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
- **Higher** application density
- **Better** resource utilization

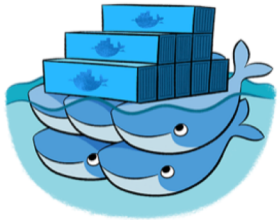
Containers are taking over the cloud

- Lightweight
- OS level virtualization
- **Higher** application density
- **Better** 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 tunneling approach like **VxLAN** protocol

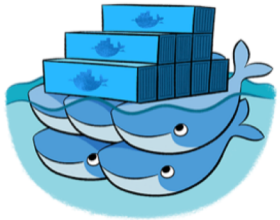


flannel

weaveworks

How do containers communicate?

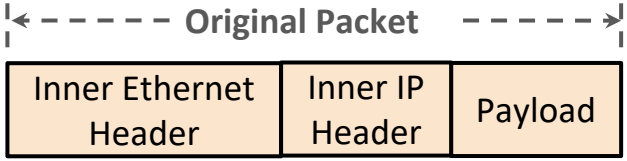
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon tunneling approach like **VxLAN** protocol



flannel

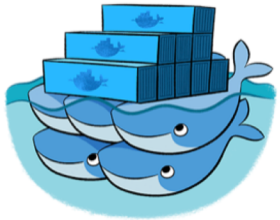
PROJECT CALICO

weaveworks



How do containers communicate?

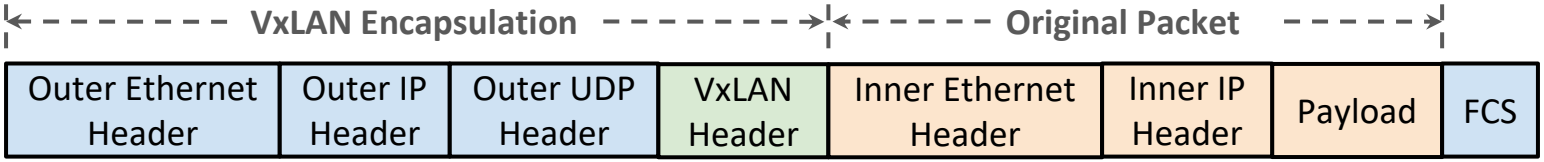
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon tunneling approach like **VxLAN** protocol



flannel

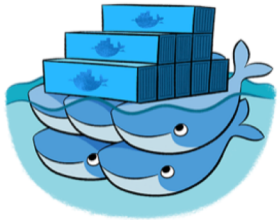
PROJECT CALICO

weaveworks



How do containers communicate?

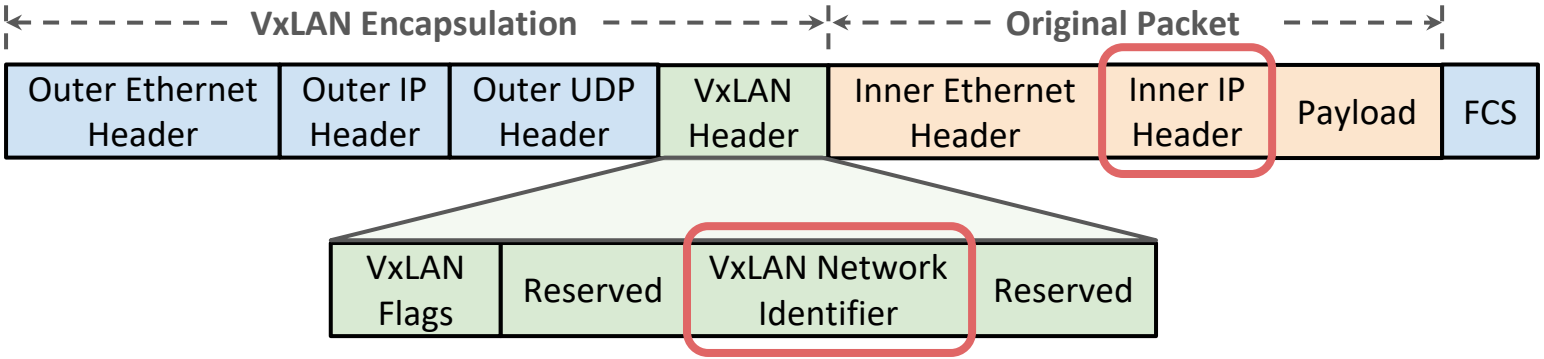
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon tunneling approach like **VxLAN** protocol



flannel

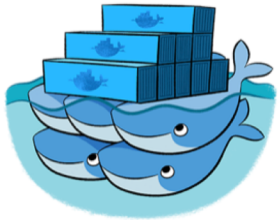
PROJECT CALICO

weaveworks



How do containers communicate?

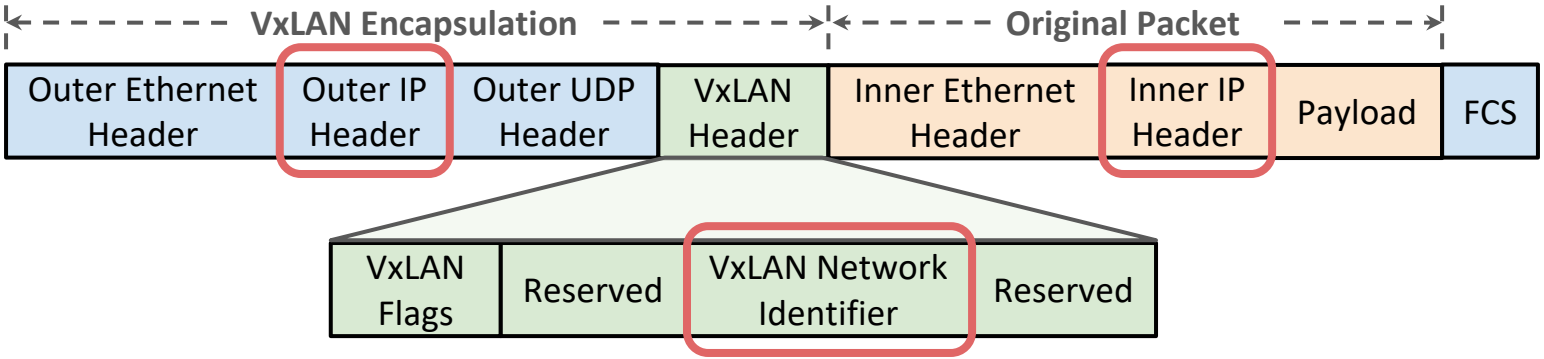
- **Overlay networks** provide connectivity
- Typical solutions: Docker Overlay, Flannel, Calico, Weave…
- Generally build upon tunneling approach like **VxLAN** protocol



flannel

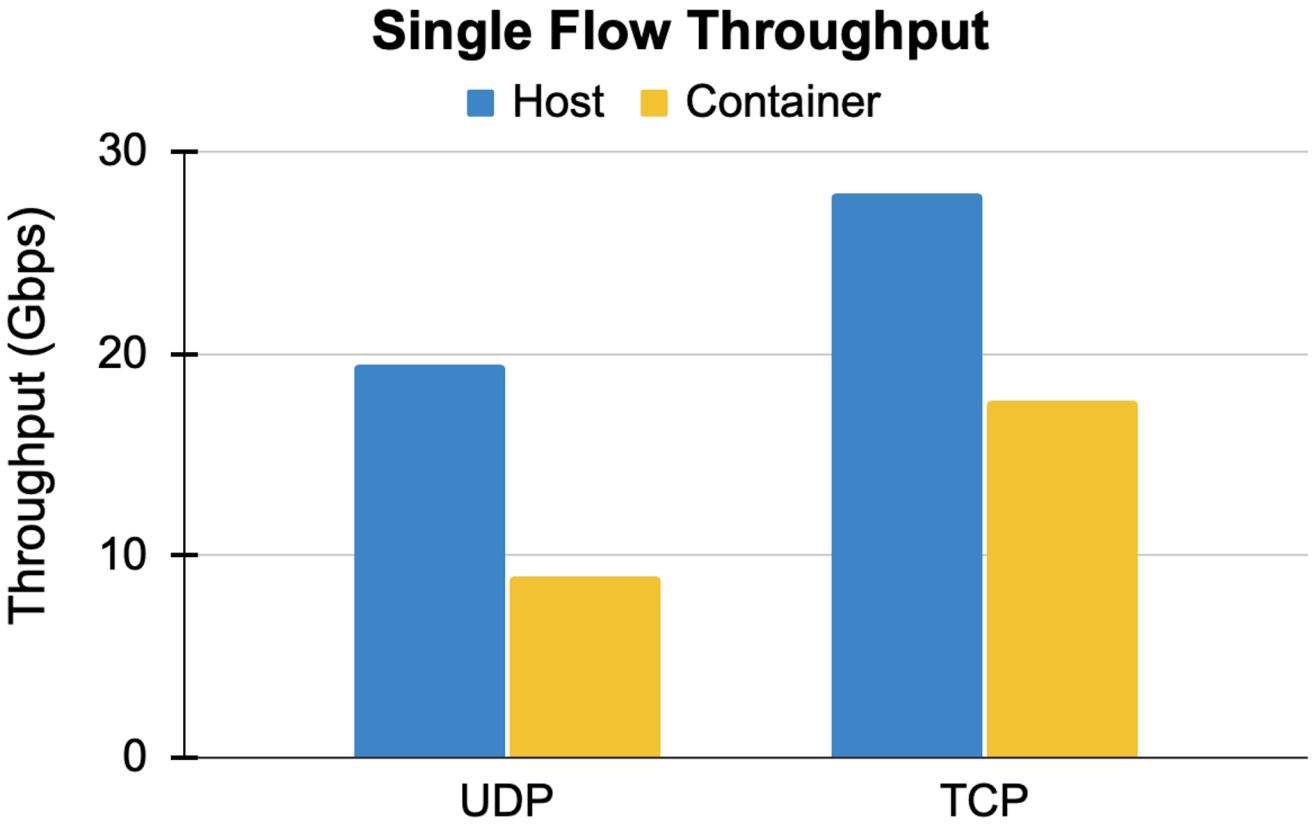
PROJECT CALICO

weaveworks



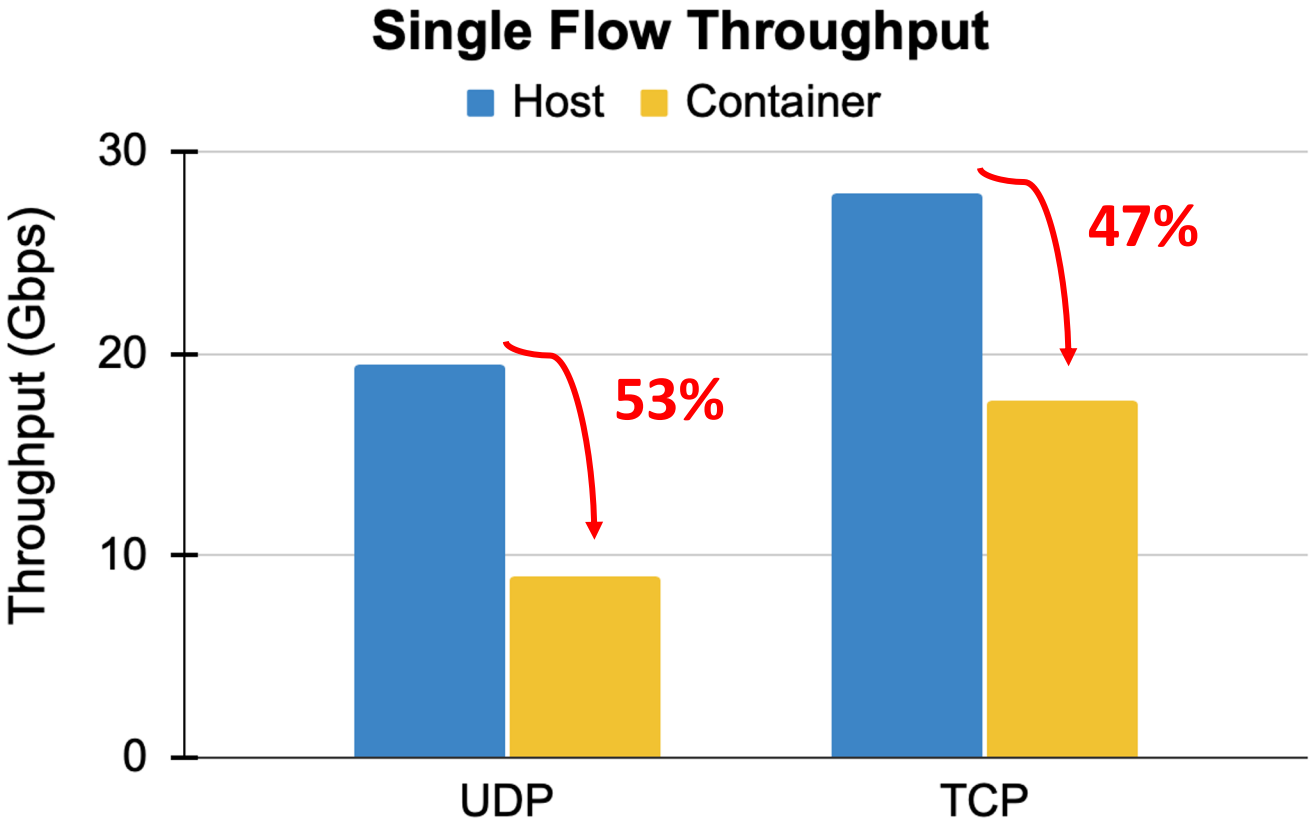
Performance overhead of overlay networks

- Machines are connected via 100 GbE NIC.



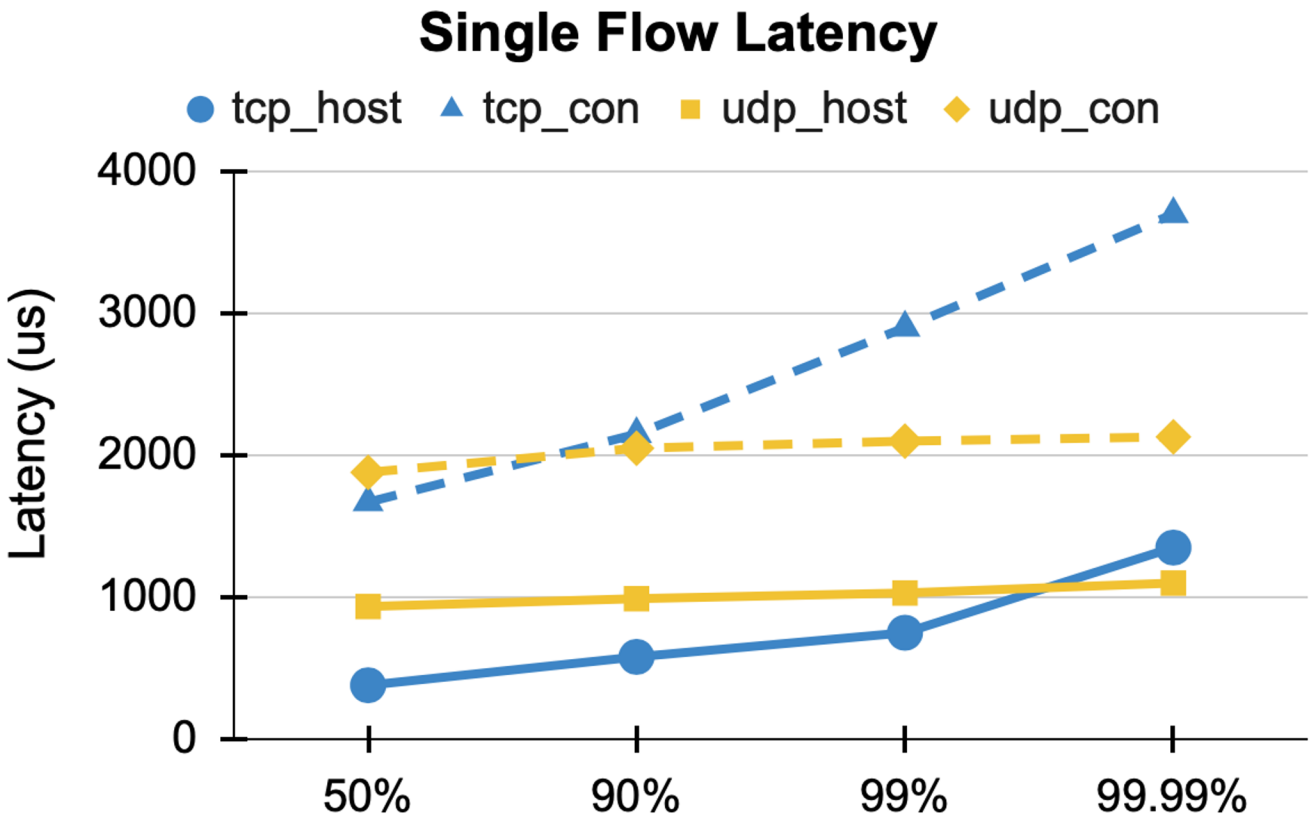
Performance overhead of overlay networks

- Machines are connected via 100 GbE NIC.



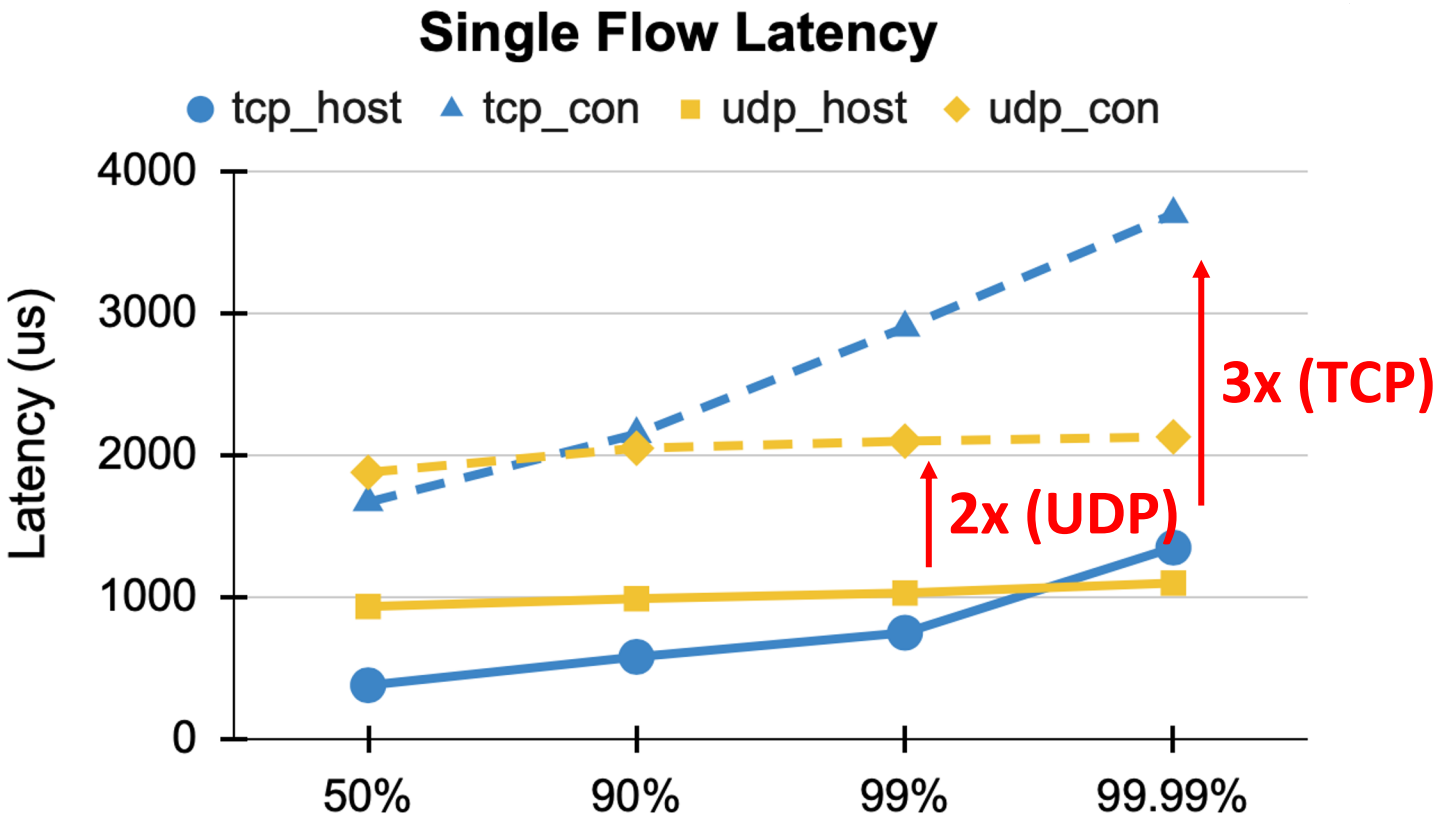
Performance overhead of overlay networks

- Machines are connected via 100 GbE NIC.



Performance overhead of overlay networks

- Machines are connected via 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?

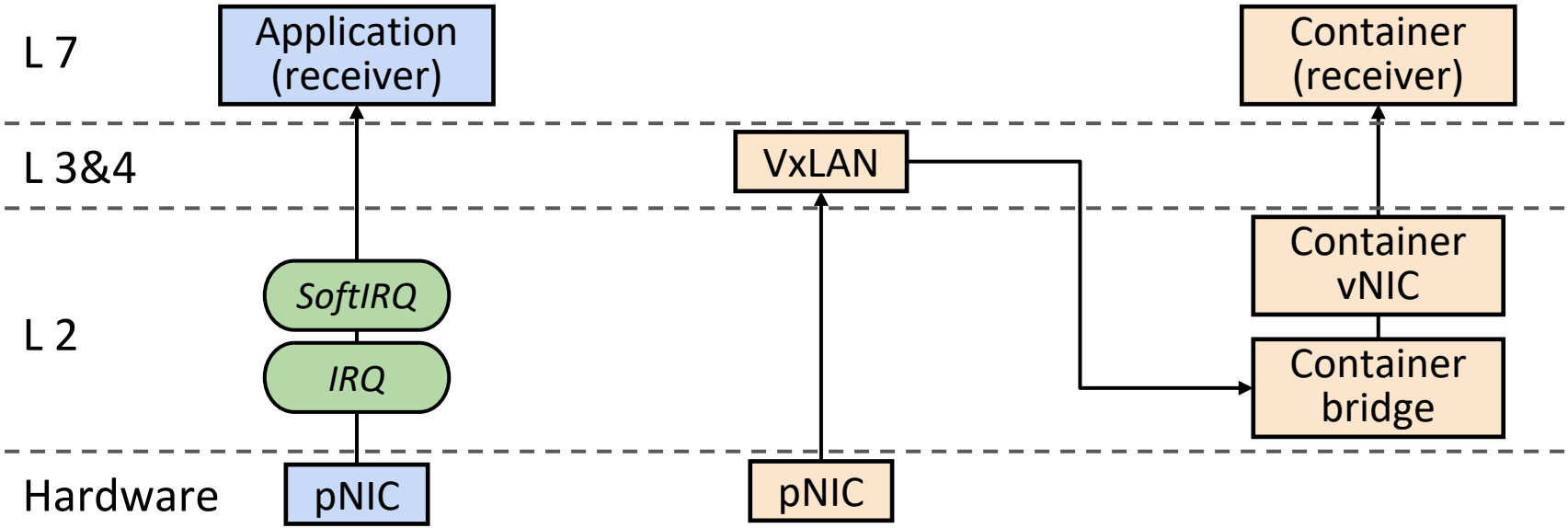
- **Host Networks**

Why is overlay network slow?

- **Host Networks**
  - IRQ + SoftIRQ

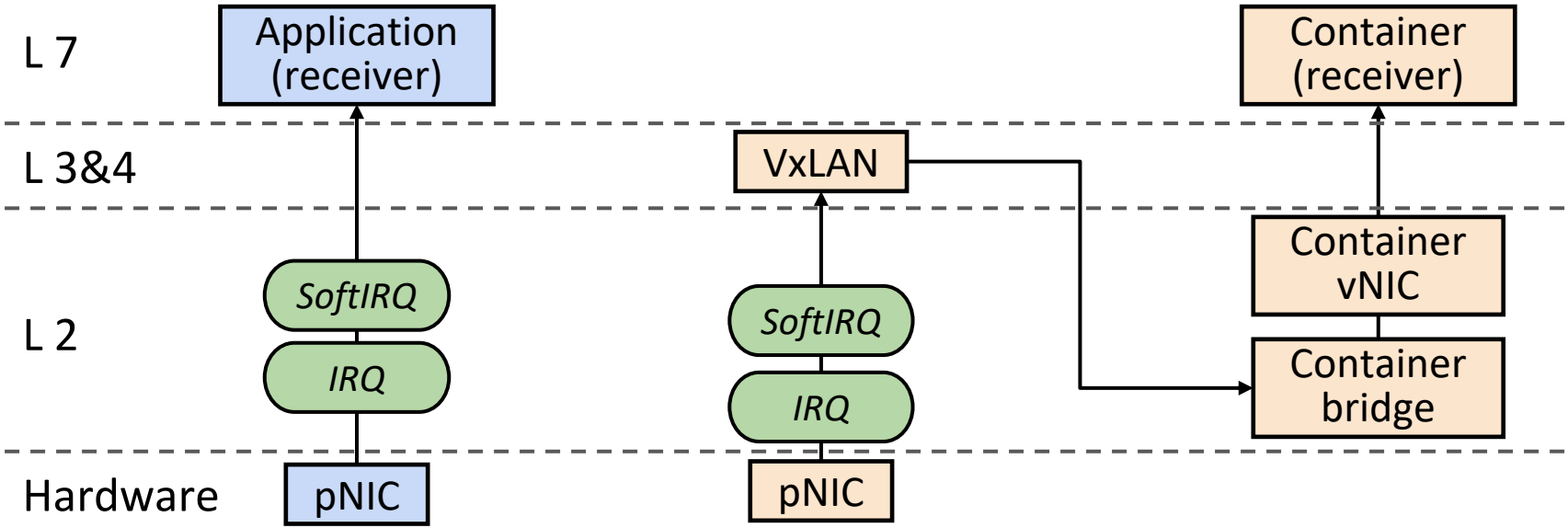
Why is overlay network slow?

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



Why is overlay network slow?

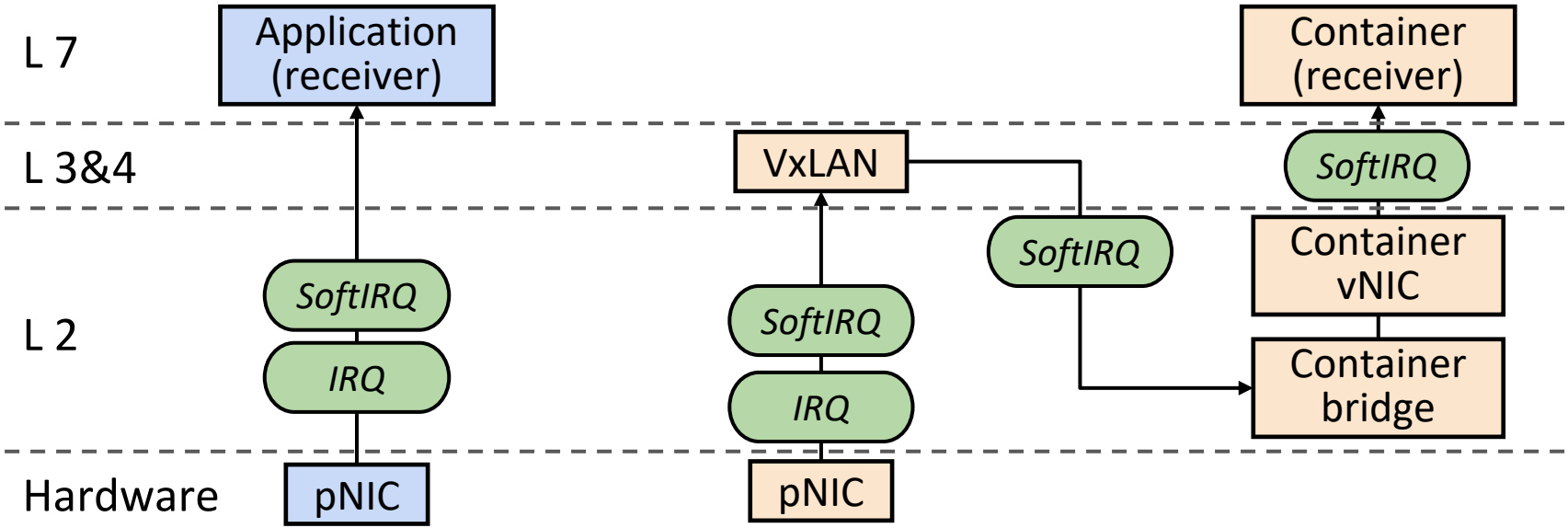
- **Host Networks**
  - IRQ + SoftIRQ
- **Overlay Networks**



Why is overlay network slow?

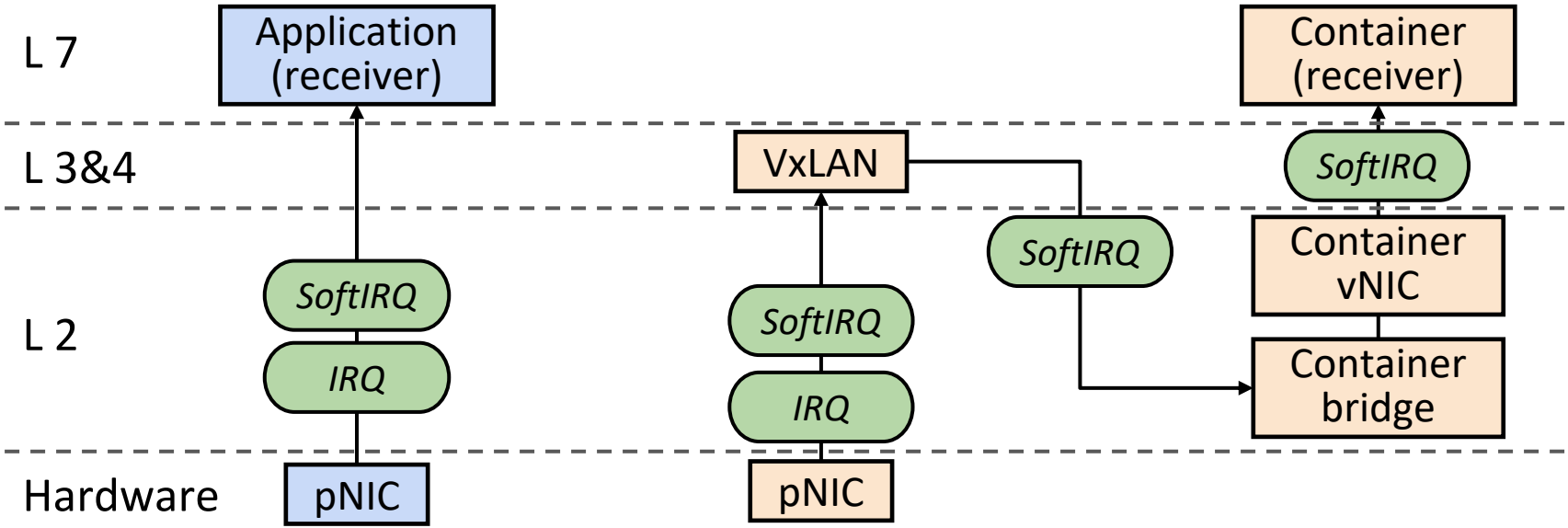
- **Host Networks**
  - IRQ + SoftIRQ

- **Overlay Networks**



Why is overlay network slow?

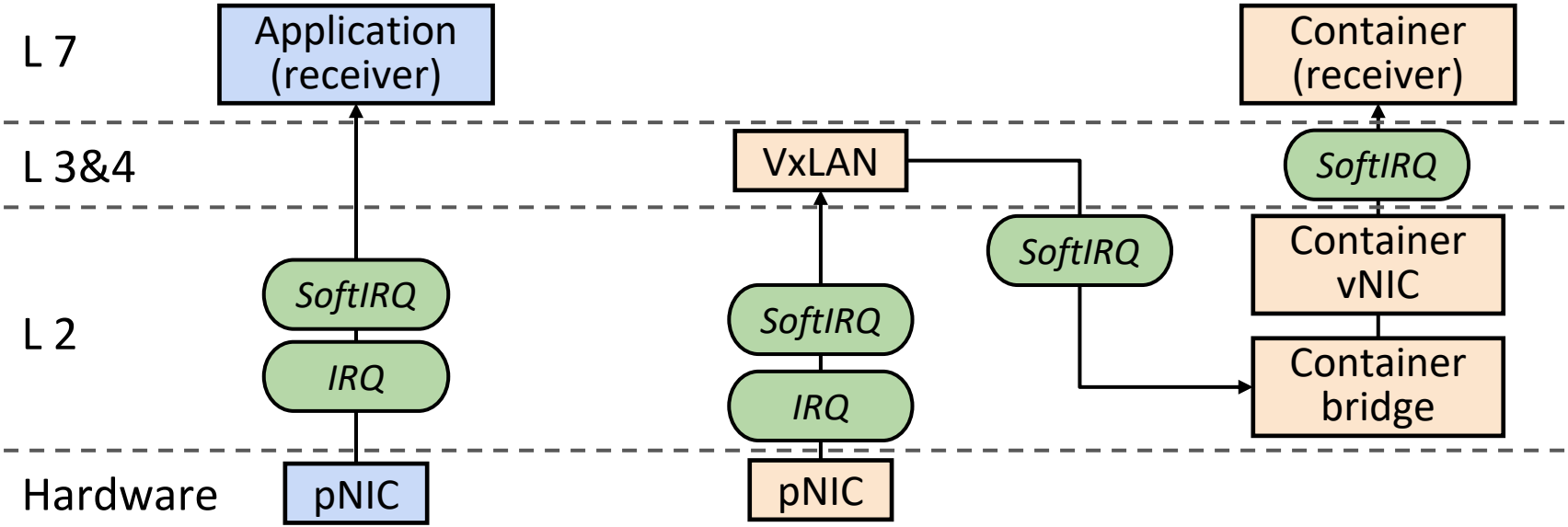
- **Host Networks**
  - IRQ + SoftIRQ
- **Overlay Networks**
  - IRQ + **3x**SoftIRQs



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)
  - ✓ Avoids OS overheads, network stack tailored to application
  - ✗ Operator has no control over network stack, have to trust application

Existing solutions to accelerate overlay networks

- Kernel bypass (DPDK+mTCP)
  - ✓ Avoids OS overheads, network stack tailored to application
  - ✗ Operator has no control over network stack, have to trust application
- Connection-level metadata transformation (Slim, FreeFlow)
  - ✓ Avoids overhead of virtual devices, fast as host network
  - ✗ Limited scalability; only for TCP; doesn't support data-plane policies

Existing solutions to accelerate overlay networks

- Kernel bypass (DPDK+mTCP)
  - ✓ Avoids OS overheads, network stack tailored to application
  - ✗ Operator has no control over network stack, have to trust application
- Connection-level metadata transformation (Slim, FreeFlow)
  - ✓ Avoids overhead of virtual devices, fast as host network
  - ✗ Limited scalability; only for TCP; doesn't support data-plane policies
- Hardware offload (Mellanox ASAP², AccelNet)
  - ✓ Fastest, completely avoids CPU overheads
  - ✗ Requires new/expensive hardware; SR-IOV limitations; limited isolation/flexibility/scalability

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

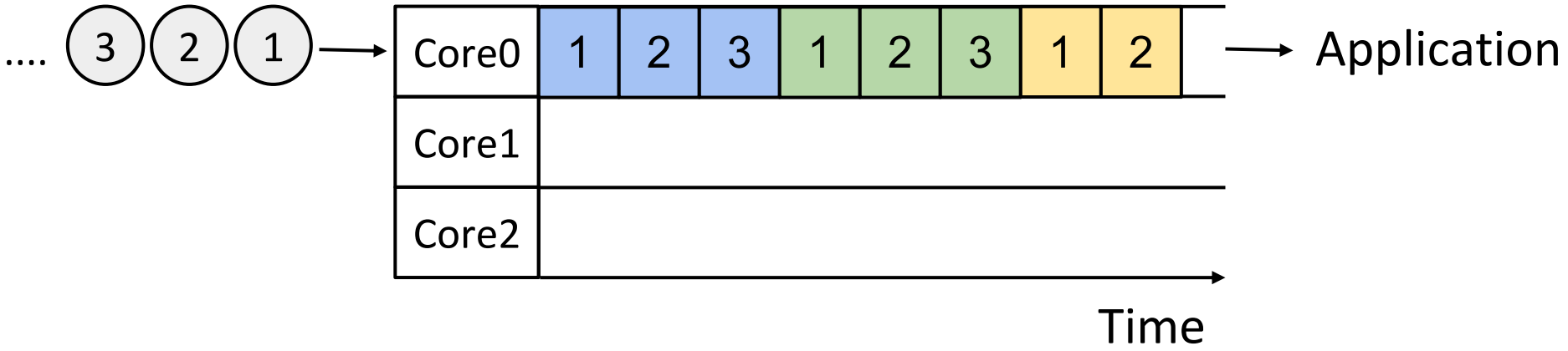
FALCON = **F**ast and B**al**anced **Co**ntainer **N**etworking

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

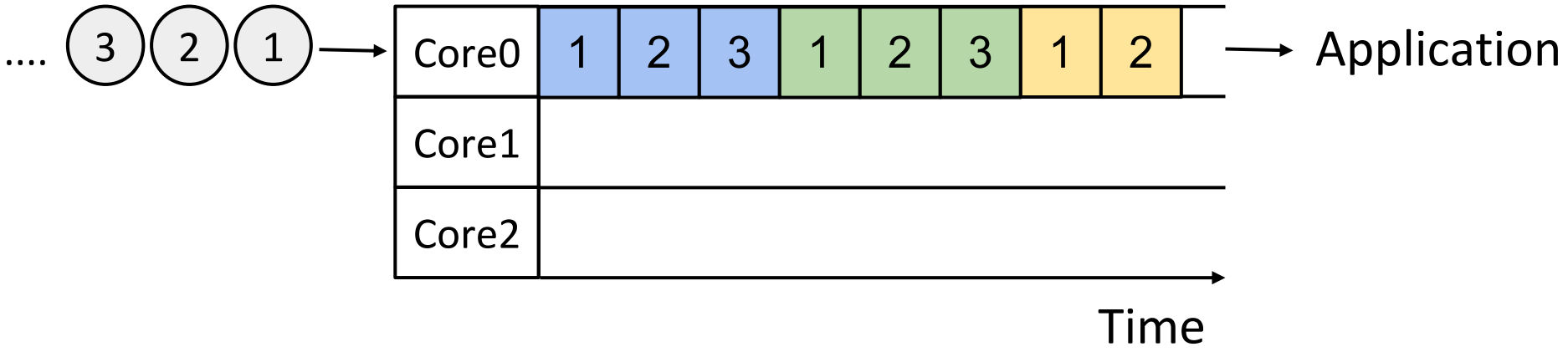
FALCON - SoftIRQs Pipelining

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



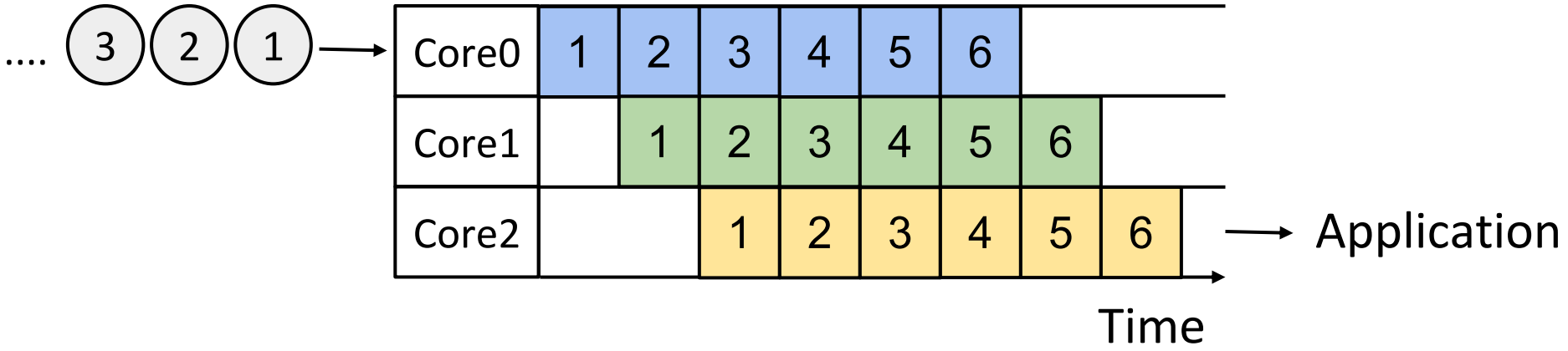
FALCON - SoftIRQs Pipelining

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



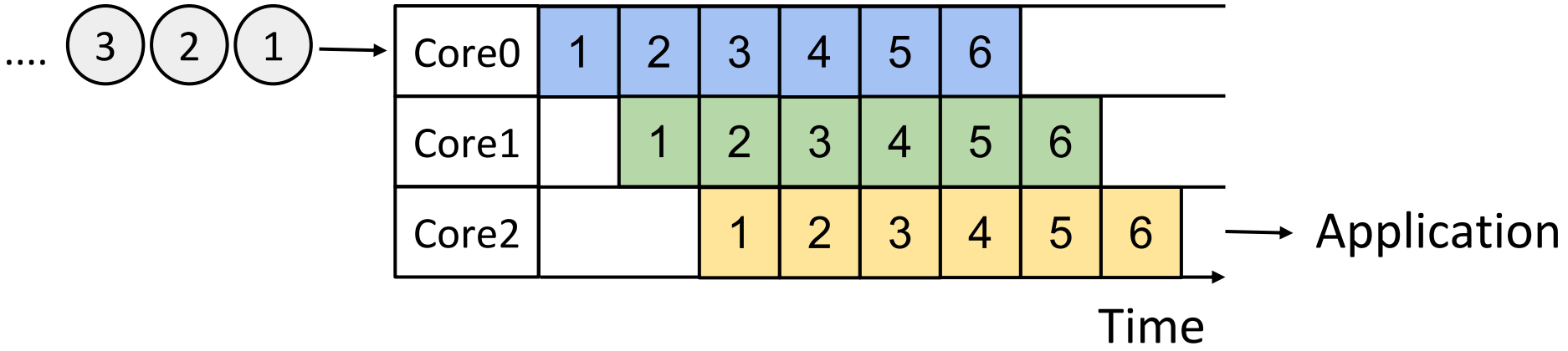
FALCON - SoftIRQs Pipelining

- Stage transition functions (Hashing)



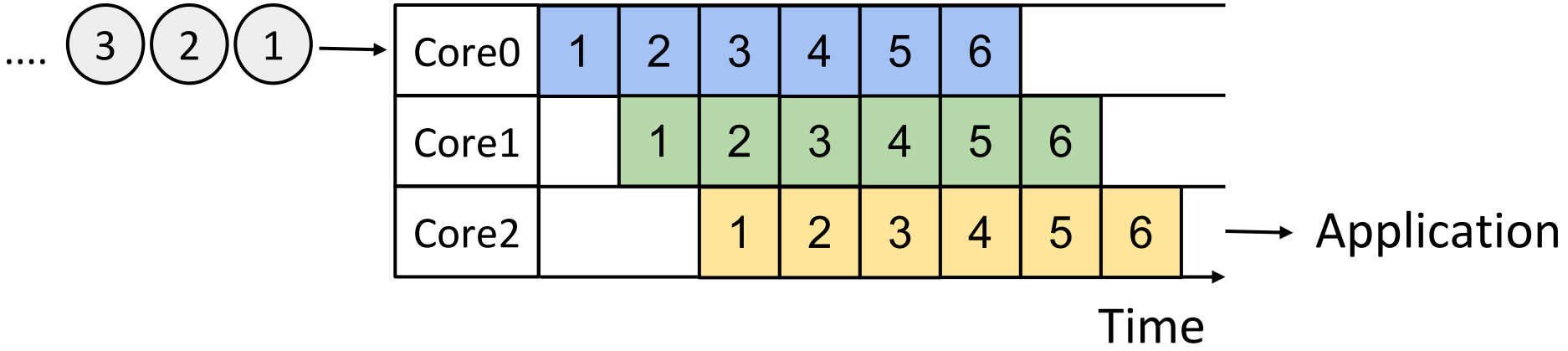
FALCON - SoftIRQs Pipelining

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



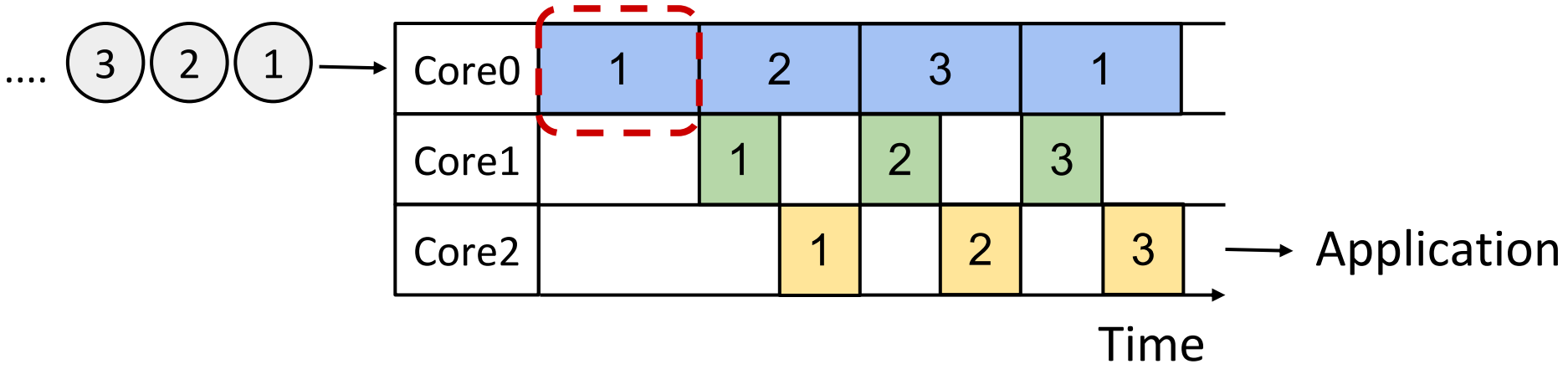
FALCON - SoftIRQs Pipelining

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



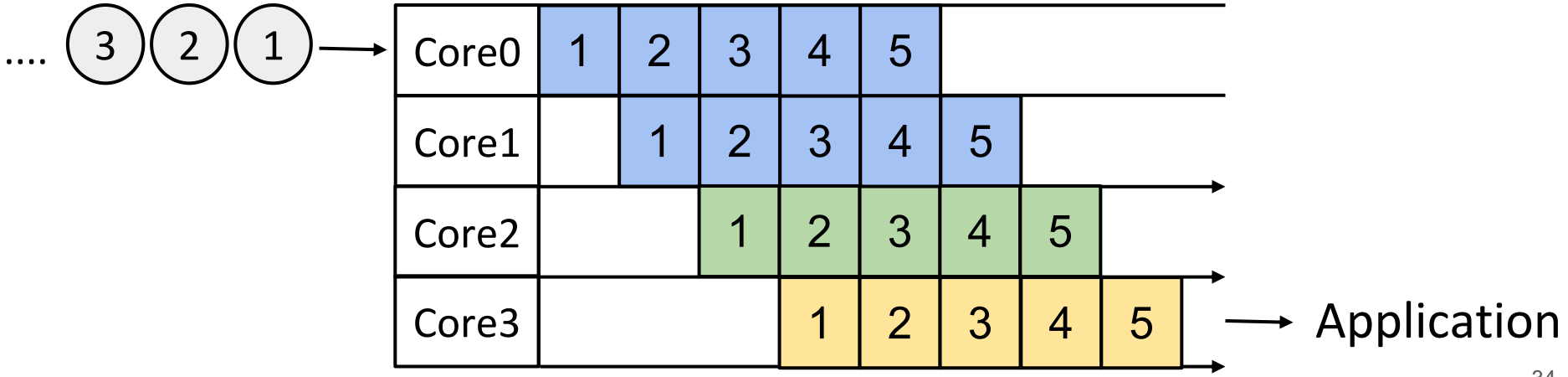
FALCON - SoftIRQs Splitting

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



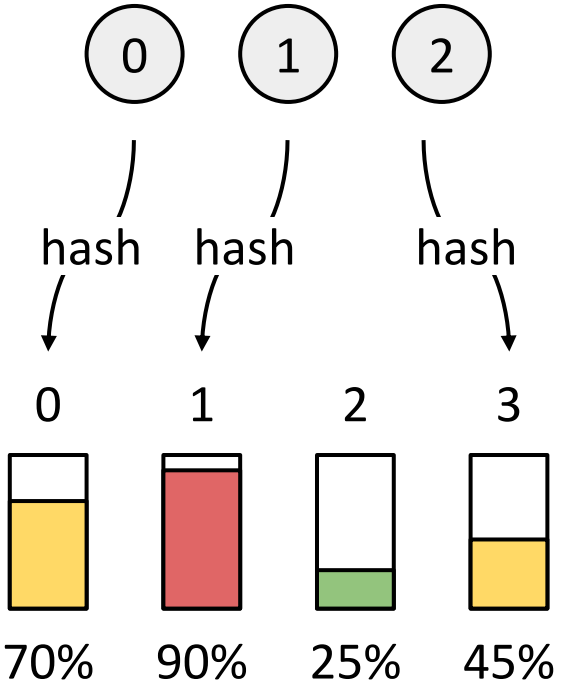
FALCON - SoftIRQs Splitting

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



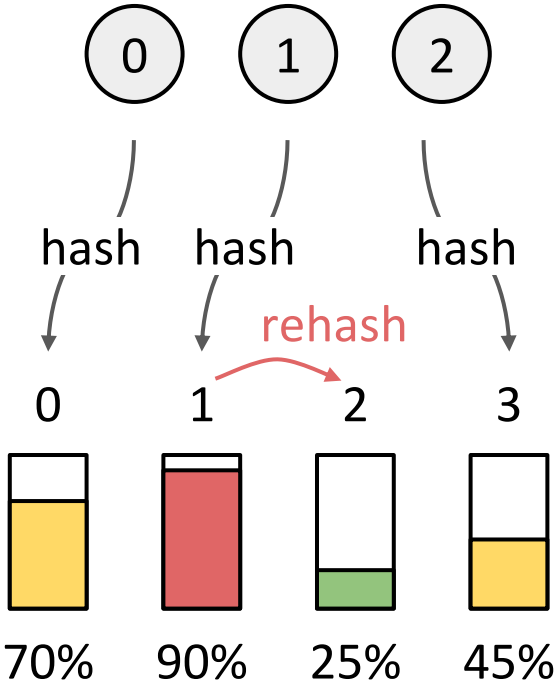
FALCON - SoftIRQs Balancing

- Dispatch SoftIRQs onto overloaded core can even make performance worse



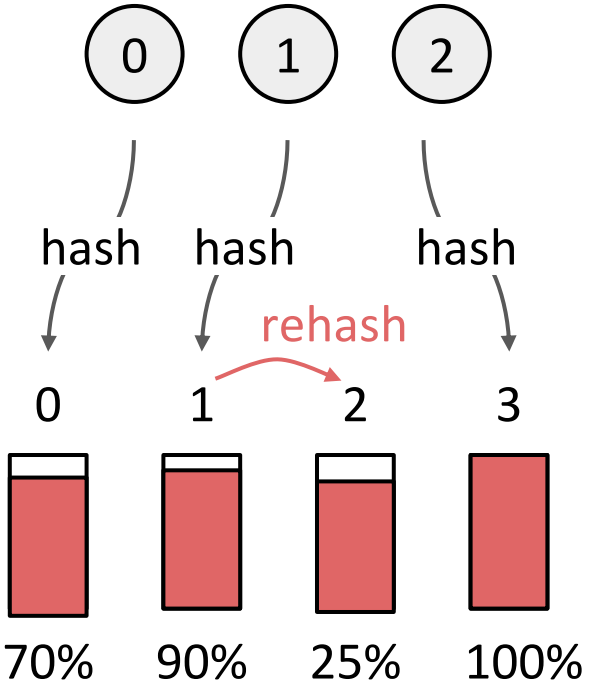
FALCON - SoftIRQs Balancing

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



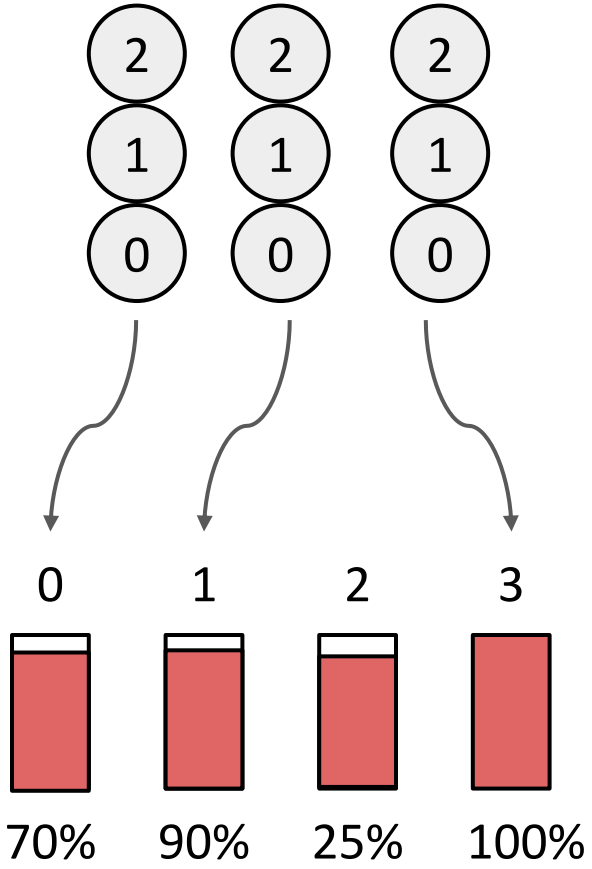
FALCON - SoftIRQs Balancing

- 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

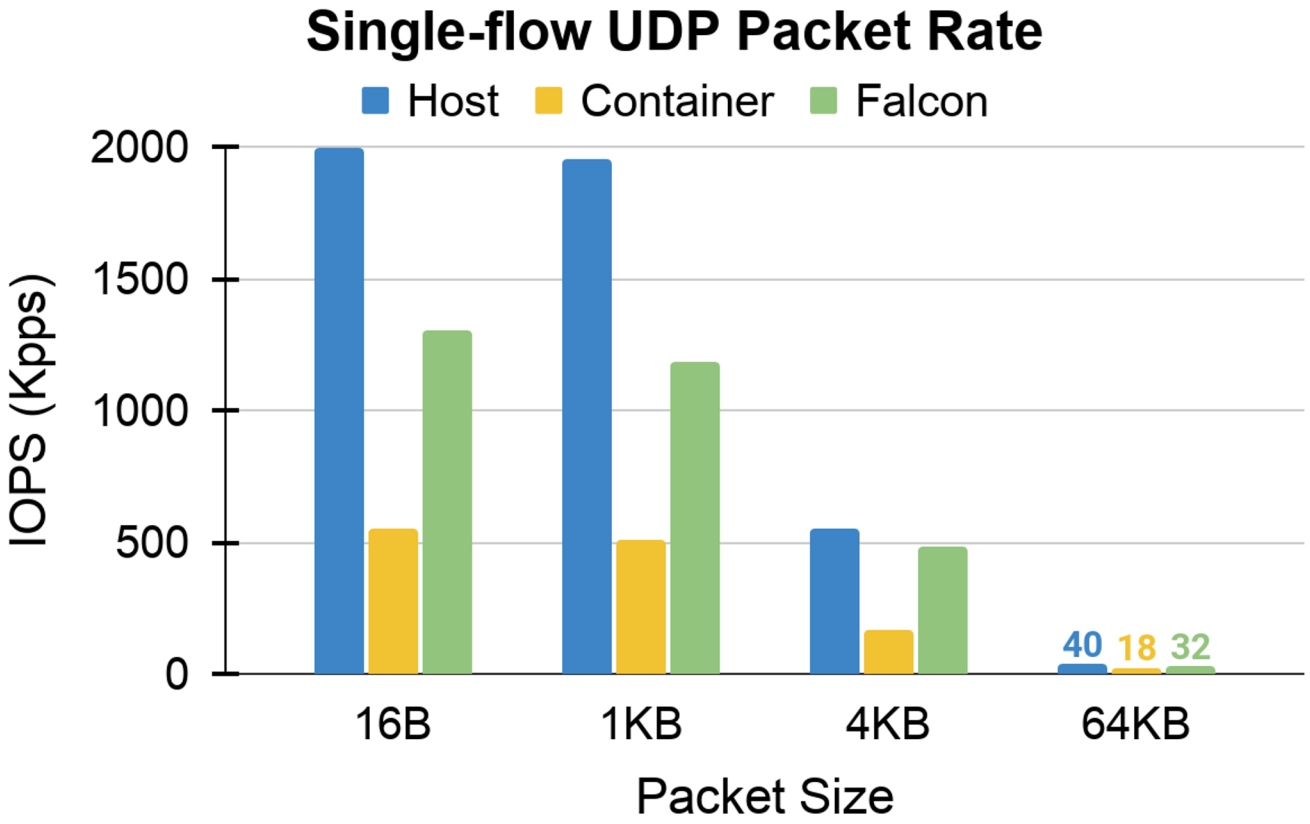
Performance comparisons: Host vs. Container vs. FALCON

Experiments:

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

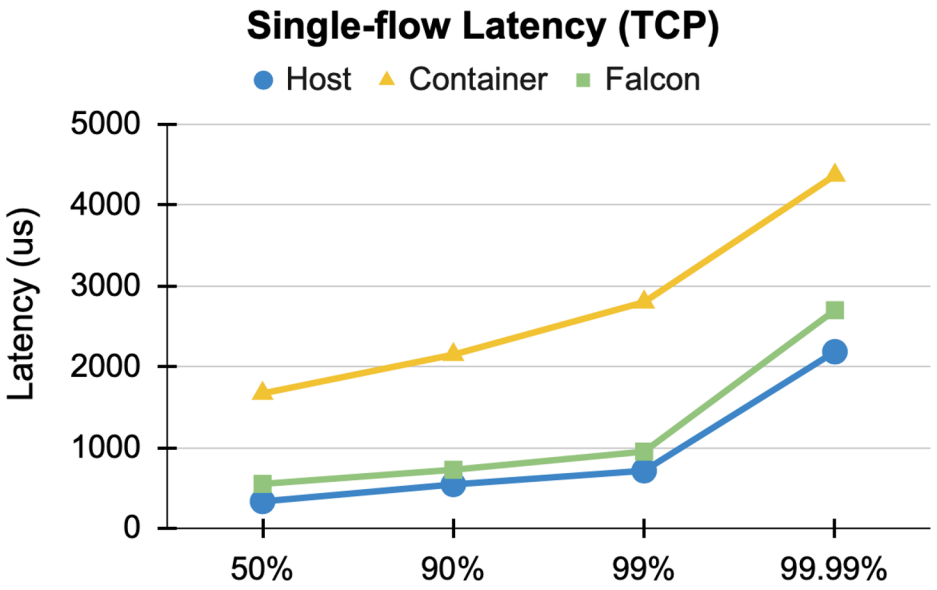
Single-flow Performance

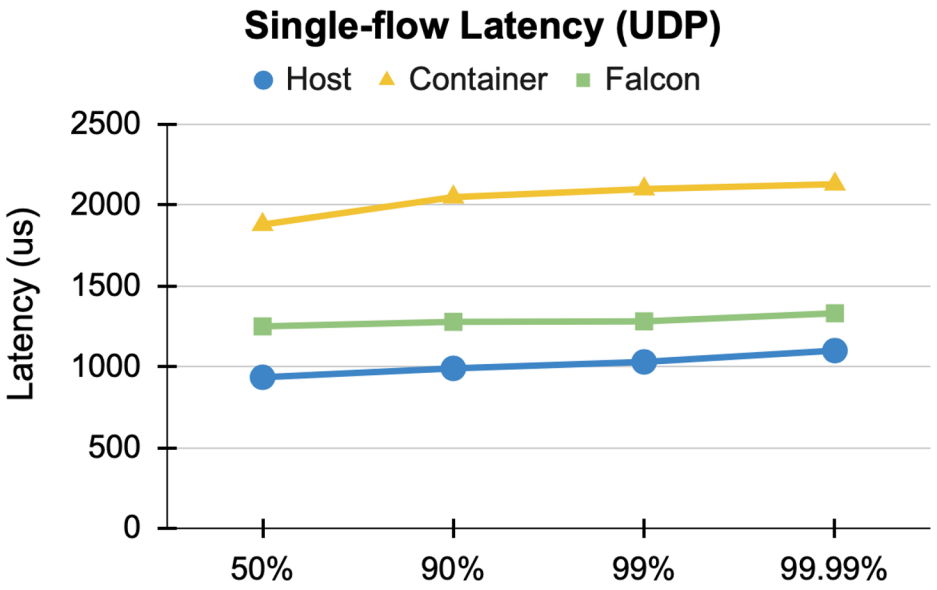
- More than **2x** improvement than vanilla container



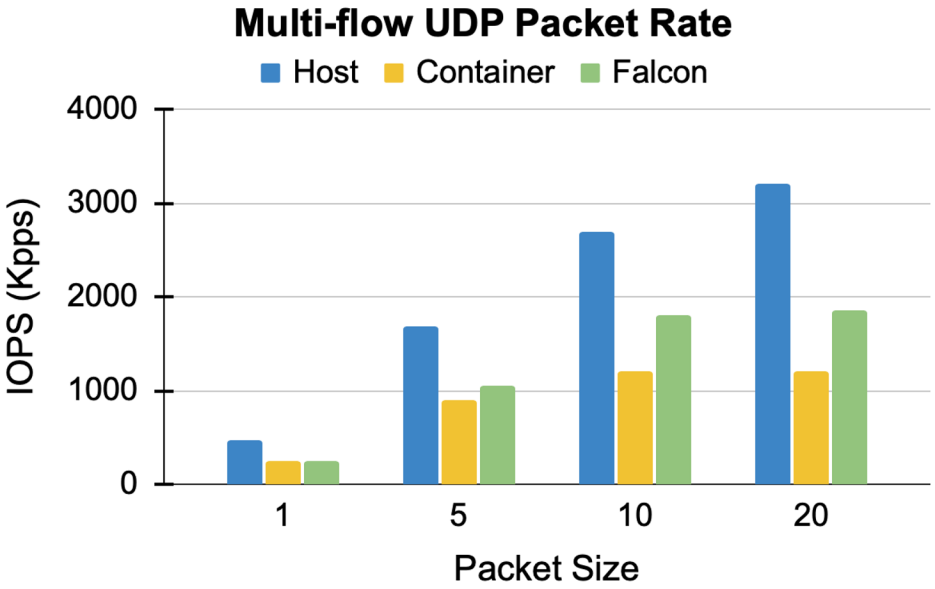
Single-flow Performance

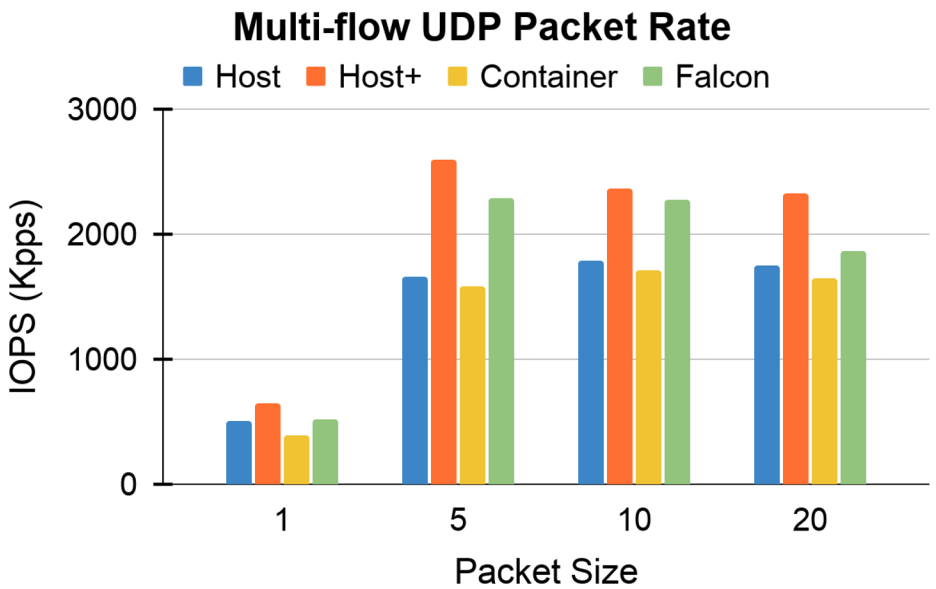
- Both median and tail latency get close to host networks



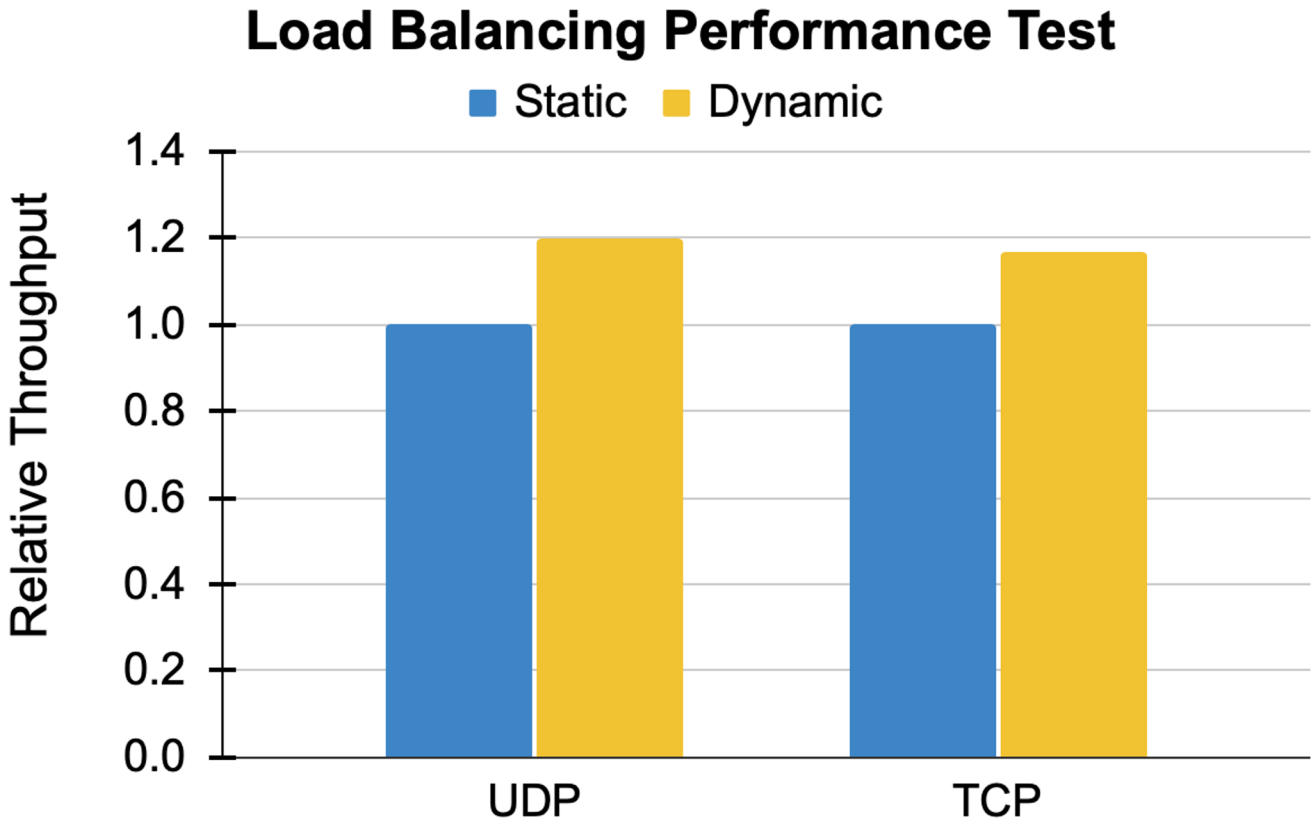


Multi-flow Performance

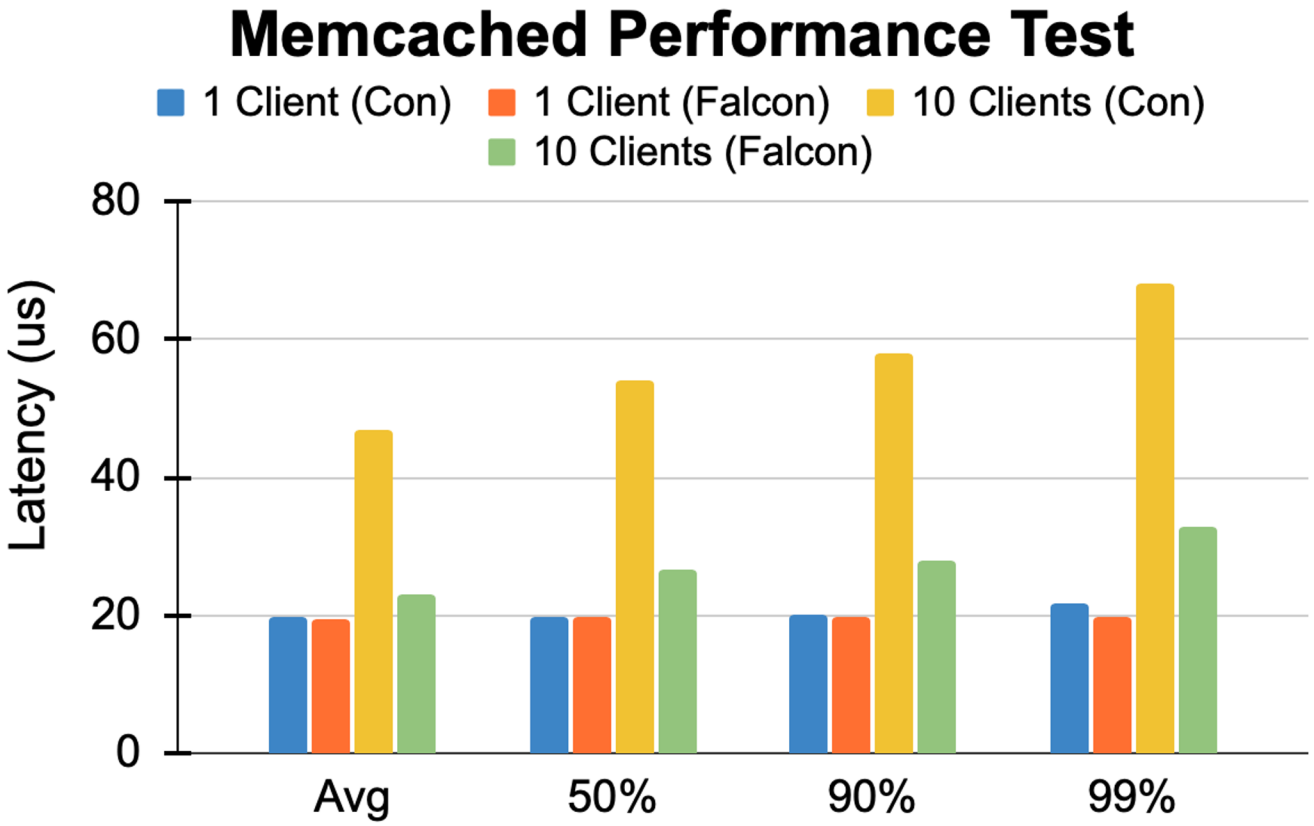




Load Balancing



Memcached Performance



Conclusion

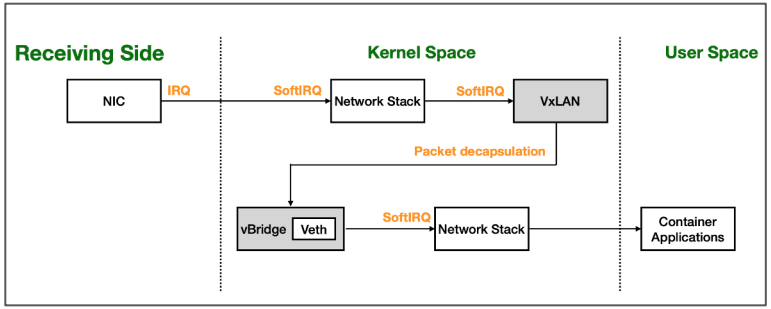
- Overlay packet processing in OS is not optimized to utilize multicore system.
- FALCON accelerates the performance of overlay networks...
  - Without losing any security, generality, or compatibility.
- Purely software-based solution that is immediately deployable.
- Our implementation is open-sourced at: github.com/munikarmanish/falcon

Thank you!

Backup Slides

Why is overlay network slow?

- Additional virtual devices
  - Prolonged packet processing path
- Serialization of single-flow processing
  - Quick saturation
  - Longer queueing delays
  - Load imbalance



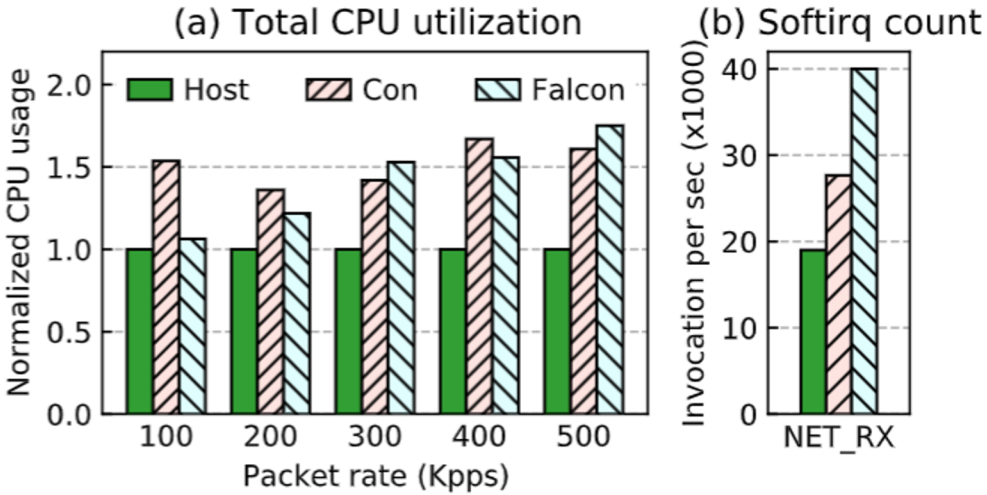
<show the packet path from NIC to Application>

How do containers communicate?

- Host mode (containers use host network namespace)
  - ✓ Simple and fast
  - ✗ No network isolation/flexibility
- Macvlan mode (assign multiple MAC to same physical NIC)
  - ✓ Fast and some isolation/flexibility
  - ✗ Complex host network routing management
- Overlay mode (packet encapsulation)
  - ✓ Full network isolation and flexibility
  - ✗ High overheads / slow

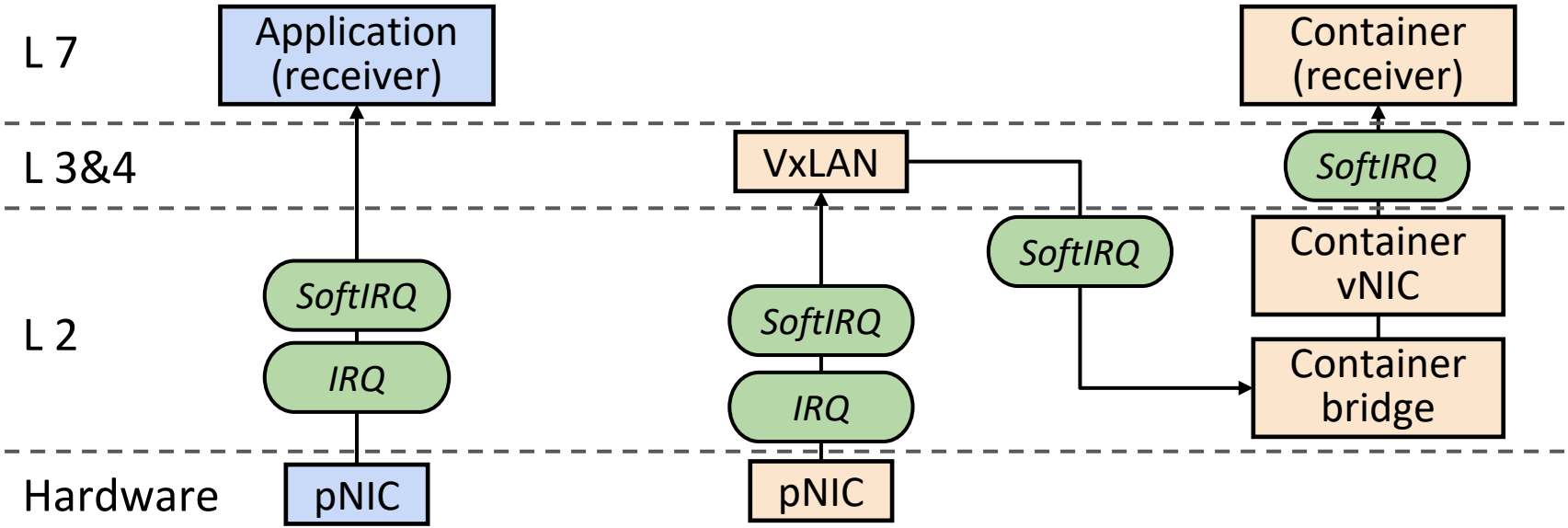
Overheads and Limitations

- Higher overall CPU utilization
  - More softirq invocations
- Loss of data locality



Existing optimizations in Linux

- IRQ coalescing
- GRO/GSO
- RSS/RPS



Evaluation - Setup

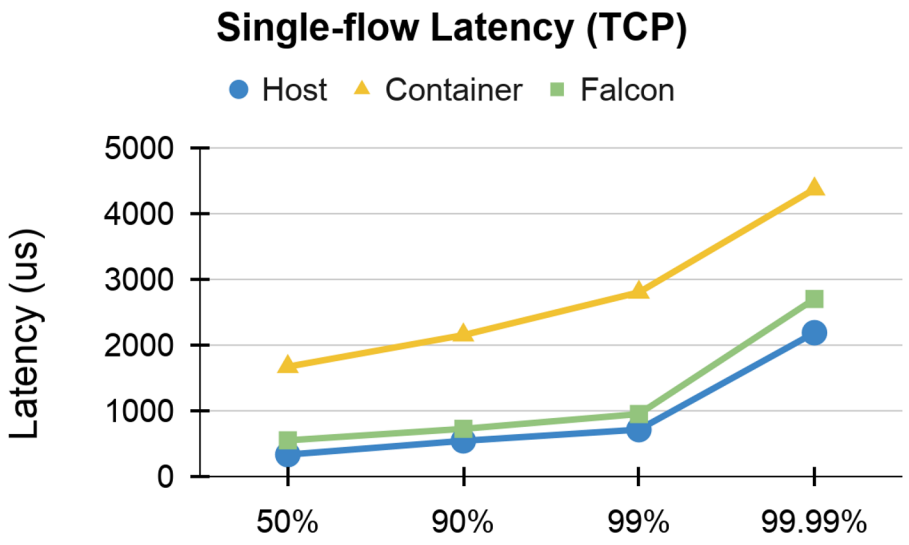
Hardware: Intel Xeon Silver 4114 with 40 logical cores @ 2.2GHz, 128 GB RAM
NIC: Mellanox ConnectX-5 EN (100 Gbps)
Software: Ubuntu 18.04 with Linux kernel v5.4

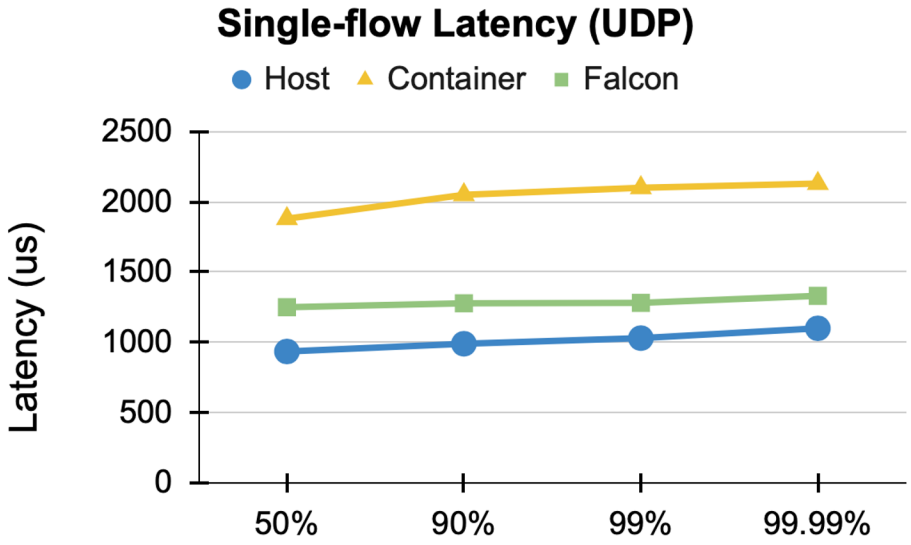
Performance comparisons: Host vs. Container vs. FALCON

Experiments:

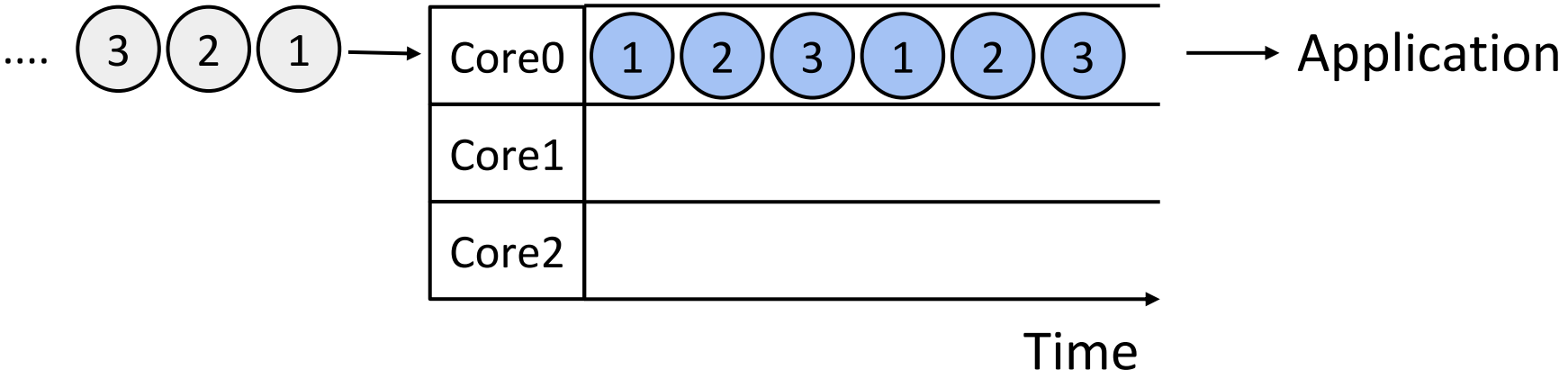
- Single-flow and multi-flow microbenchmarks
- Cloudsuite benchmark (memcached & nginx)
- Load balancing

Single-flow performance

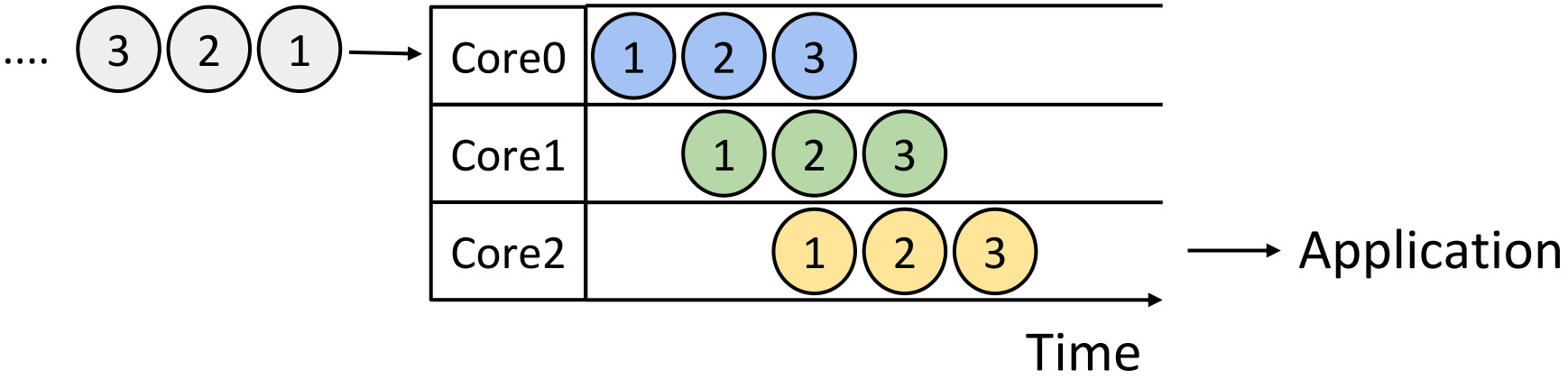




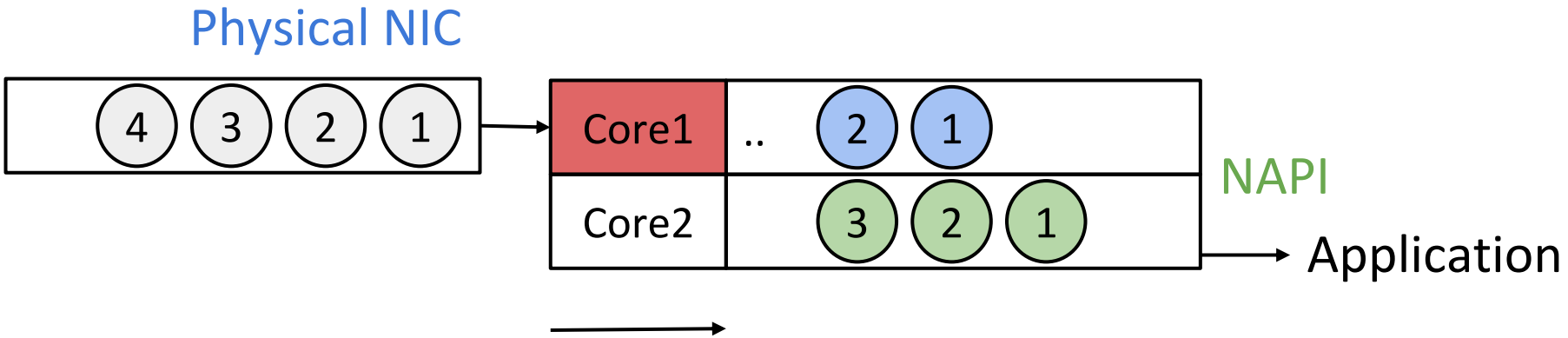
FALCON Design - SoftIRQ Pipelining



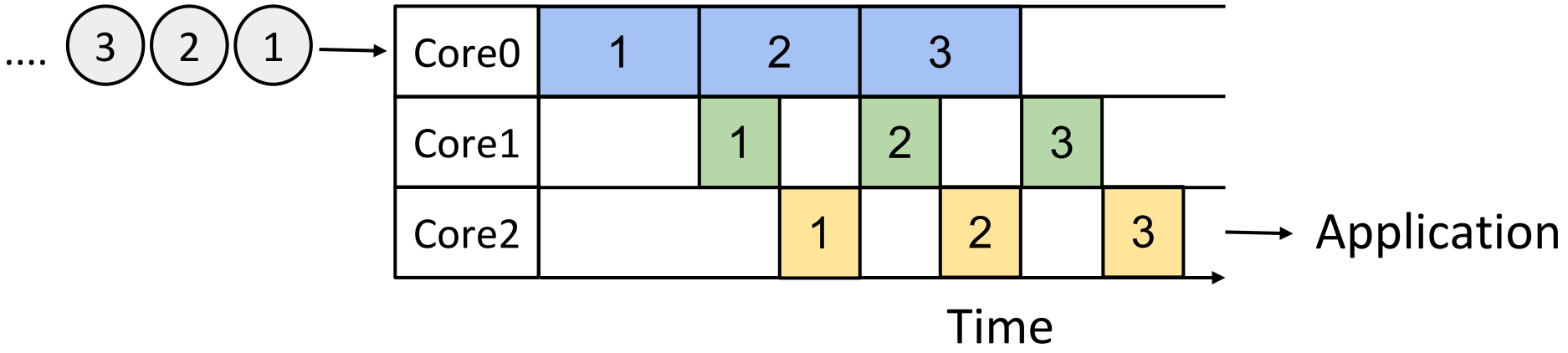
FALCON Design - SoftIRQ Pipelining



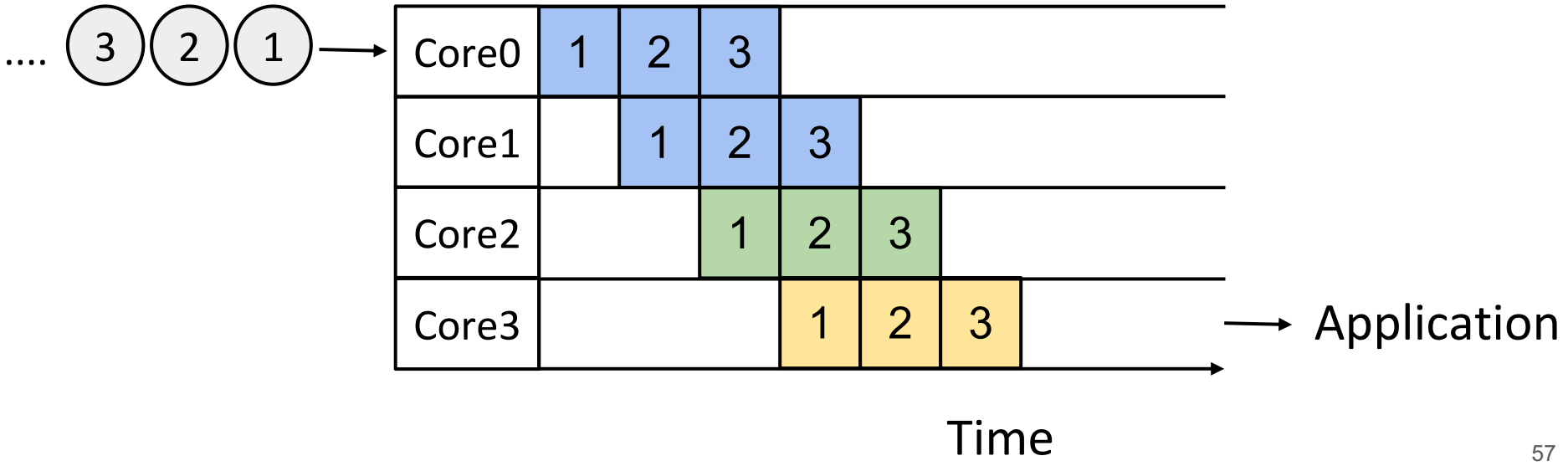
FALCON Design - SoftIRQ Splitting



FALCON Design - SoftIRQ Splitting



FALCON Design - SoftIRQ Splitting



Load Balancing