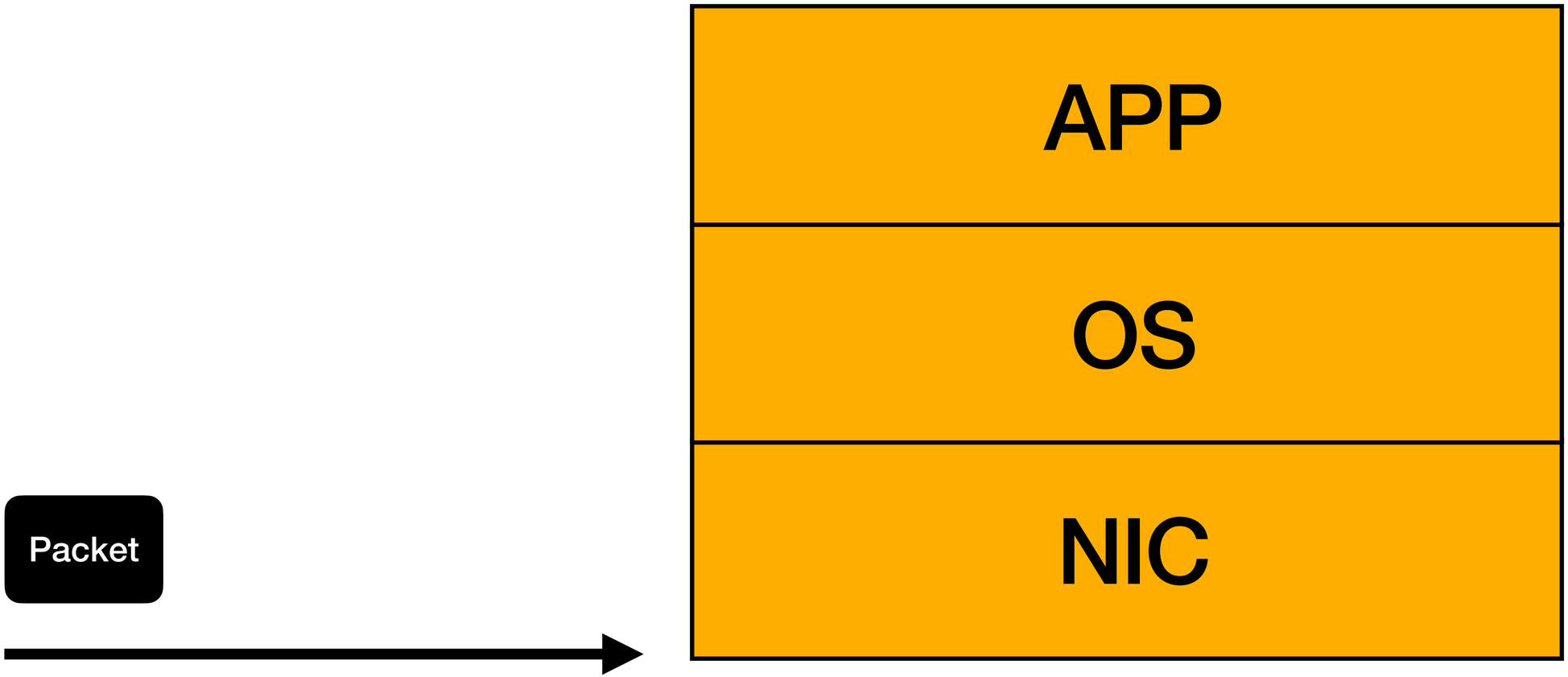
A generic ML-enhanced controller that optimizes network application energy efficiency

Han Dong, Sanjay Arora, Jonathan Appavoo

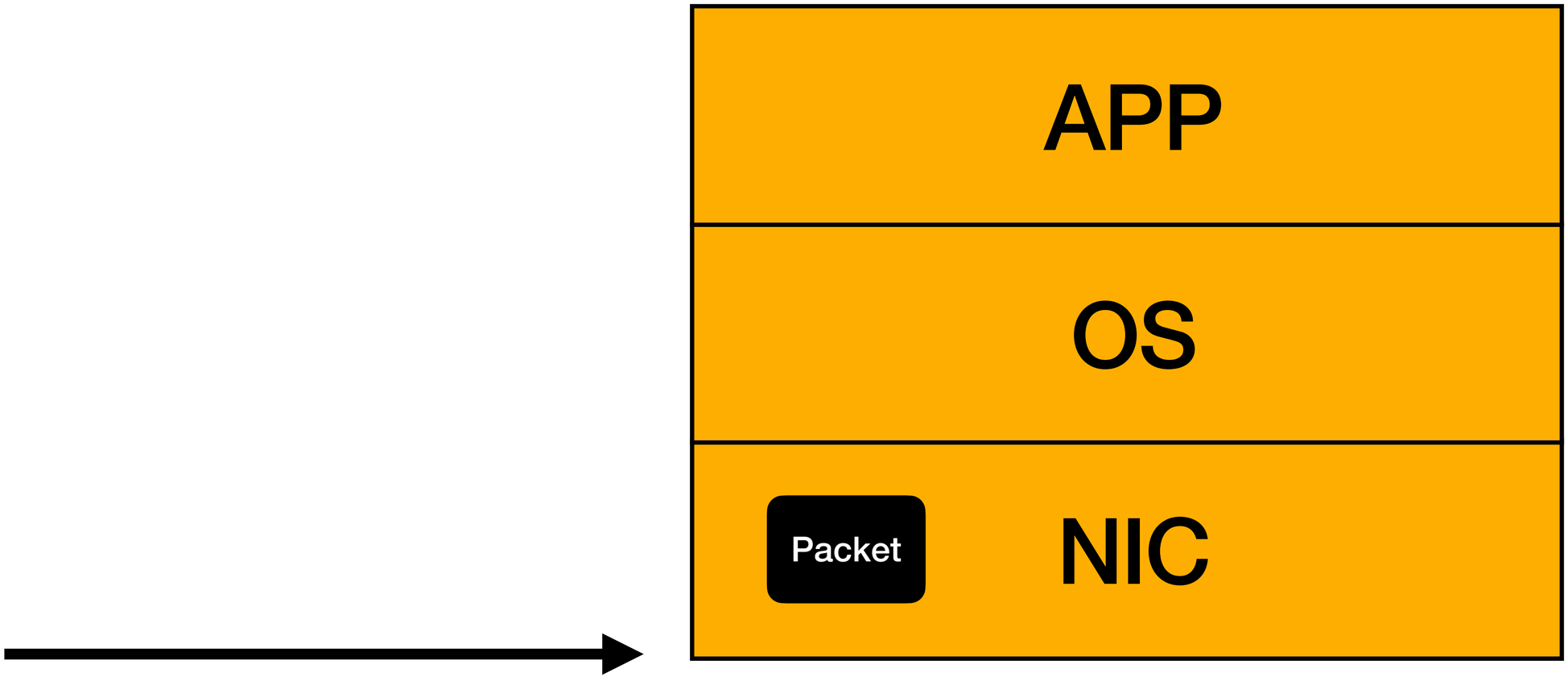
handong@bu.edu

BOSTON UNIVERSITY

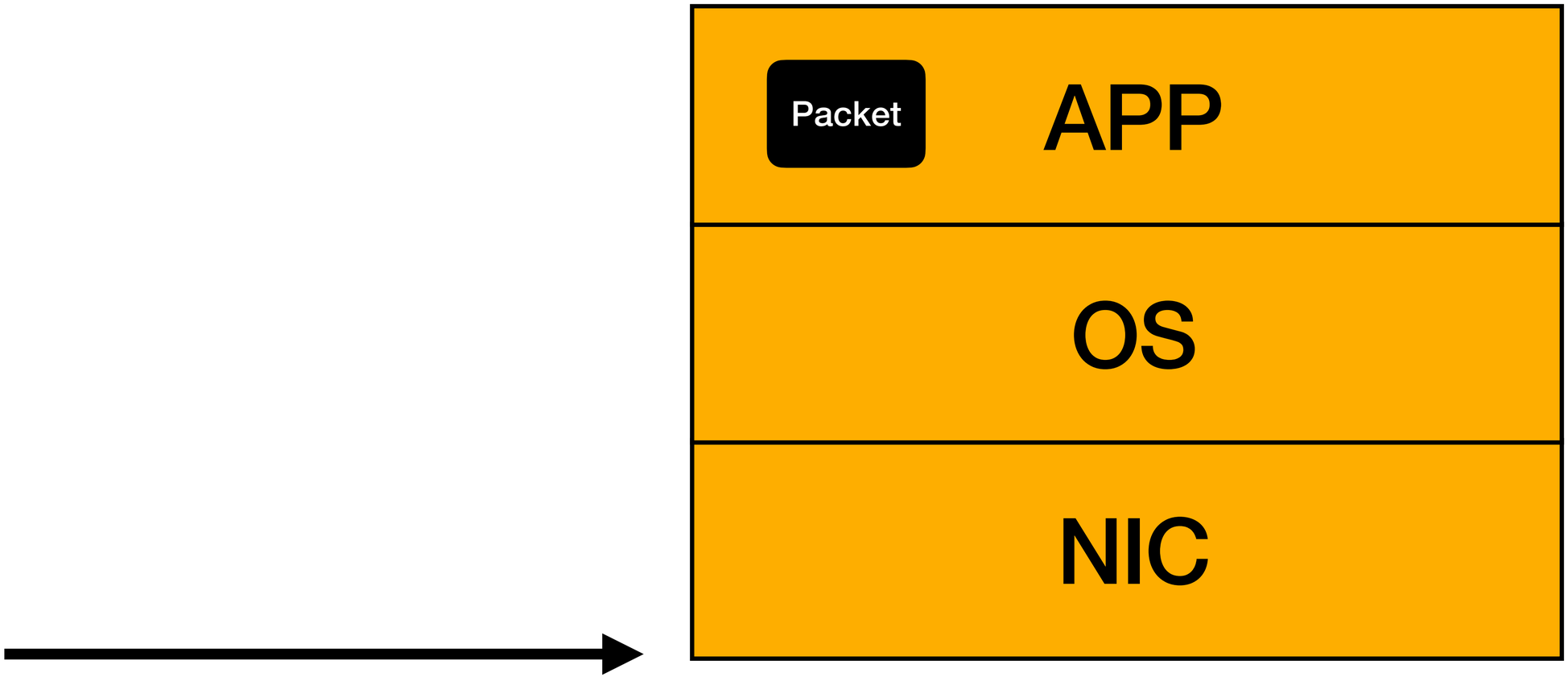
A simple network processing example



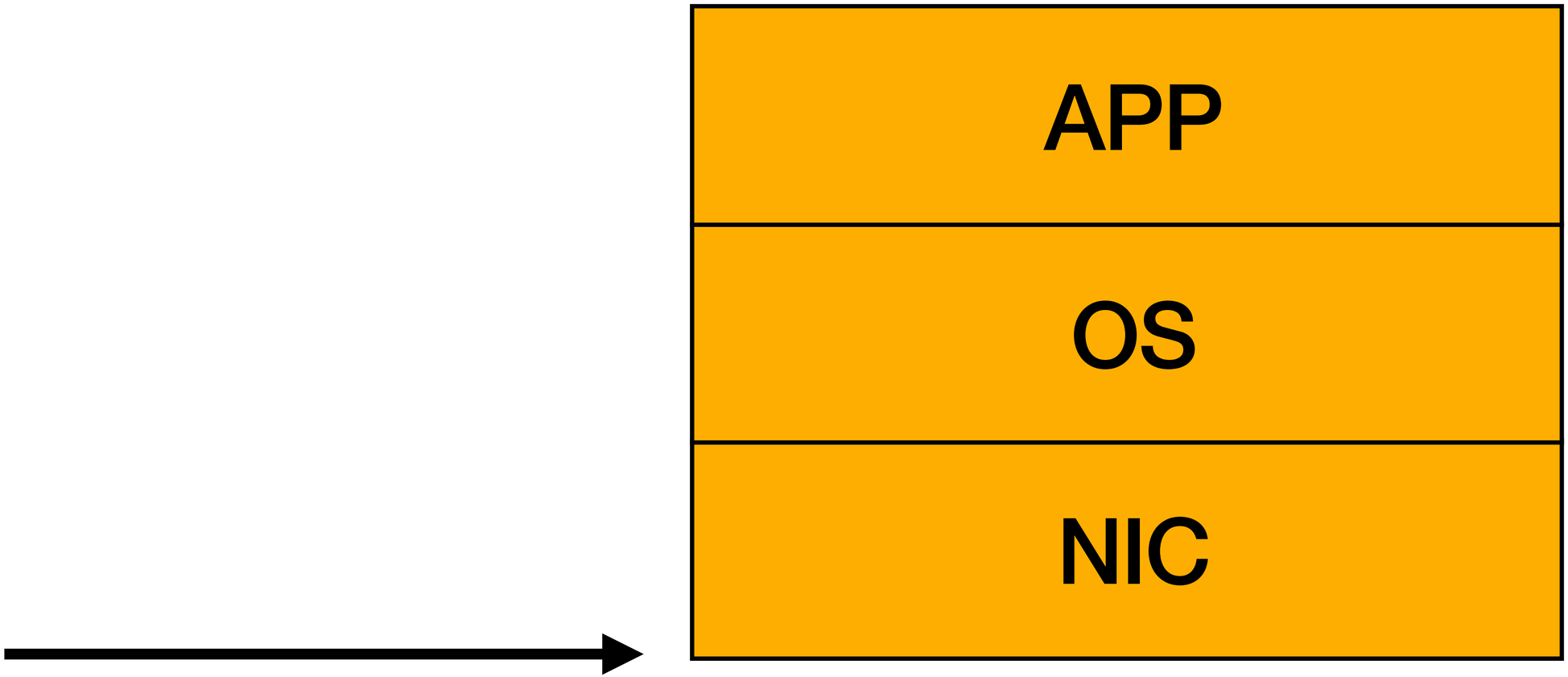
A simple network processing example



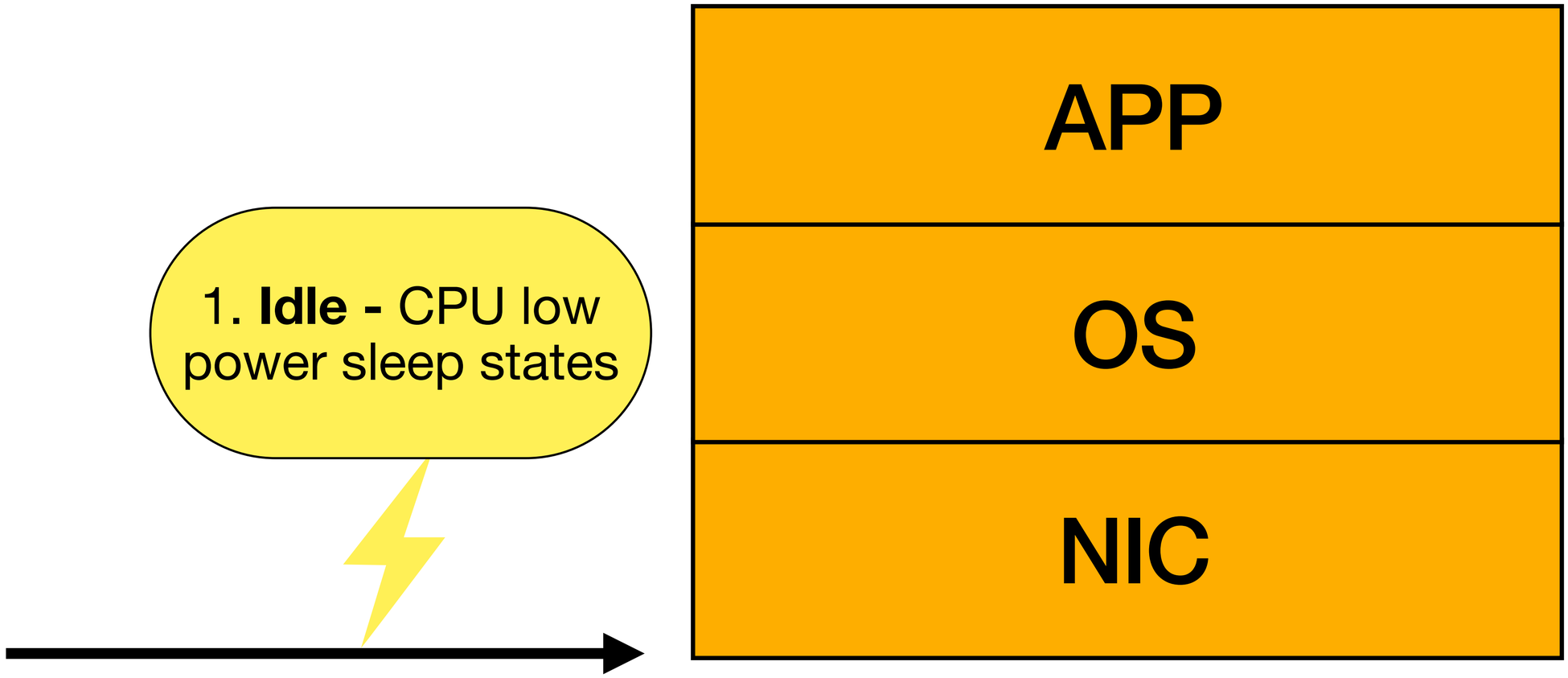
A simple network processing example



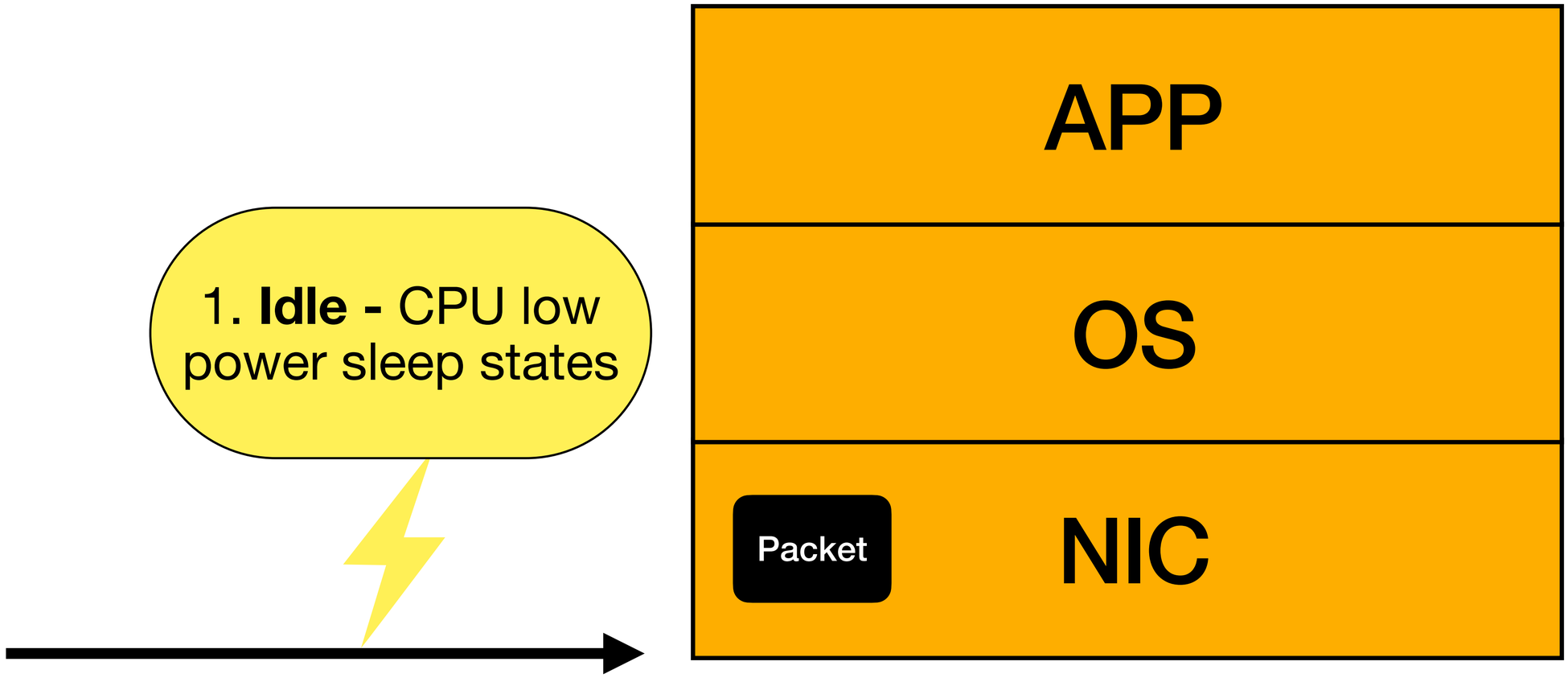
Where does energy consumption happen?



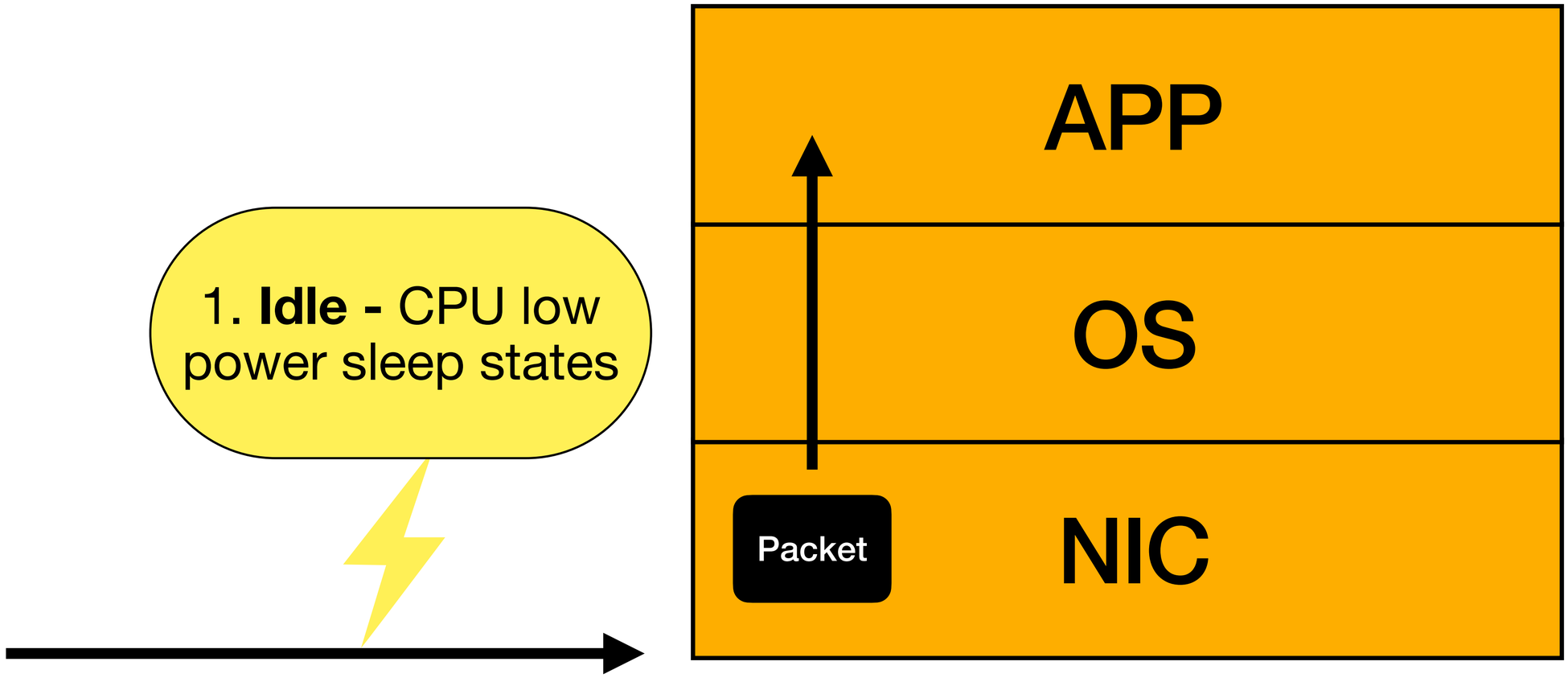
Where does energy consumption happen?



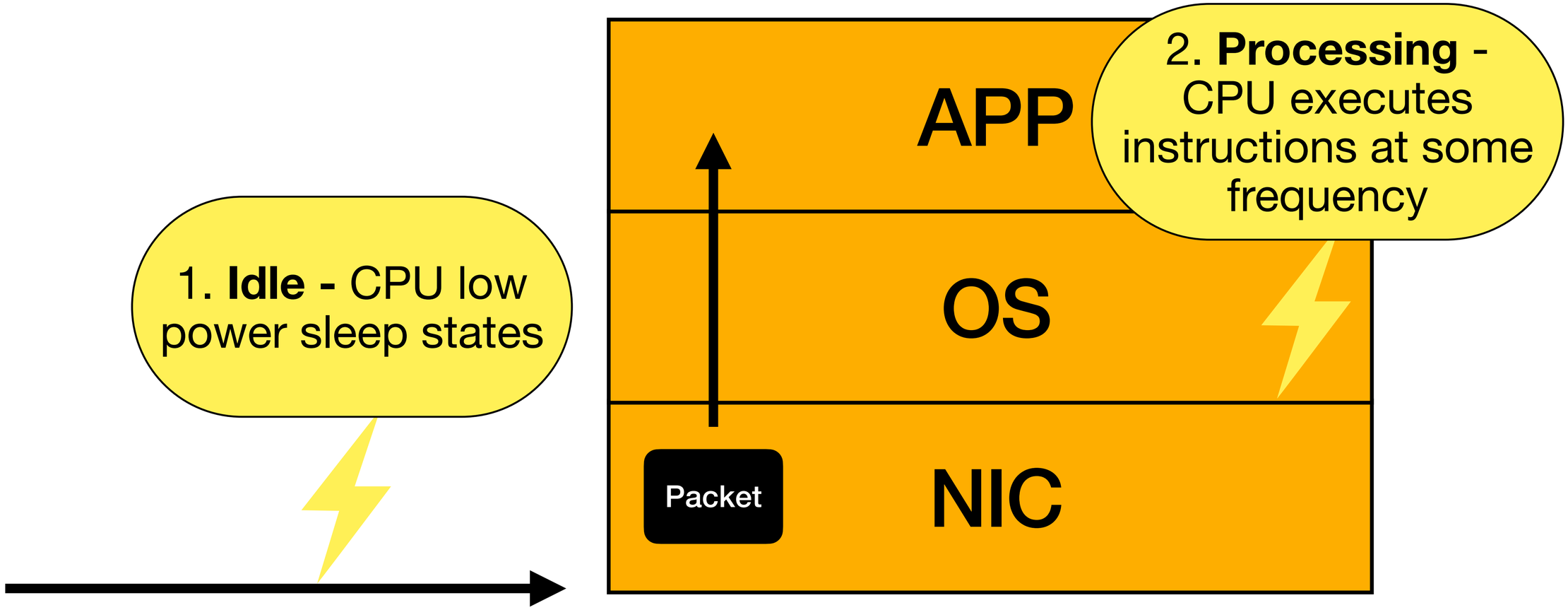
Where does energy consumption happen?



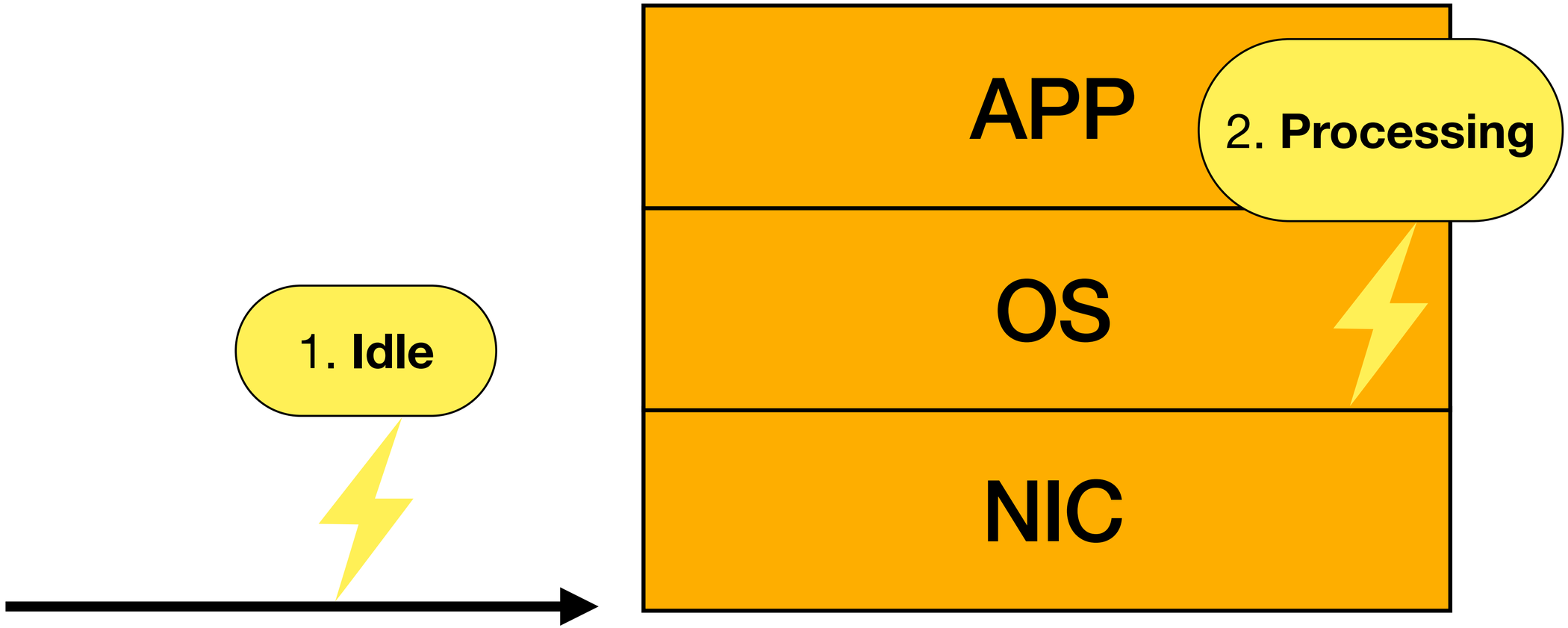
Where does energy consumption happen?



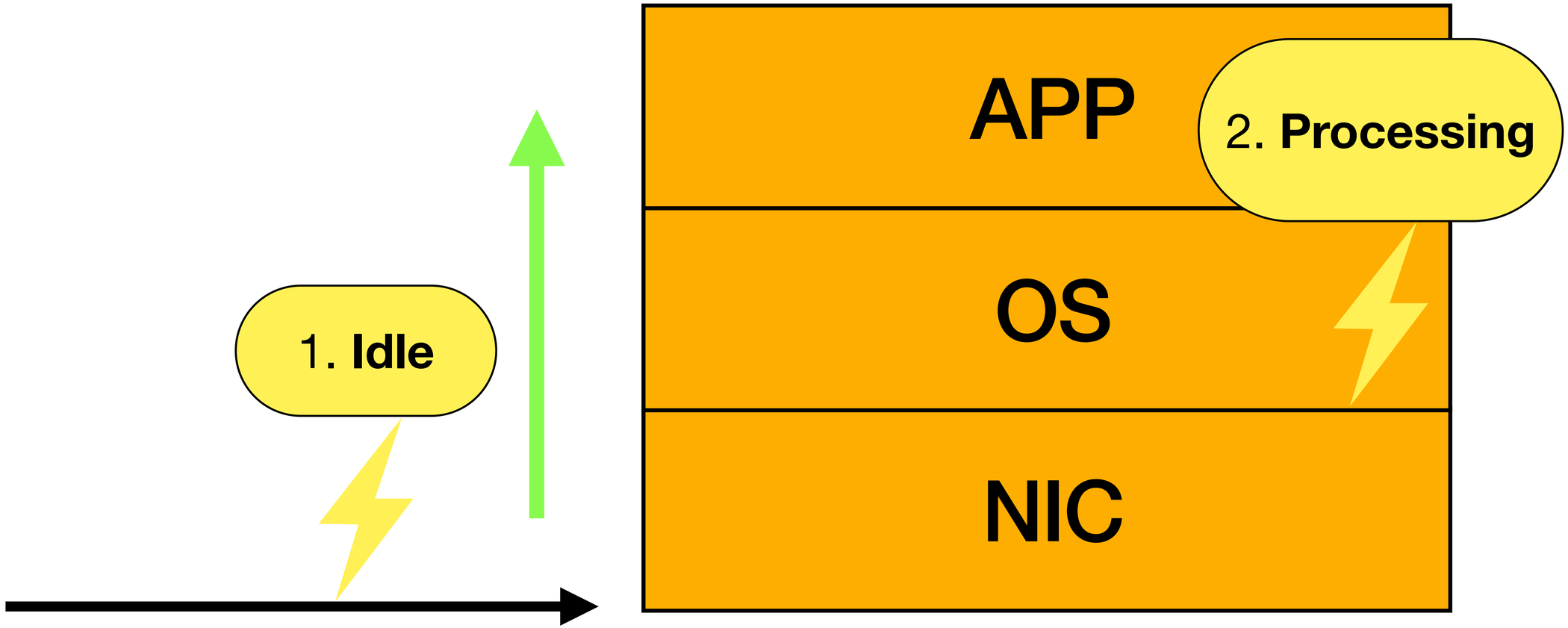
Where does energy consumption happen?



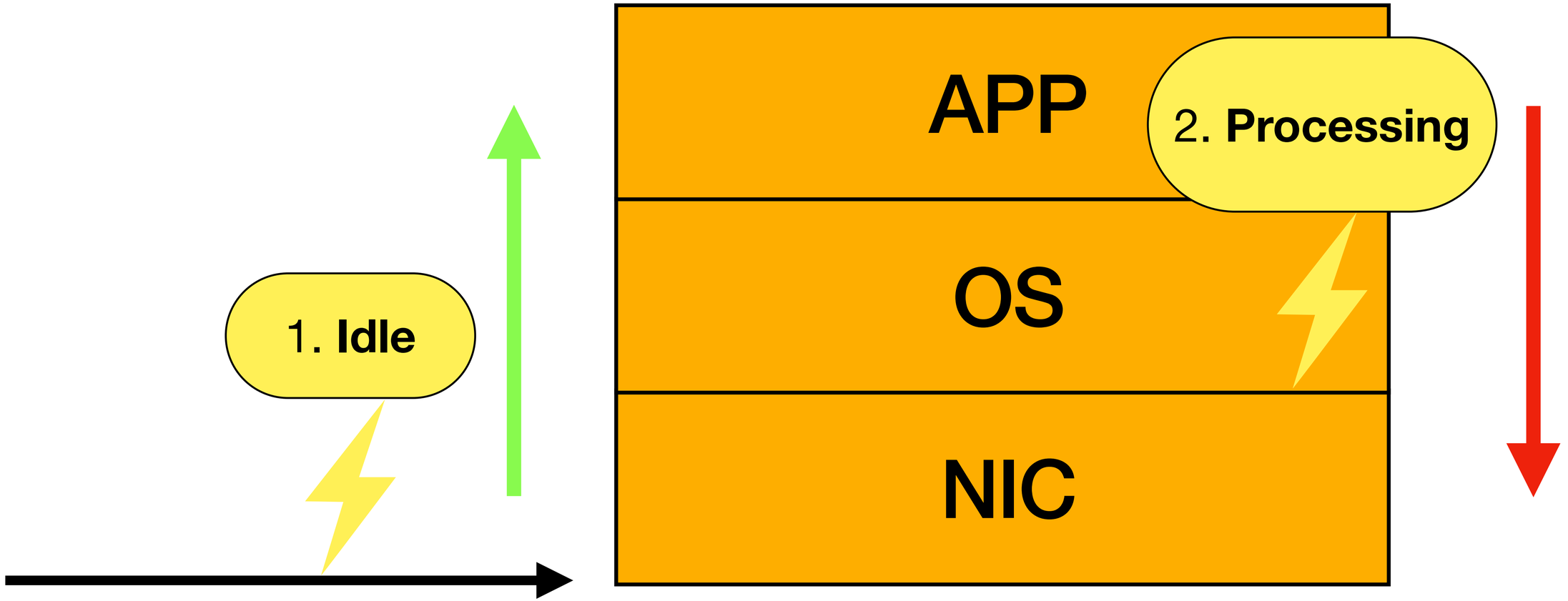
Can we reduce energy consumption while maintaining performance objectives?



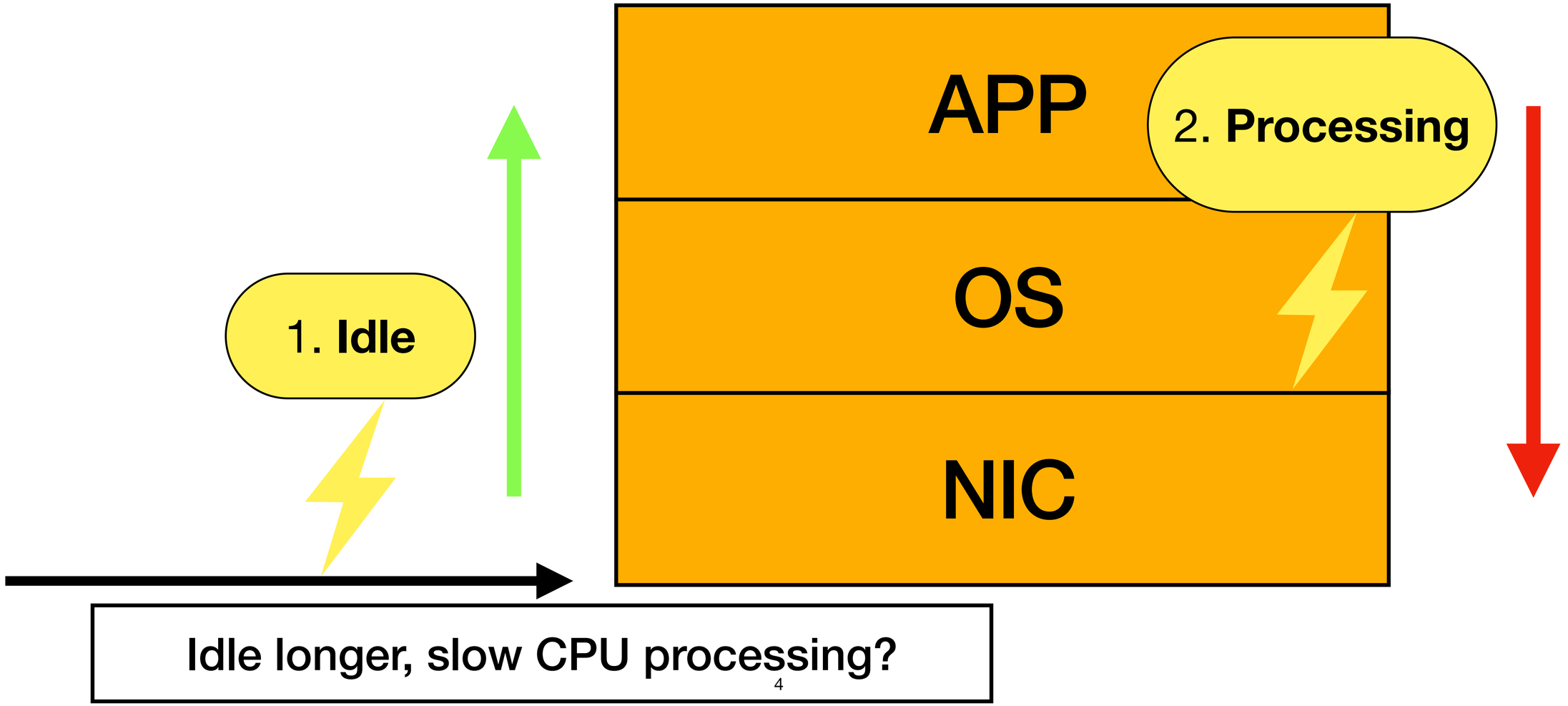
Can we reduce energy consumption while maintaining performance objectives?



Can we reduce energy consumption while maintaining performance objectives?

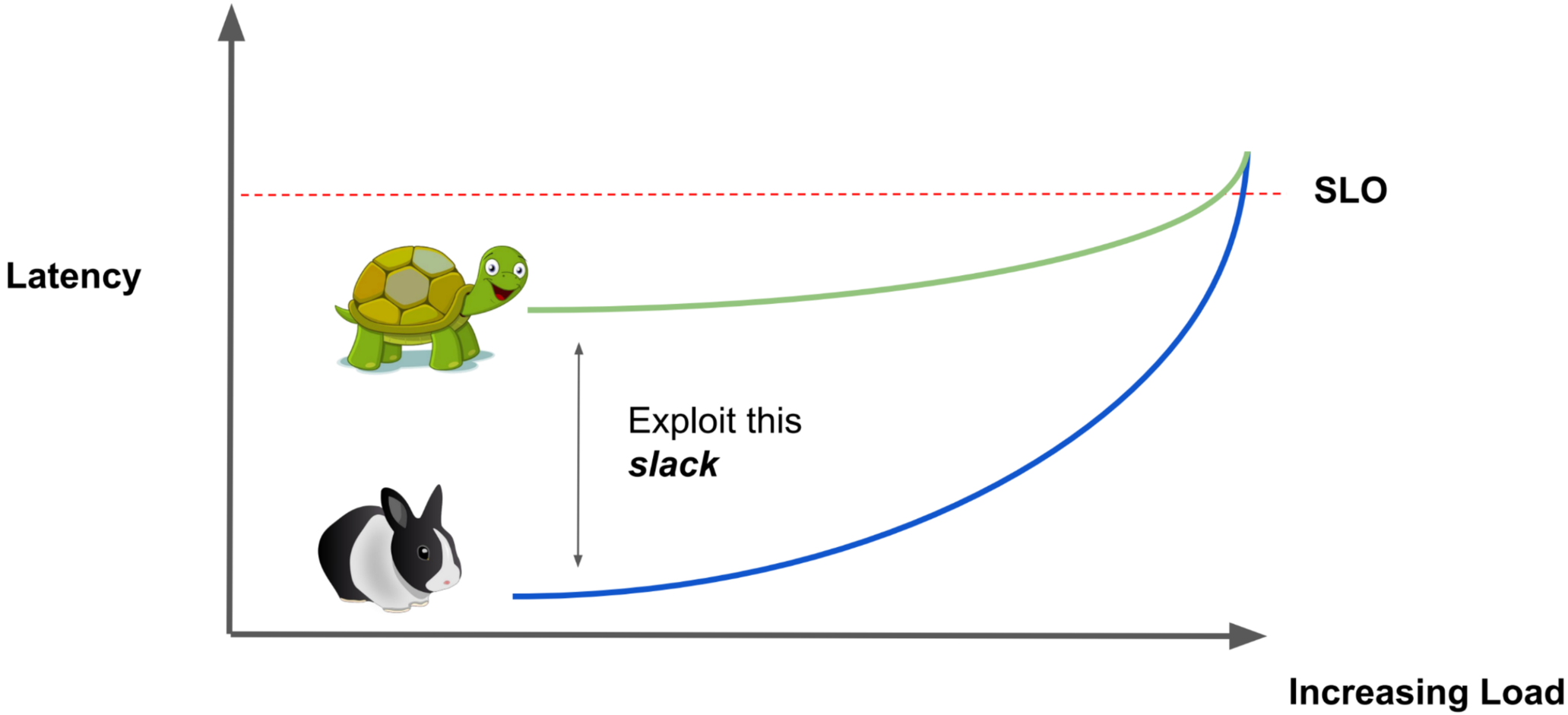


Can we reduce energy consumption while maintaining performance objectives?



Web services with service-level objectives (SLO) i.e. 99% tail latency < 1 ms

Web services with service-level objectives (SLO) i.e. 99% tail latency < 1 ms

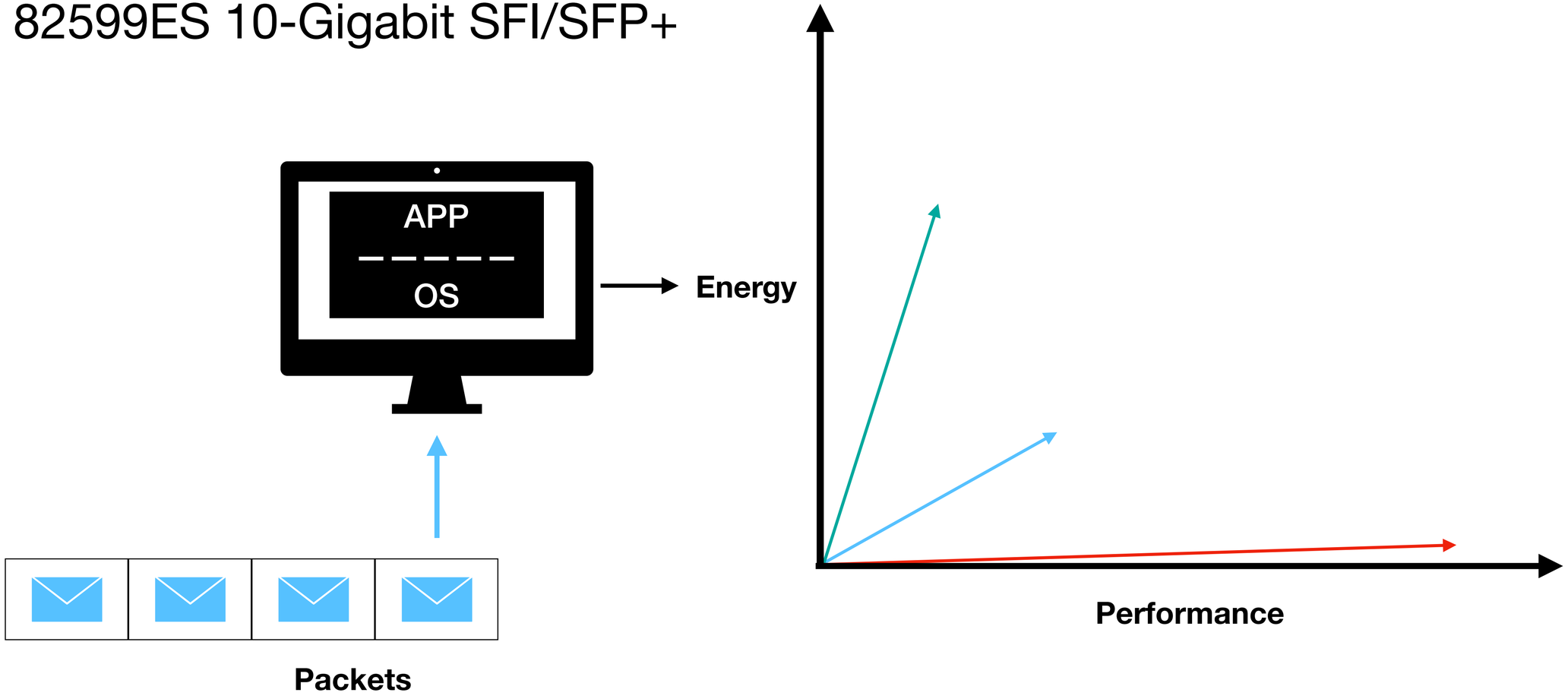


Our work: tuning of two hardware knobs 1) **interrupt coalescing (ITR)** and 2) **processor speeds (DVFS)** to reduce energy while maintaining SLO across diverse sets of hardware/software

1. Performance and Energy Study

Defining the measurement problem

- Intel Xeon E5-2690 @ 2.90 GHz
- Intel 82599ES 10-Gigabit SFI/SFP+ NIC

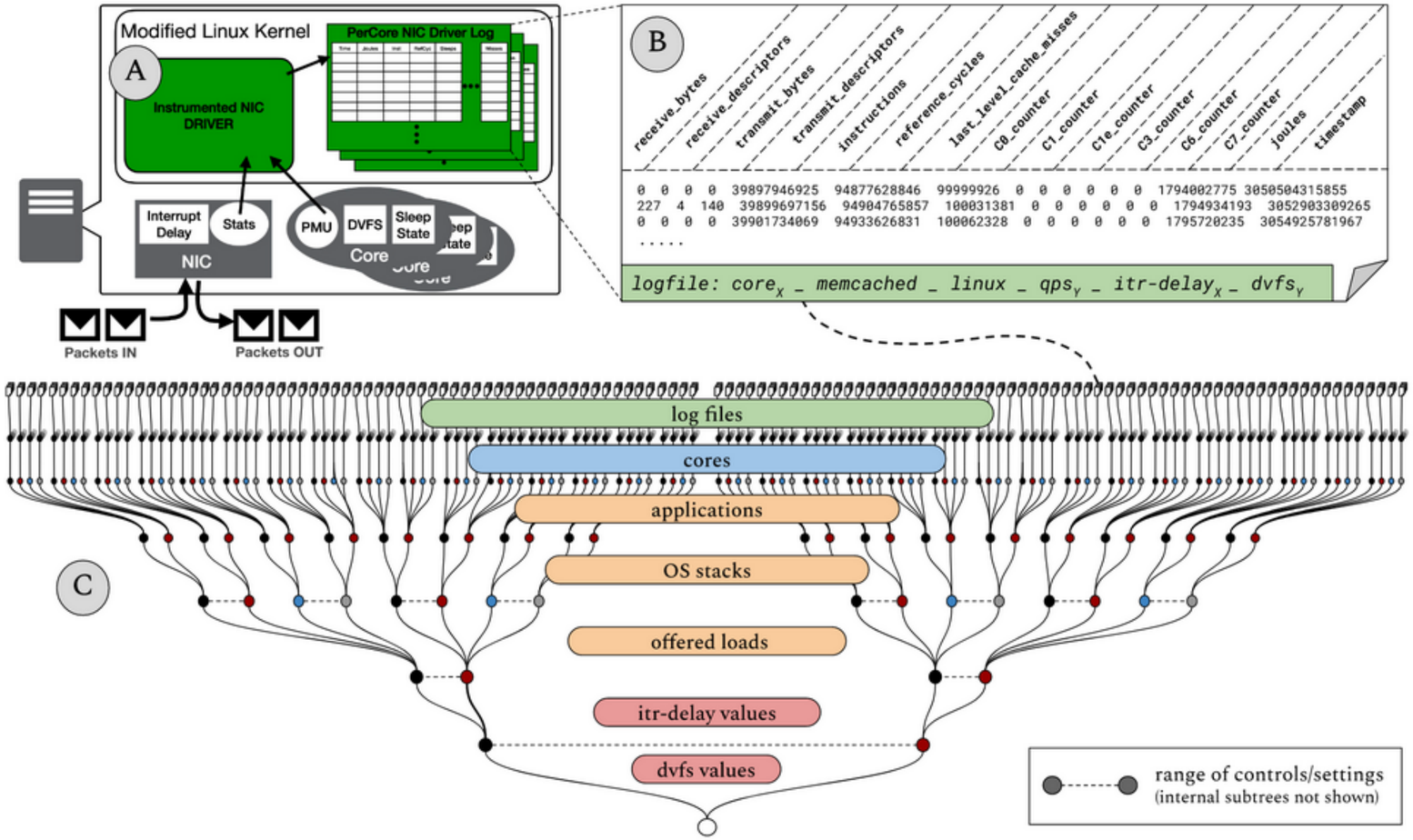


Data Collection Framework for Systems

A. Instrumented on network interrupt path

B. Per-core /procfs entry

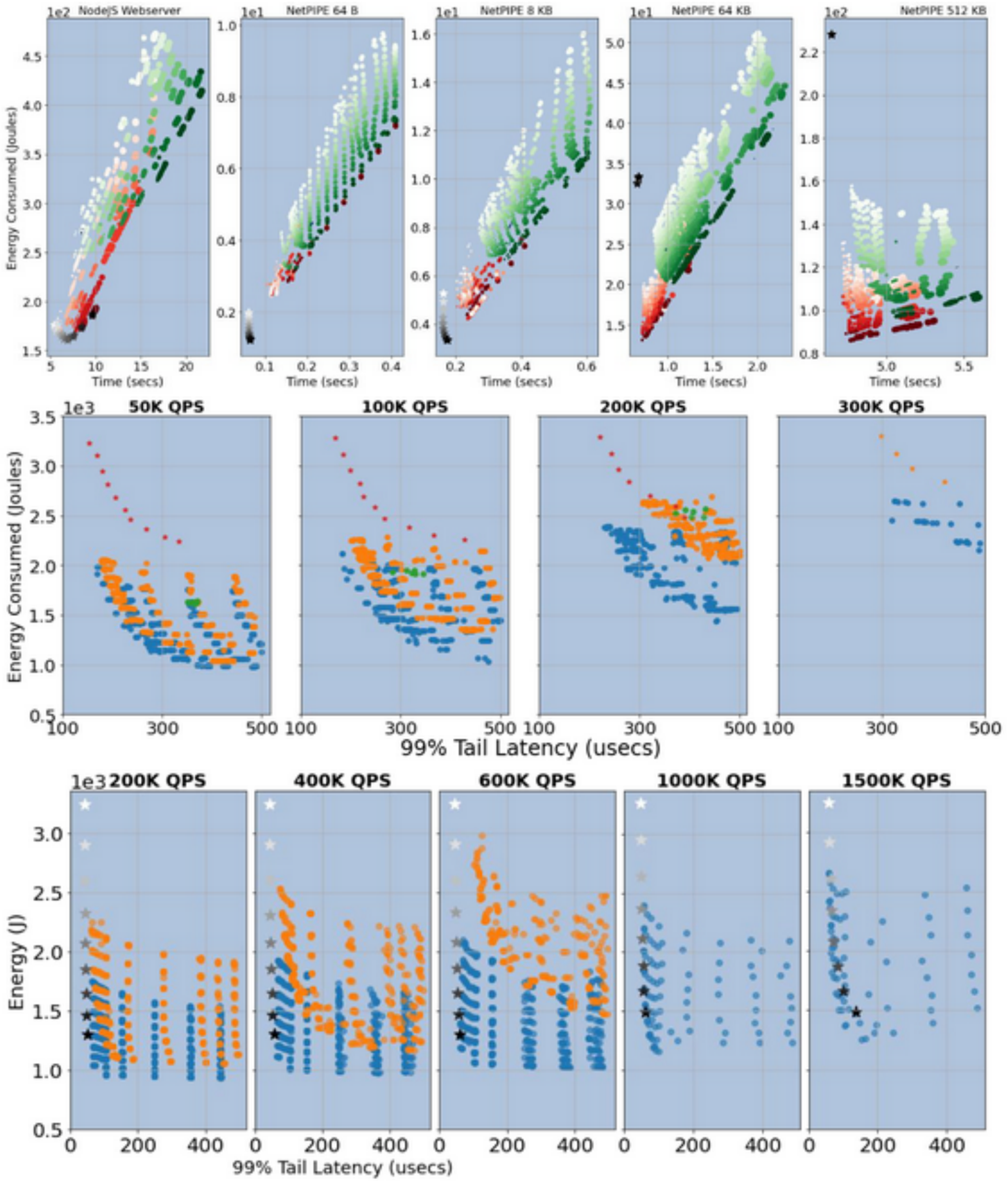
C. Agnostic across application/OS



Details: https://github.com/sesa/intlog

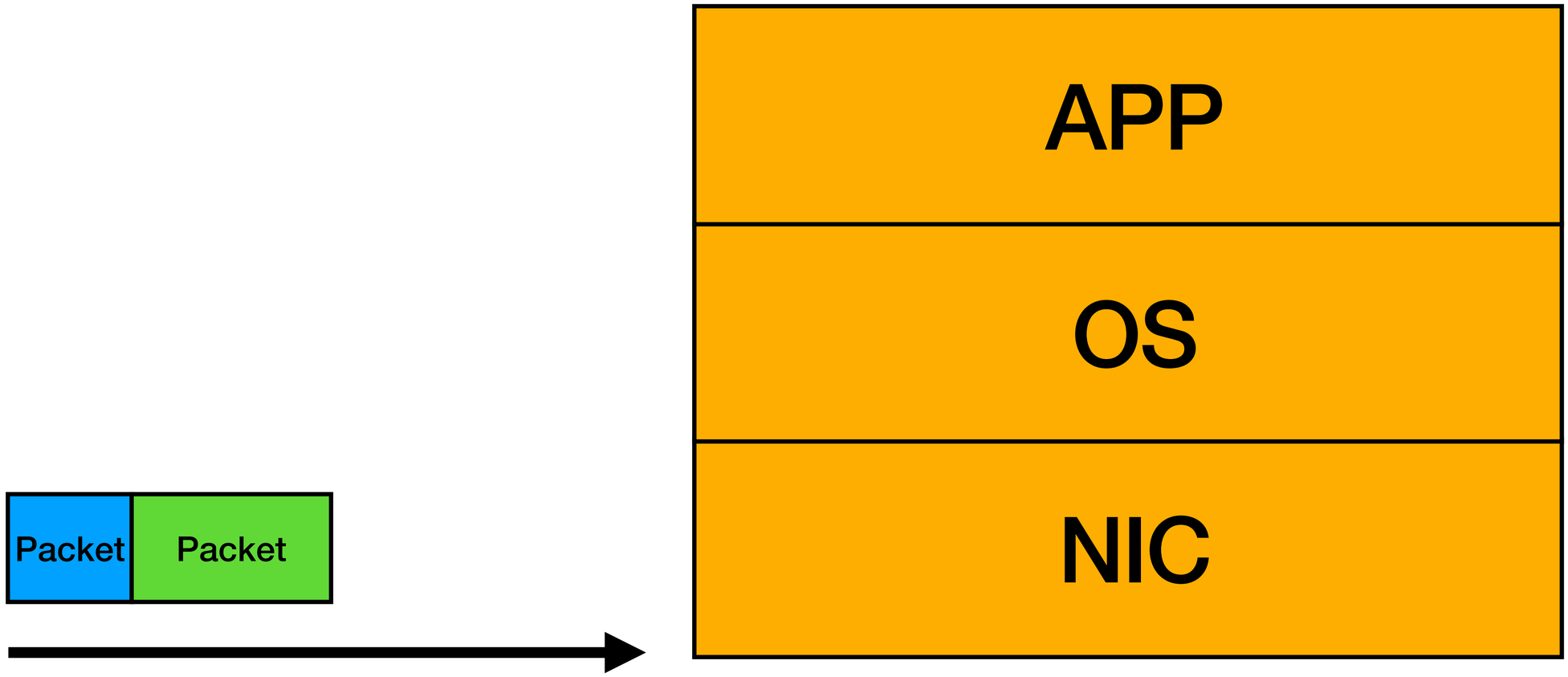
Data Collection Results

- Explored up to 340 unique ITR, DVFS combinations
- Repeated up to 10 times for experimental stability
- Collected TBs of systems logs data
- **Finding**: Linux can save up to 50% energy while meeting SLO of memcached.

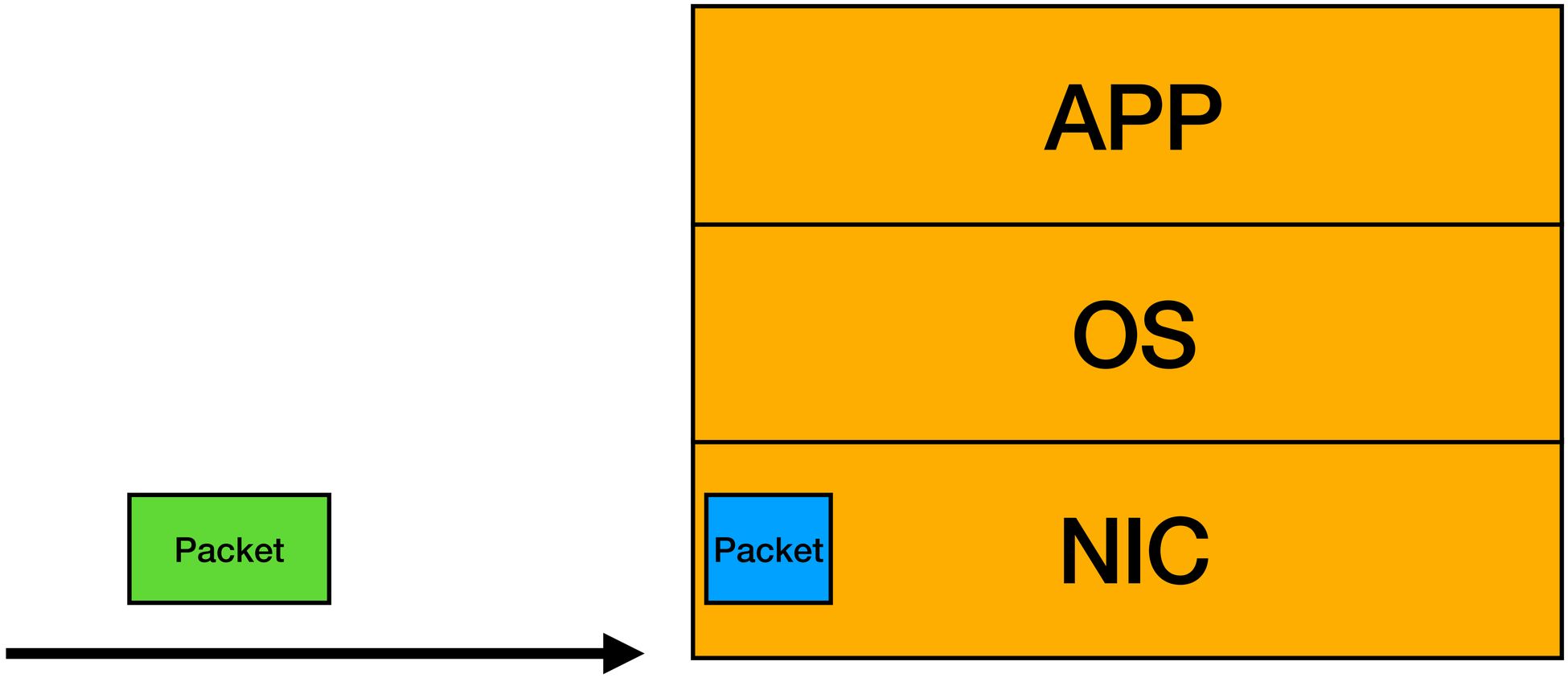


Details: https://arxiv.org/abs/2112.07010

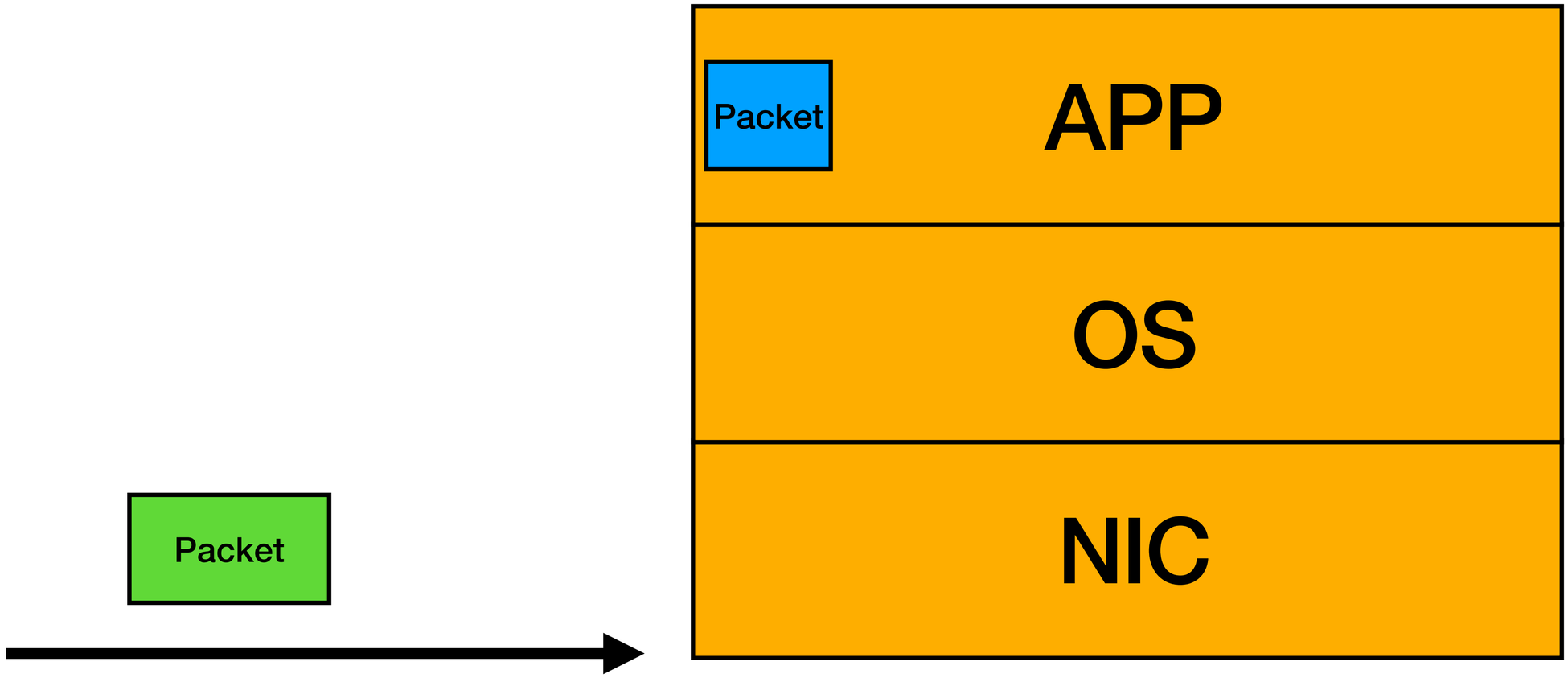
Insight



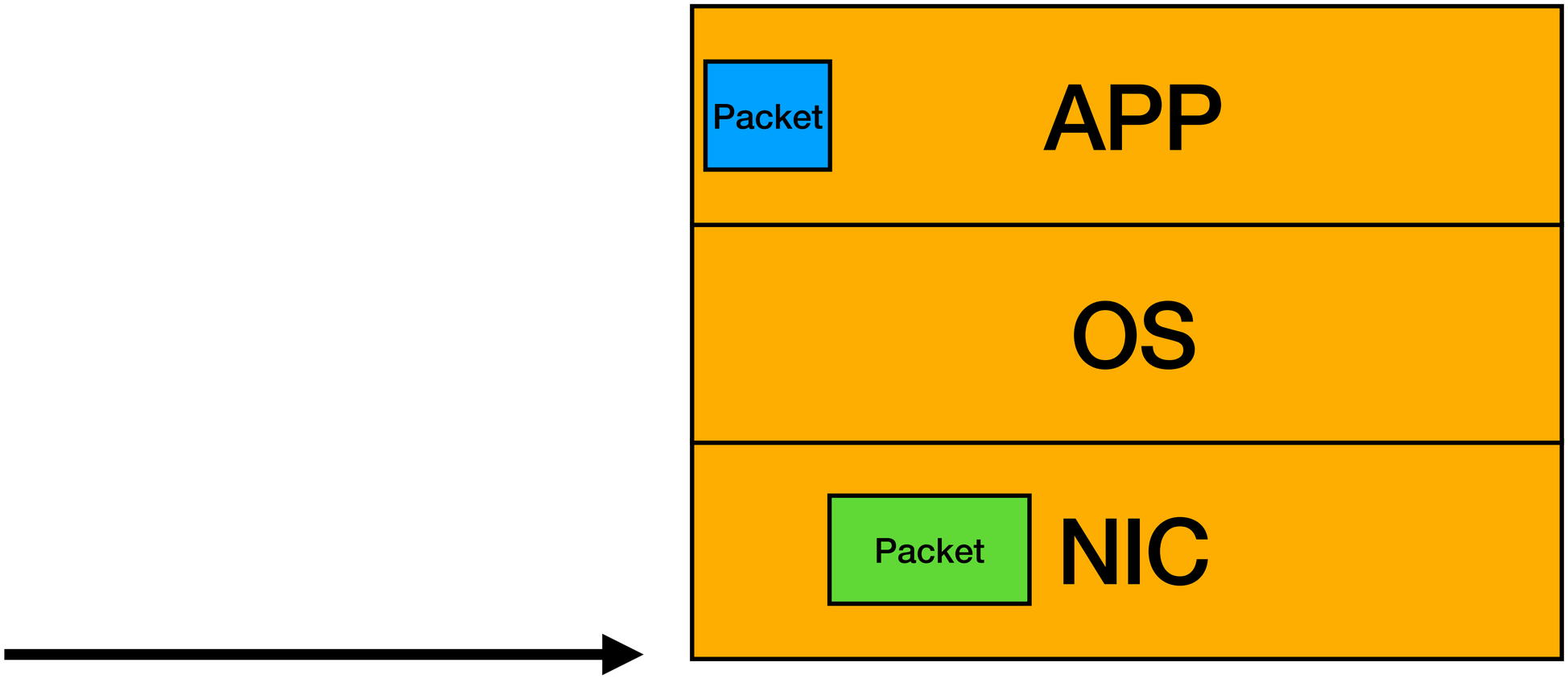
Insight



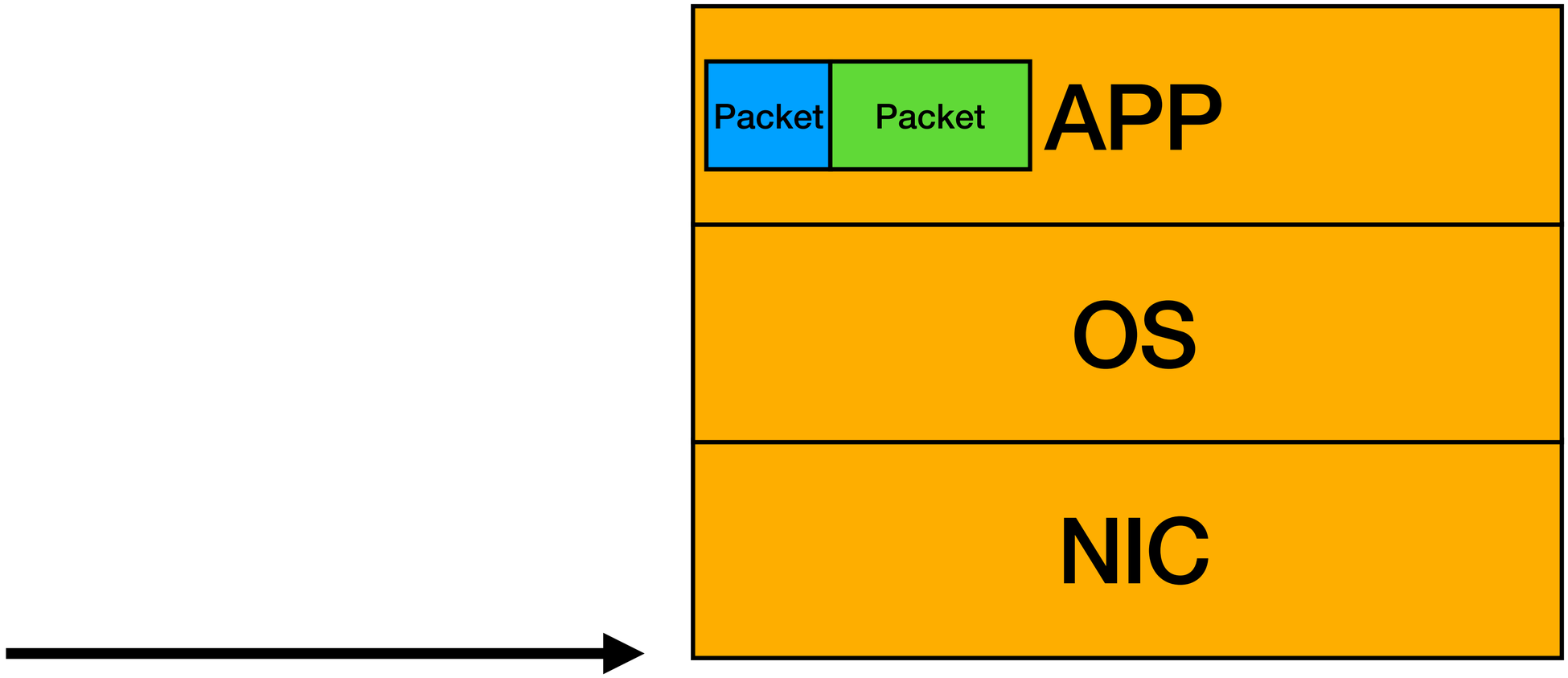
Insight



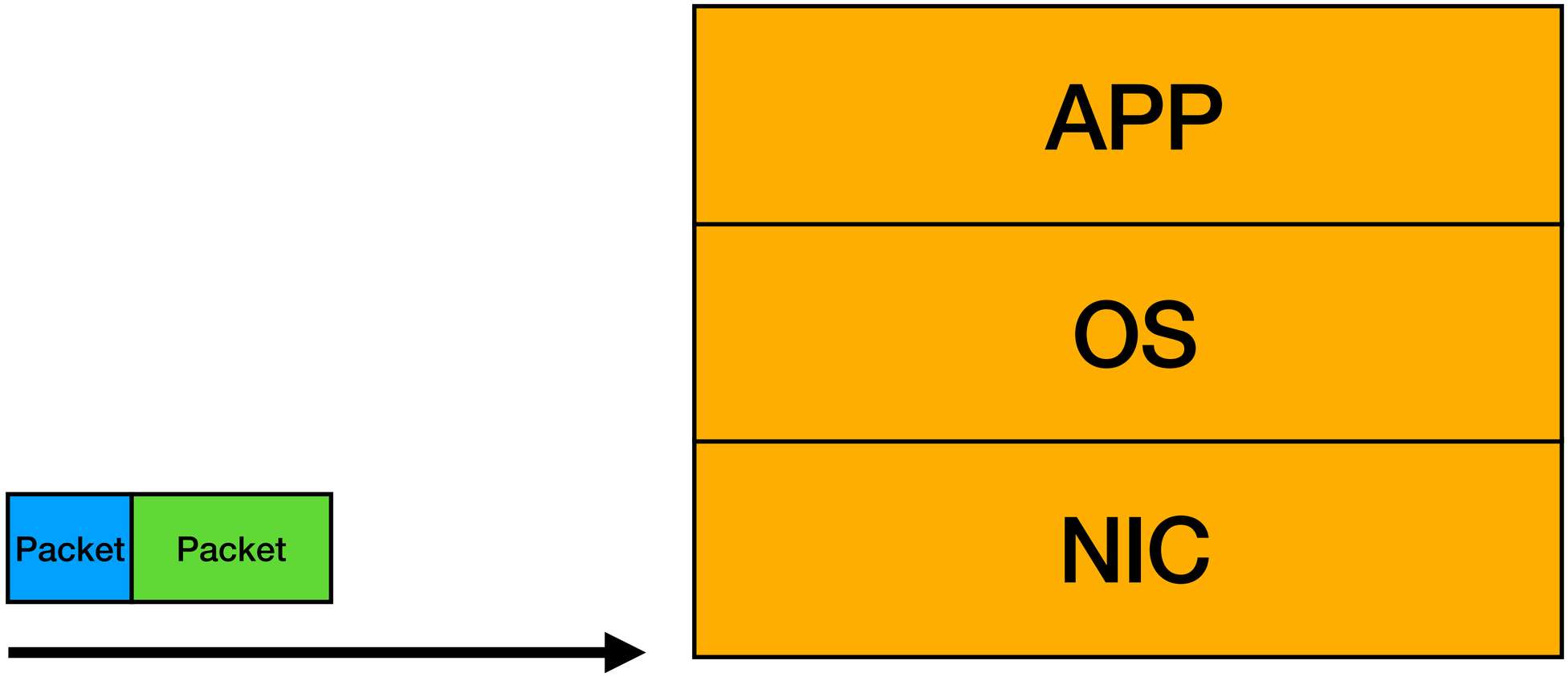
Insight



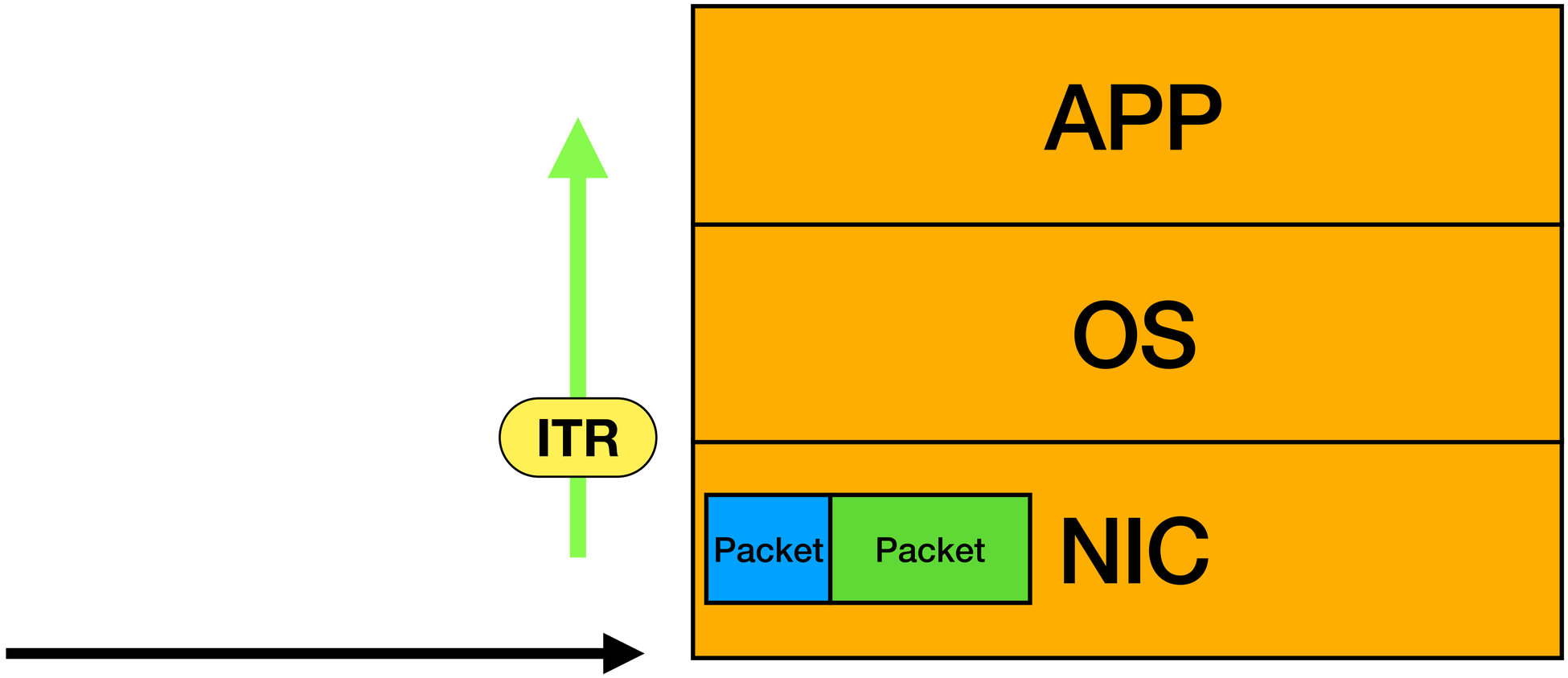
Insight



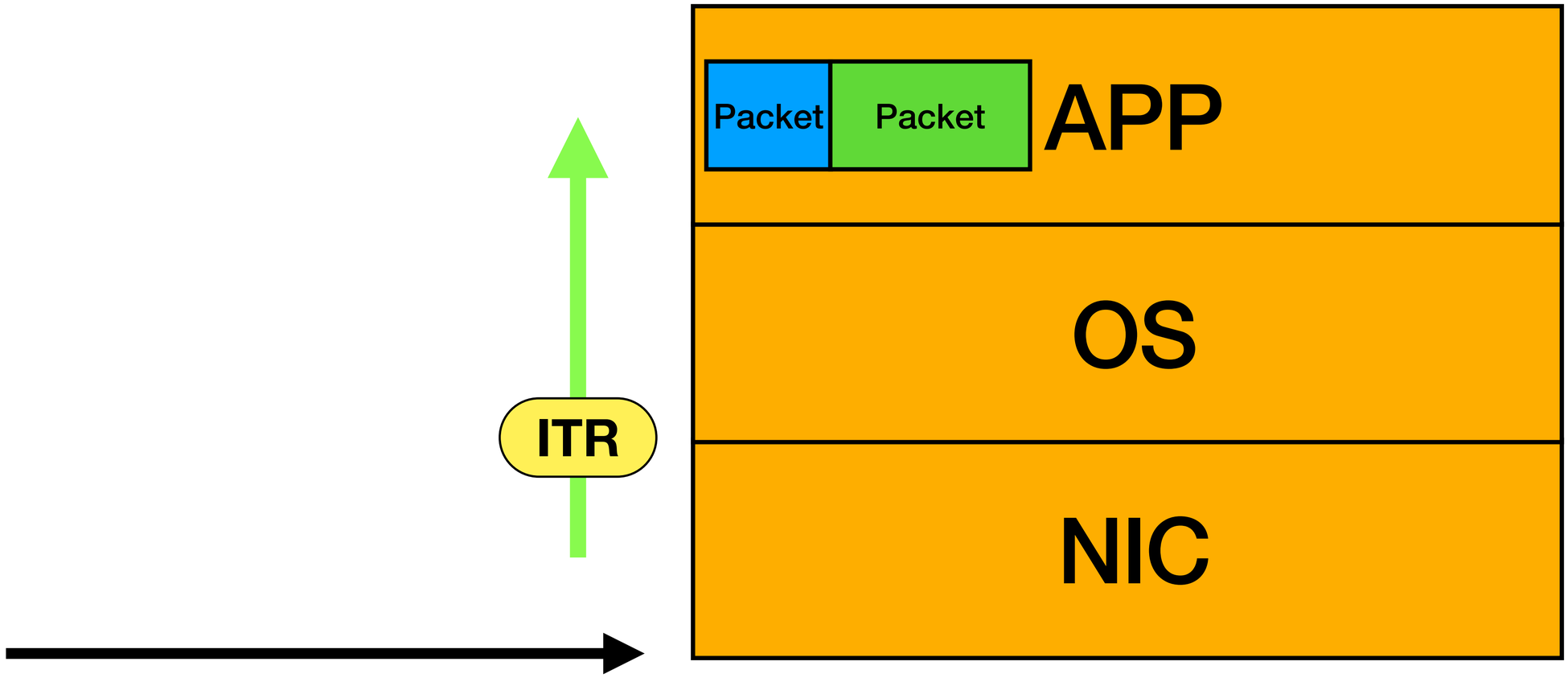
Insight



Insight

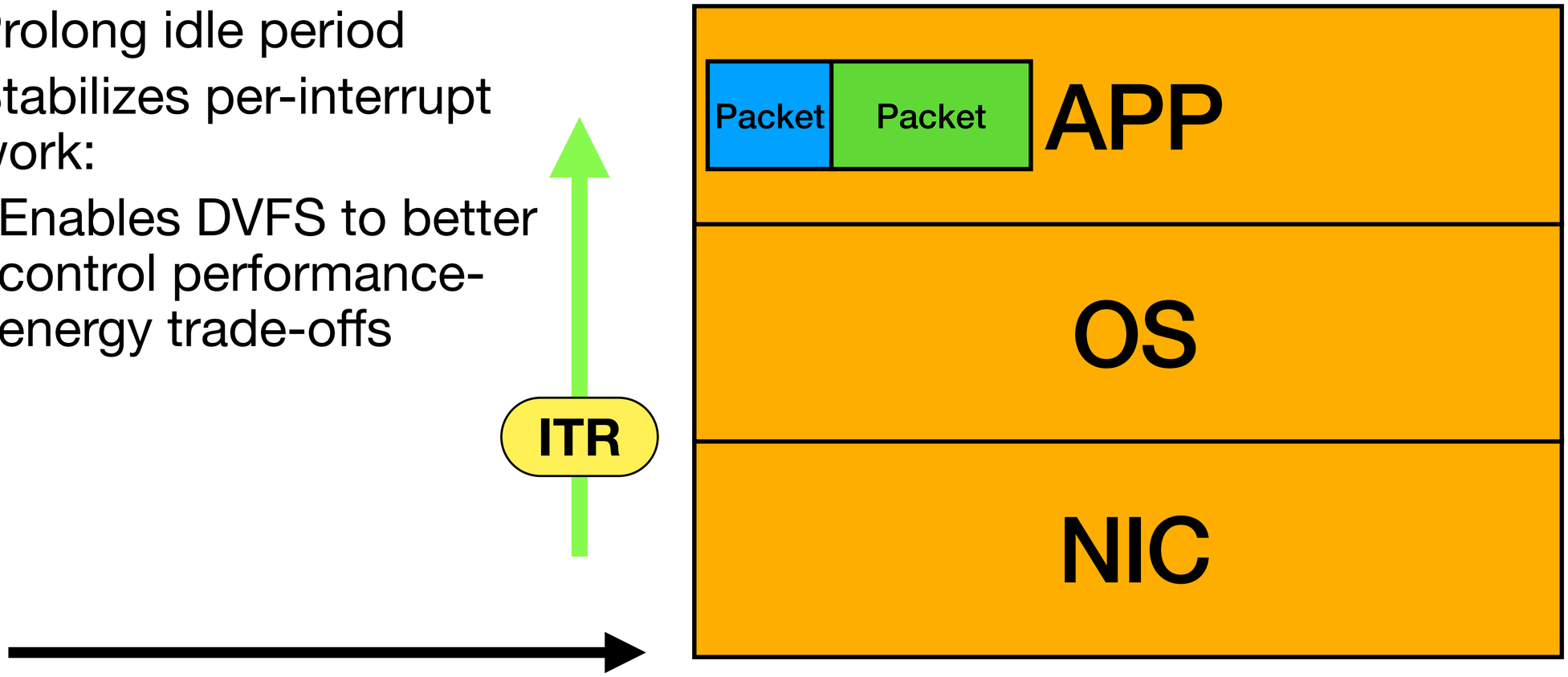


Insight



Insight

1. Reduces interrupt handling costs
2. Prolong idle period
3. Stabilizes per-interrupt work:
   - Enables DVFS to better control performance-energy trade-offs



2. Mathematical Modeling and Fitting

Motivation

- Study reveals **common and stable structure** in response to changes to ITR, DVFS

- Implication is that one can model performance and energy profiles in a formal way.

Modeling Result

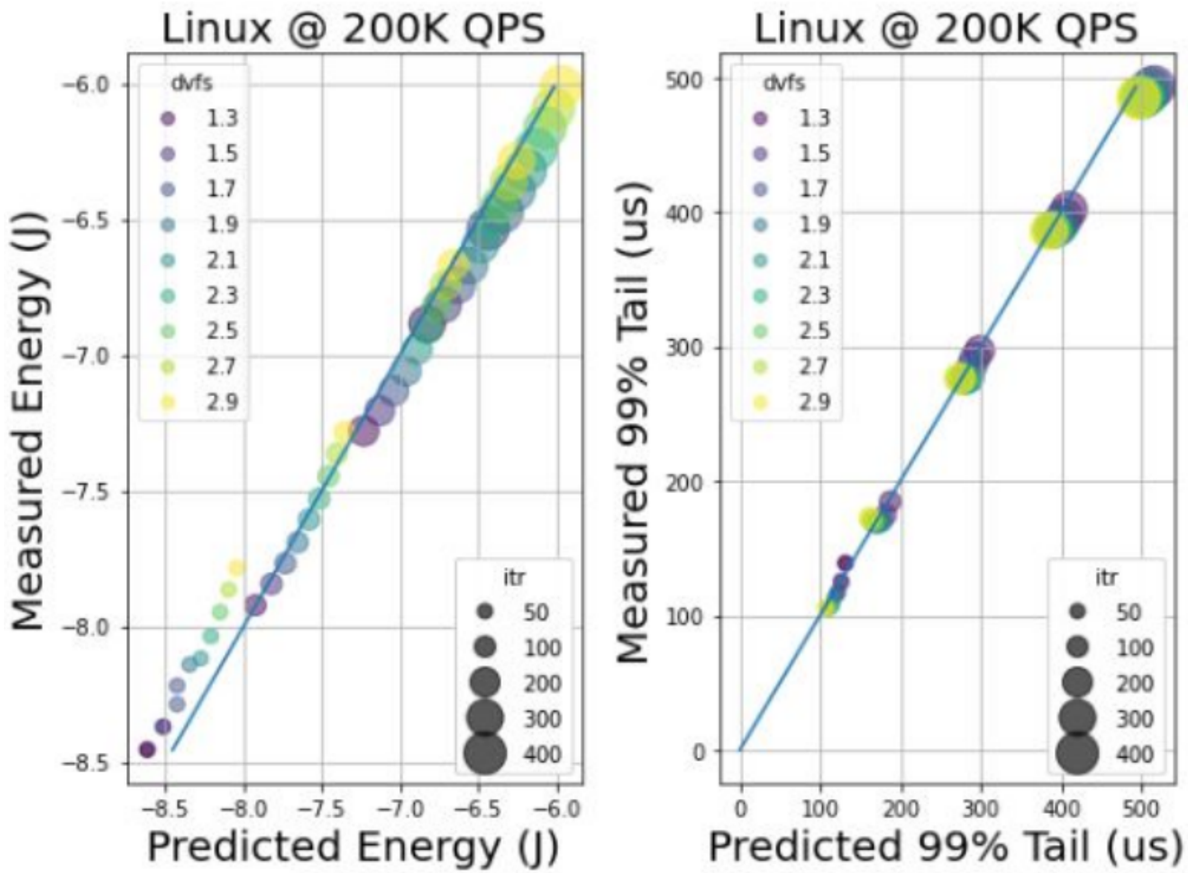
e.g.

$$\triangle t = \frac{Z}{DVFS^{1+\alpha}} + (\phi * ITR)$$

…

…

…



Details: https://handong32.github.io/PhDThesis.pdf

Modeling Result

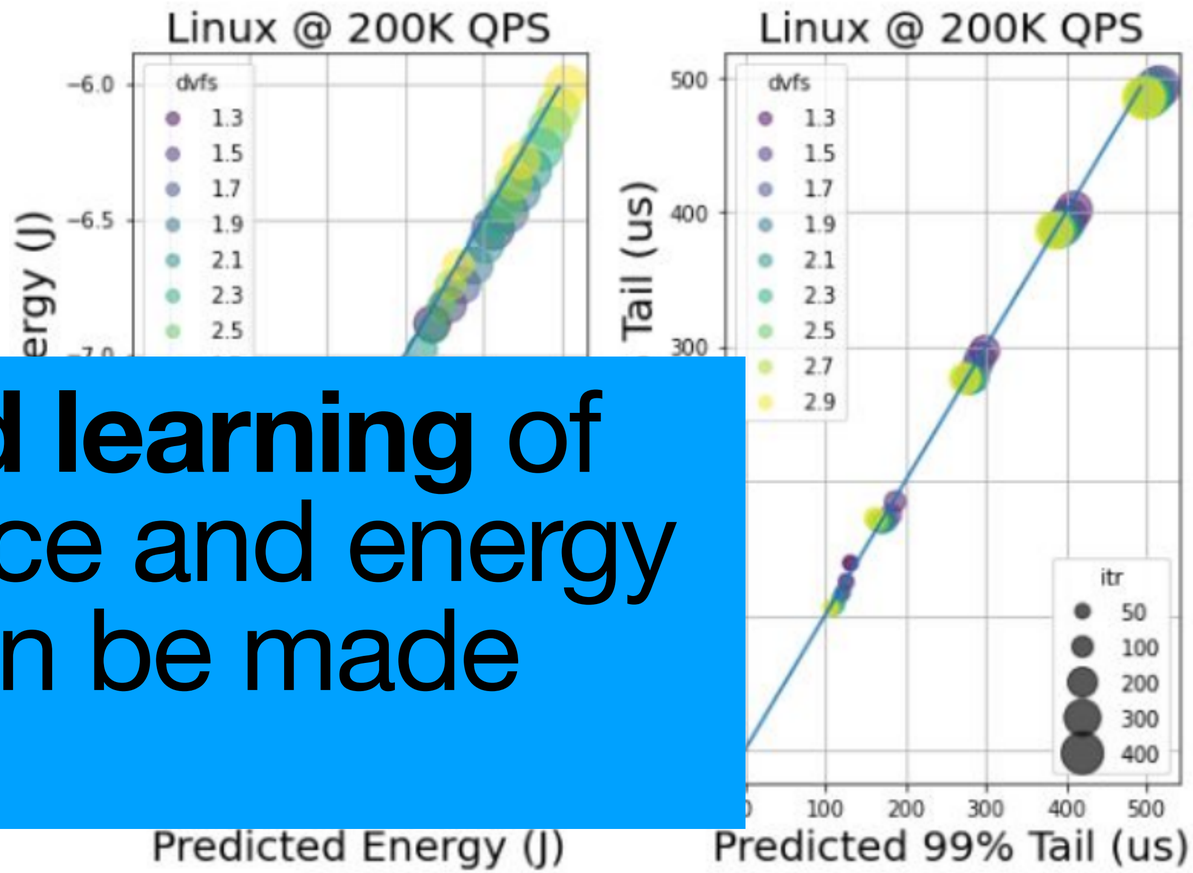
e.g.

$$\triangle t = \frac{Z}{DVFS^{1+\alpha}} + (\phi * ITR)$$

…

…

…



Controlled learning of performance and energy policies can be made feasible.

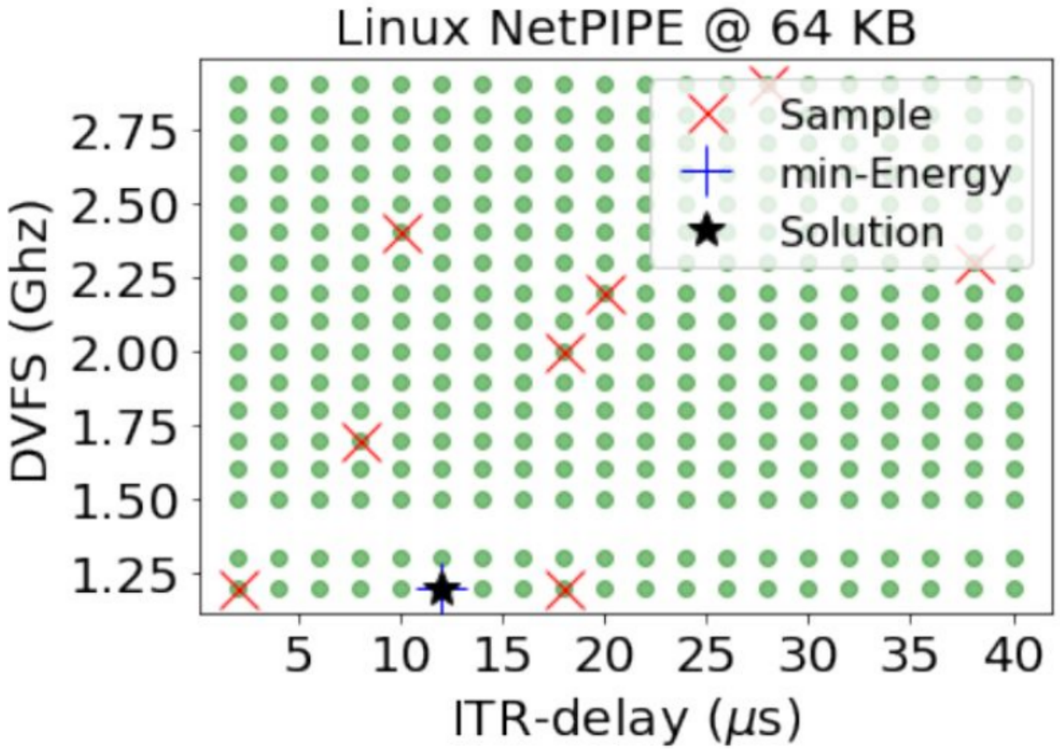
Details: https://handong32.github.io/PhDThesis.pdf

3. Applying ML to ITR, DVFS tuning

Building Block: Sample Efficient Machine Learning

e.g. Bayesian Optimization

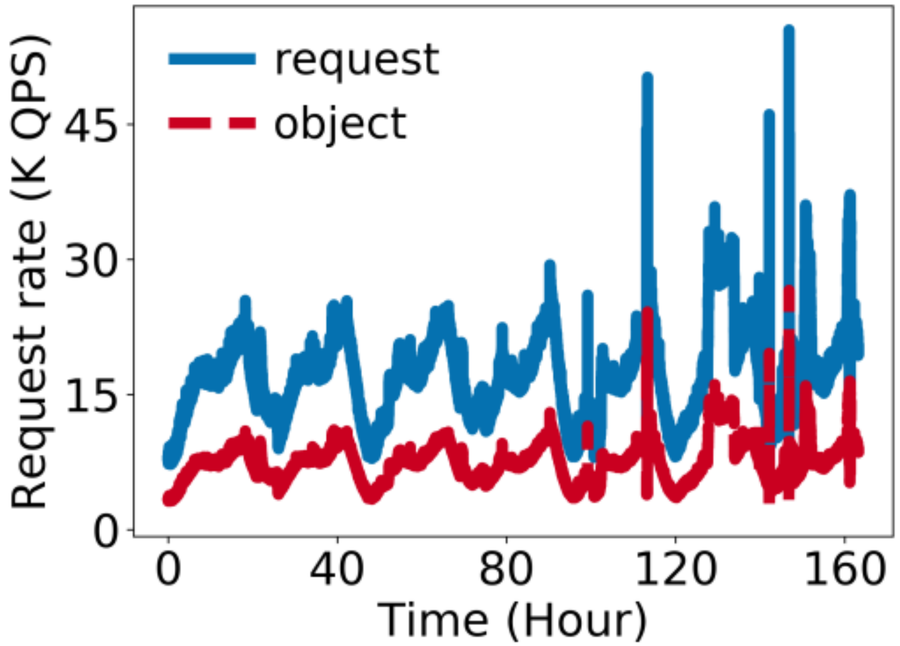
- Reduces number of samples to find some optimal configuration:



Bayesian Optimization Controller

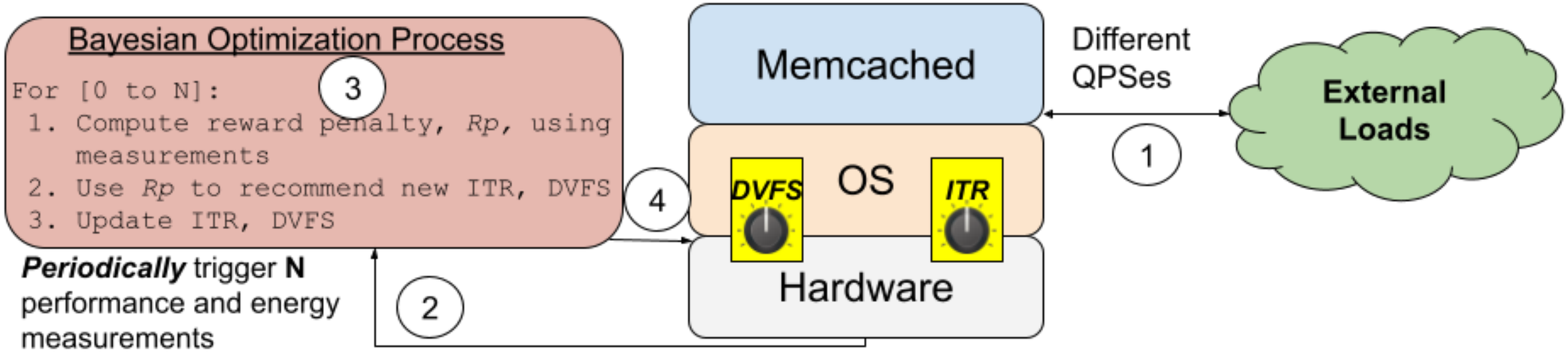
Motivating use case: in-memory KV stores

- Stable structure
- Repetitive request rates
- Times scale in hours/days

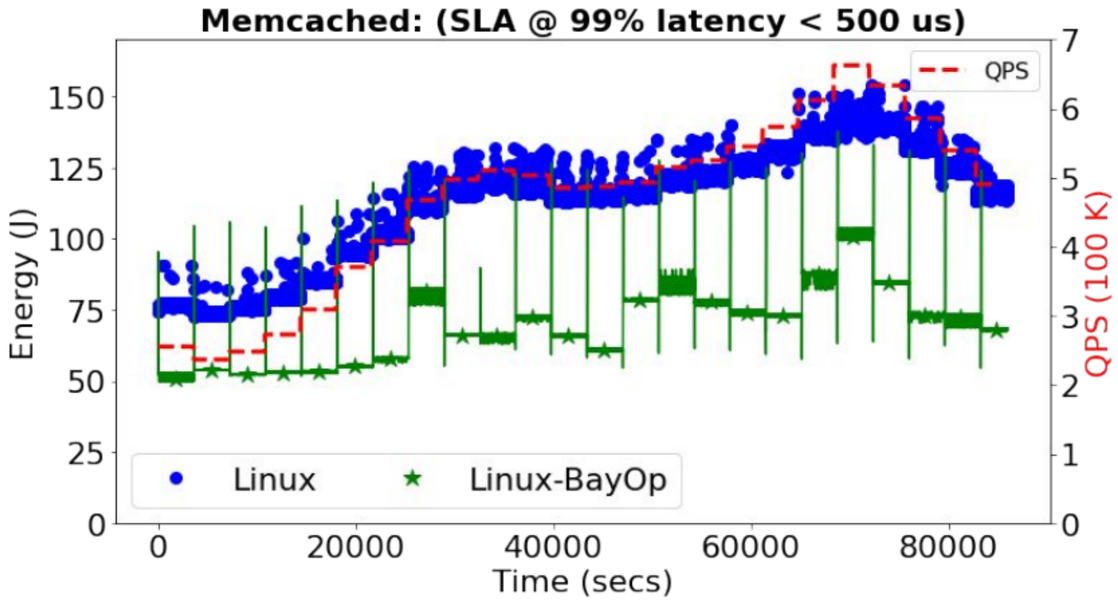


Yang et al. A large scale analysis of hundreds of in-memory cache clusters at Twitter. OSDI'20

Bayesian Optimization Controller

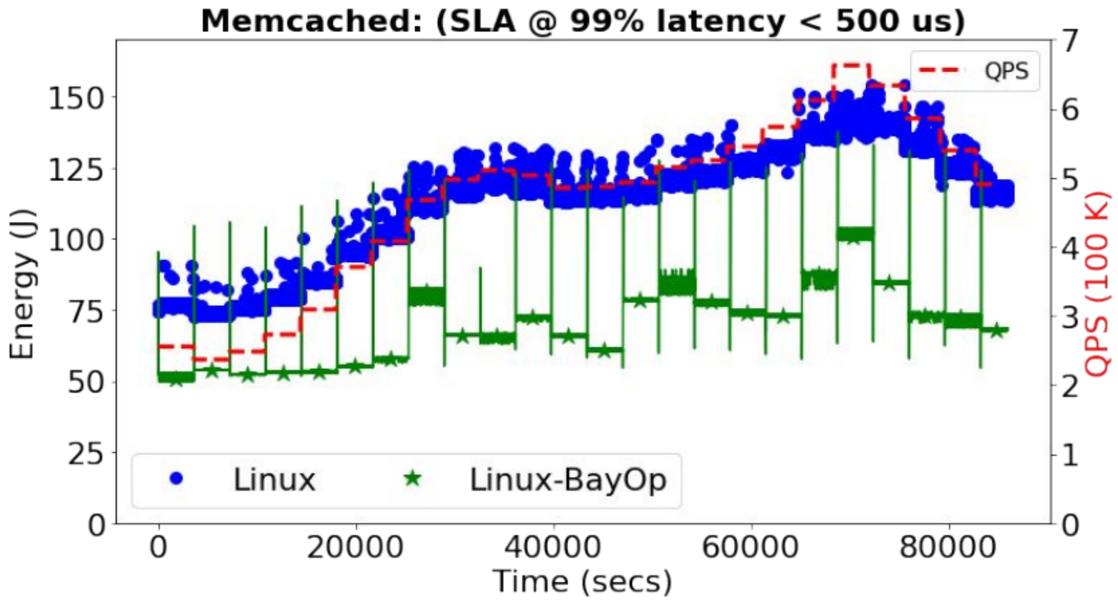


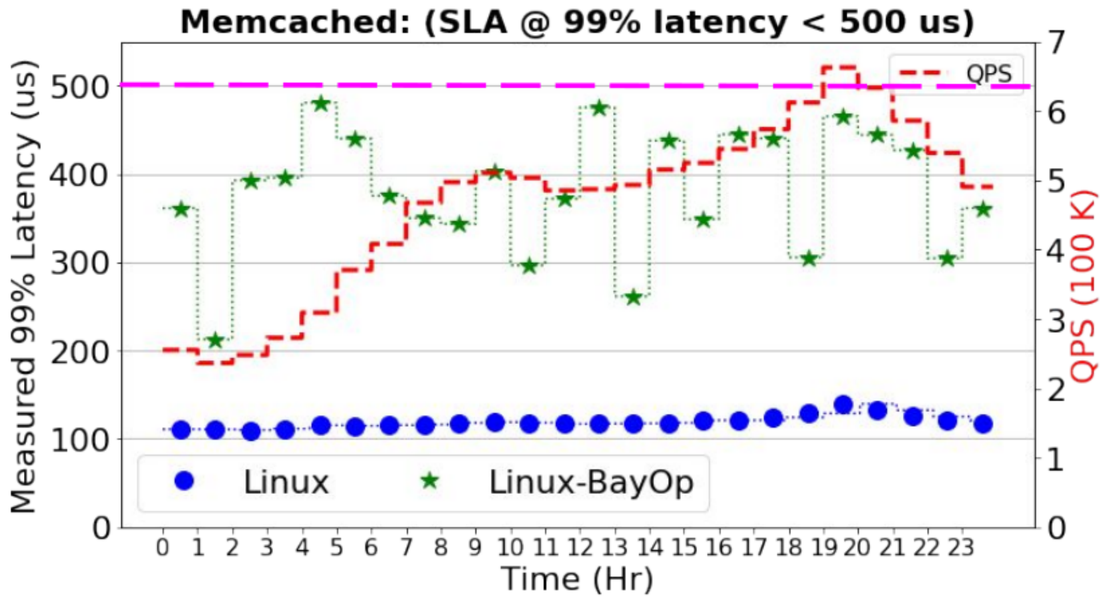
Applying Controller to Twitter `cache-trace` Dataset



Details: https://research.redhat.com/blog/article/tuning-linux-kernel-policies-for-energy-efficiency-with-machine-learning/

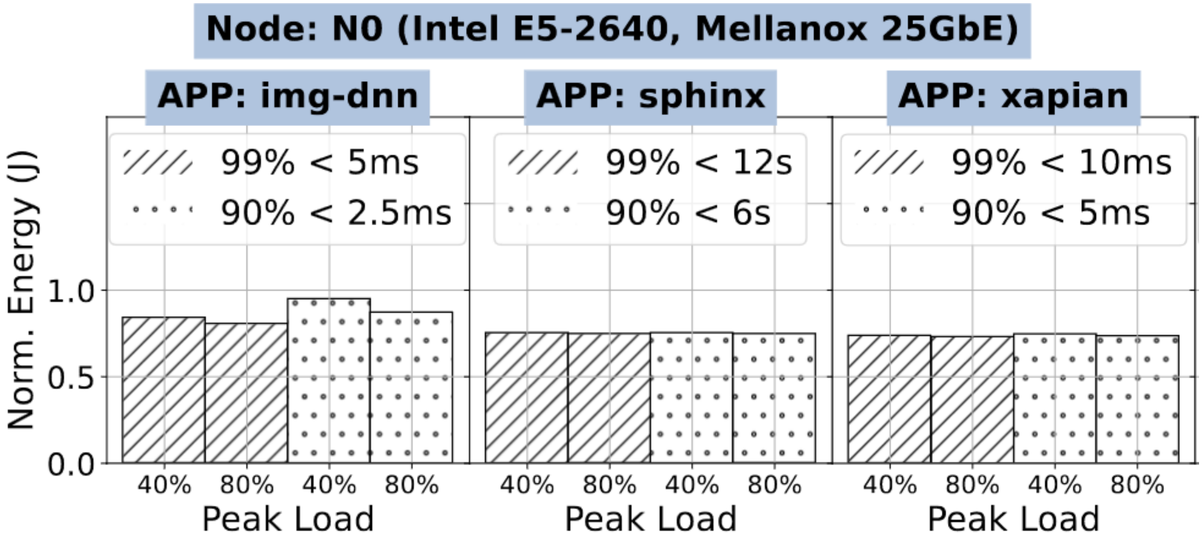
Applying Controller to Twitter `cache-trace` Dataset



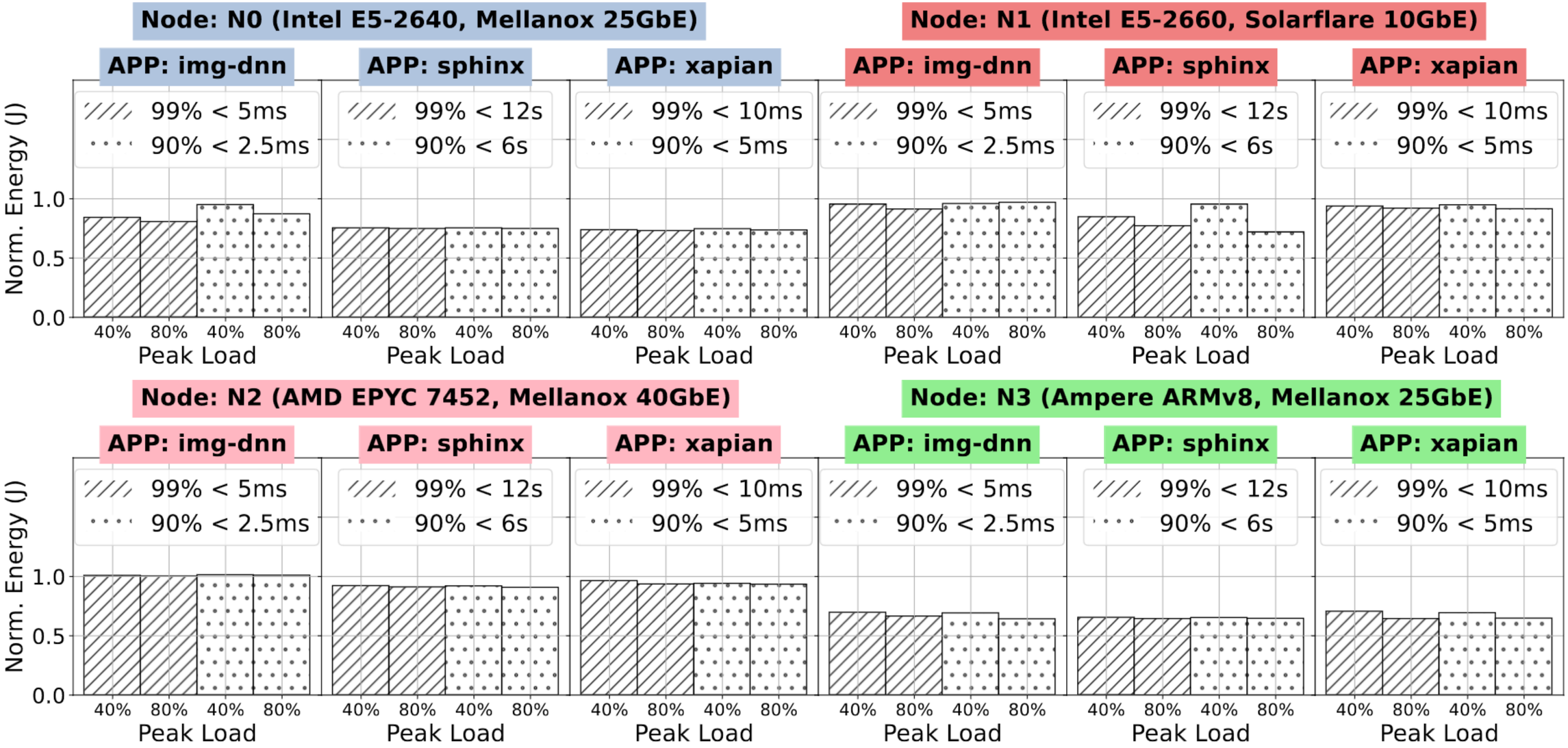


Details: https://research.redhat.com/blog/article/tuning-linux-kernel-policies-for-energy-efficiency-with-machine-learning/

Generality of ITR, DVFS



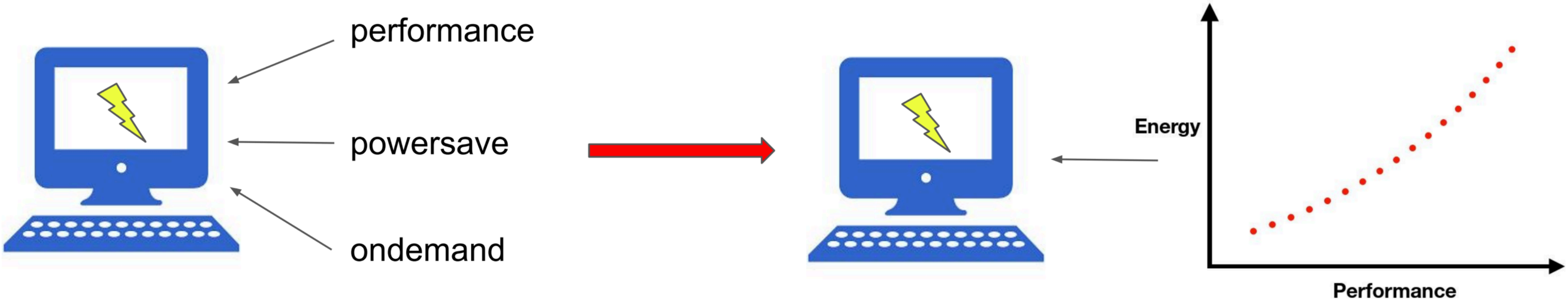
Generality of ITR, DVFS



Future Work

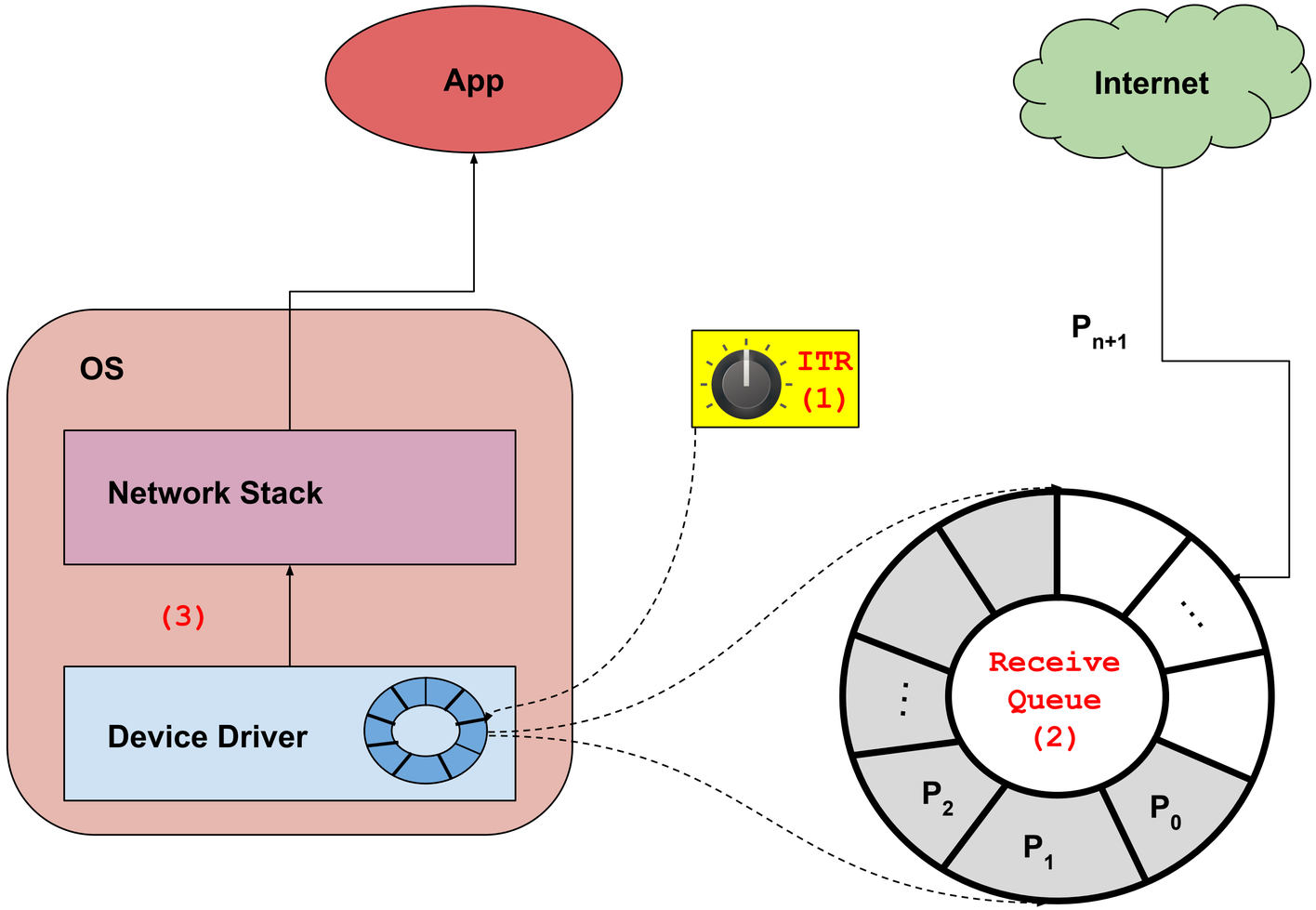
- Currently limited to a single application on a single node:
  - Per-core ITR, DVFS, multiple SLOs
- Explore other ML techniques and reward functions
- Generality of **batching**:
  - Receive/Send buffer windows, disk readahead, NFS rsize, quantums, preemption granularity, writeback throttle, LRO/GRO, NAPI poll budget, swapiness, etc

Conclusion



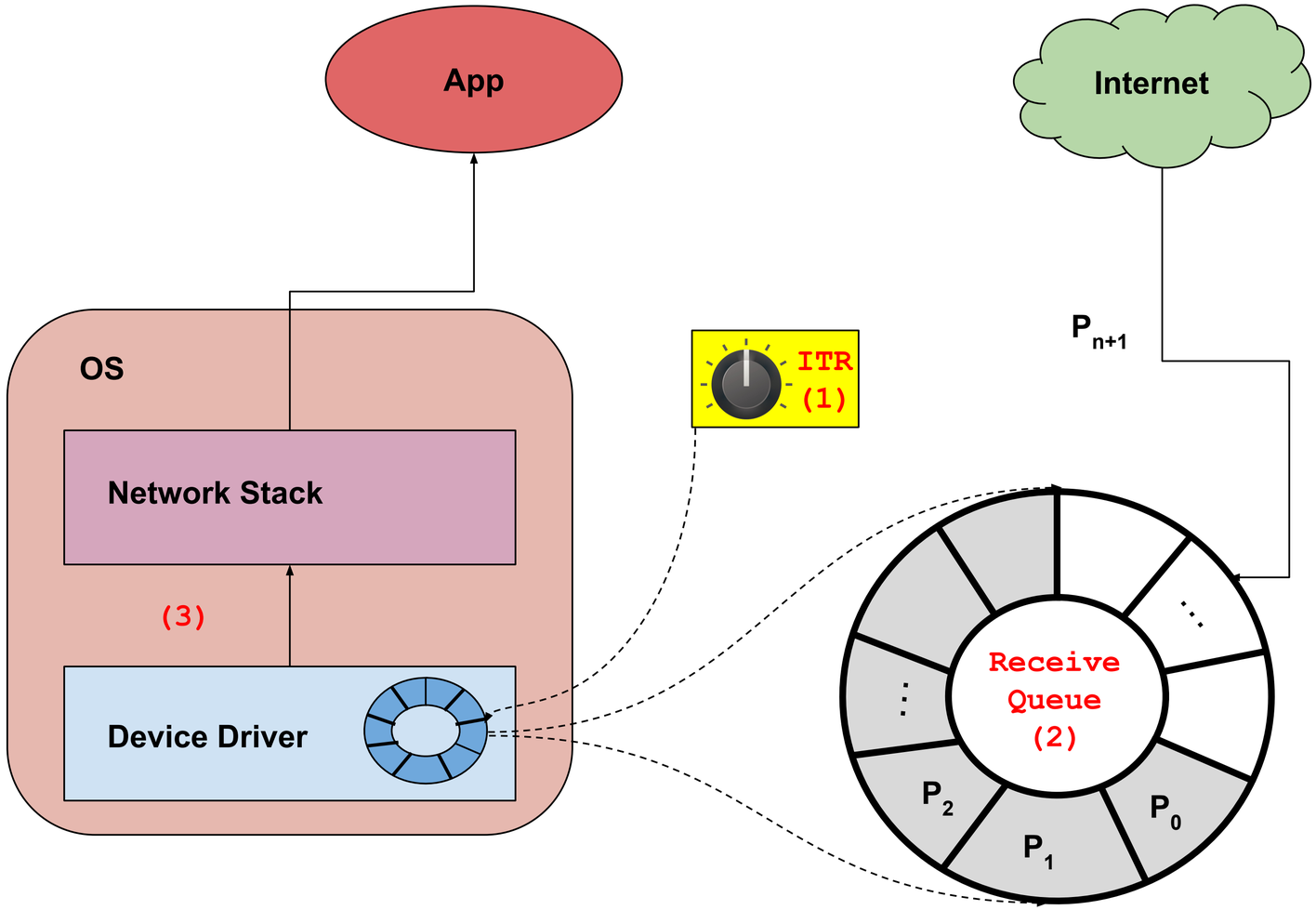
Backups

Interrupt Coalescing (ITR)



Interrupt Coalescing (ITR)

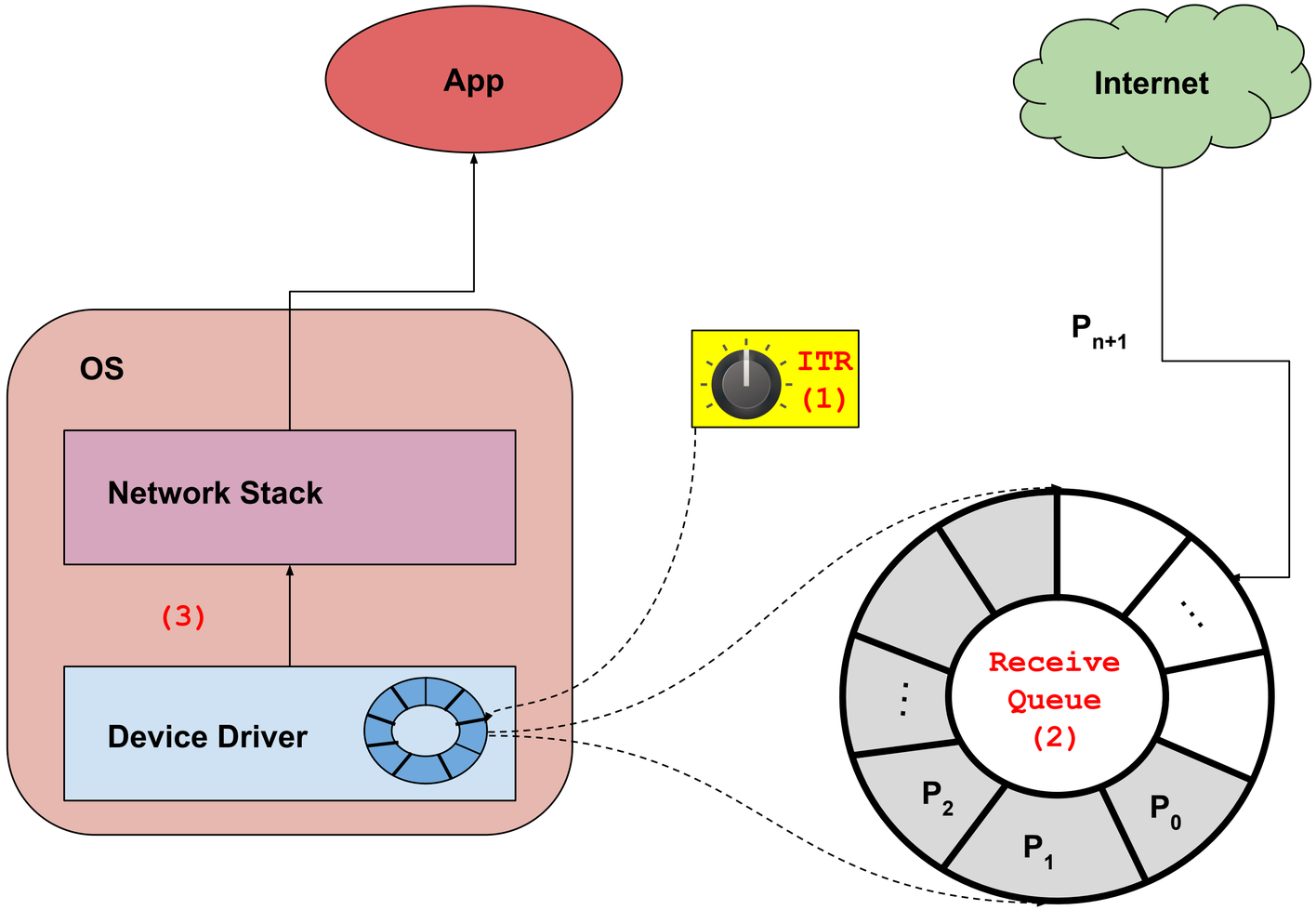
Control packet batching behavior via *ethtool*:



Interrupt Coalescing (ITR)

Control packet batching behavior via *ethtool*:

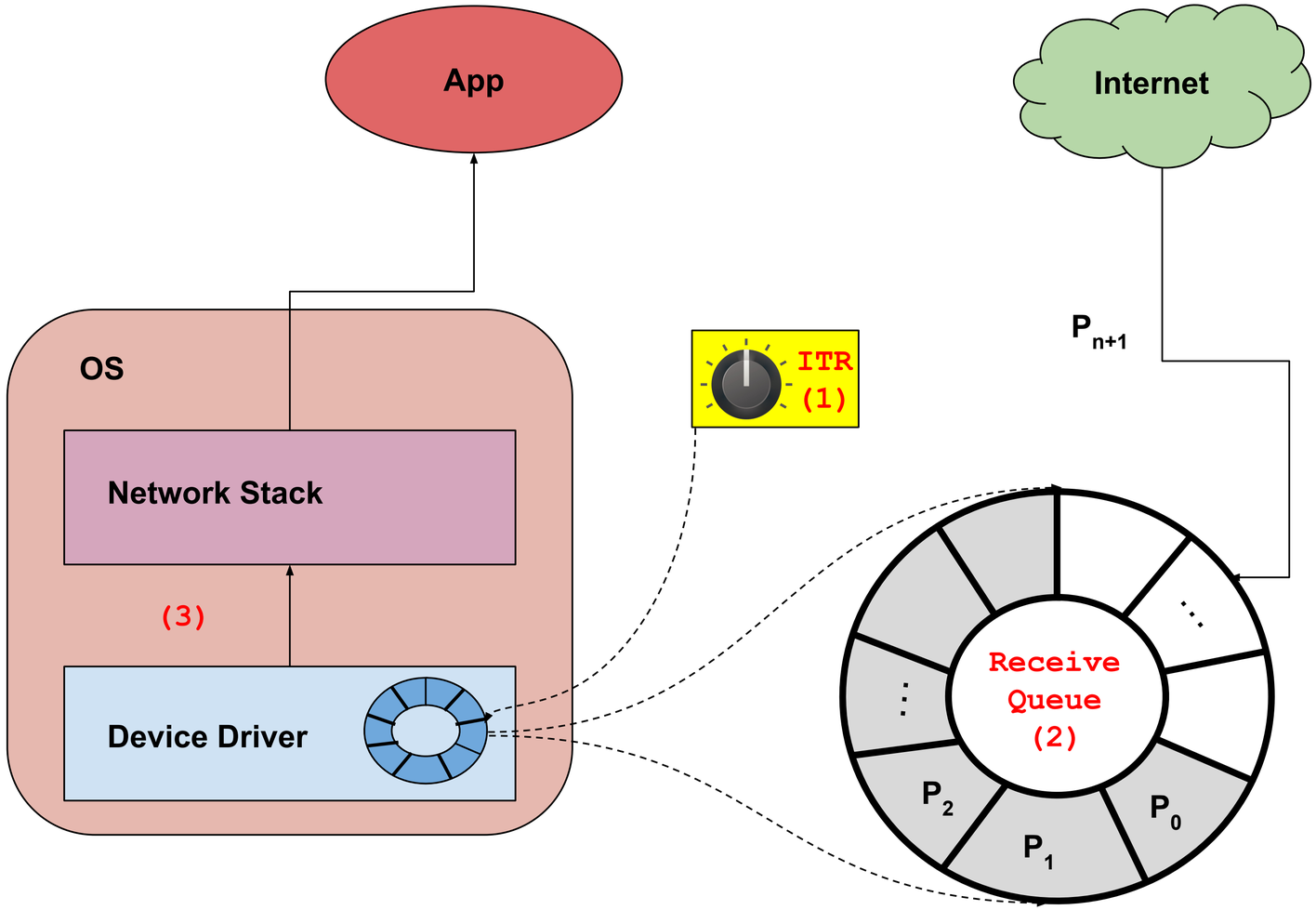
1. Set ITR value



Interrupt Coalescing (ITR)

Control packet batching behavior via *ethtool*:

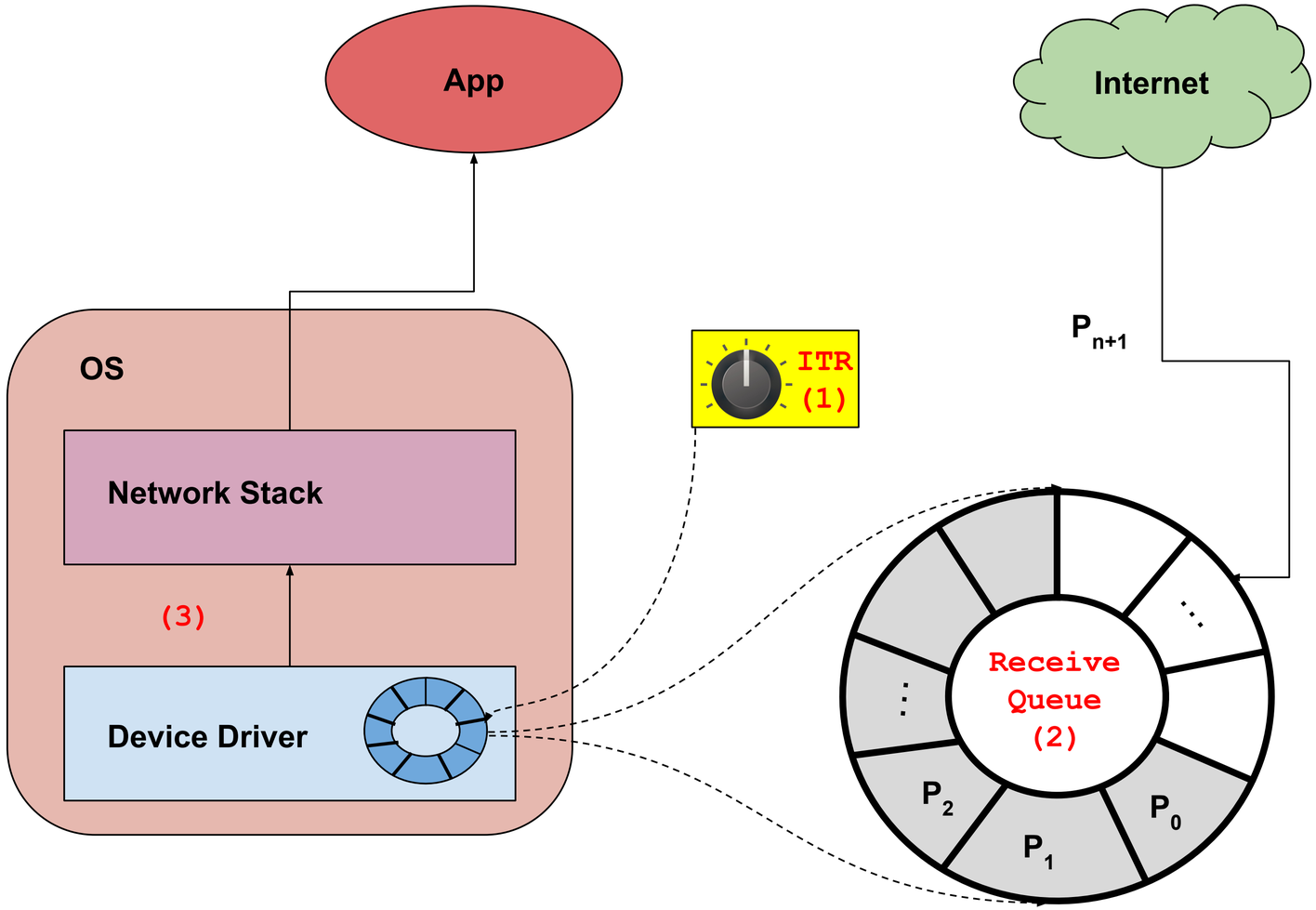
1. Set ITR value
2. Incoming packets are buffered on receive queue until ITR value has been reached



Interrupt Coalescing (ITR)

Control packet batching behavior via *ethtool*:

1. Set ITR value
2. Incoming packets are buffered on receive queue until ITR value has been reached
3. Network device asserts interrupt for packet processing



Processor speed: Dynamic Voltage Frequency Scaling (DVFS)

- $P = C * V^2 * f$
  - P = dynamic power
  - C = switching capacitance
  - V = operational voltage
  - f = operational frequency

Processor speed: Dynamic Voltage Frequency Scaling (DVFS)

- $P = C * V^2 * f$
  - P = dynamic power
  - C = switching capacitance
  - V = operational voltage
  - f = operational frequency

1. Set **dynamically** by Linux policy governors
2. Our work explores **static** frequencies on a per-workload basis