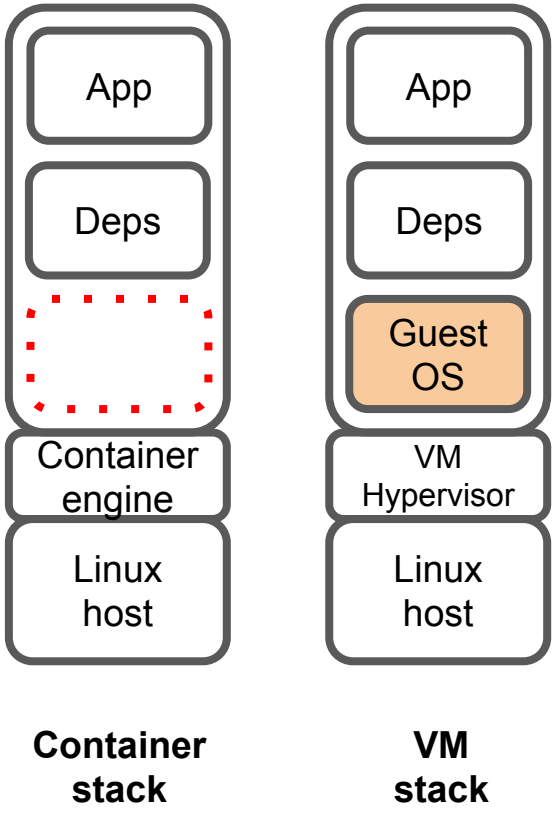
- Digital data sharing
- Supporting multi-organisation collaboration



Containers - quick recap

Why to use?

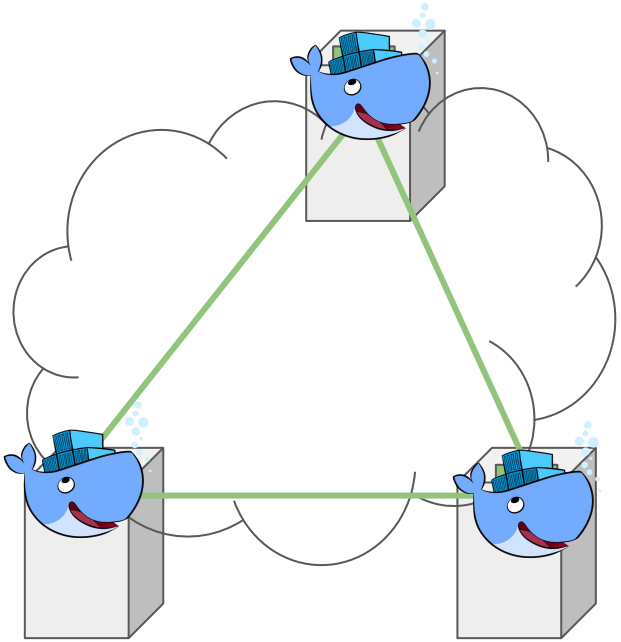
- Lightweight (when comparing to a VM)
- Makes application more portable
- Fast startup



Containers - virtual networks

Why do containers need virtual networks?

- Service may consist of groups of containers
- Each group can have tens, hundreds of them
- Imagine containers are spread across different hosts…
  - Different networks… data-centers… cloud providers...



It’s simply useful to provide a flat network not bound up with the underlay infrastructure

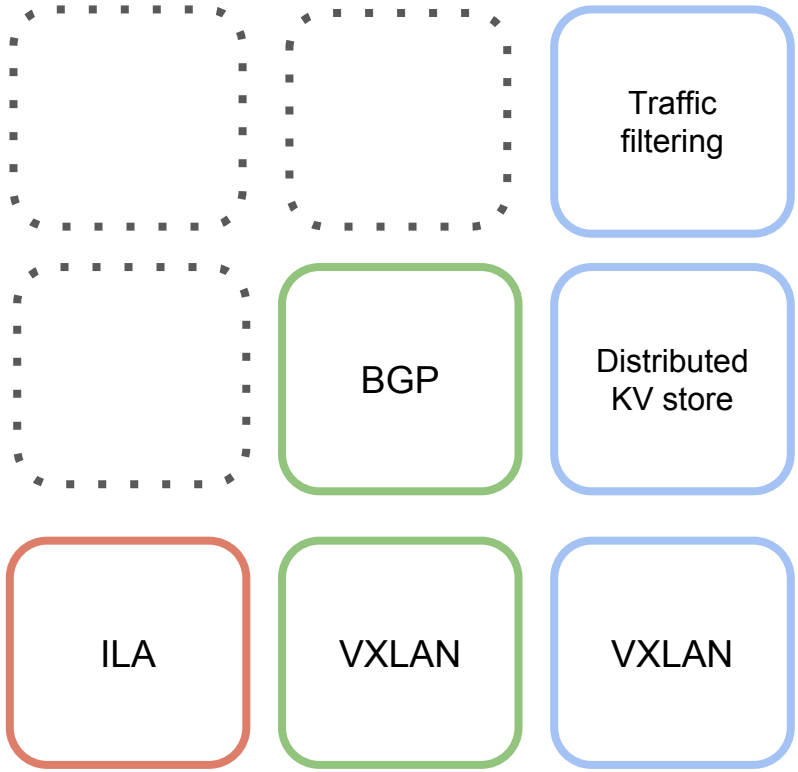
Research scope

ILA and **EVPN**:

- Addressing
- Solution complexity
- Usability

Cilium:

- Performance
- Traffic policies

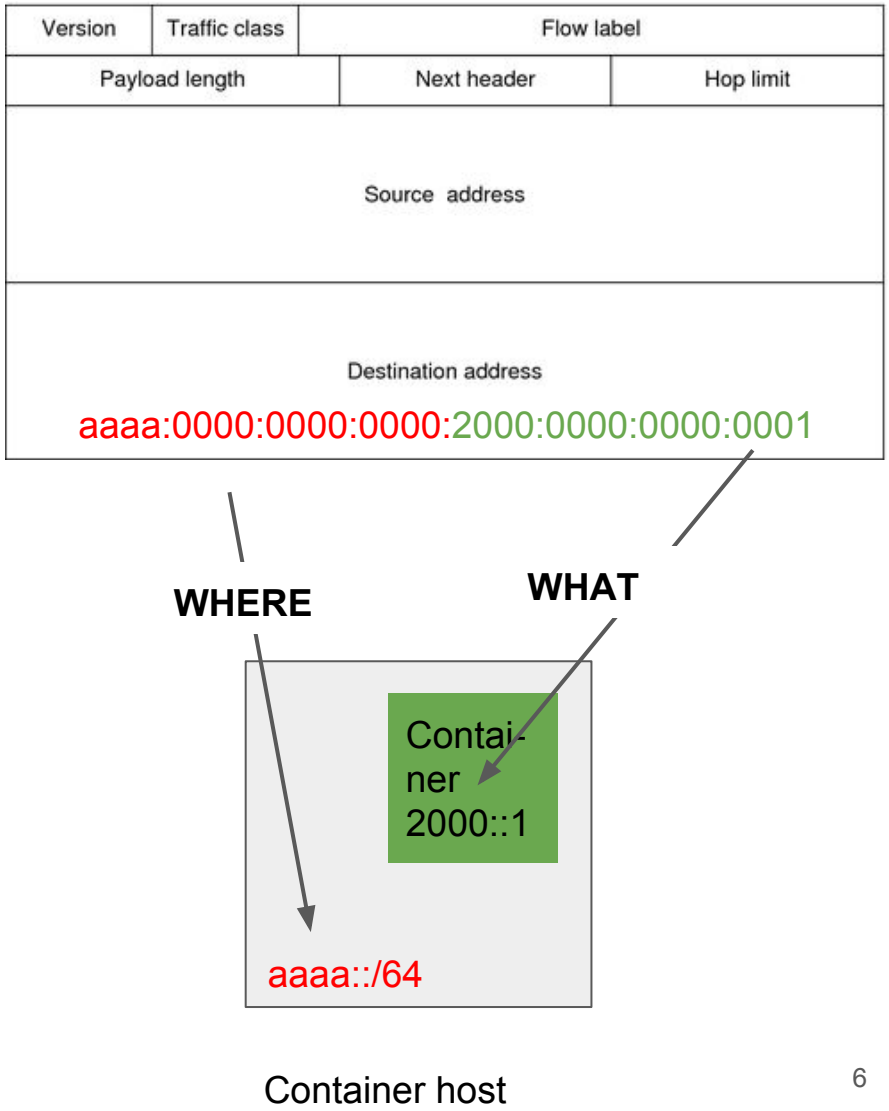


ILA (Identifier-Locator Addressing)

- **Data-plane:** does not use any encapsulation

  "Overloads" IPv6 address to convey two attributes:

  - Locator (where the destination is)
  - Identifier (which container are we specifically trying to contact)
- **Control-plane:** not specified (i.e. Do-It-Yourself)



ILA (Identifier-Locator Addressing): SIR prefix

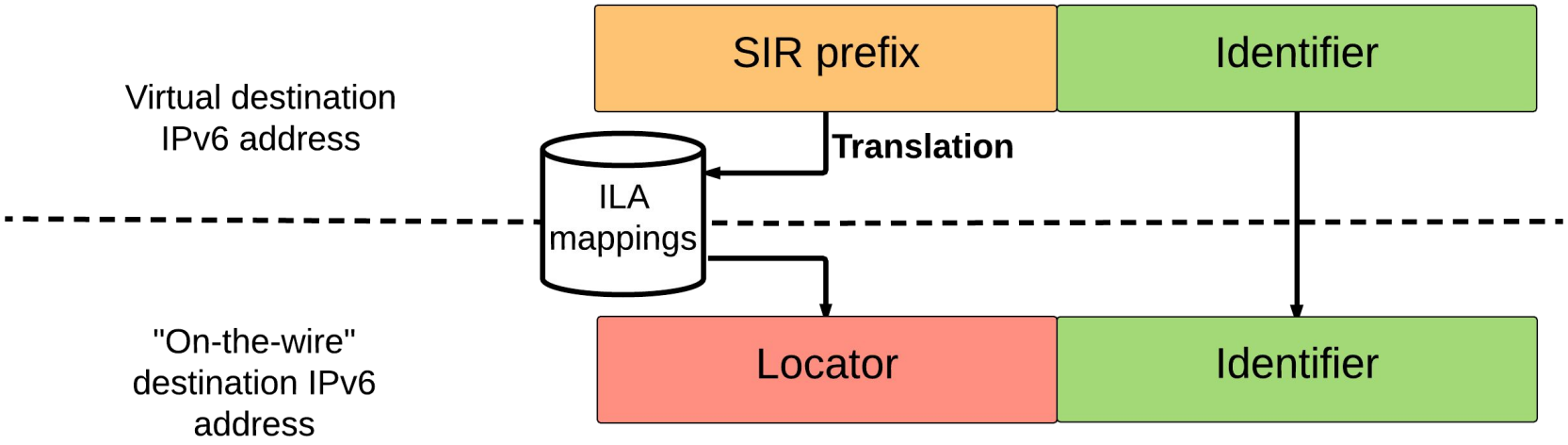
Mobility requirement:

Locator is by definition not mobile.

How the container keep its address?

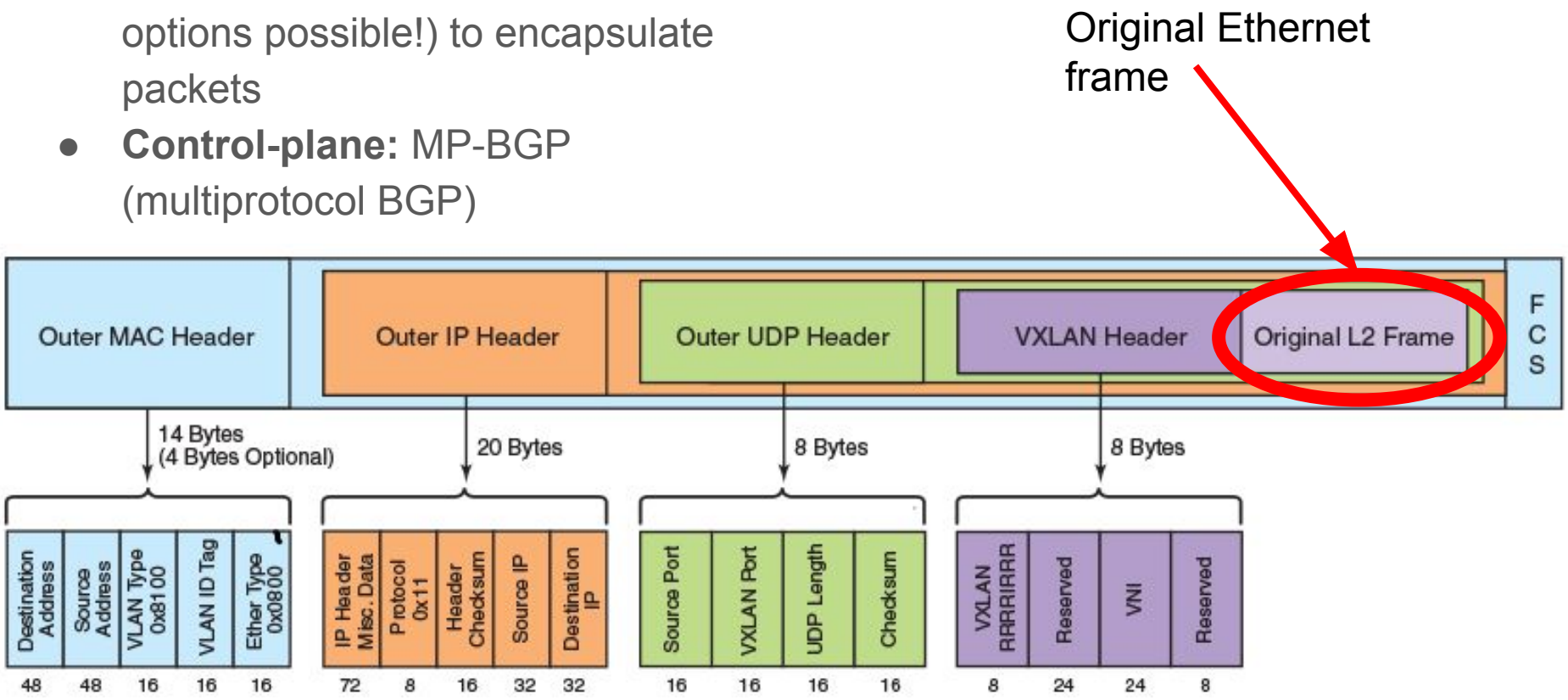
Solution:

Locator is not exposed to the endpoints (swap it with a virtual prefix: SIR)



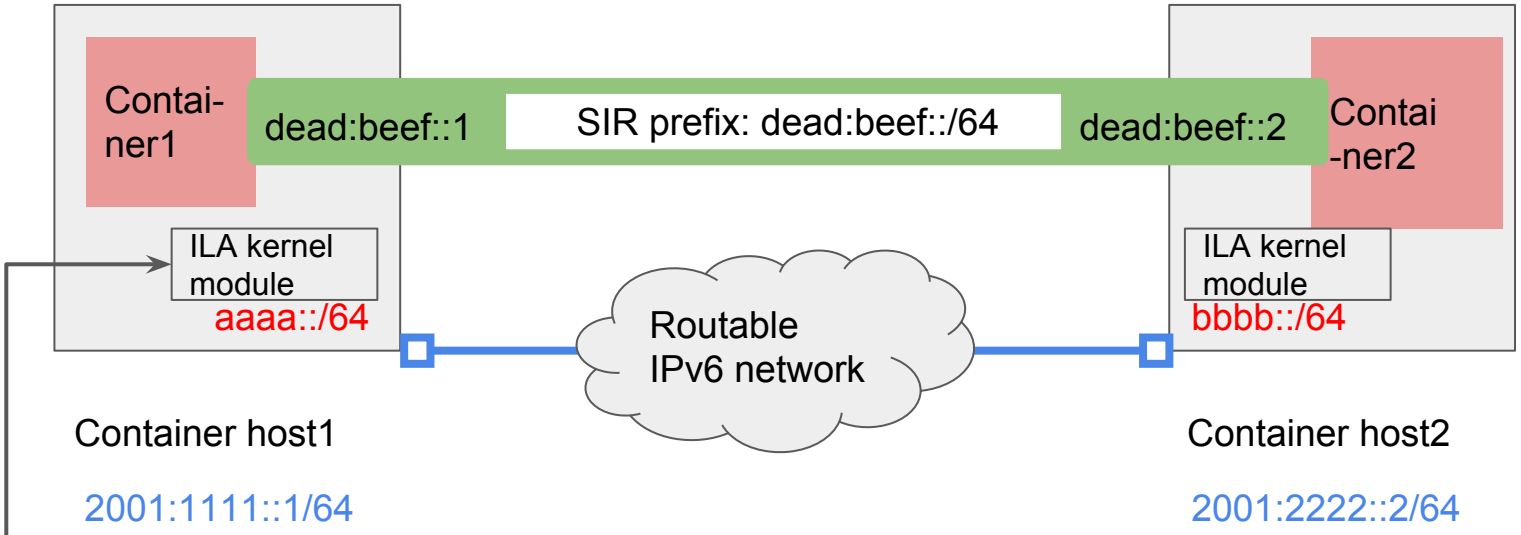
EVPN (Ethernet-VPN)

- **Data-plane:** VXLAN (other options possible!) to encapsulate packets
- **Control-plane:** MP-BGP (multiprotocol BGP)



http://www.brocade.com/content/html/en/deployment-guide/brocade-vcs-gateway-vmware-dp/GUID-5A5F6C36-E03C-4CA6-9833-1907DD928842.html

ILA: test environment



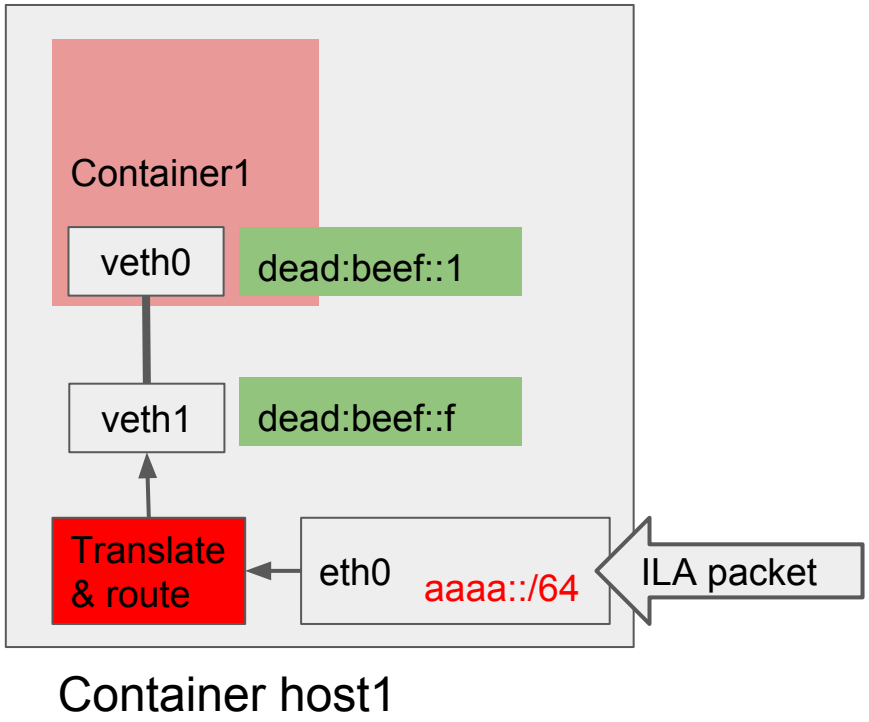

```
#egress route
dead:beef::0:0:0:2 encap ila bbbb:0:0:0  csum-mode no-action \
via 2001:2222::2/64

#ingress route
aaaa:0:0:0 encap ila dead:beef:0:0 csum-mode no-action \
via dead:beef::0:0:1/64
```

*Examples use simplified Identifier addresses

ILA: test environment

- Ingress ILA route conflicted with kernel generated routes in the “local” routing table
- Container needs to fill its NDP table (create NDP proxy or create static entries)
- After the ILA translation, TCP header checksum is incorrect*
  - In our environment we ended up disabling network device offloading to make the packets through
- First 4 bits of Identifier are reserved bits (used for scoping)

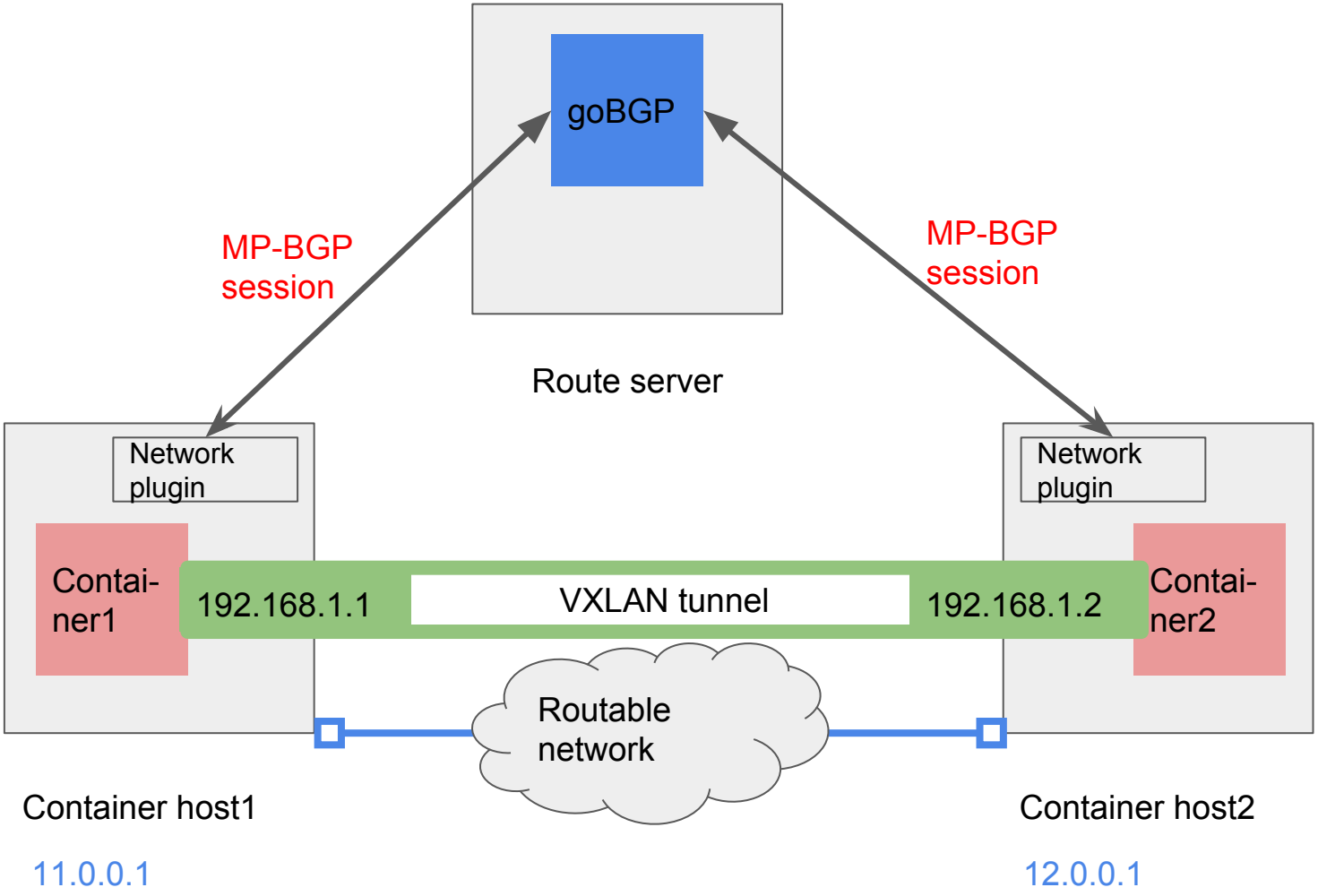


*Could be circumvented with ILA’s checksum-neutral adjustment mode

ILA: Results

- Feasible to be used as a virtual IPv6 container network
- Quite some caveats in regard to data-plane operations
- We did not get to the stage to think about developing a proper control-plane. All the setup was half-manual

EVPN: test environment



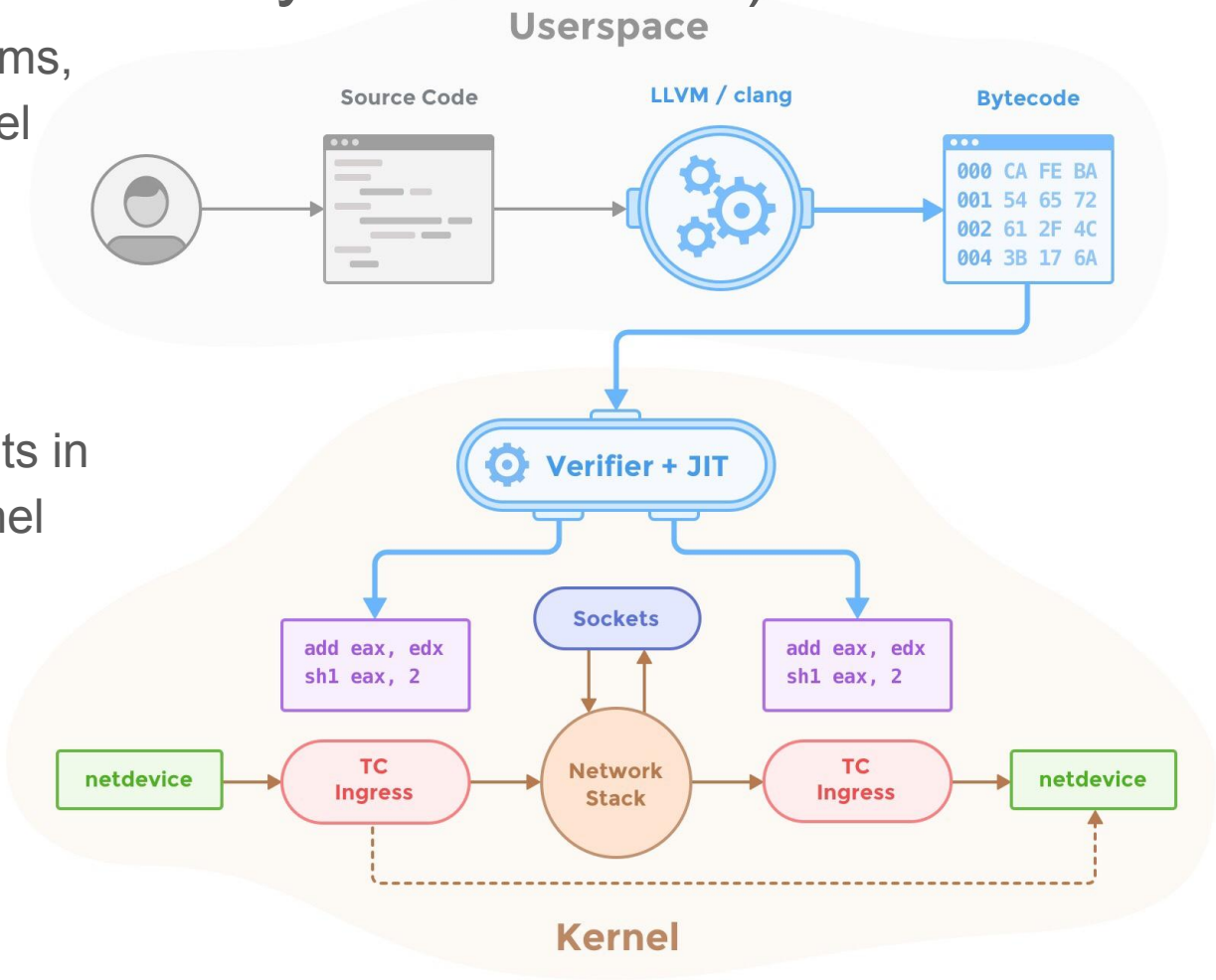
http://murat1985.github.io/kubernetes/cni/2016/05/15/bagpipe-gobgp.html

EVPN: Results

- Feasible as a container network to create virtual L2 networks
- The main challenge we see is the programmatic integration with container orchestration systems
- Setup was straightforward: bridging container veth interfaces to VXLAN adapter

Cilium foreword:
eBPF (extended Berkeley Packet Filter)

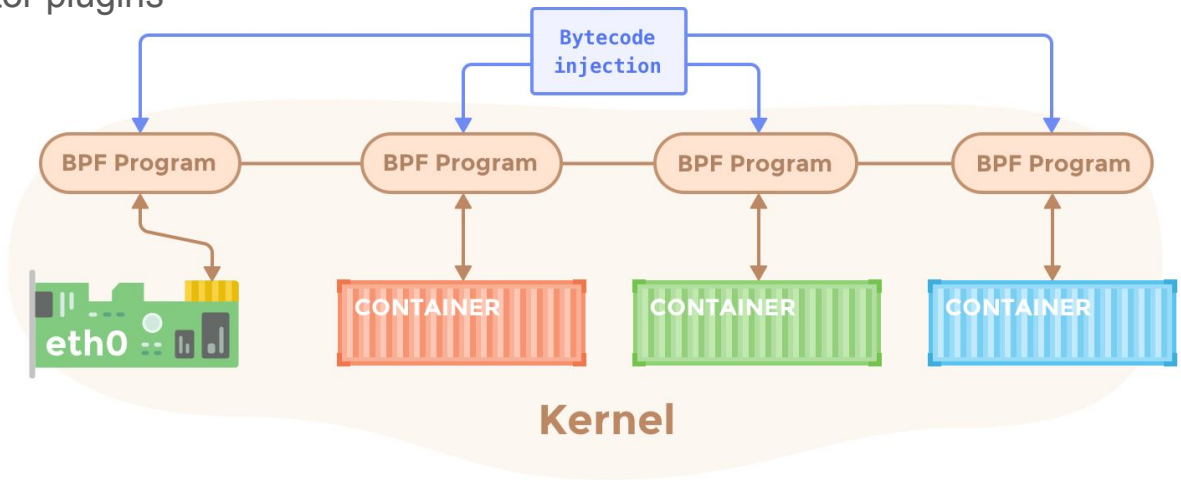
- Small, limited programs, executed in-the kernel space
- Can be used to manipulate and filter packets
- Allow to take shortcuts in the regular linux kernel networking stack



http://cilium.readthedocs.io/en/latest/architecture/

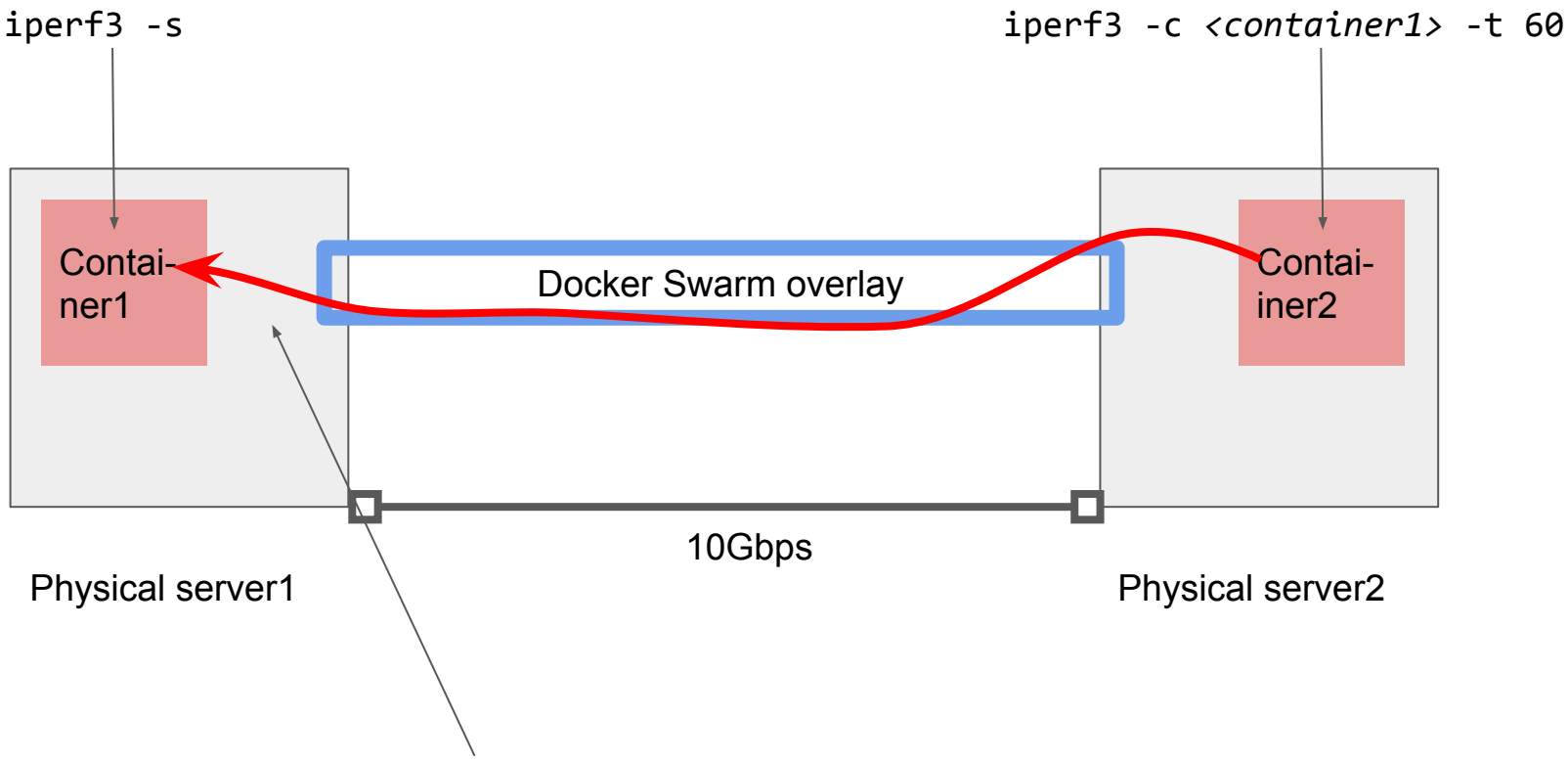
Cilium

- **Data-plane:** VXLAN (or Geneve) to encapsulate packets
- **Control-plane:** distributed KV store (e.g. Consul)
- **Special ingredients:**
  - eBPF
  - container orchestrator plugins
  - traffic policies



http://cilium.readthedocs.io/en/latest/architecture/

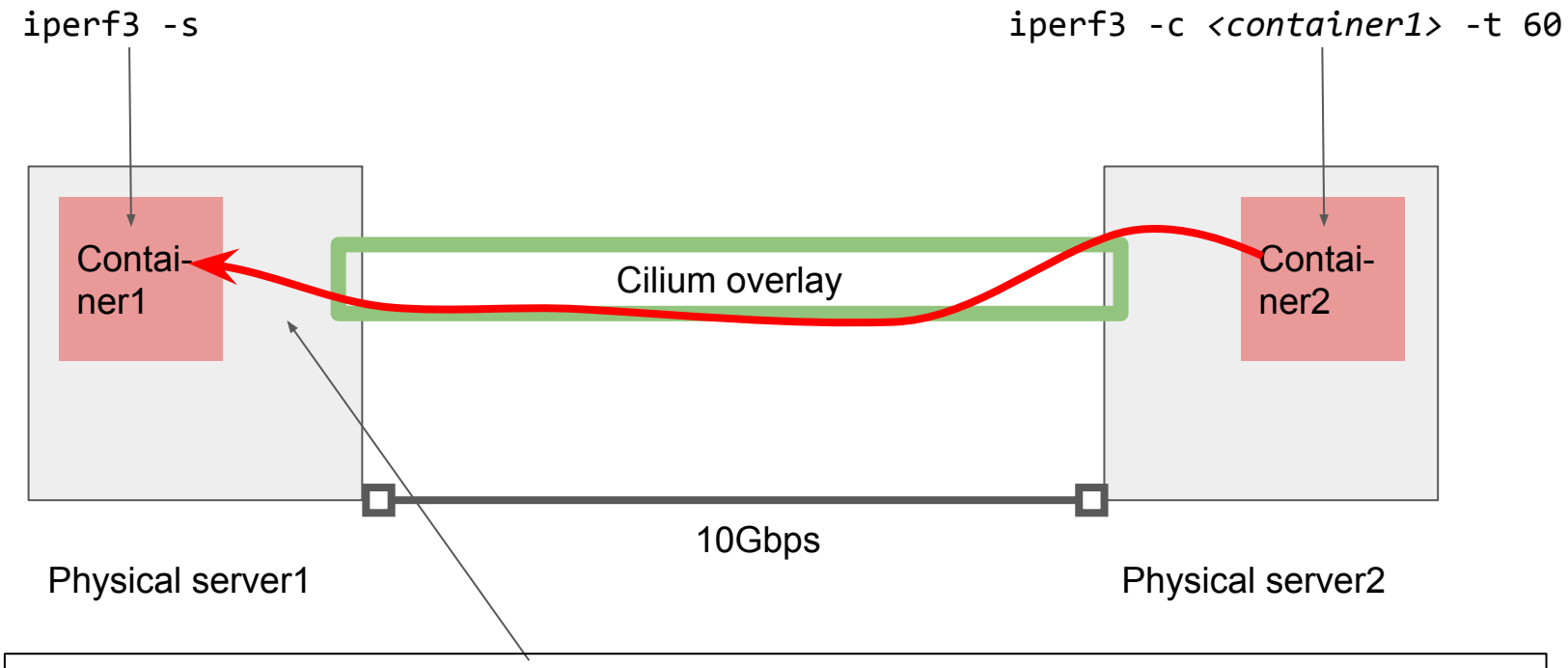
Overlay filtering topology: Docker Swarm + netfilter



Hit by a vast majority of traffic

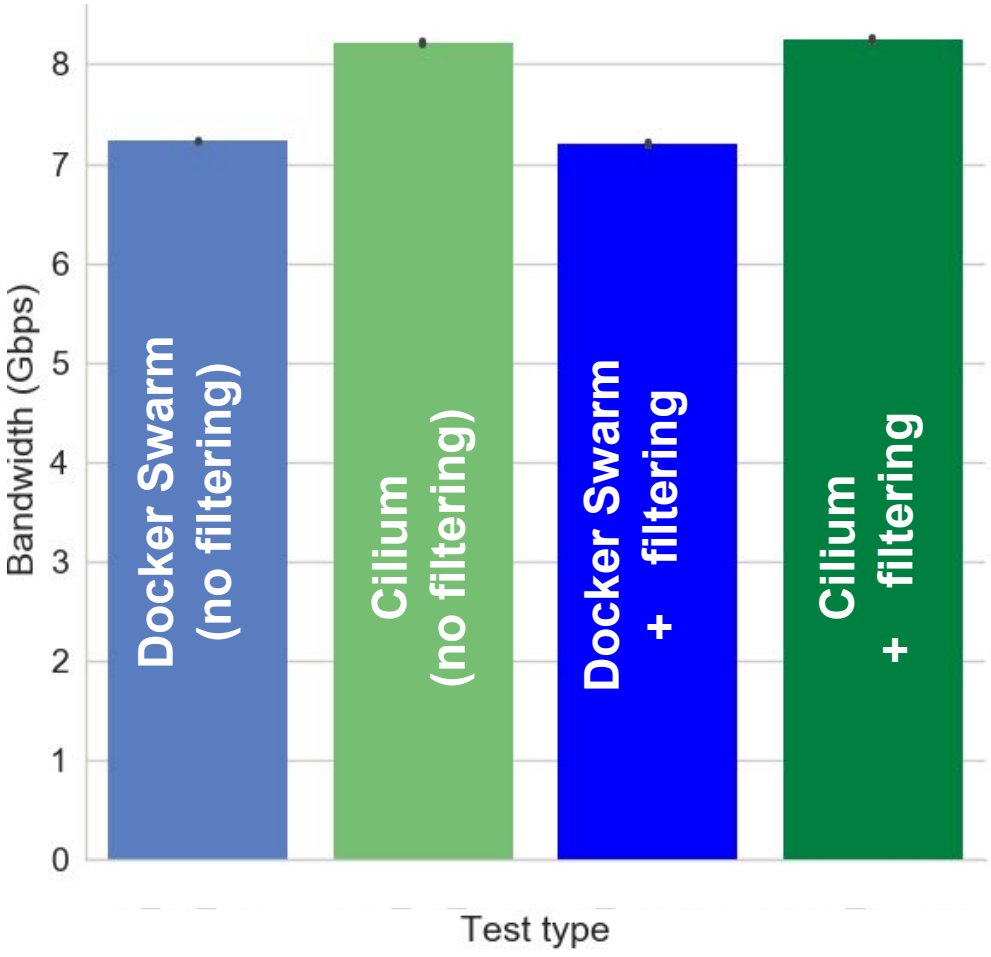
```
iptables -t filter -A FORWARD -m state --state
ESTABLISHED ,RELATED -j ACCEPT
iptables -t filter -A FORWARD -m tcp -p tcp --dport 5201
-j ACCEPT
iptables -t filter -P FORWARD DROP
```

Overlay filtering topology: Cilium + eBPF




```
"endpointSelector": {"matchLabels":{"id":"service1"}}, "ingress": [{
"fromEndpoints": [ {"matchLabels":{"id":"service1"}}
],
"toPorts": [{
                        "ports": [{"protocol": "tcp", "port" : "5201"}]
 }]
}]
```

Overlay filtering topology: Results



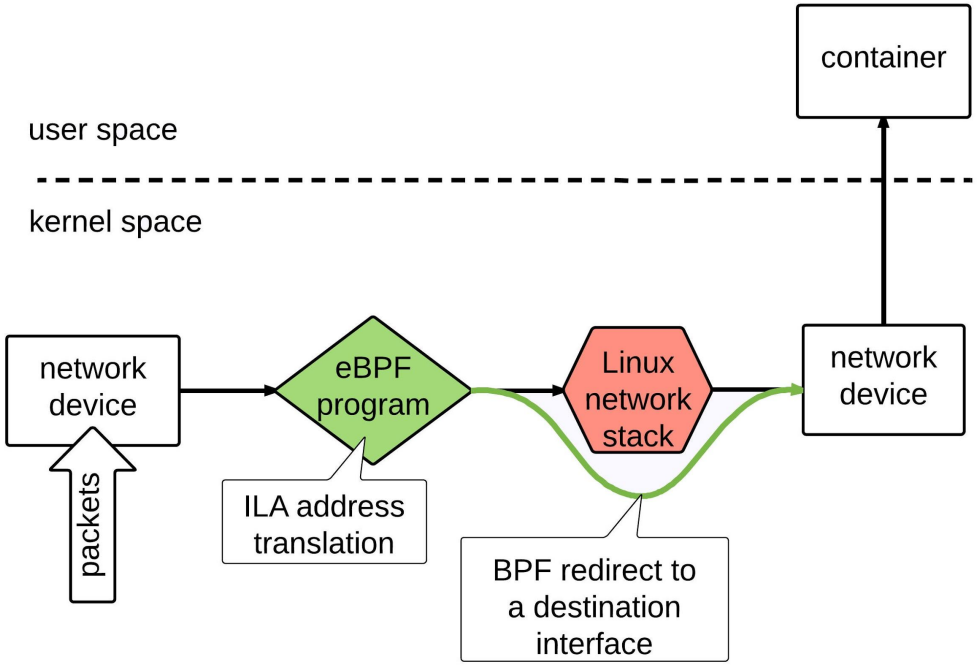
- Cilium was more performant than Docker Swarm (7.22 Gbps vs 8.22 Gbps)
- There was **no significant difference after the traffic filters has been applied** (7.20 Gbps, 8.24 Gbps)
- Both **networks required manual tuning to achieve high speeds** (MTU increasing, enabling GRO, GSO, TSO)

Overall conclusions

- ILA offers an alternative to encapsulation based world
  - However, it comes at a price of complicated setup and addressing limitations
- EVPN is more flexible in regard to addressing and set-up
  - It also has the potential to satisfy more use-cases
- Cilium with its broad use of eBPF outperforms the “classical” kernel-based network
  - Single-flow filtering did not have notable performance impact in tested scenarios

Demo at SURF booth (#857)

PoC ILA implementation with extended Berkley Packet Filter (eBPF)



Future work

- Extend tests on Cilium’s performance

- Implement multi-tenancy scenarios for the test-topologies