Parallelizing Packet Processing in Container Overlay Networks

(EuroSys '21)

Jiaxin Lei[1], Manish Munikar[2], Kun Suo[3], Hui Lu[1], Jia Rao[2]

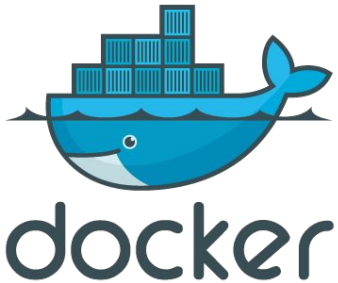
[1]*Binghamton University*
[2]*The University of Texas at Arlington*
[3]*Kennesaw State University*

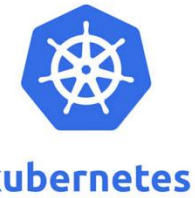
Containers are taking over the cloud

- Lightweight
- OS level virtualization
- **High** application density
- **Efficient** resource utilization

Containers are taking over the cloud

- Lightweight
- OS level virtualization
- **High** application density
- **Efficient** resource utilization

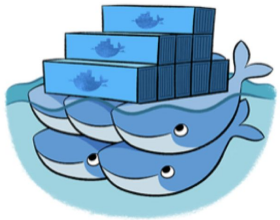




Based on report, Google launches over **7k+** containers **per sec** for its searching service.

How do containers communicate?

- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon a tunneling approach like **VxLAN** protocol



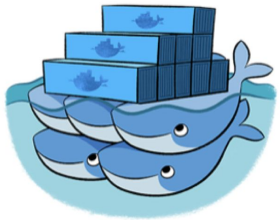
flannel

PROJECT CALICO

weaveworks

How do containers communicate?

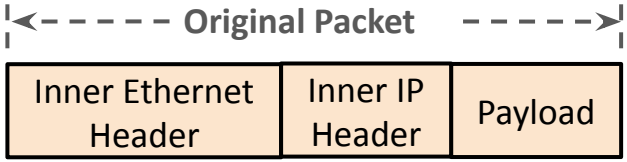
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon a tunneling approach like **VxLAN** protocol



flannel

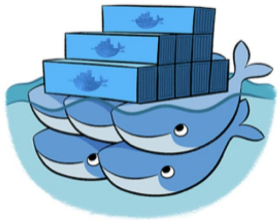
PROJECT CALICO

weaveworks



How do containers communicate?

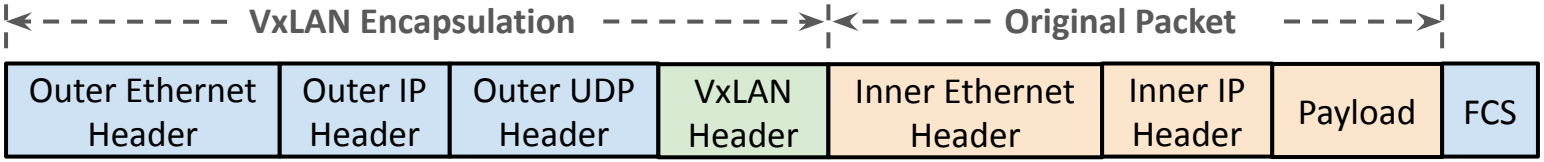
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon a tunneling approach like **VxLAN** protocol



flannel

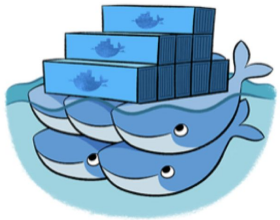
PROJECT CALICO

weaveworks



How do containers communicate?

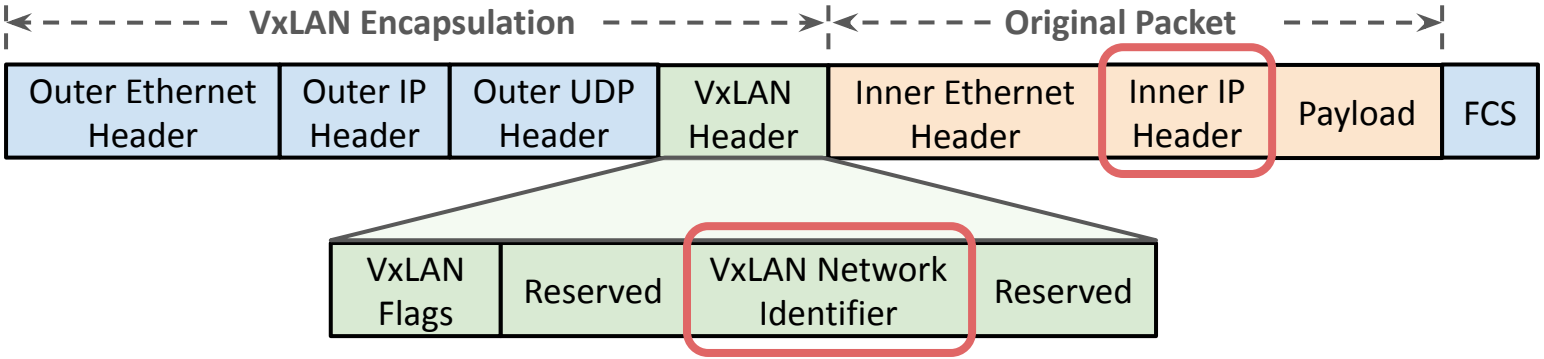
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon a tunneling approach like **VxLAN** protocol



flannel

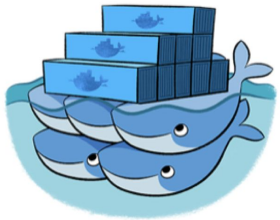
PROJECT CALICO

weaveworks



How do containers communicate?

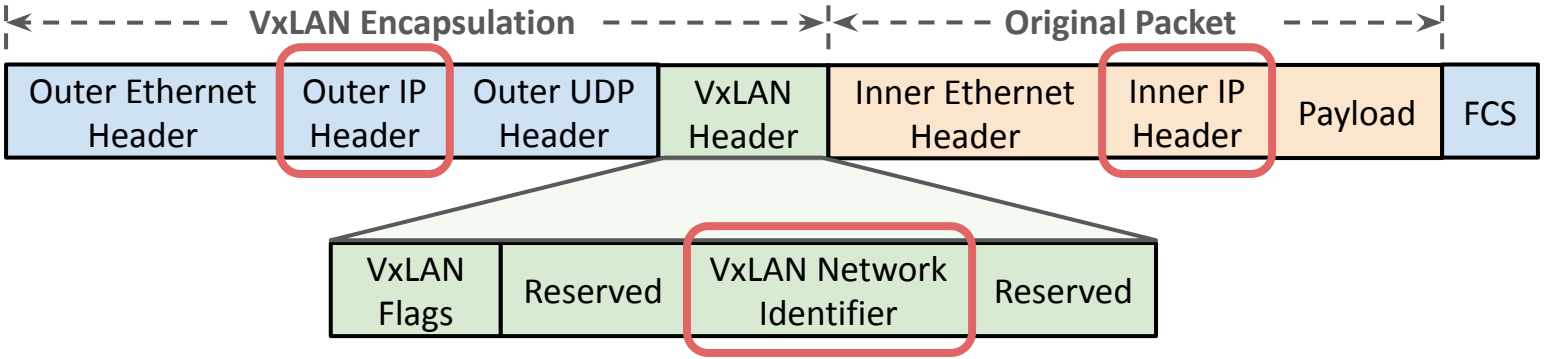
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon a tunneling approach like **VxLAN** protocol



flannel

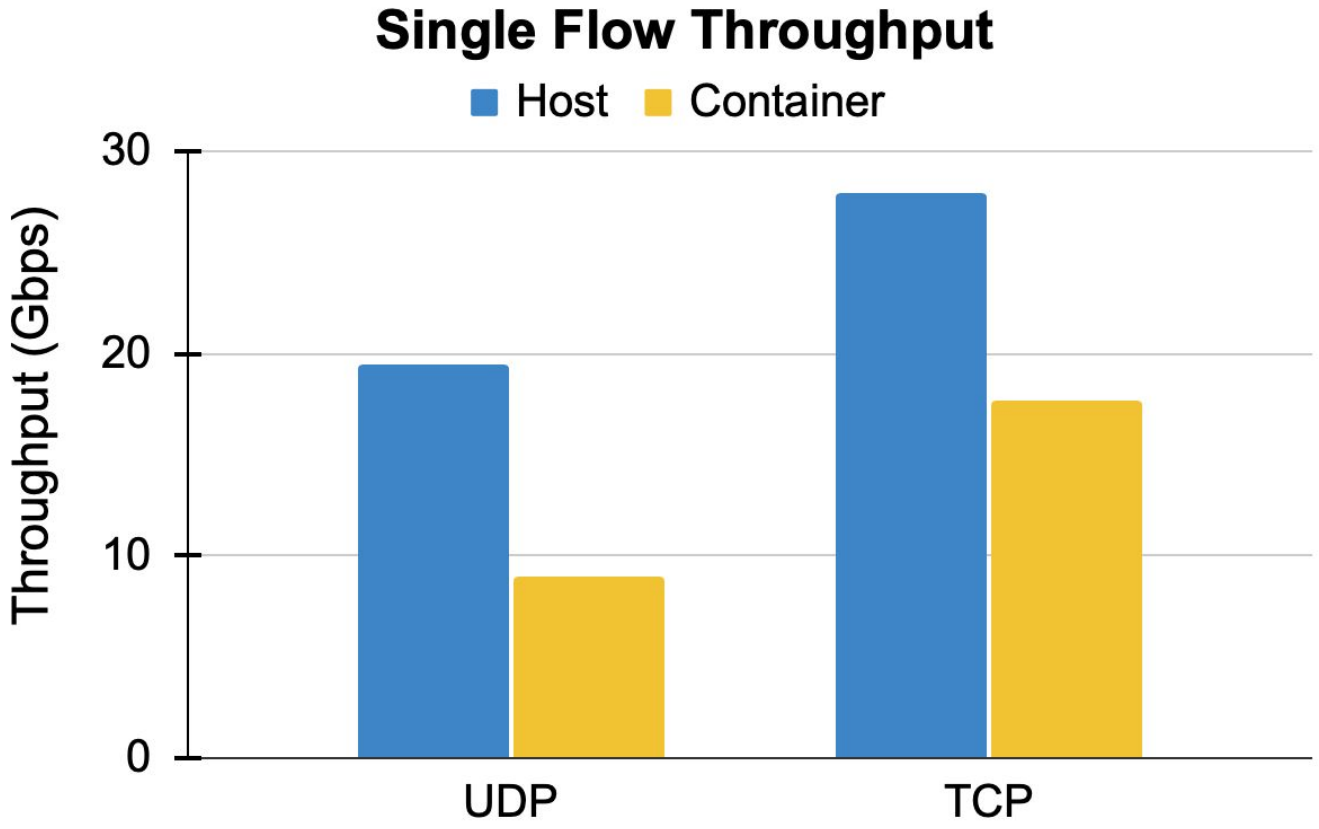
PROJECT CALICO

weaveworks



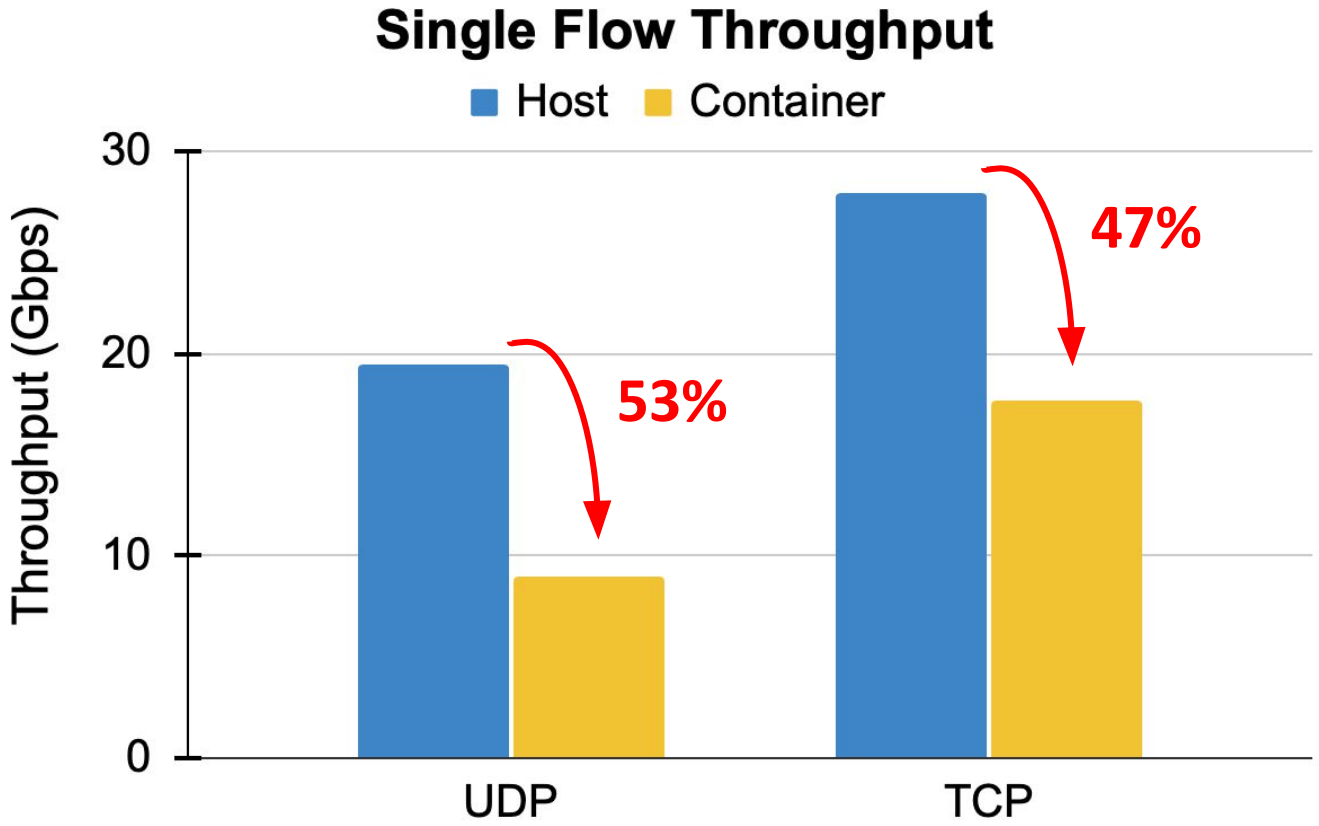
Performance overhead of overlay networks

- Machines are connected via a 100 GbE NIC.



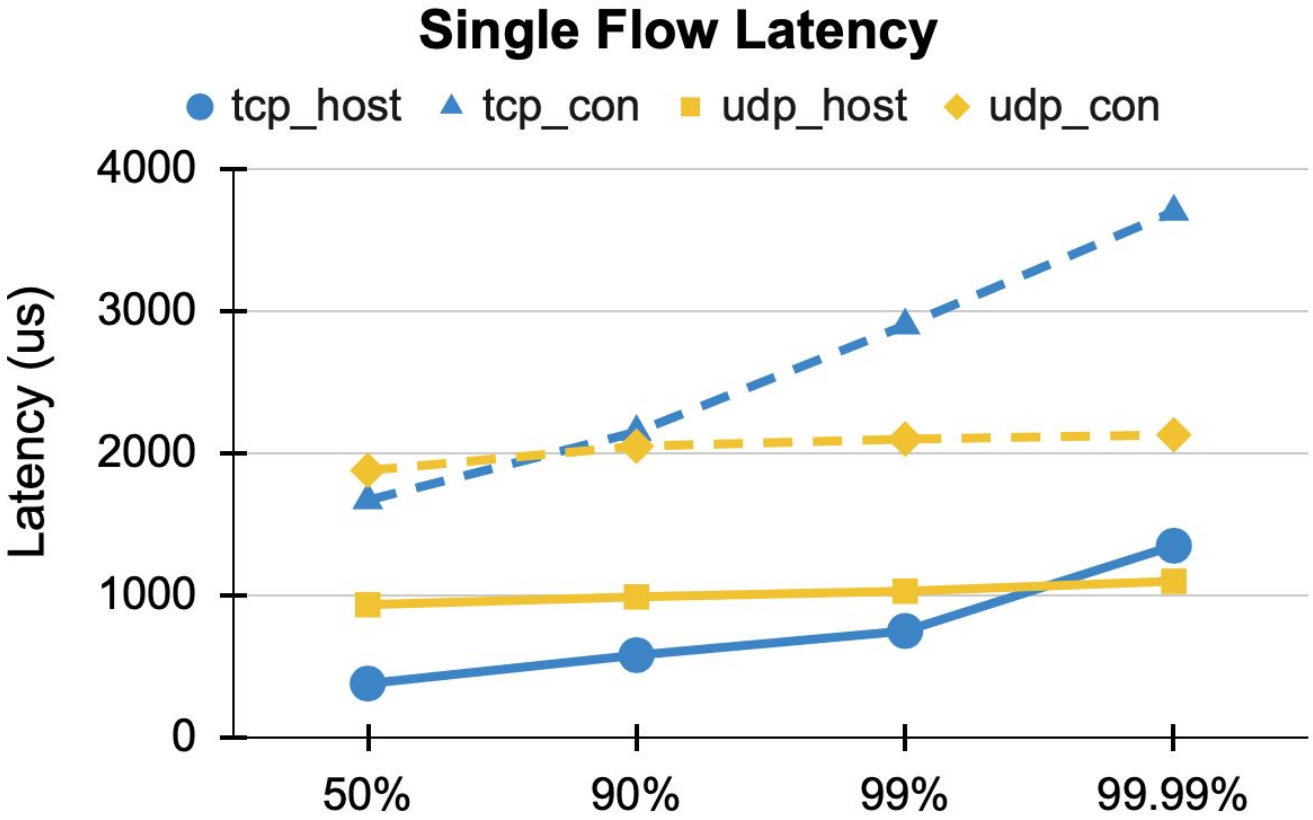
Performance overhead of overlay networks

- Machines are connected via a 100 GbE NIC.



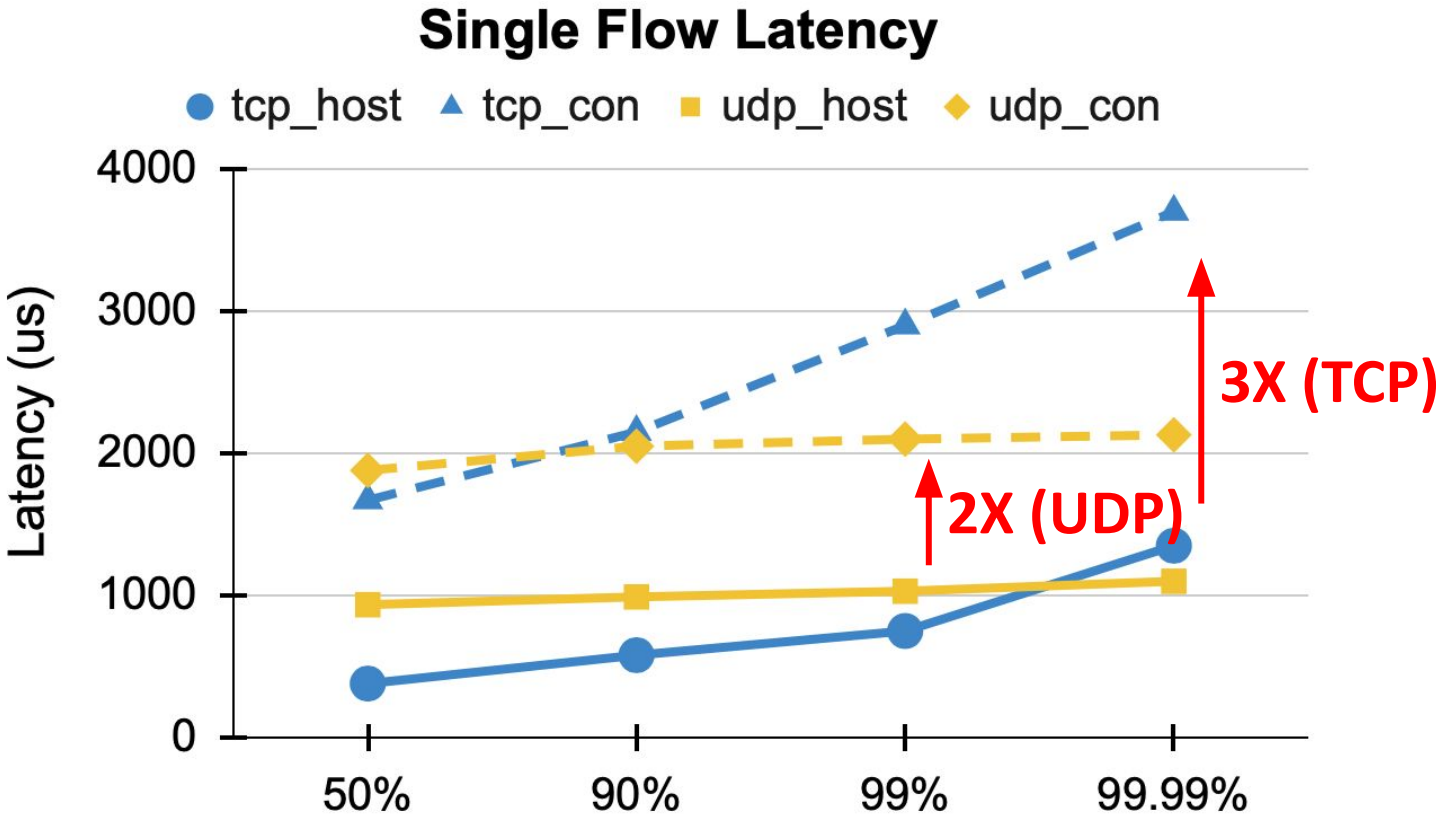
Performance overhead of overlay networks

- Machines are connected via a 100 GbE NIC.



Performance overhead of overlay networks

- Machines are connected via a 100 GbE NIC.



Why is overlay network slow?

L 7

L 3&4

L 2

Hardware

Why is overlay network slow?

- **Host Networks**

Why is overlay network slow?

- **Host Networks**

Why is overlay network slow?

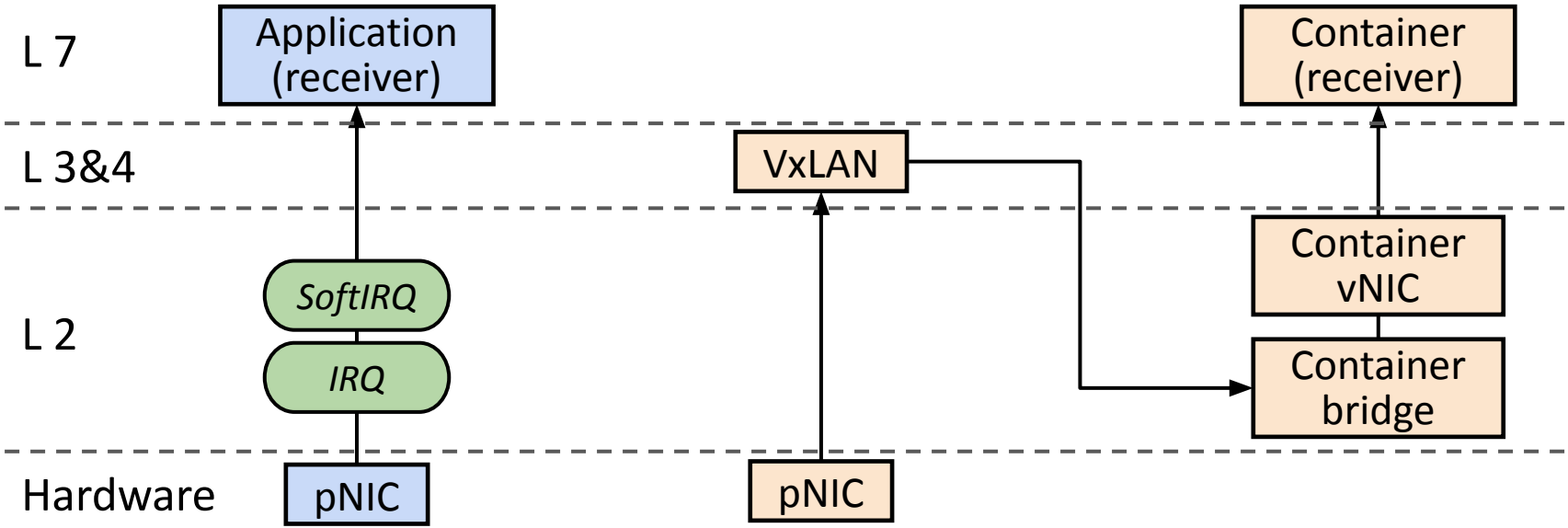
- **Host Networks**

Why is overlay network slow?

- **Host Networks**
  - IRQ + SoftIRQ

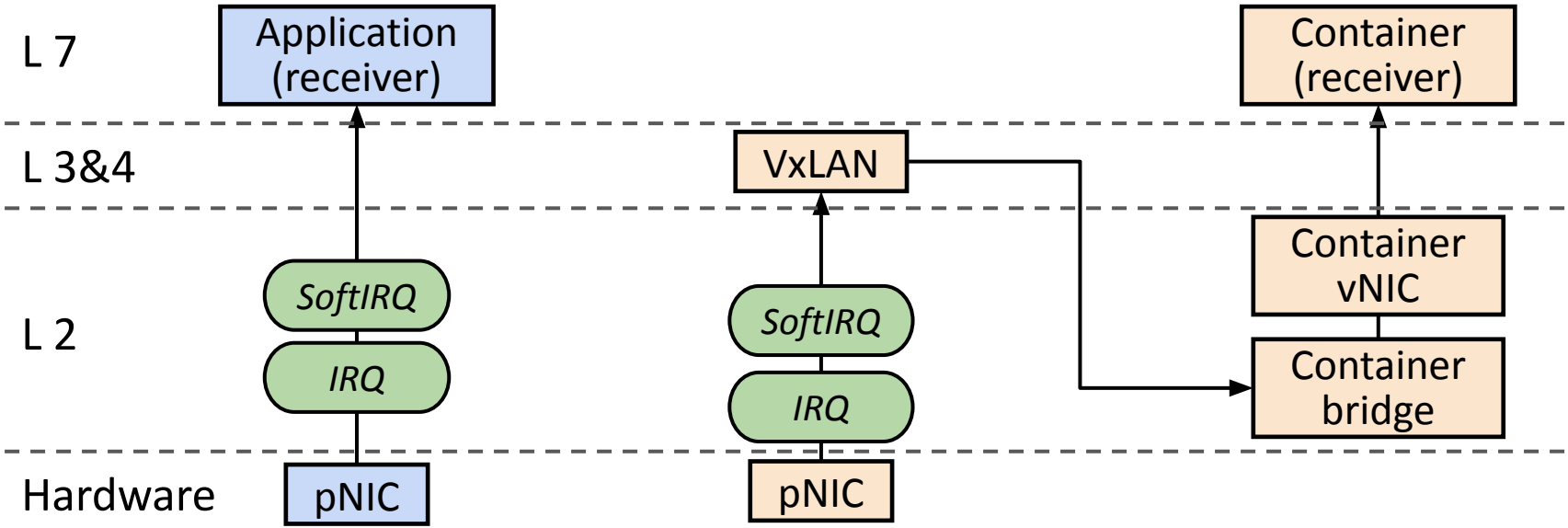
Why is overlay network slow?

- **Host Networks**
  - IRQ + SoftIRQ
- **Overlay Networks**



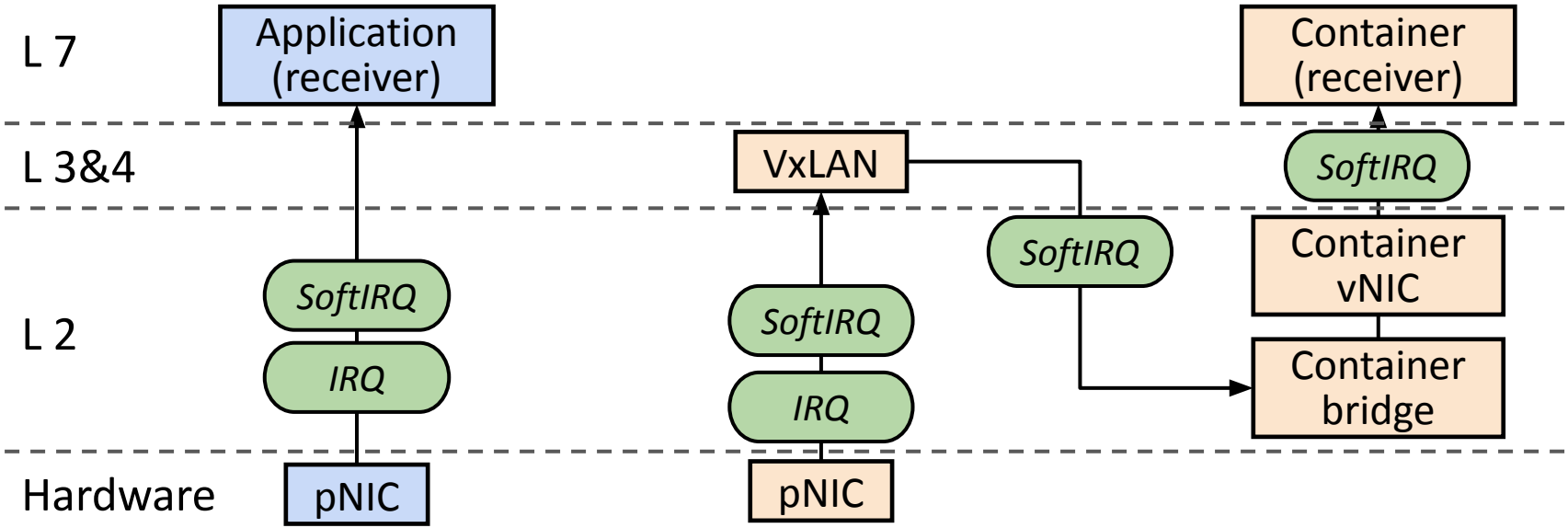
Why is overlay network slow?

- **Host Networks**
  - IRQ + SoftIRQ
- **Overlay Networks**



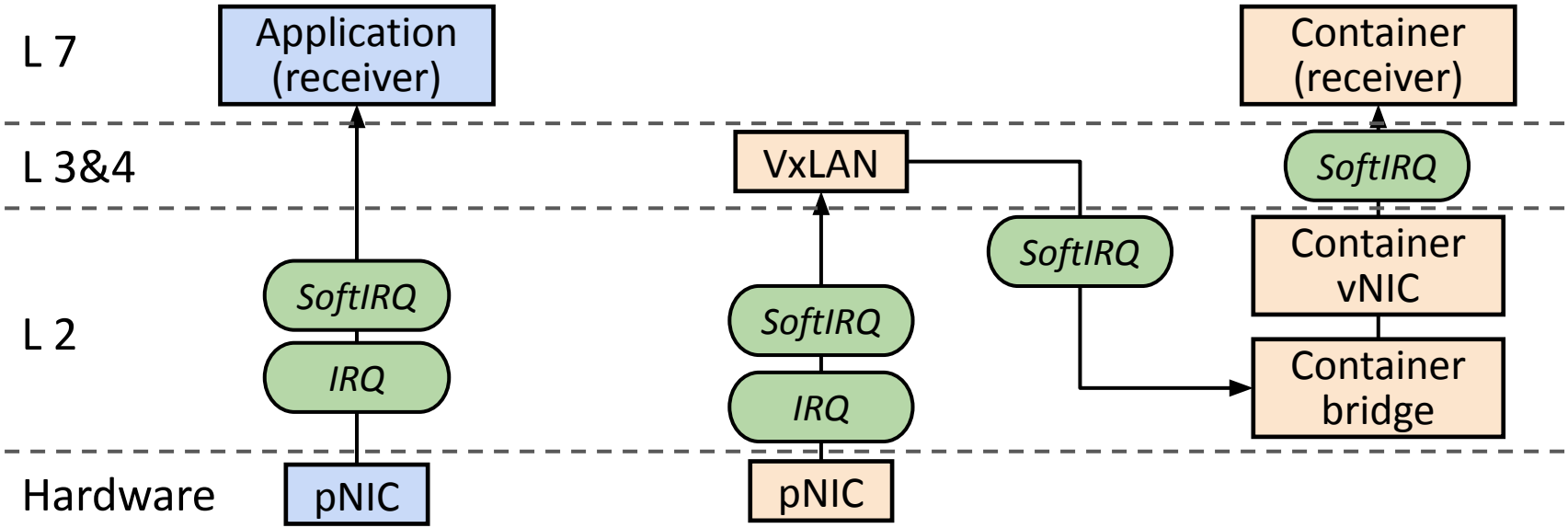
Why is overlay network slow?

- **Host Networks**
  - IRQ + SoftIRQ
- **Overlay Networks**



Why is overlay network slow?

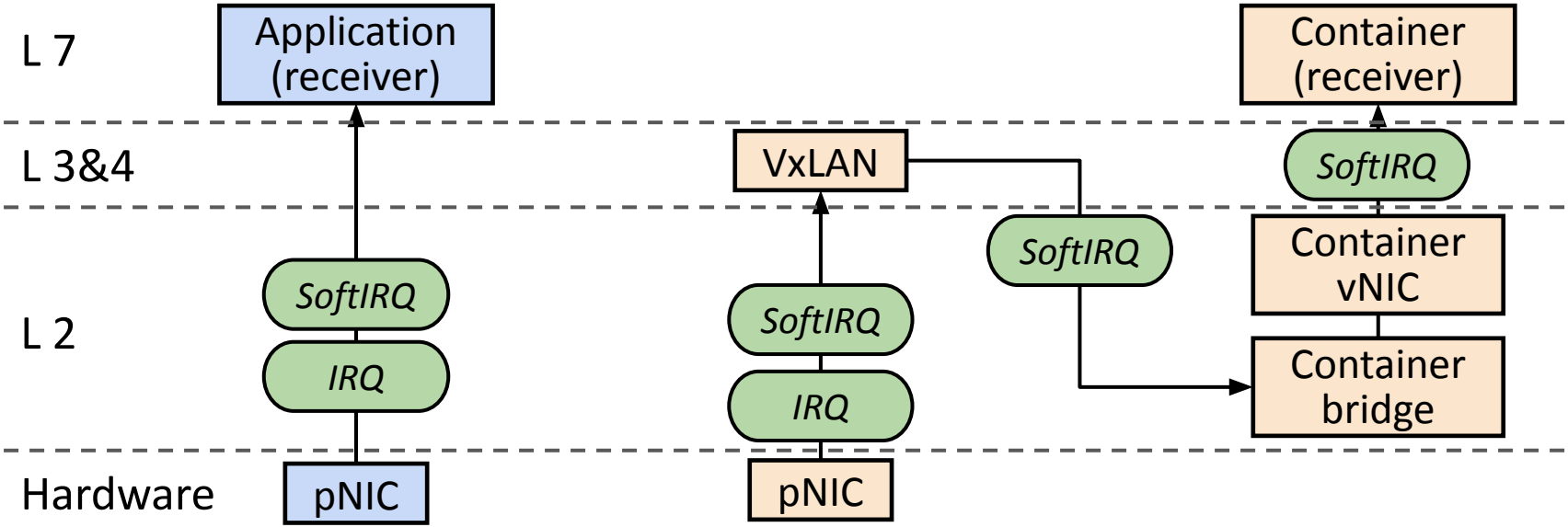
- **Host Networks**
  - IRQ + SoftIRQ
- **Overlay Networks**
  - IRQ + **3X** SoftIRQs



Why is overlay network slow?

- **Host Networks**
  - IRQ + SoftIRQ
- **Overlay Networks**
  - IRQ + **3X** SoftIRQs

- **Overhead**
  - **Additional** devices
  - **Prolonged** path
  - **Excessive and serialized** softIRQs



Existing solutions to accelerate overlay networks

- Kernel bypass (DPDK+mTCP*[NSDI'14]*)
  - ✔ Avoids OS overheads with network stack tailored to applications
  - ❌ Operators have no control over network stack; have to trust applications

Existing solutions to accelerate overlay networks

- Kernel bypass (DPDK+mTCP*[NSDI'14]*)
  - ✔ Avoids OS overheads with network stack tailored to applications
  - ❌ Operators have no control over network stack; have to trust applications
- Connection-level metadata transformation (Slim*[NSDI'19]*, FreeFlow*[NSDI'19]*)
  - ✔ Avoids overhead of virtual devices; fast as host network
  - ❌ Limited scalability; only for TCP; not support data-plane policies

Existing solutions to accelerate overlay networks

- Kernel bypass (DPDK+mTCP*[NSDI'14]*)
  - ✔ Avoids OS overheads with network stack tailored to applications
  - ❌ Operators have no control over network stack; have to trust applications
- Connection-level metadata transformation (Slim*[NSDI'19]*, FreeFlow*[NSDI'19]*)
  - ✔ Avoids overhead of virtual devices; fast as host network
  - ❌ Limited scalability; only for TCP; not support data-plane policies
- Hardware offload (Mellanox ASAP^2, AccelNet*[NSDI'18]*)
  - ✔ Fastest, completely avoids CPU overheads
  - ❌ Requires new/expensive hardware; SR-IOV limitations

Our solution - FALCON

FALCON = **F**ast and B**al**anced **Co**ntainer **N**etworking

Key idea: Utilize idle CPU resources to accelerate packet processing

Our solution - FALCON

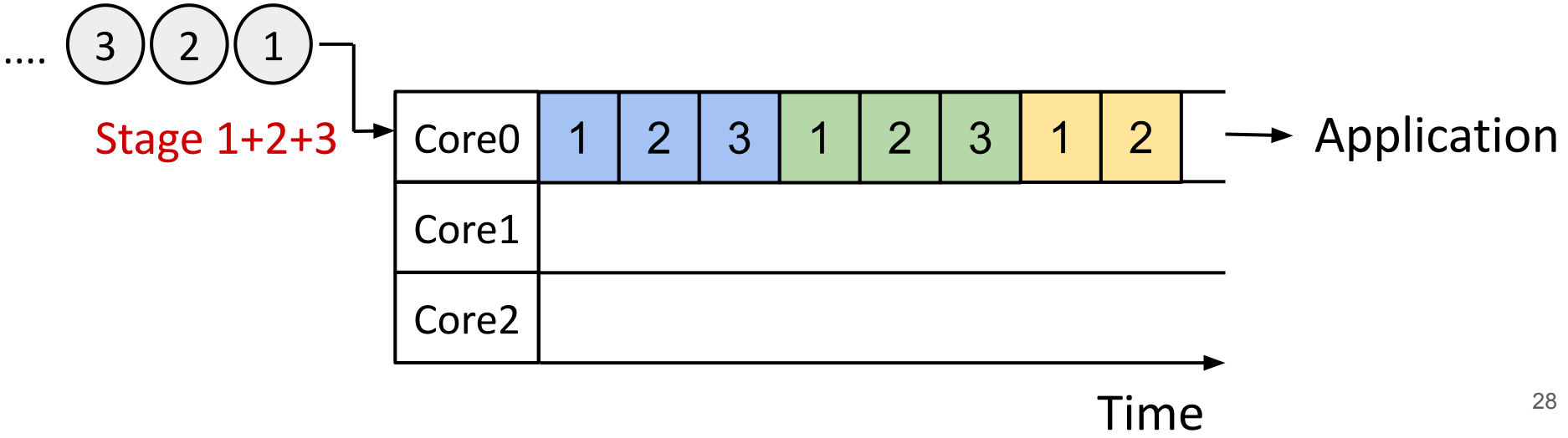
FALCON = **F**ast and B**al**anced **Co**ntainer **N**etworking

Key idea: Utilize idle CPU resources to accelerate packet processing

- Software-based solution
- Full network isolation / flexibility
- Completely backward compatible
    - Immediately deployable in real systems
- Better performance

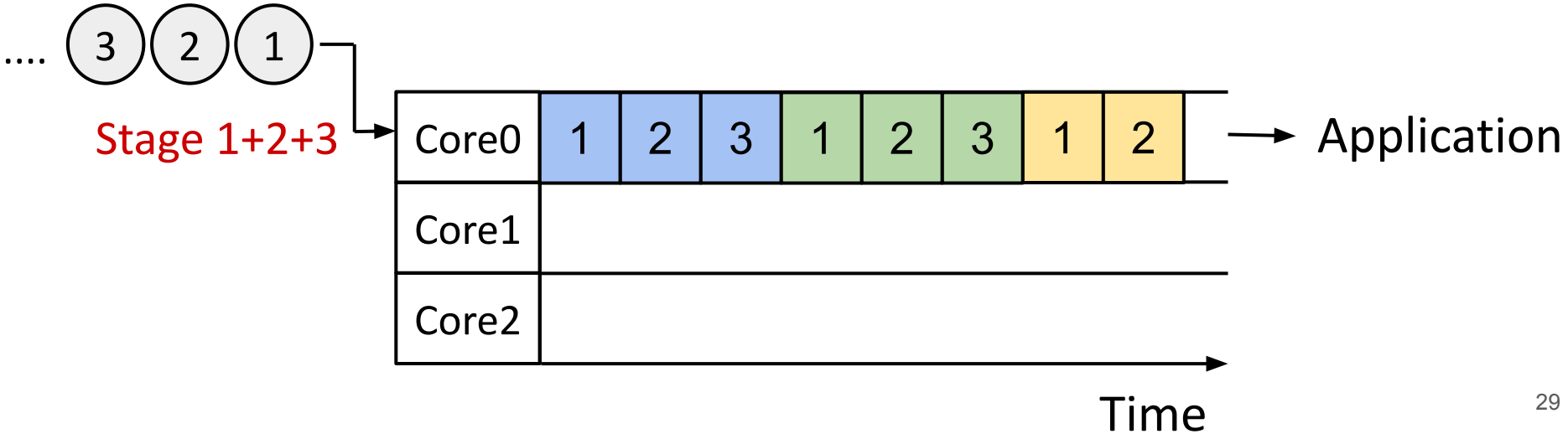
FALCON - SoftIRQs Pipelining

- **Blue** - 1st SoftIRQ ; **Green** - 2nd SoftIRQ; **Yellow** - 3rd SoftIRQ



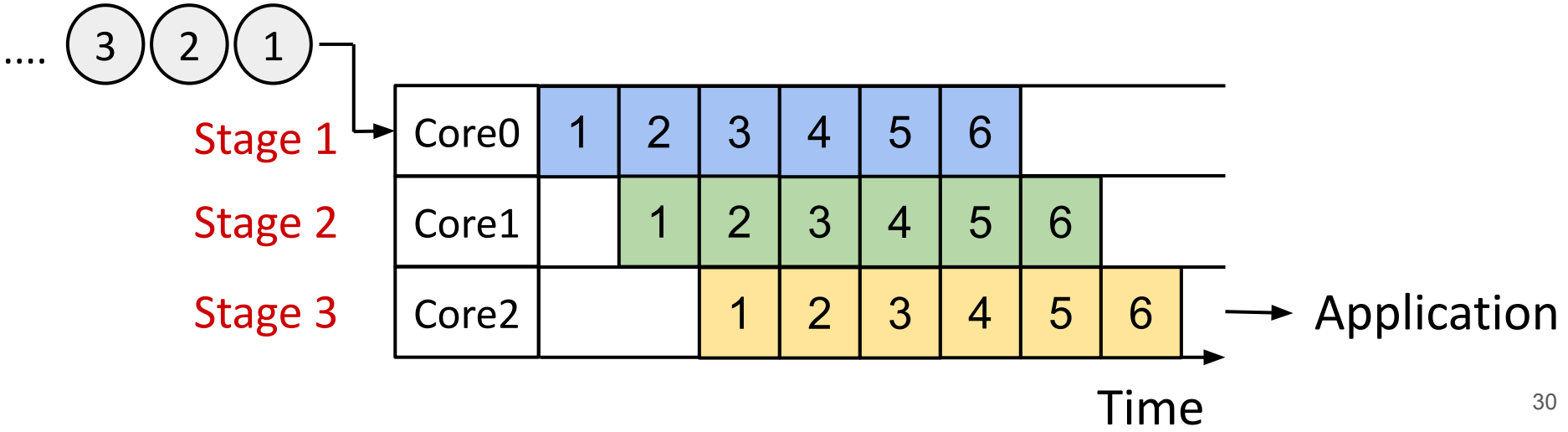
FALCON - SoftIRQs Pipelining

- **Blue** - 1st SoftIRQ ; **Green** - 2nd SoftIRQ; **Yellow** - 3rd SoftIRQ
- These 3 stages can be dispatched and parallelized



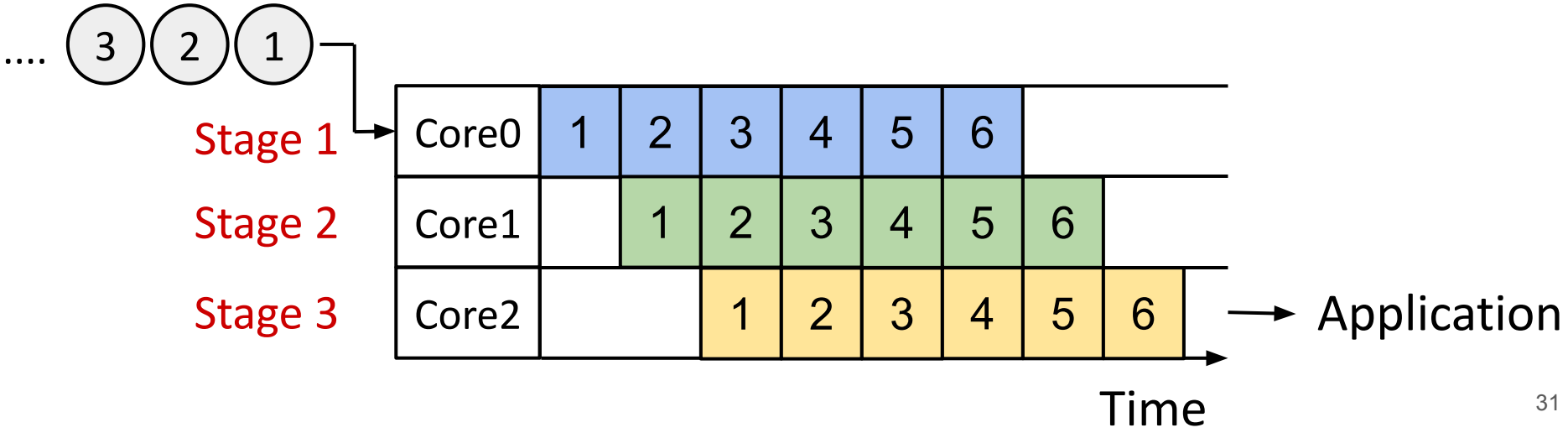
FALCON - SoftIRQs Pipelining

- Stage transition functions (Hashing)



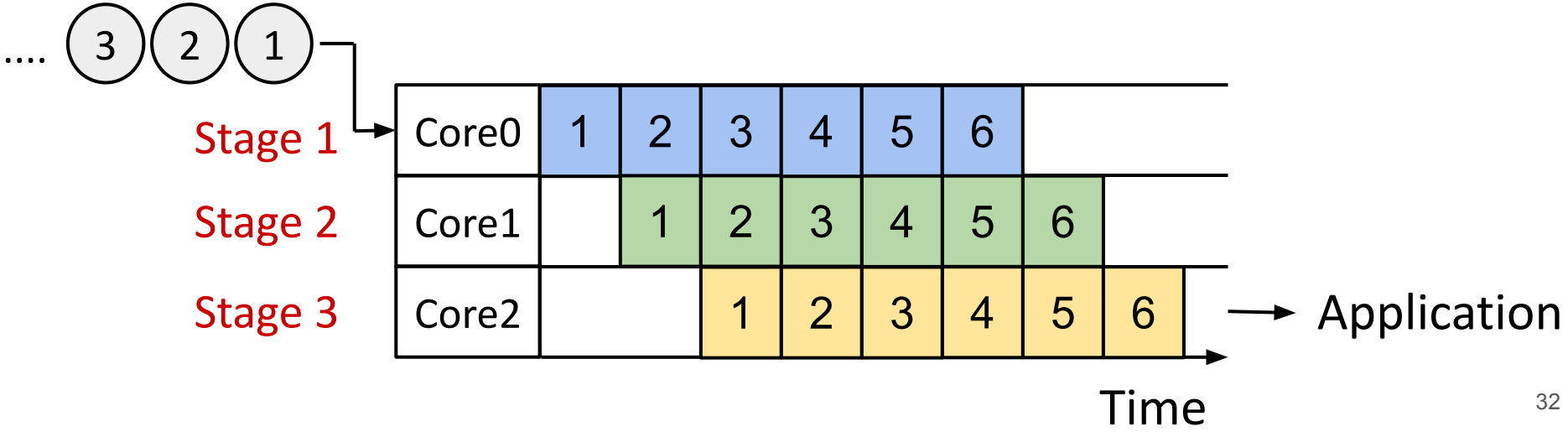
FALCON - SoftIRQs Pipelining

- Stage transition functions (Hashing)
  - **4** tuples (IPs+Ports) -> **5** tuples (IPs+Ports+**DeviceID**)



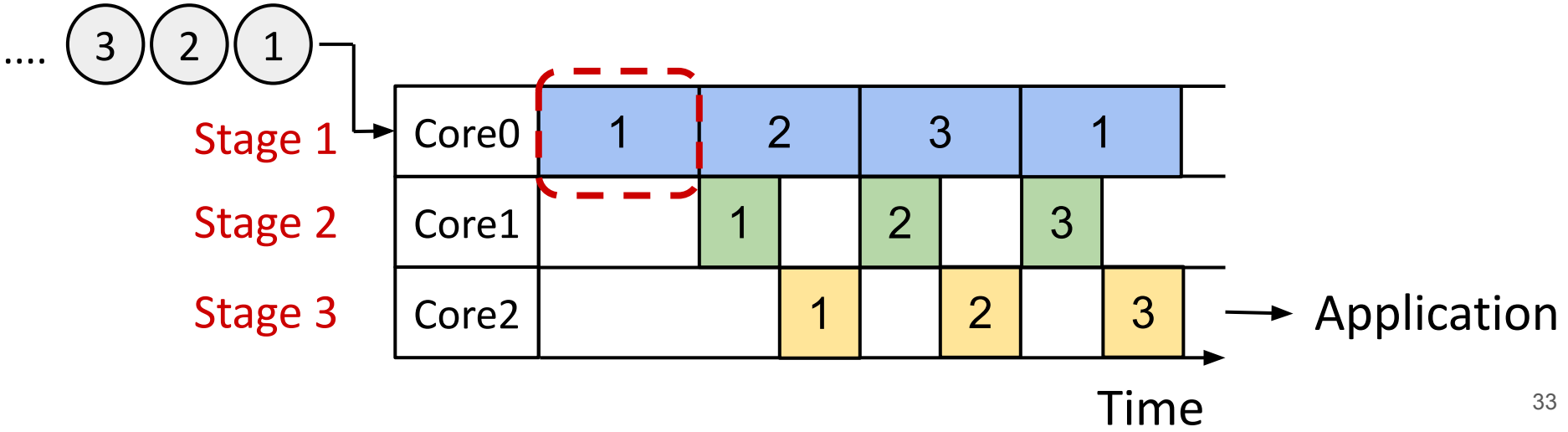
FALCON - SoftIRQs Pipelining

- Stage transition functions (Hashing)
  - **4** tuples (IPs+Ports) -> **5** tuples (IPs+Ports+**DeviceID**)
- Parallelization (Overlapping SoftIRQs)
- Maintain In-order



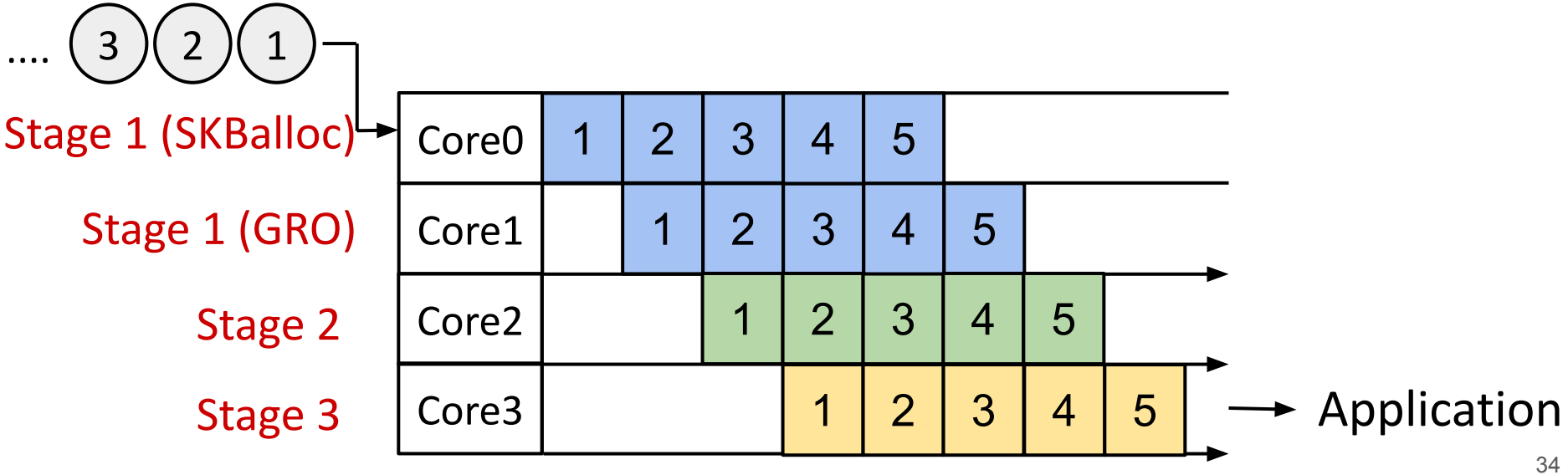
FALCON - SoftIRQs Splitting

- For TCP, the 1st stage is heavily loaded
  - SKB allocation + GRO processing



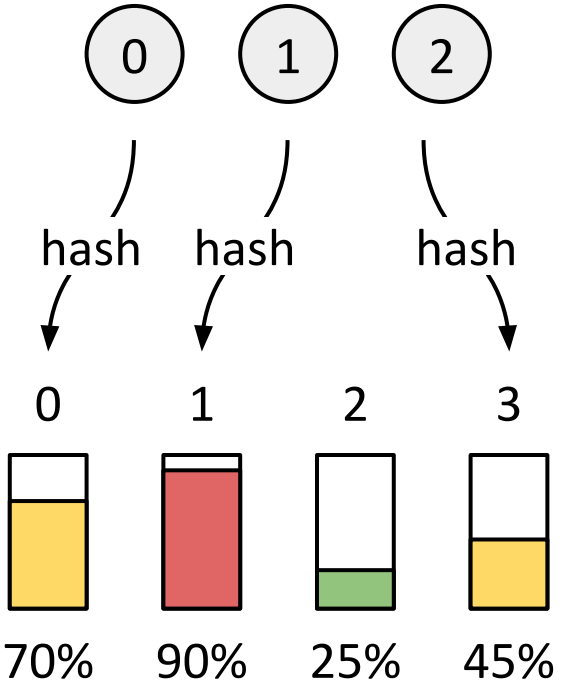
FALCON - SoftIRQs Splitting

- For TCP, the 1st stage is heavily loaded
  - SKB allocation + GRO processing
- Enable the transition function when doing GRO



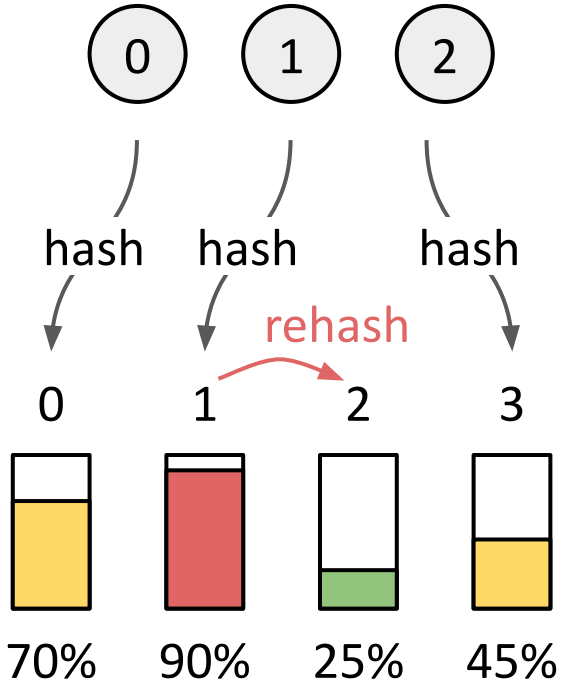
FALCON - SoftIRQs Balancing

- Dispatch SoftIRQs onto overloaded core can even make performance worse



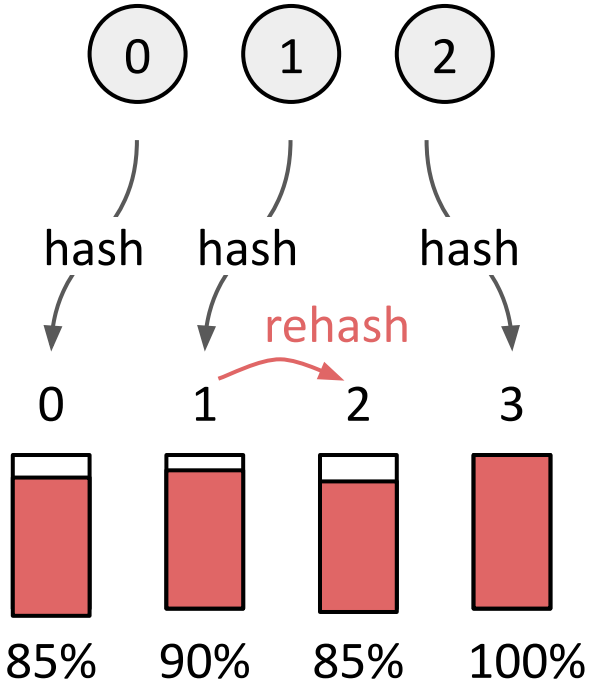
FALCON - SoftIRQs Balancing

- Dispatch SoftIRQs onto overloaded core can even make performance worse
- Rehashing based on load



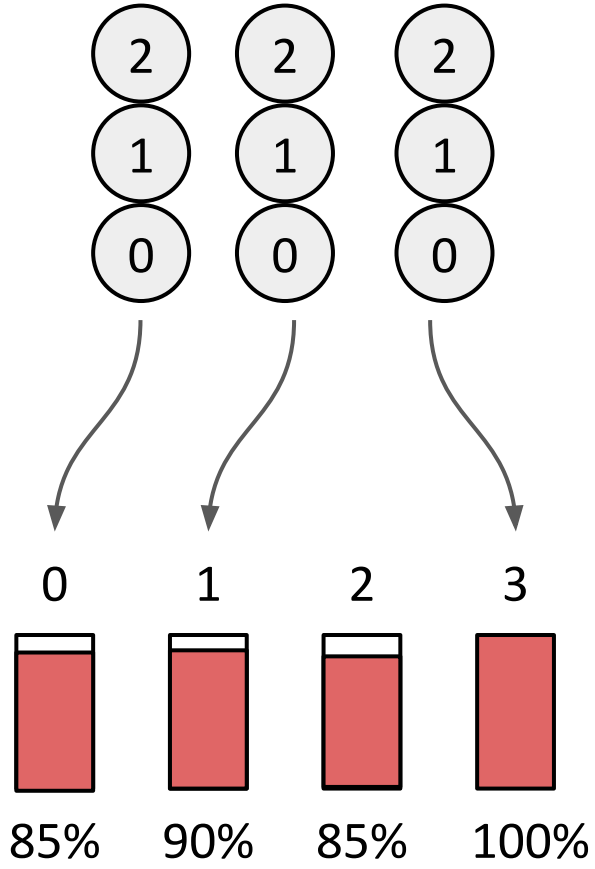
FALCON - SoftIRQs Balancing

- Dispatch SoftIRQs onto overloaded core can even make performance worse
- Rehashing based on load



FALCON - SoftIRQs Balancing

- Dispatch SoftIRQs onto overloaded core can even make performance worse
- Rehashing based on load
- When whole system is overloaded, FALCON can be dynamically disabled.



Evaluation - Setup

Hardware: Intel 40 logical cores @ 2.2GHz, 128 GB RAM
NIC: Mellanox ConnectX-5 EN (100 Gbps)
Software: Ubuntu 18.04 with Linux kernel v5.4

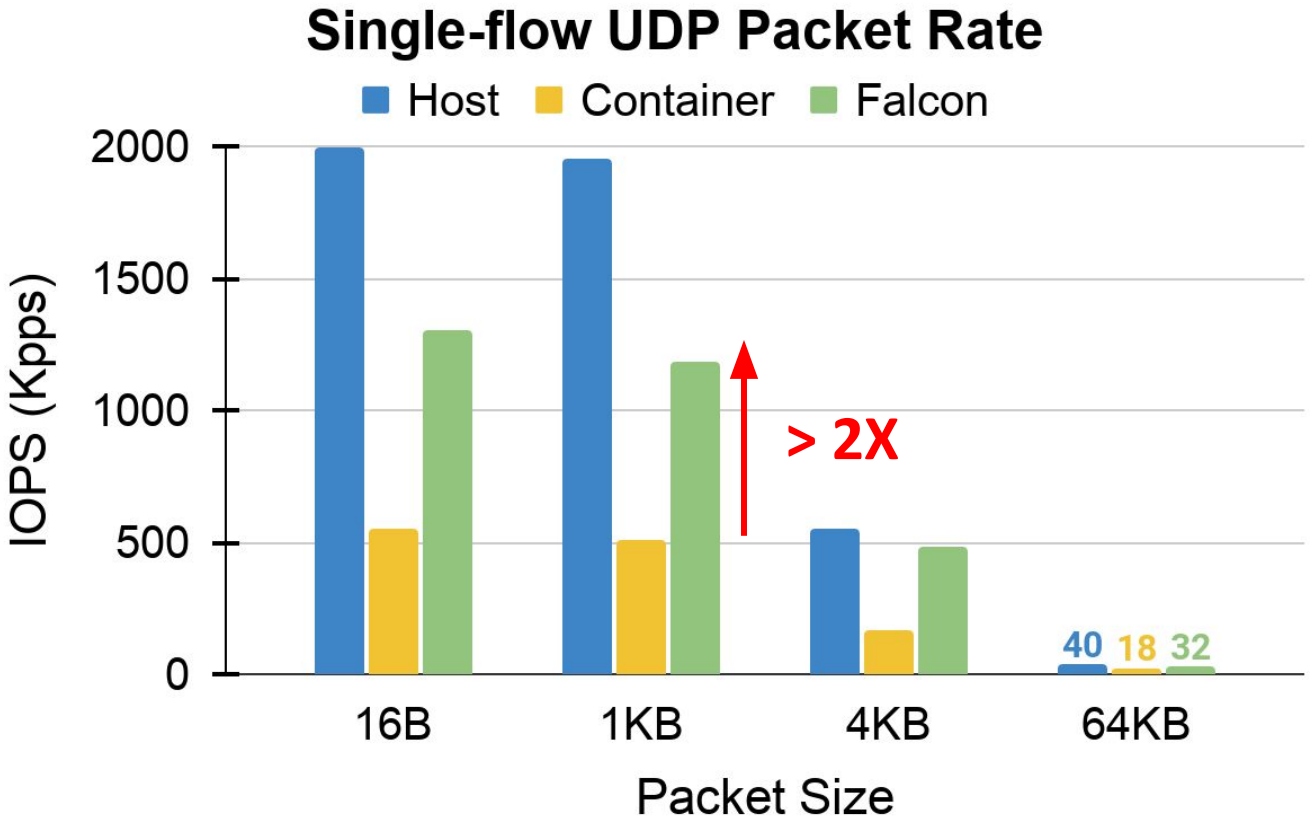
Performance comparisons: Host vs. Container vs. FALCON

Experiments:

- Single-flow and multi-flow microbenchmarks
- Load balancing
- Cloudsuite benchmark (memcached)

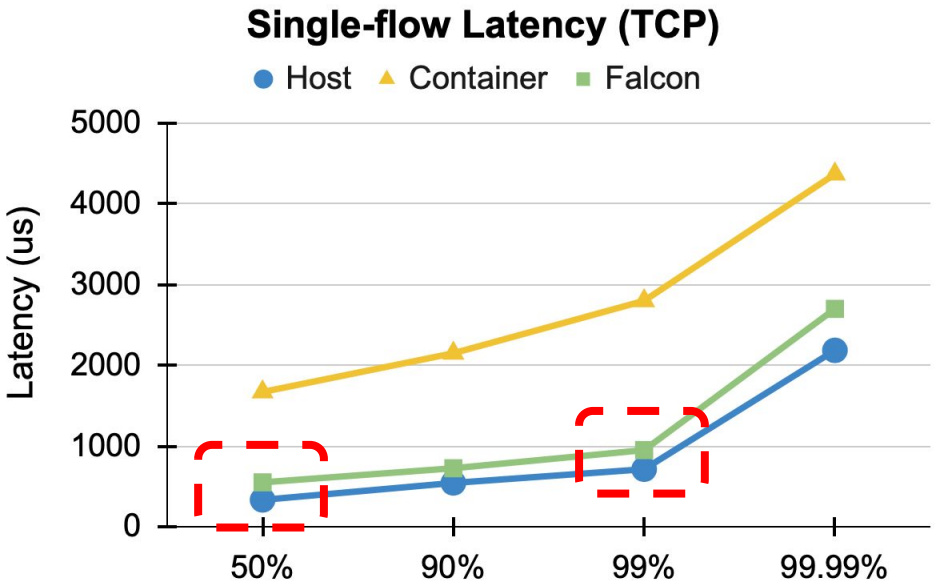
Single-flow Performance

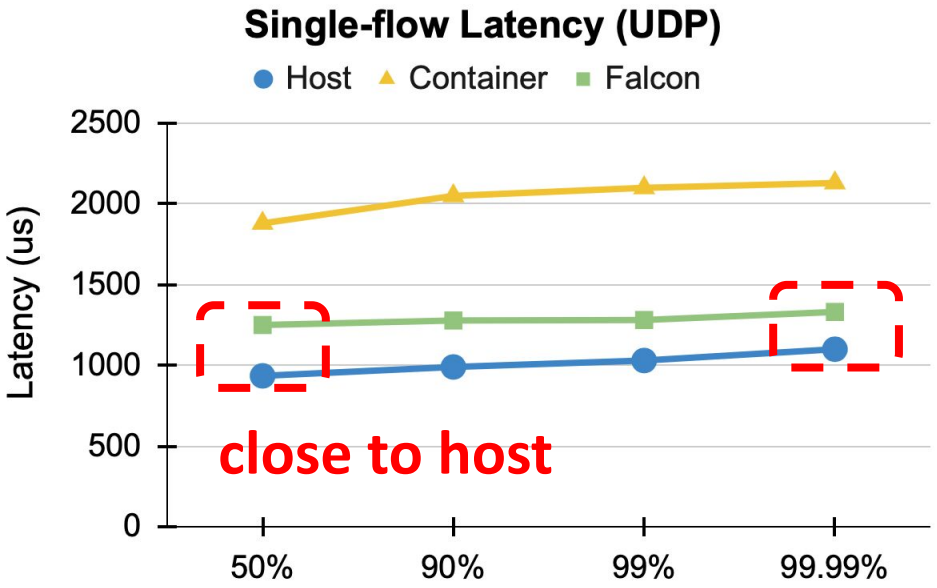
- More than **2X** improvement than vanilla container



Single-flow Performance

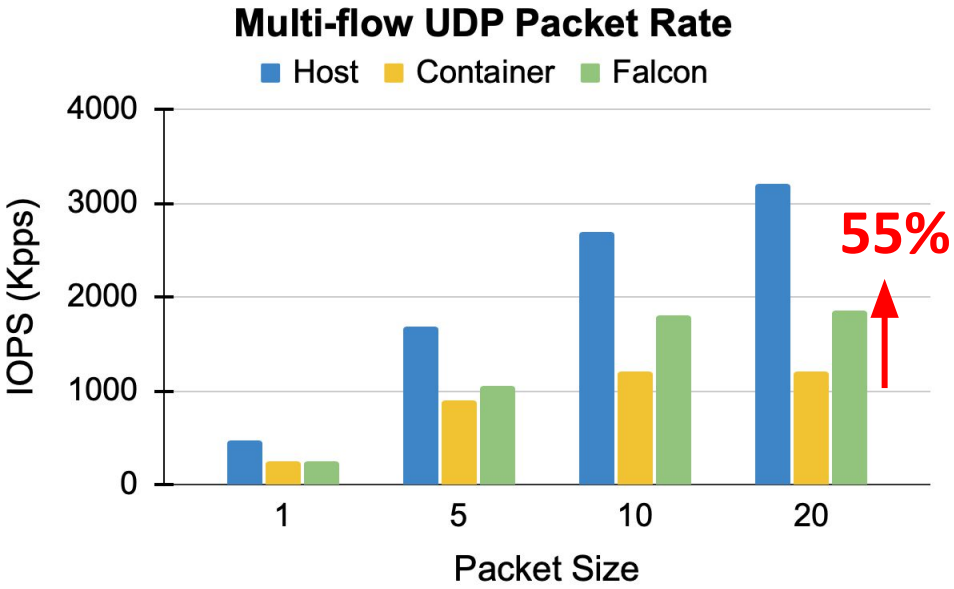
- Both median and tail latency get **close to host** networks.

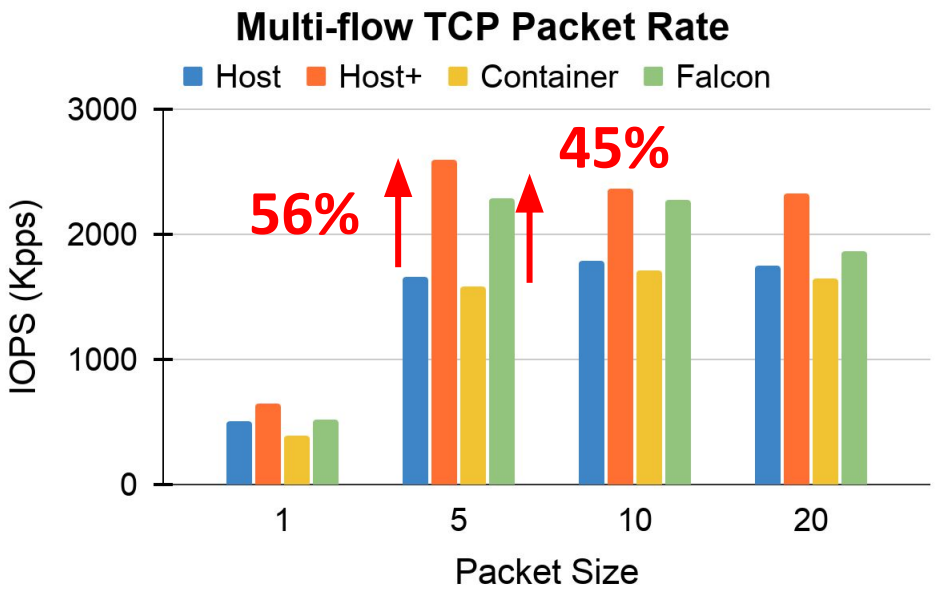




Multi-flow Performance

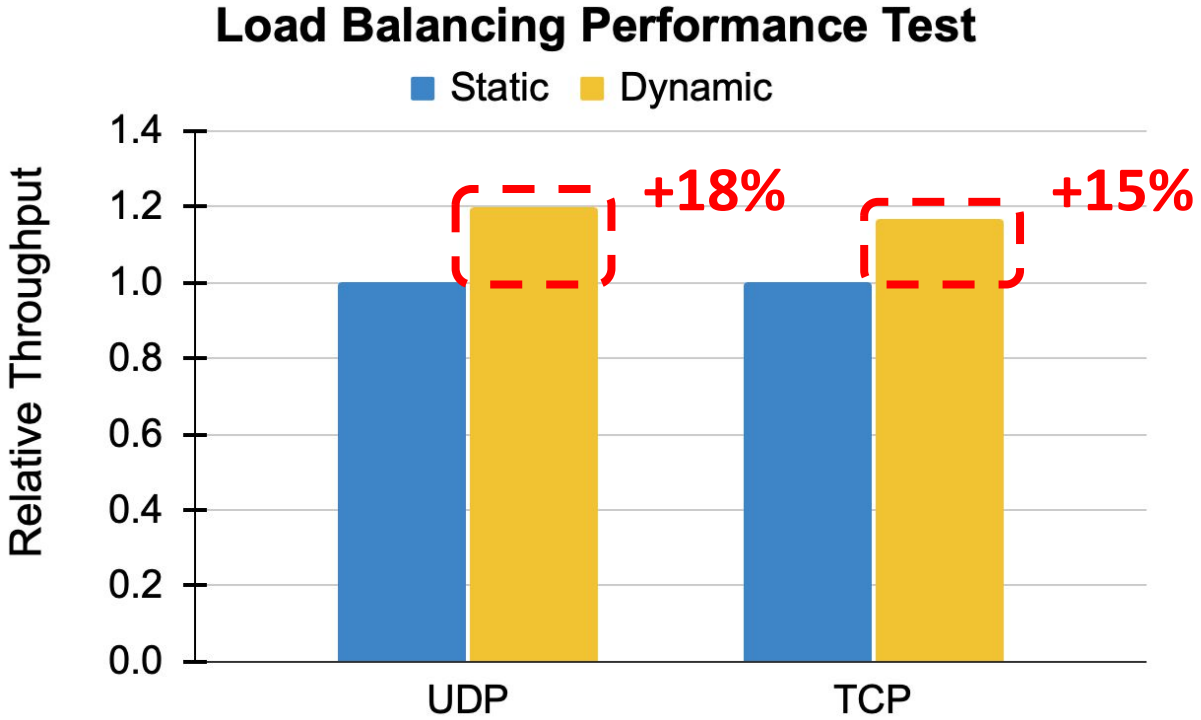
- UDP: as much as **55%** improvement
- TCP: improve host network by **56%**, overlay network by **45%**





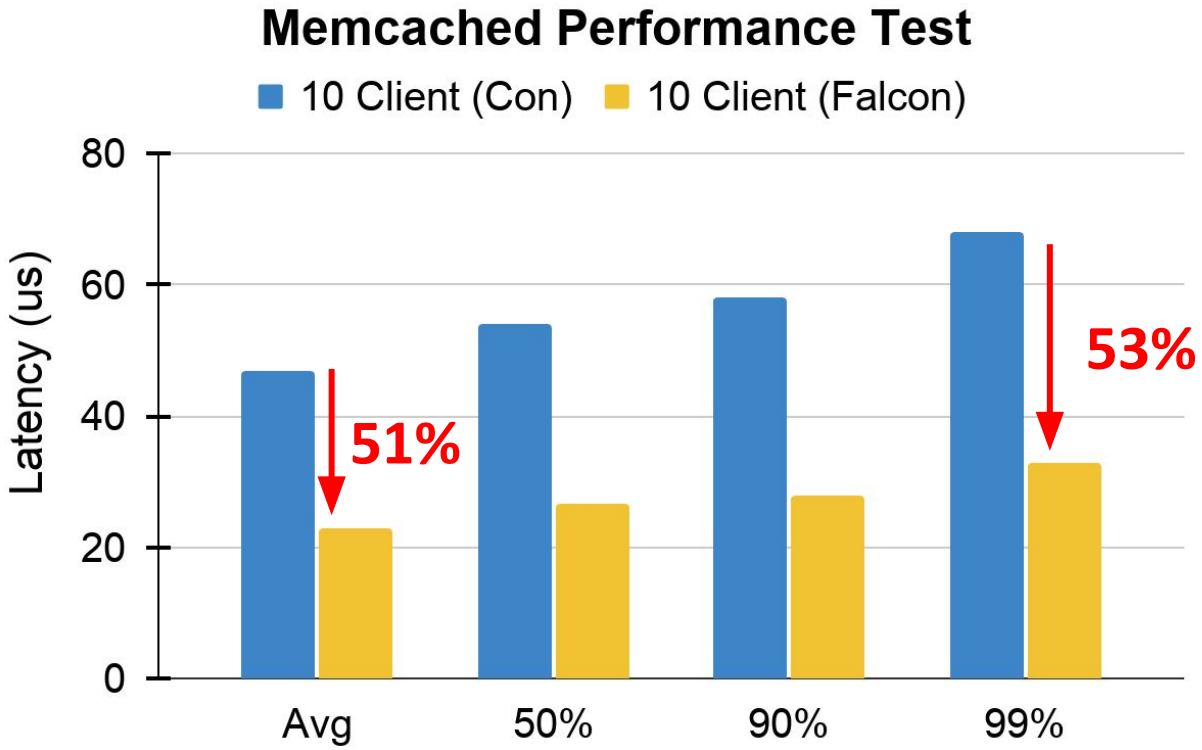
Load Balancing

- Increase the intensity of certain flows suddenly
- Improvement: UDP - **18%** ; TCP - **15%**



Memcached Performance

- With 10 clients, Falcon reduces the average and tail latency by **51%** and **53%**



Conclusions

- Excessive, serialized and expensive SoftIRQs incur significant performance loss for overlay networks.
- FALCON parallelizes the SoftIRQs processing by utilizing the multicore architecture.
- FALCON can significantly improve the performance of container overlay networks.
- Our implementation is open-sourced at: github.com/munikarmanish/falcon

Thank you!