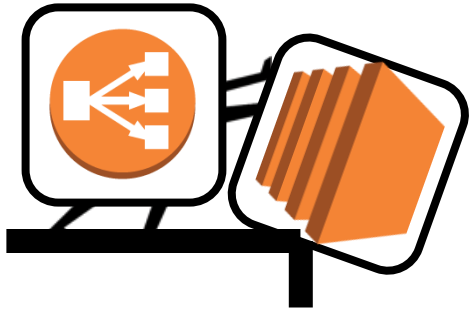
SIGCOMM 2018 Student Research Competition (Undergraduate Category)
Also in *Proceedings of SIGCOMM Posters and Demos 2018*



PAM: When Overloaded, Push Your Neighbor Aside!

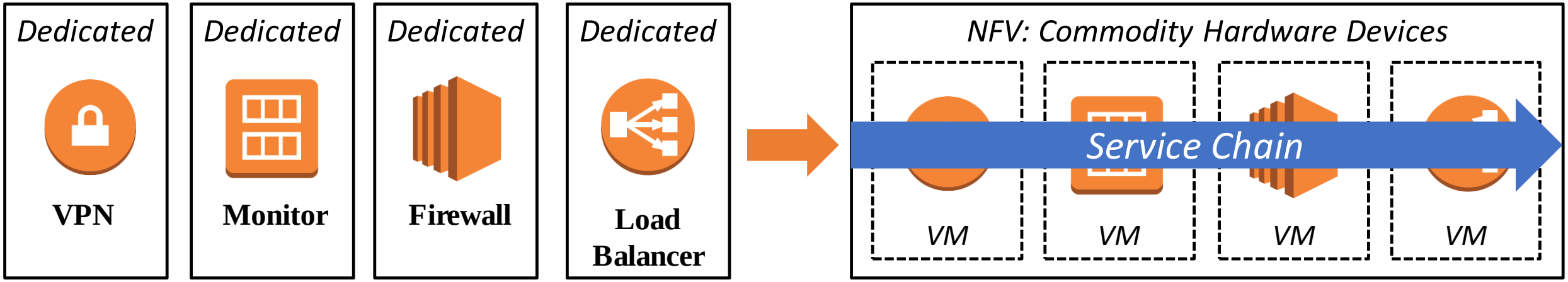
Zili Meng Jun Bi Chen Sun

Shuhe Wang Minhu Wang Hongxin Hu

Tsinghua University

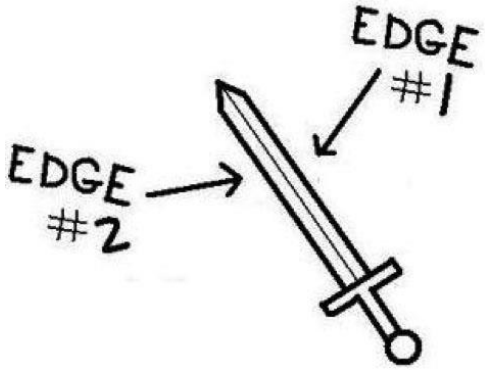
CLEMSON UNIVERSITY

NFV — Bright Side *vs.* Dark Side



Virtualization Techniques

- *Low Cost*
- *Flexibility*
- *Scalability*
- ……

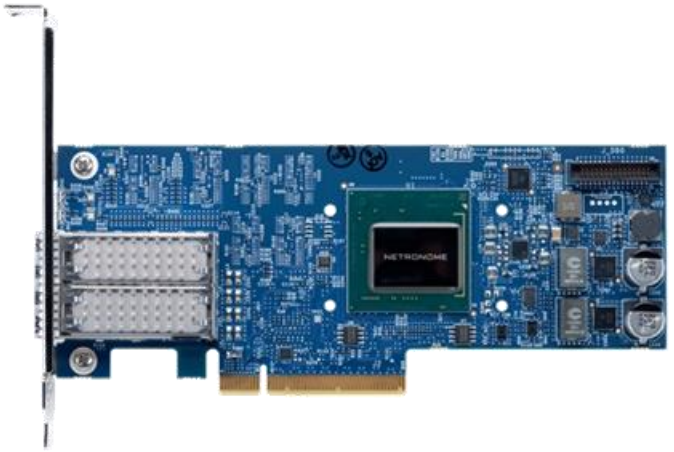


High Latency

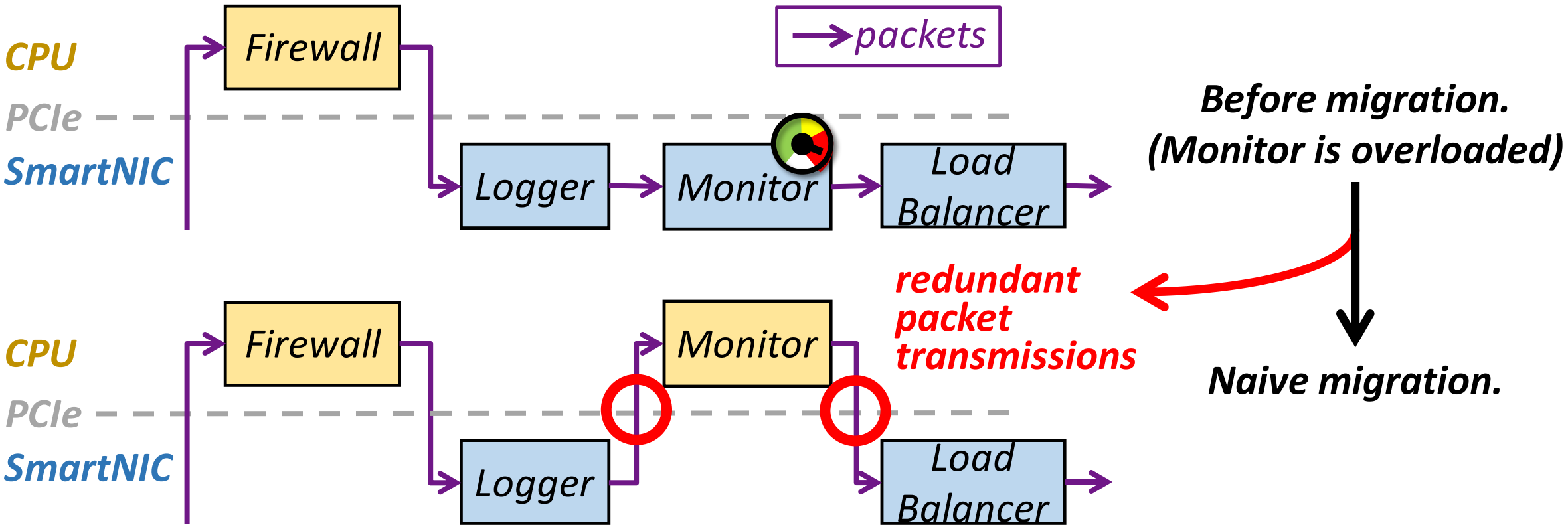
200 μs ~ 1 ms × 7

Accelerating NFV with SmartNICs

- NPU-based Multicore SmartNICs
  - Netronome, Mellanox
  - Offloading NFs to SmartNIC to improve performance.
  - Easy to develop & debug
- NF Migration between SmartNIC and CPU
  - SmartNIC may also be overloaded.
  - UNO [SoCC'17]: Ensure consistency.



Existing Solutions Cause Performance Degradation



Measurement of Transmission Latency

Packet transmission time is
comparable *to the packet processing time.*

Service Chain Latency

644 µs

Measurement of Transmission Latency

Packet transmission time is
comparable *to the packet processing time.*

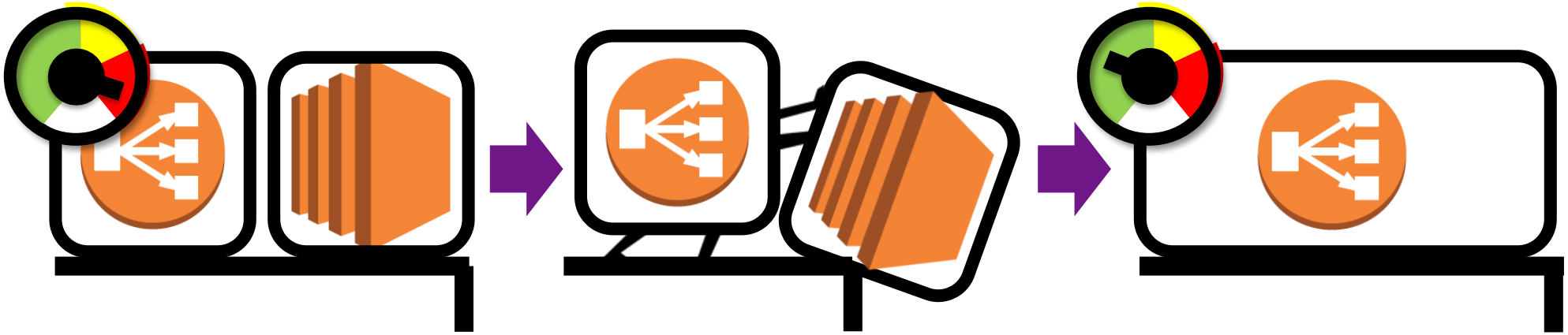
Service Chain Latency

One round-trip transmission

Can we reduce the additional transmission latency due to NF migration?

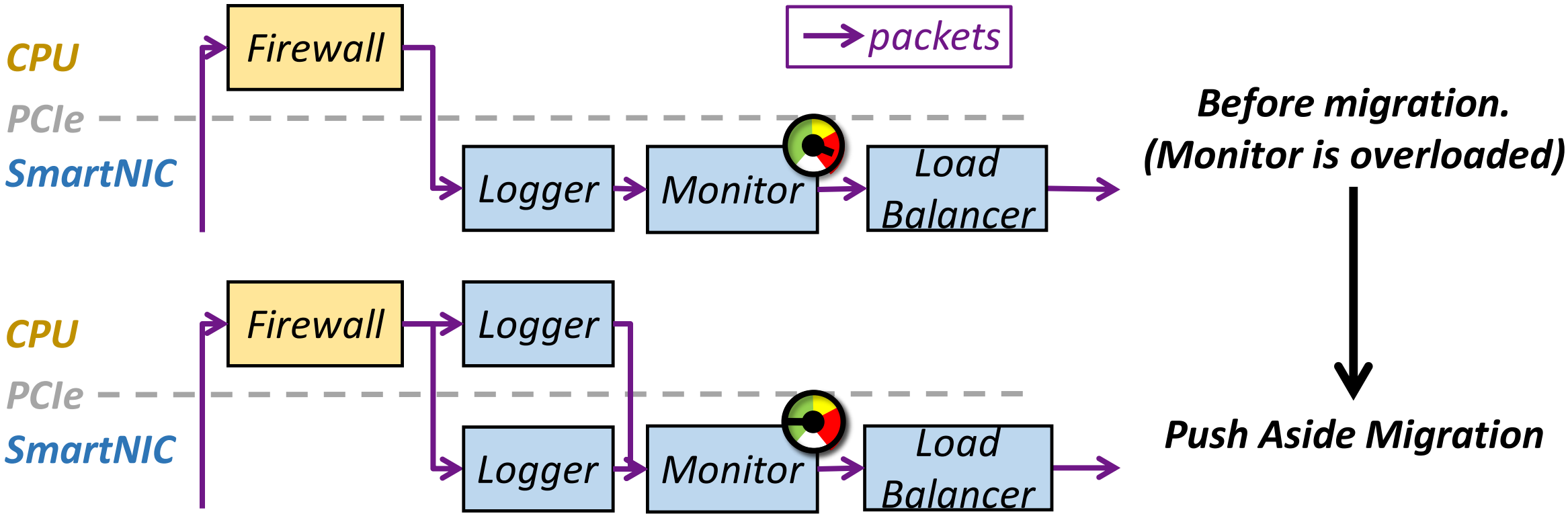
Key Novelty – Push Aside Migration

When overloaded, push your neighbor aside and occupy its resources.



Push Aside Migration

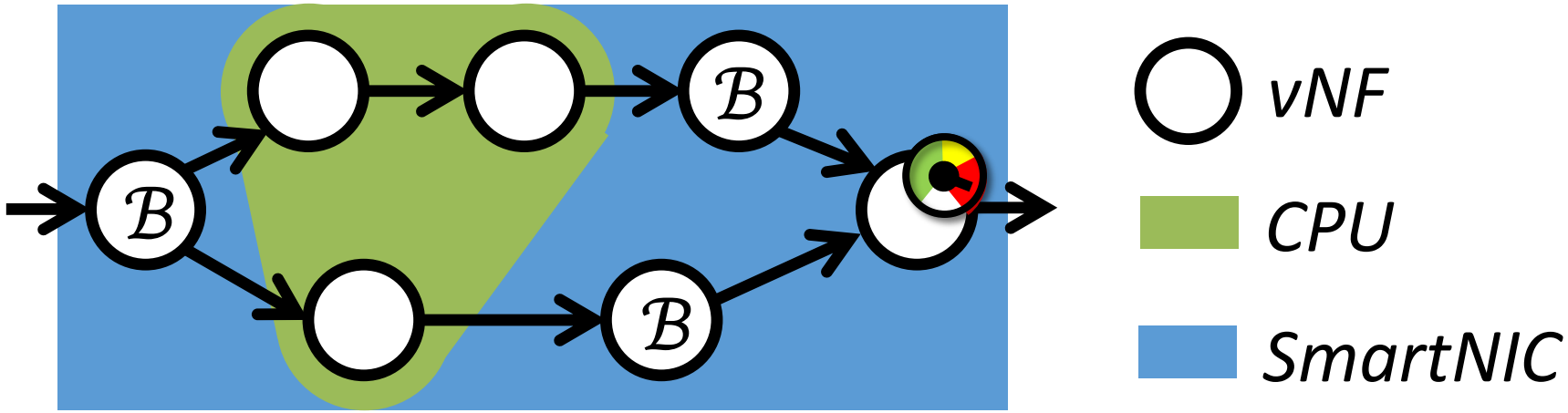
*At the **service chain** scope…*



Can we reduce the additional transmission latency due to NF migration?

Yes, push border NFs away to make space for the overloaded NF.

Dynamic Scaling of Service Chains



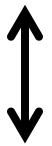
Which border NF to migrate?

Greedy-based Border vNF Selection Algorithm

Goal: Minimize the Number of vNF to Migrate

- Always select the border vNF with minimum capacity.

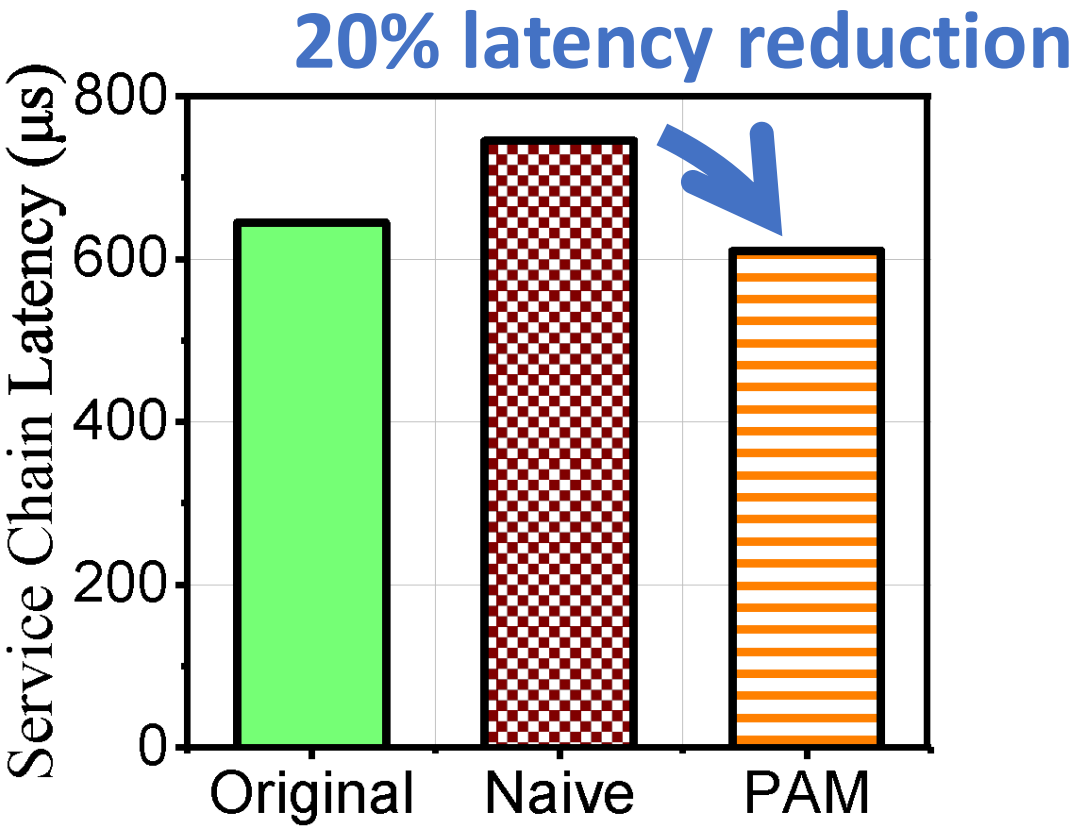
Minimum capacity given fixed resource

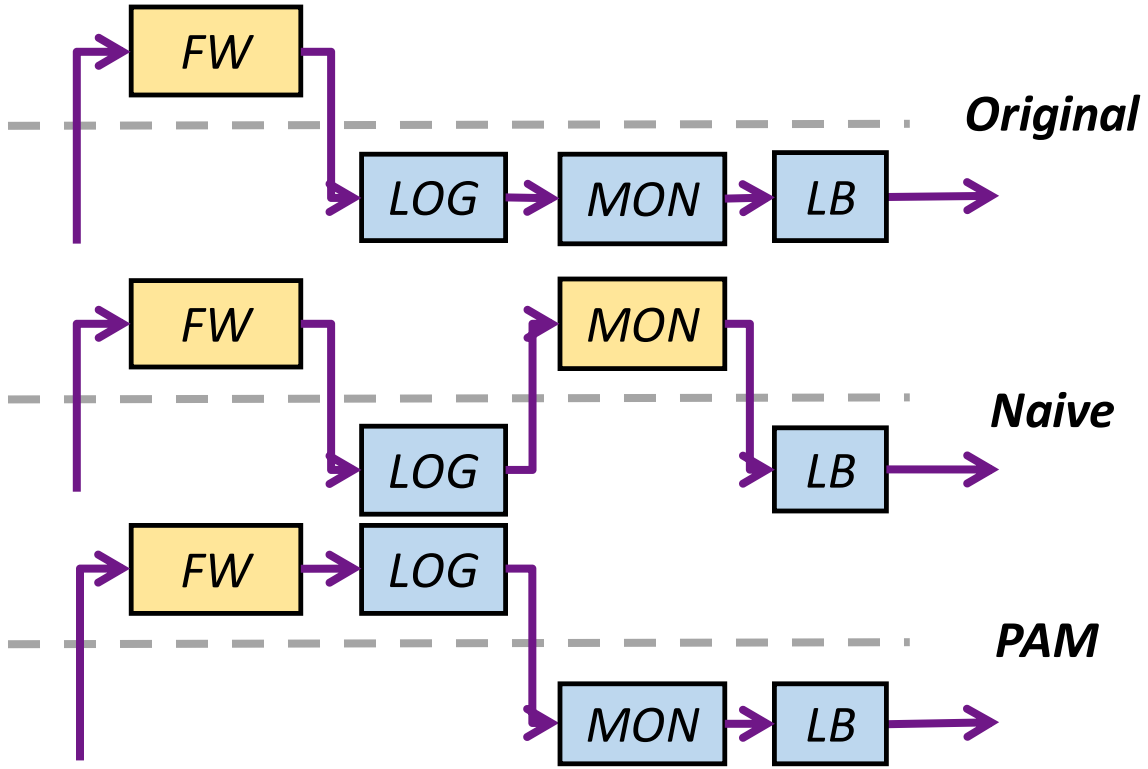


Maximum resource consumed given fixed throughput

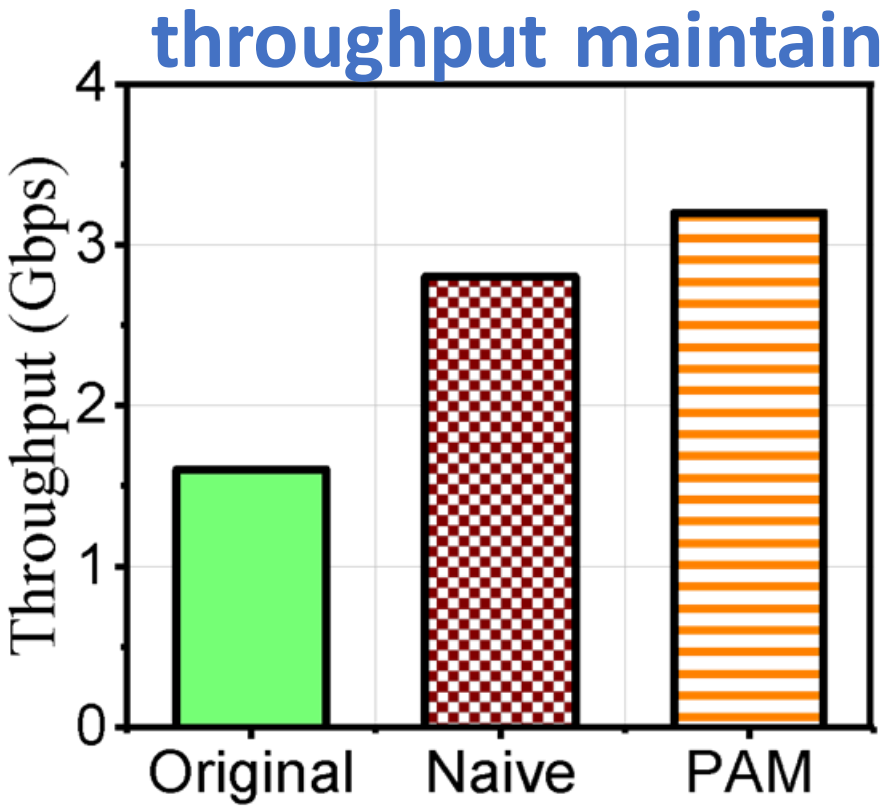
- Constraints ensured:
  - Overload on SmartNIC will be alleviated.
  - Migration should not create new hot spots on CPU.
- Please refer to our paper for more details.

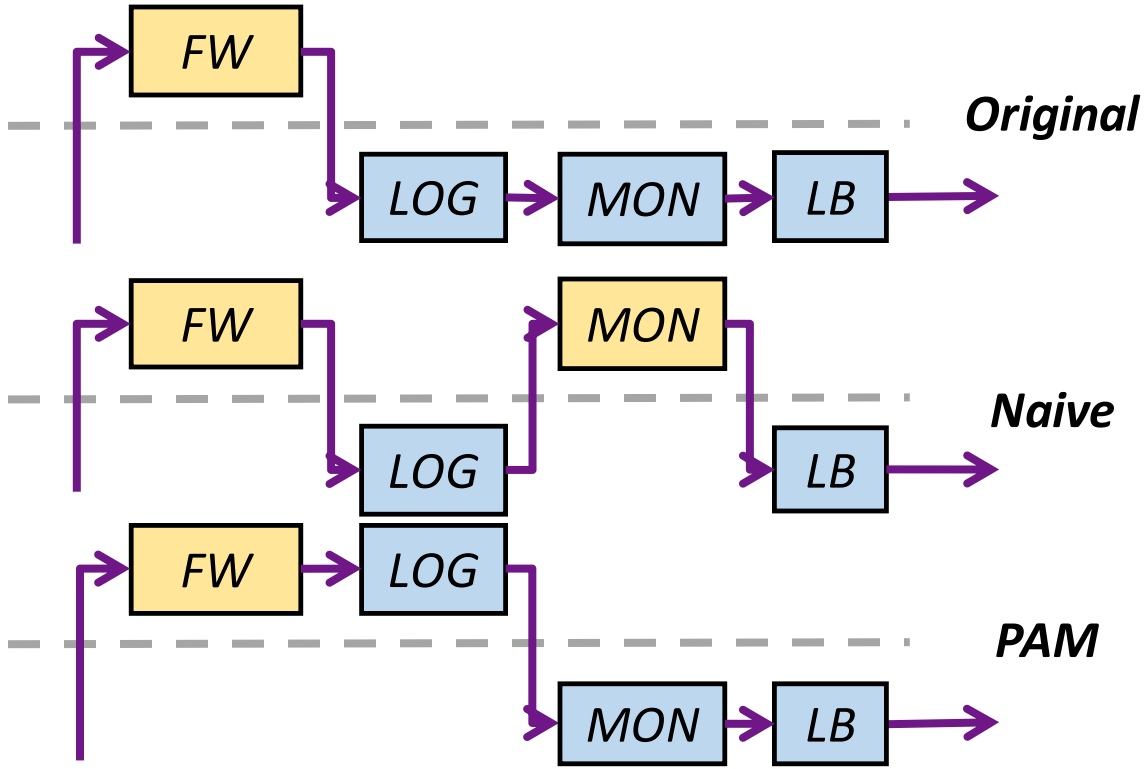
Evaluation – Latency Reduction





Evaluation – Throughput Maintenance





Discussions

- Suitability of vNF on different devices.
  - Some kinds of NFs may be suitable only to SmartNIC or CPU.
  - Potential solution (ongoing work): Introduce ***suitability*** of NFs.

Discussions

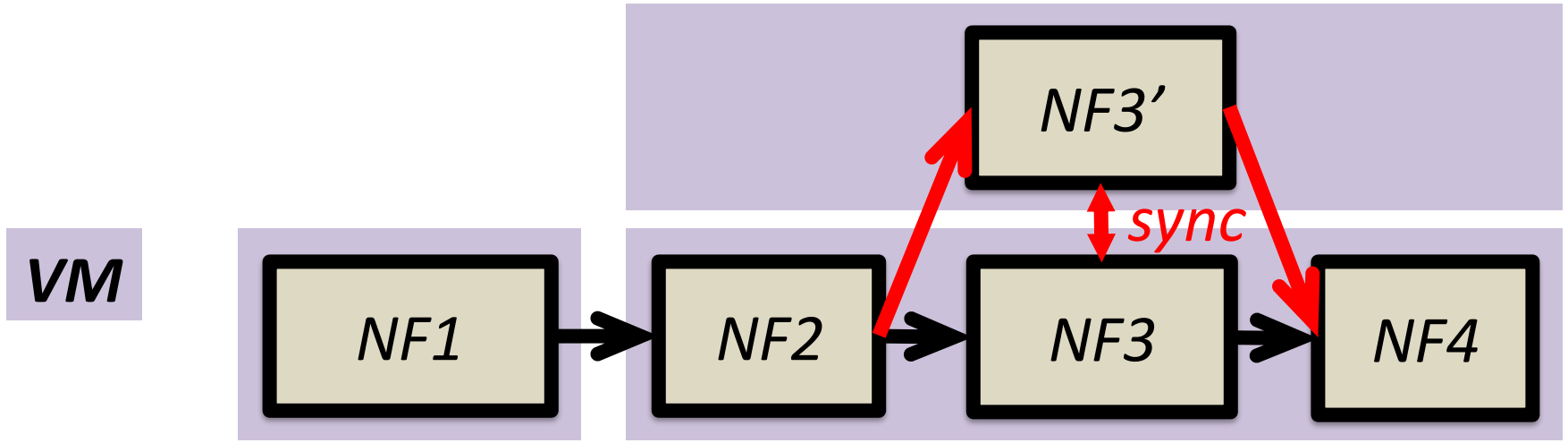
- Suitability of vNF on different devices.
  - Some kinds of NFs may be suitable only to SmartNIC or CPU.
  - Potential solution (ongoing work): Introduce ***suitability*** of NFs.
- Isolation on SmartNICs.
  - Unlike CPU, there is no mature isolation mechanisms on SmartNIC.
  - Potential solution (future work): Software isolation (NetBricks [OSDI'16])

Discussions

- Suitability of vNF on different devices.
  - Some kinds of NFs may be suitable only to SmartNIC or CPU.
  - Potential solution (ongoing work): Introduce ***suitability*** of NFs.
- Isolation on SmartNICs.
  - Unlike CPU, there is no mature isolation mechanisms on SmartNIC.
  - Potential solution (future work): Software isolation (NetBricks [OSDI'16])
- Precise analysis on PCIe and SmartNIC resource.
  - Potential solution (future work): PCIe modelling (pcie-bench [Sigcomm'18])

Applying PAM to Other Scenarios

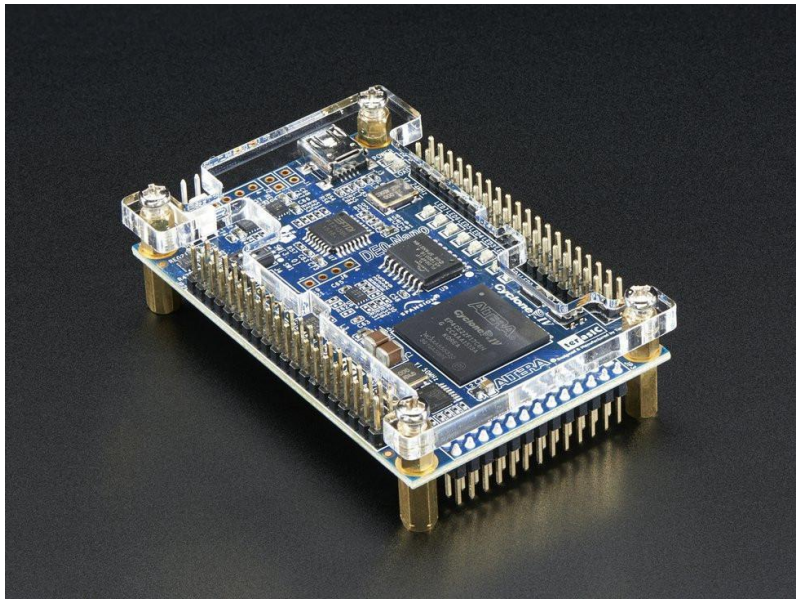
- PAM aims to bring a new direction for NF scaling.
- When multiple NFs share resources, by pushing other NFs away, the overload NF could automatically preempt resource for scaling.



Applying PAM to Other Scenarios

- PAM is designed for the scenario of SmartNIC-CPU cooperation.
- Can it be extended to other application scenarios?
  - Multiple kinds of devices.

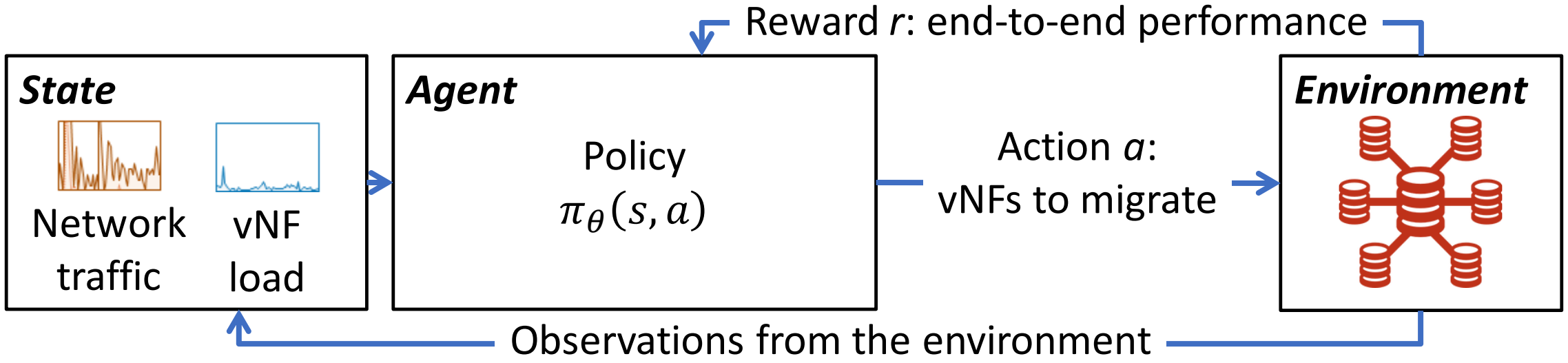
GPU



FPGA

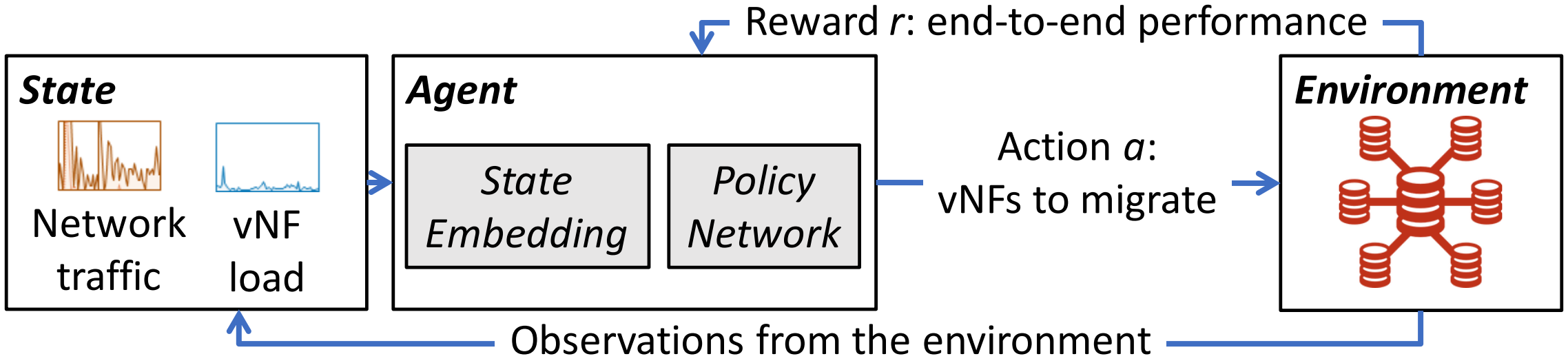
Future Thoughts on Selecting NFs to Migrate

- PAM is heuristic, but ***which NF to migrate*** is a problem.
- Can we improve the performance further and globally?
  - Inspired by scheduling problem on cluster jobs, using reinforcement learning for further performance improvement (DeepRM [HotNets'16]).



Future Thoughts on Selecting NFs to Migrate

- PAM is heuristic, but ***which NF to migrate*** is a problem.
- Can we improve the performance further and globally?
  - Inspired by scheduling problem on cluster jobs, using reinforcement learning for further performance improvement (DeepRM [HotNets'16]).



Conclusion & Takeaway

Problem: Migration between SmartNIC and CPU degrades performance.

Intuition: When one NF is overloaded, we can migrate other NFs away and grab their resources to alleviate the hot spot.

Question: Which NFs to migrate?

Answer: Migrate NFs on the border between SmartNIC and CPU with minimum capacity.

Evaluation: **18%** latency benefits.

Thank you!

www.zilimeng.info

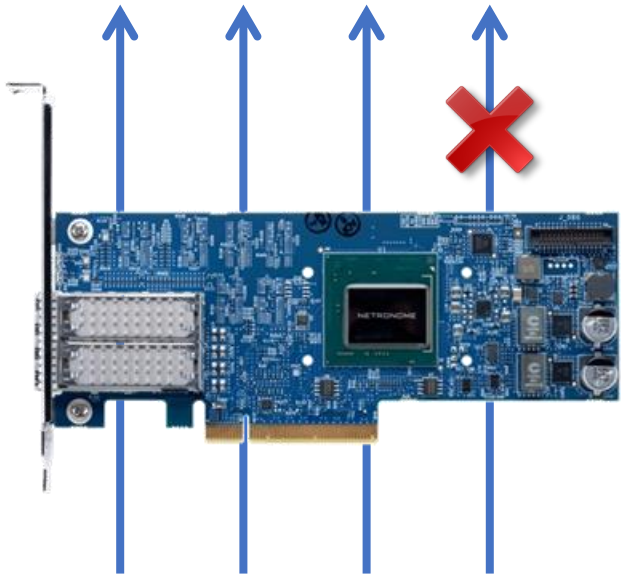
mengzl15@mails.tsinghua.edu.cn

Backup Slides

Resource Analysis

Throughput Capacity

$\theta_i^{\mathcal{C}}, \theta_i^{\mathcal{S}}$: throughput capacity of vNF i on CPU ($\mathcal{C}$) or SmartNIC ($\mathcal{S}$).



$\theta_i^{\mathcal{S}}$

vNF i	$\theta_i^{\mathcal{S}}$	$\theta_i^{\mathcal{C}}$
Firewall	10Gbps	4Gbps
Logger	2Gbps	4Gbps
Monitor	3.2Gbps	10Gbps
Load Balancer	>10Gbps	4Gbps
Payload Analyzer	5Gbps	200Mbps

Resource Analysis

Assumption

*Resource utilization of a vNF increases **linearly** with its throughput:*

$$r_i^S = \frac{\theta_{cur}}{\theta_i^S}, \qquad r_i^C = \frac{\theta_{cur}}{\theta_i^C}$$

Resource Analysis

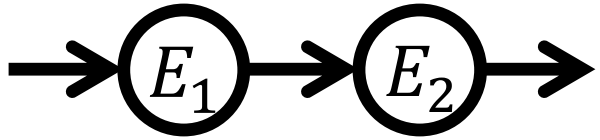
Assumption

*Resource utilization of a vNF increases **linearly** with its throughput:*

$$r_i^{\mathcal{S}} = \frac{\theta_{cur}}{\theta_i^{\mathcal{S}}}, \qquad r_i^{\mathcal{C}} = \frac{\theta_{cur}}{\theta_i^{\mathcal{C}}}$$

Deduction

The capacity θ' of the chain $E_1 \rightarrow E_2$:



$$\frac{\theta'}{\theta_1^{\mathcal{S}}} + \frac{\theta'}{\theta_2^{\mathcal{S}}} = 1 \Rightarrow \theta' = \frac{\theta_1^{\mathcal{S}}\theta_2^{\mathcal{S}}}{\theta_1^{\mathcal{S}} + \theta_2^{\mathcal{S}}}$$

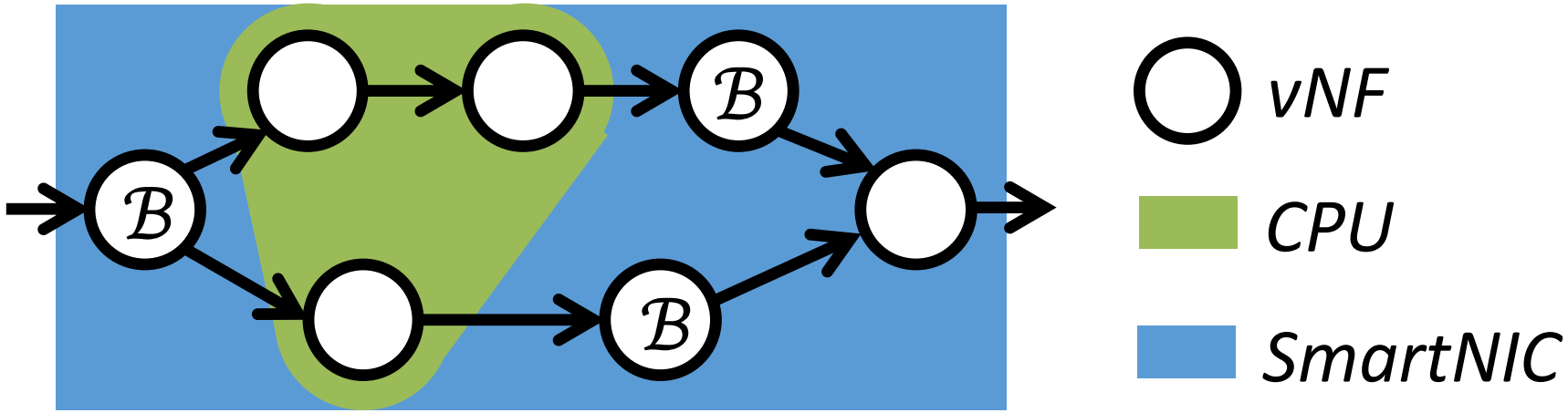
For "*Payload Analyzer → Monitor*":

$$\theta'_{measure} = 1.8\text{Gbps} \approx \theta'_{theory} = 1.9\text{Gbps}$$

Border vNF Selection Algorithm

Step 1: Border vNFs Identification

- $\mathcal{B}$: border elements on SmartNIC in a service chain (graph).
- Check whether a NF is placed together with its upstream/downstream NFs.

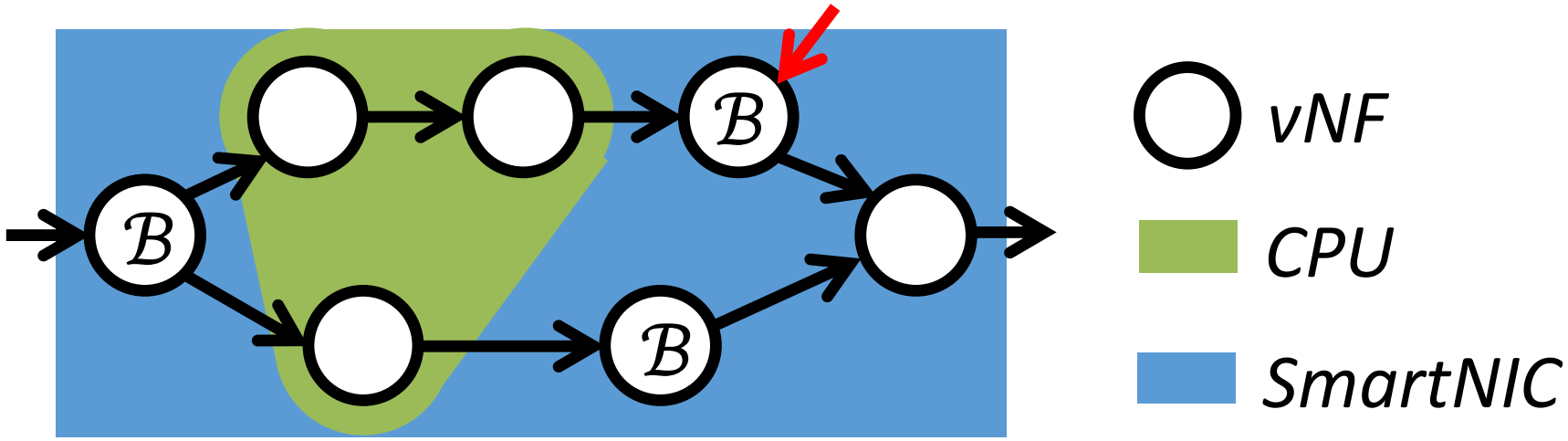


Border vNF Selection Algorithm

Step 2: Migration vNFs Selection

- Select the NF with minimum capacity.
  - Intuition: migrating the NF with minimum capacity will alleviate overload more efficiently.

$$b_0 = \underset{b \in \mathcal{B}}{\operatorname{argmin}}\, \theta_b^{\mathcal{S}}$$



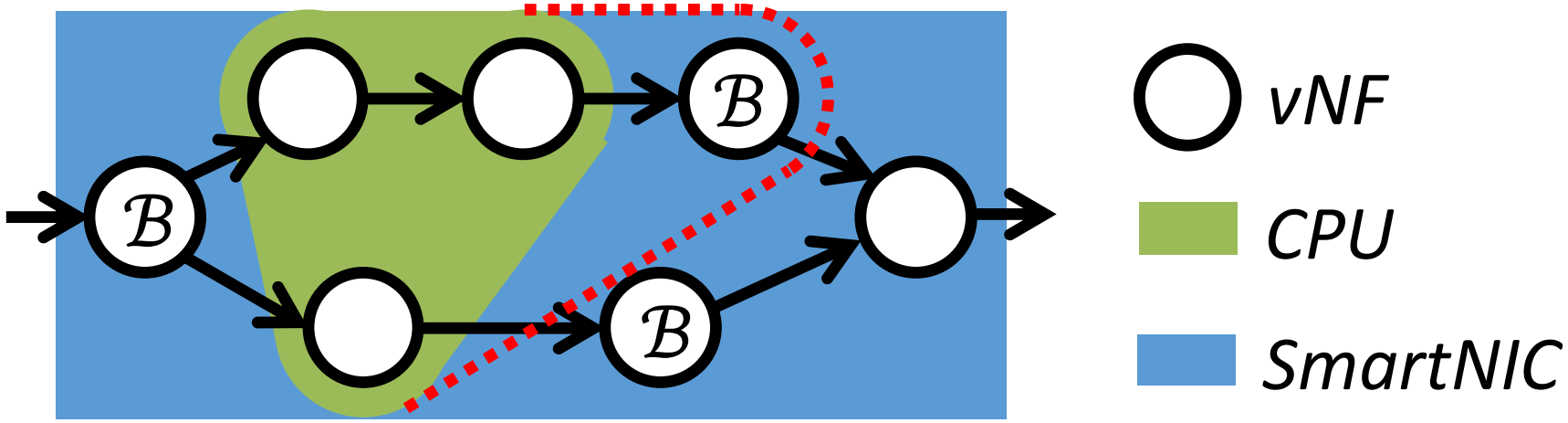
Border vNF Selection Algorithm

Step 3: Overload Alleviation Check

- $(\mathbb{C}_1)$: Migration should not cause new hot spots on CPU.

$$\sum_{i \in \{NFs\ on\ \mathcal{C}\}} \frac{\theta_{cur}}{\theta_i^{\mathcal{C}}} + \frac{\theta_{cur}}{\theta_{b_0}^{\mathcal{C}}} < 1$$

- Migrate b_0 if $(\mathbb{C}_1)$ is satisfied. Otherwise go back to **Step 2**.



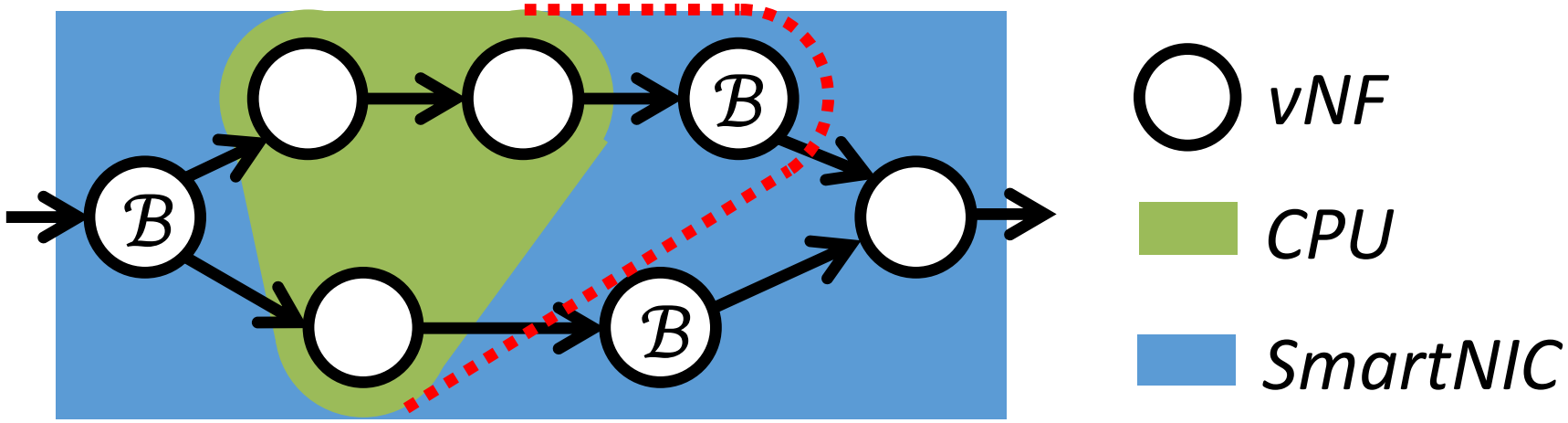
Border vNF Selection Algorithm

Step 3: Overload Alleviation Check

- $(\mathbb{C}_2)$: The overload on SmartNIC should be alleviated.

$$\sum_{i \in \{NFs\ on\ \mathcal{S}\}, i \neq b_0} \frac{\theta_{cur}}{\theta_i^{\mathcal{S}}} < 1$$

- Algorithm ends if $(\mathbb{C}_2)$ is satisfied. Otherwise go back to **Step 2**.



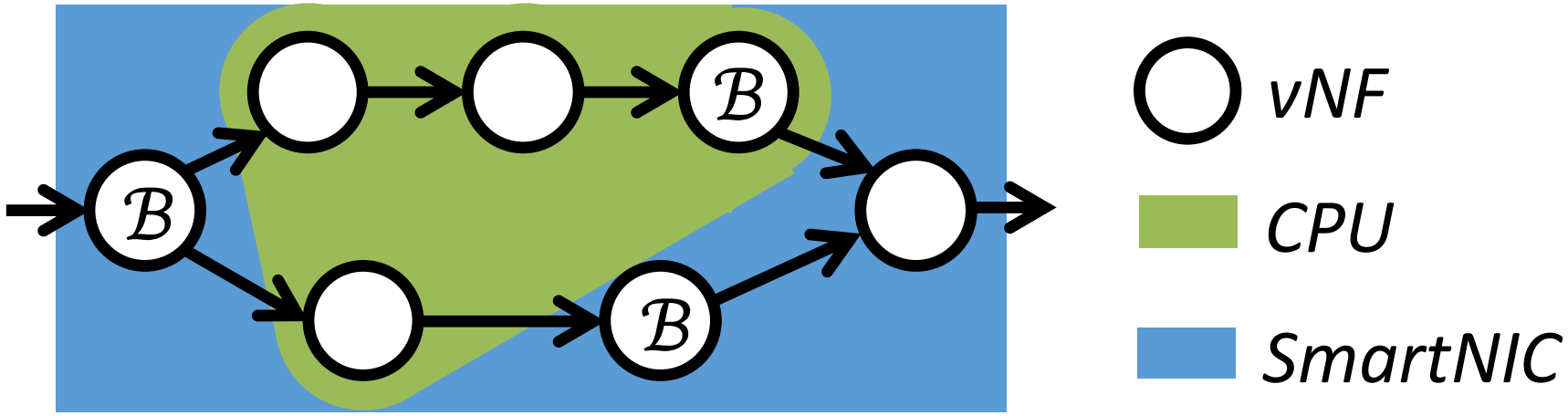
Border vNF Selection Algorithm

Step 3: Overload Alleviation Check

- $(\mathbb{C}_2)$: The overload on SmartNIC should be alleviated.

$$\sum_{i \in \{NFs\ on\ \mathcal{S}\}, i \neq b_0} \frac{\theta_{cur}}{\theta_i^{\mathcal{S}}} < 1$$

- Algorithm ends if $(\mathbb{C}_2)$ is satisfied. Otherwise go back to **Step 2**.



Policy Gradient Algorithm – REINFORCE

REINFORCE (Monte-Carlo policy gradient) relies on an estimated return by Monte-Carlo methods using episode samples to update the policy parameter θ. REINFORCE works because the expectation of the sample gradient is equal to the actual gradient:

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_\pi[Q^\pi(s,a)\nabla_\theta \ln \pi_\theta(a|s)] \\ &= \mathbb{E}_\pi[G_t \nabla_\theta \ln \pi_\theta(A_t|S_t)] \qquad ; \text{Because } Q^\pi(S_t,A_t)=\mathbb{E}_\pi[G_t|S_t,A_t]\end{aligned}$$

Therefore we are able to measure G_t from real sample trajectories and use that to update our policy gradient. It relies on a full trajectory and that's why it is a Monte-Carlo method.

The process is pretty straightforward:

1. Initialize the policy parameter θ at random.
2. Generate one trajectory on policy π_θ: $S_1, A_1, R_2, S_2, A_2, \ldots, S_T$.
3. For t=1, 2, … , T:
    1. Estimate the the return G_t;
    2. Update policy parameters: $\theta \leftarrow \theta + \alpha\gamma^t G_t \nabla_\theta \ln \pi_\theta(A_t|S_t)$

From https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#reinforce