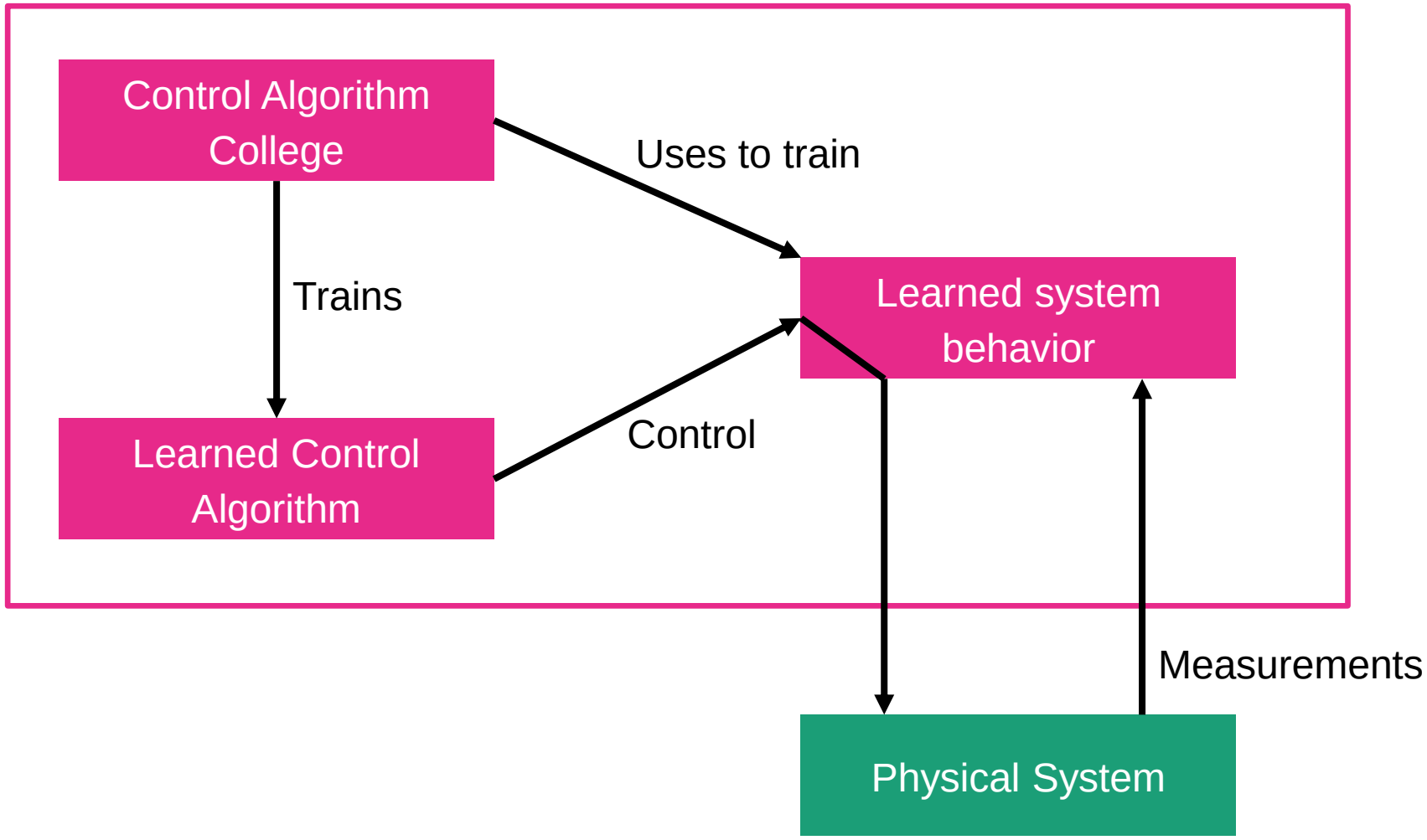
Chair of Communication Networks
TUM School of Computation, Information, and Technology
Technical University of Munich

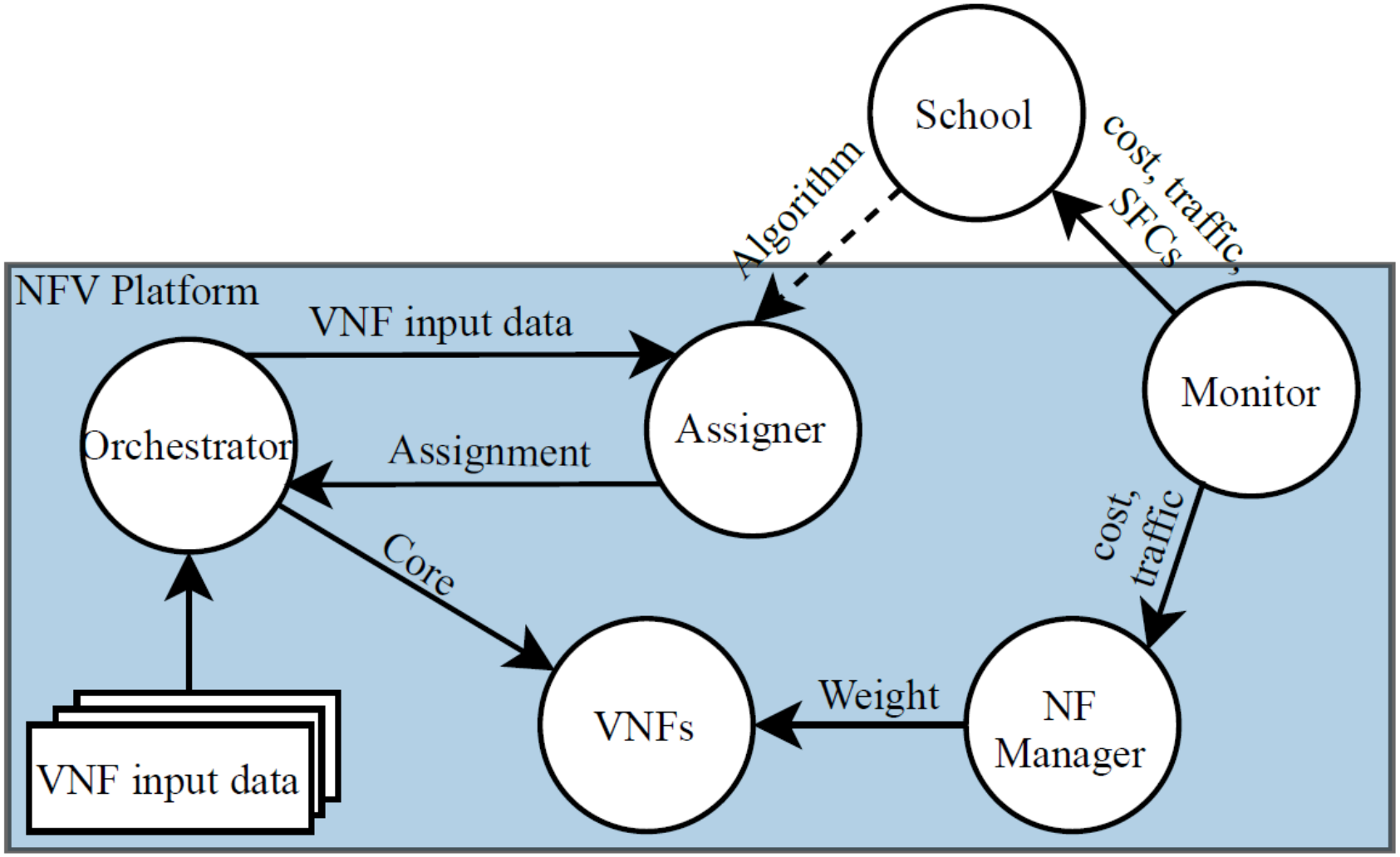
D_2A: Operating a Service Function Chain Platform with Data-Driven Scheduling Policies

Patrick Krämer
Patrick.Kraemer@tum.de

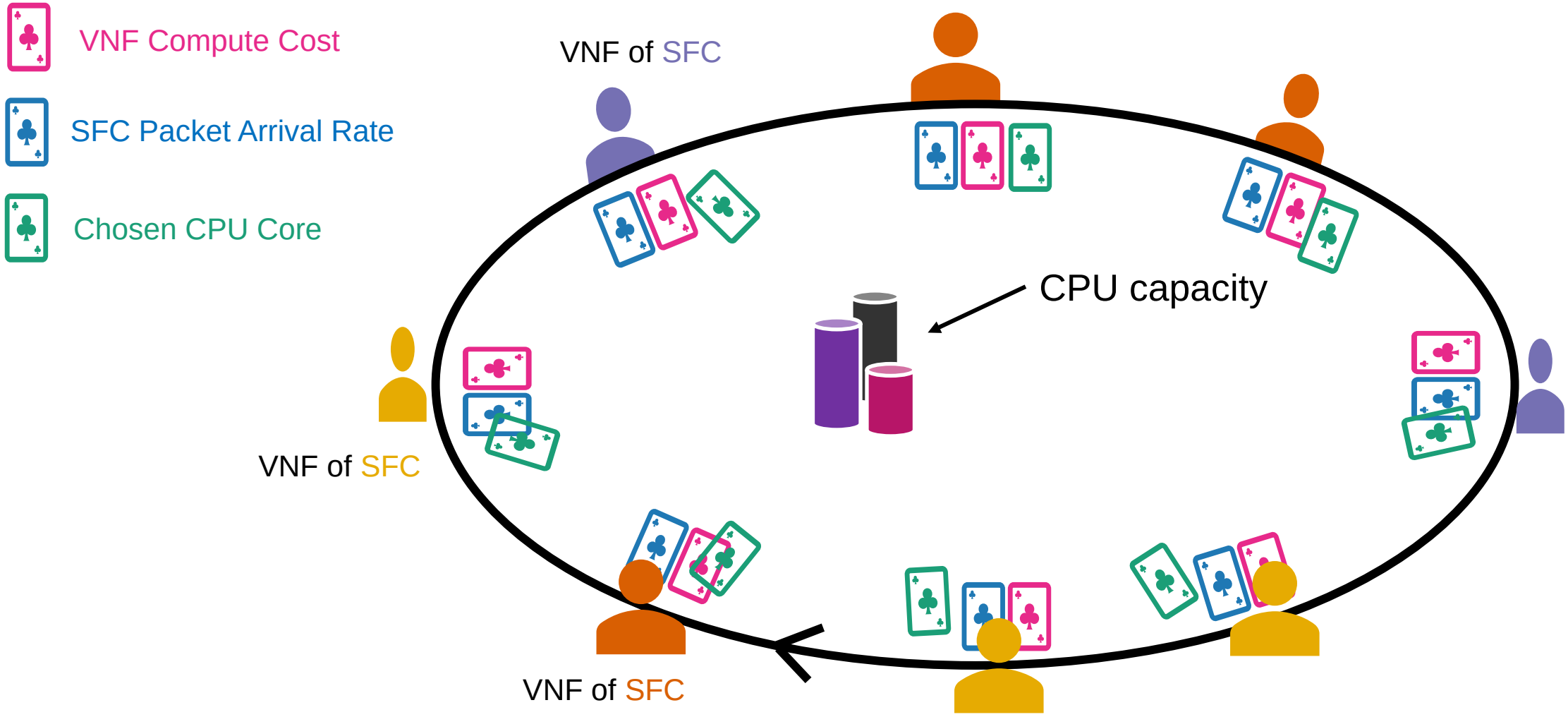
D_2A learns the system behavior and trains control algorithm with it.



Learned control algorithms assign dockerized VNFs to CPU cores.



The formulation as sequential game makes D_2A flexible and the learned behavior “predictable”.

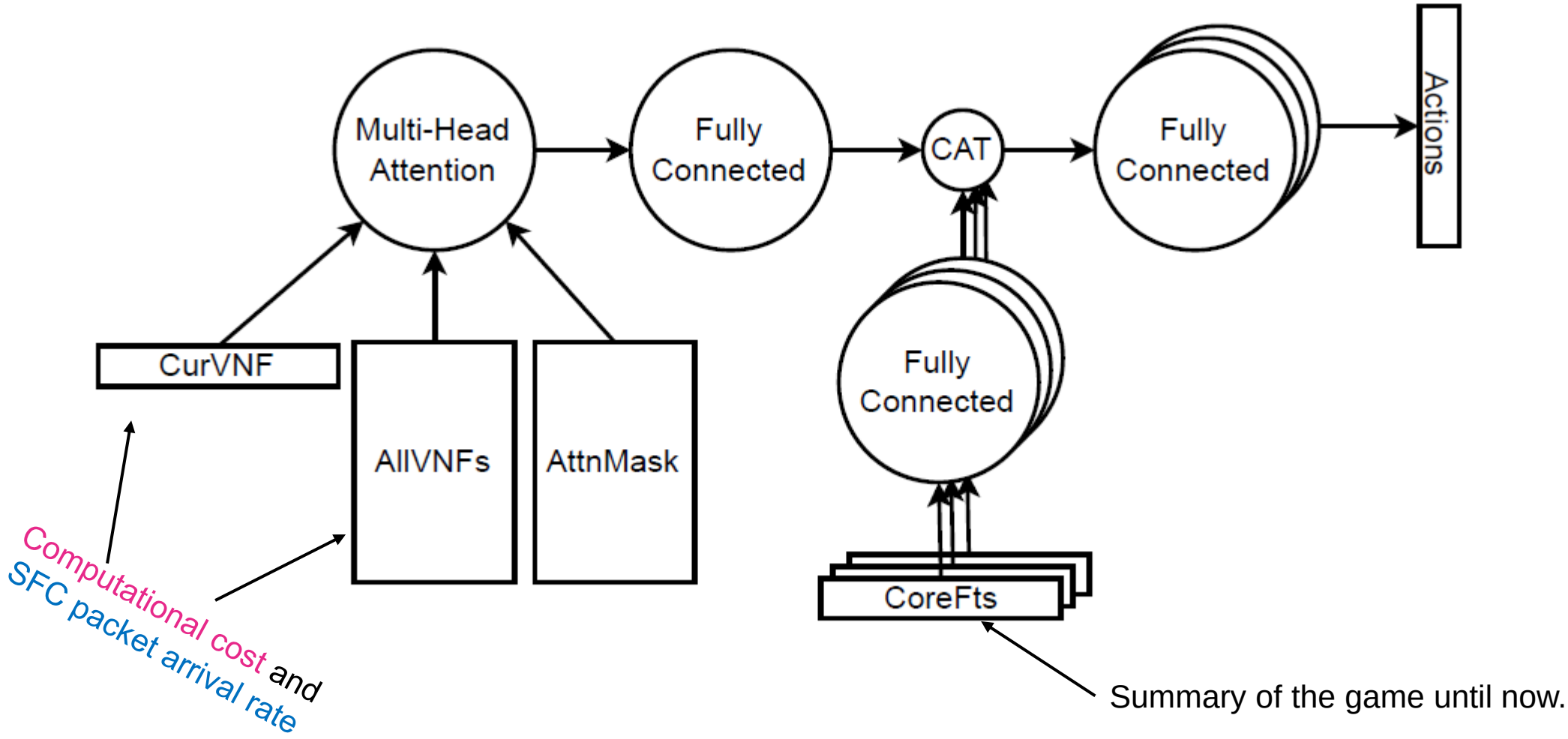


Learning best-responses with Reinforcement Learning

The reward function incentives core sharing and includes a penalty for overutilizing.

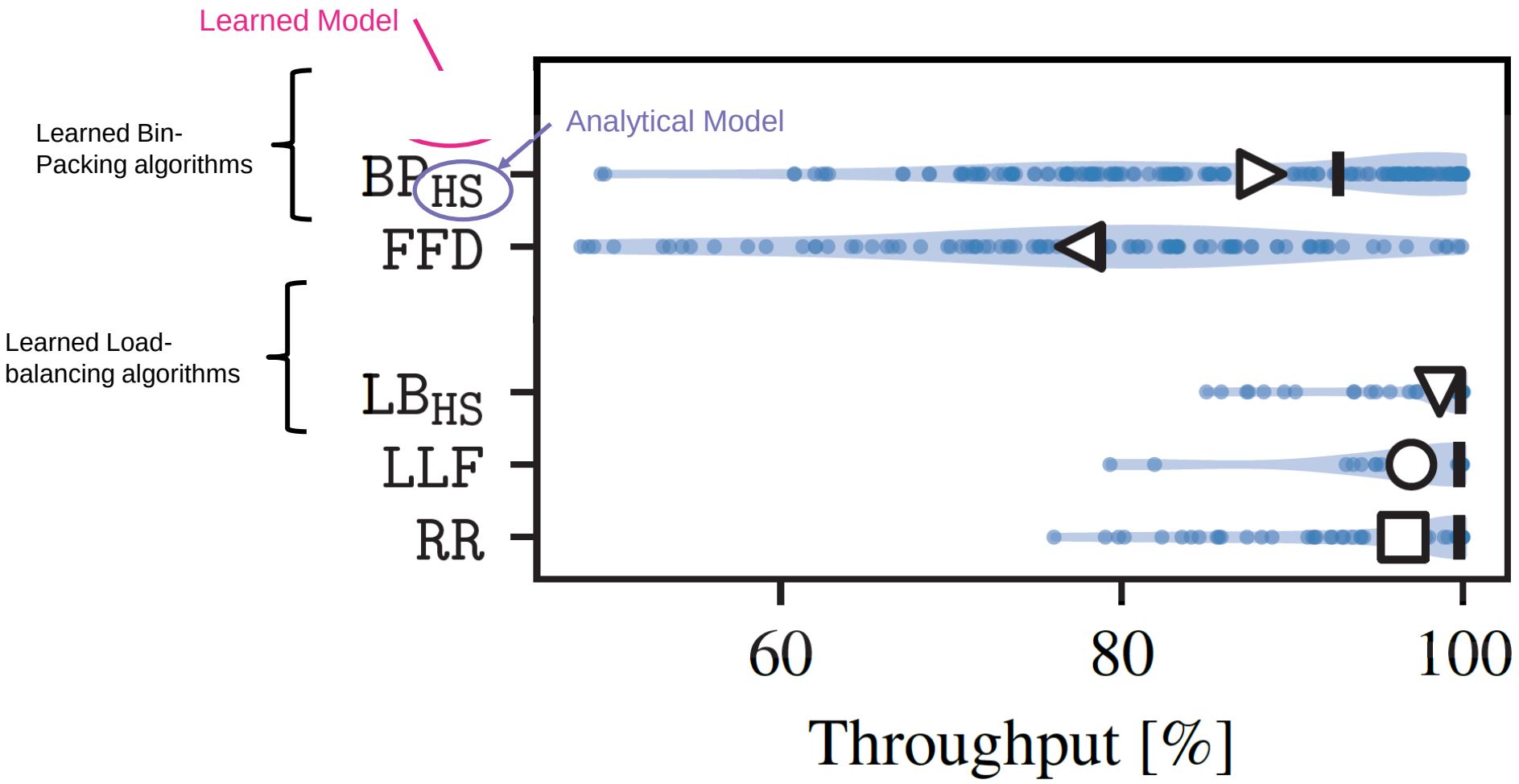
$$\text{Reward} \begin{cases} -10 & \text{If the chosen core is over-utilized after assignment} \\ -1 \dfrac{\text{Player's Load}}{\text{Total Load on Core}} & \text{else.} \end{cases}$$

The Neural Network maps the current state of the game to an action.

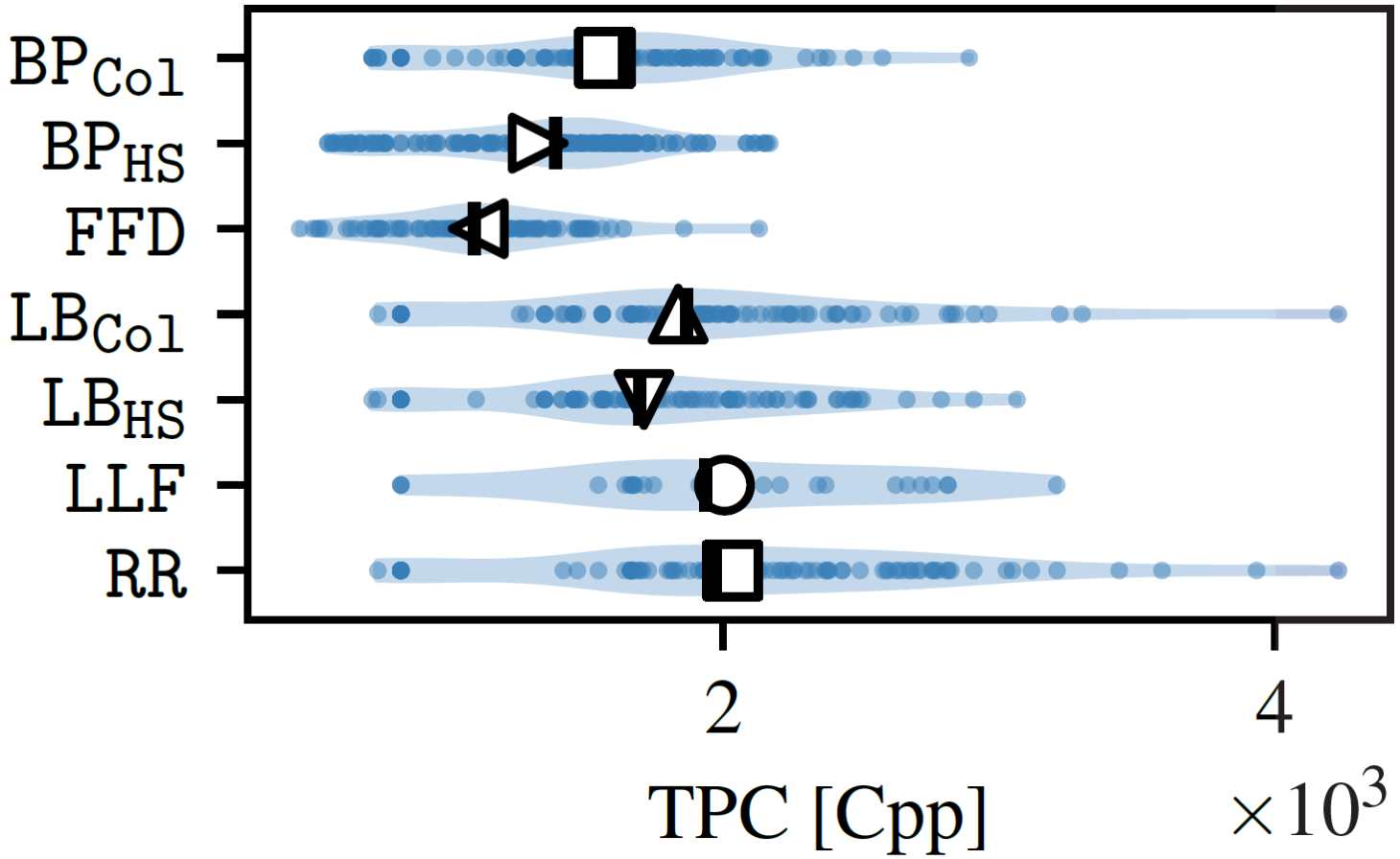


Testbed evaluation.

Bin packing achieves the same throughput as load balancing.



Bin packing needs fewer resources than load balancing.



Summary

- Integrate Neural Combinatorial Optimization and Game Theory

- Learn hard to model system behaviors through ML

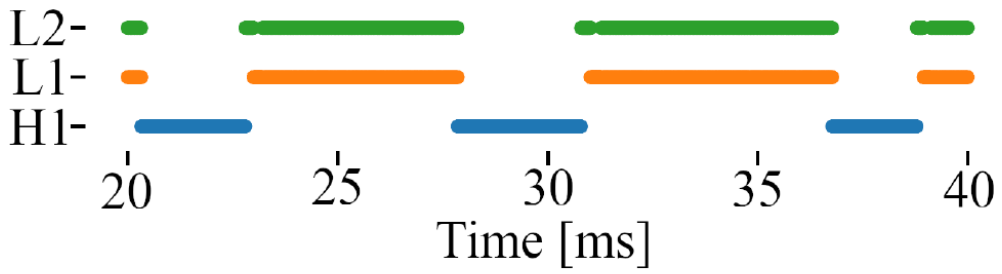
- Integrate the learned behaviors into the training process

→ Framework to automatically tune assignment algorithms to specific workloads

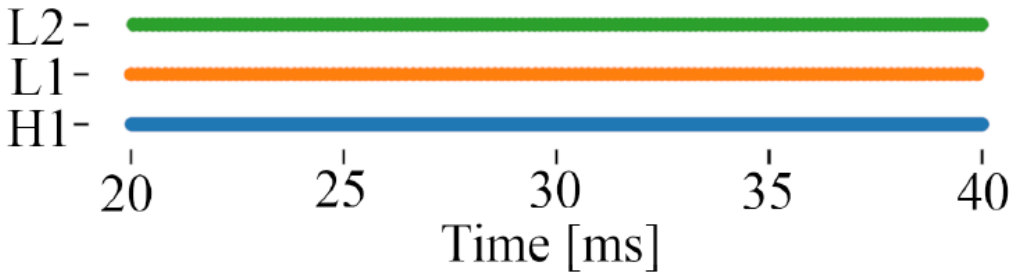
Thank you!

Backup

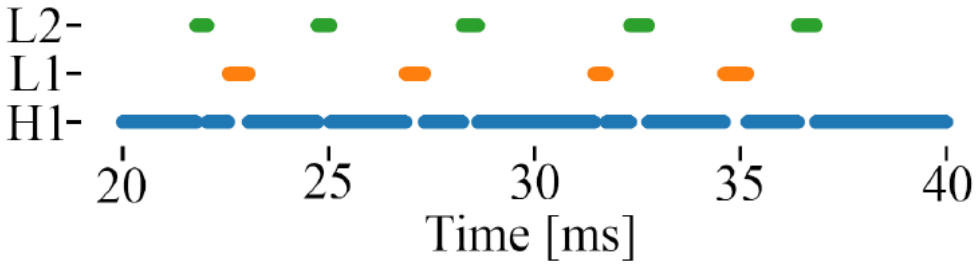
The Completely Fair Scheduler result in a hard to model behavior.



(a) CFS- 1.5 Mpps.



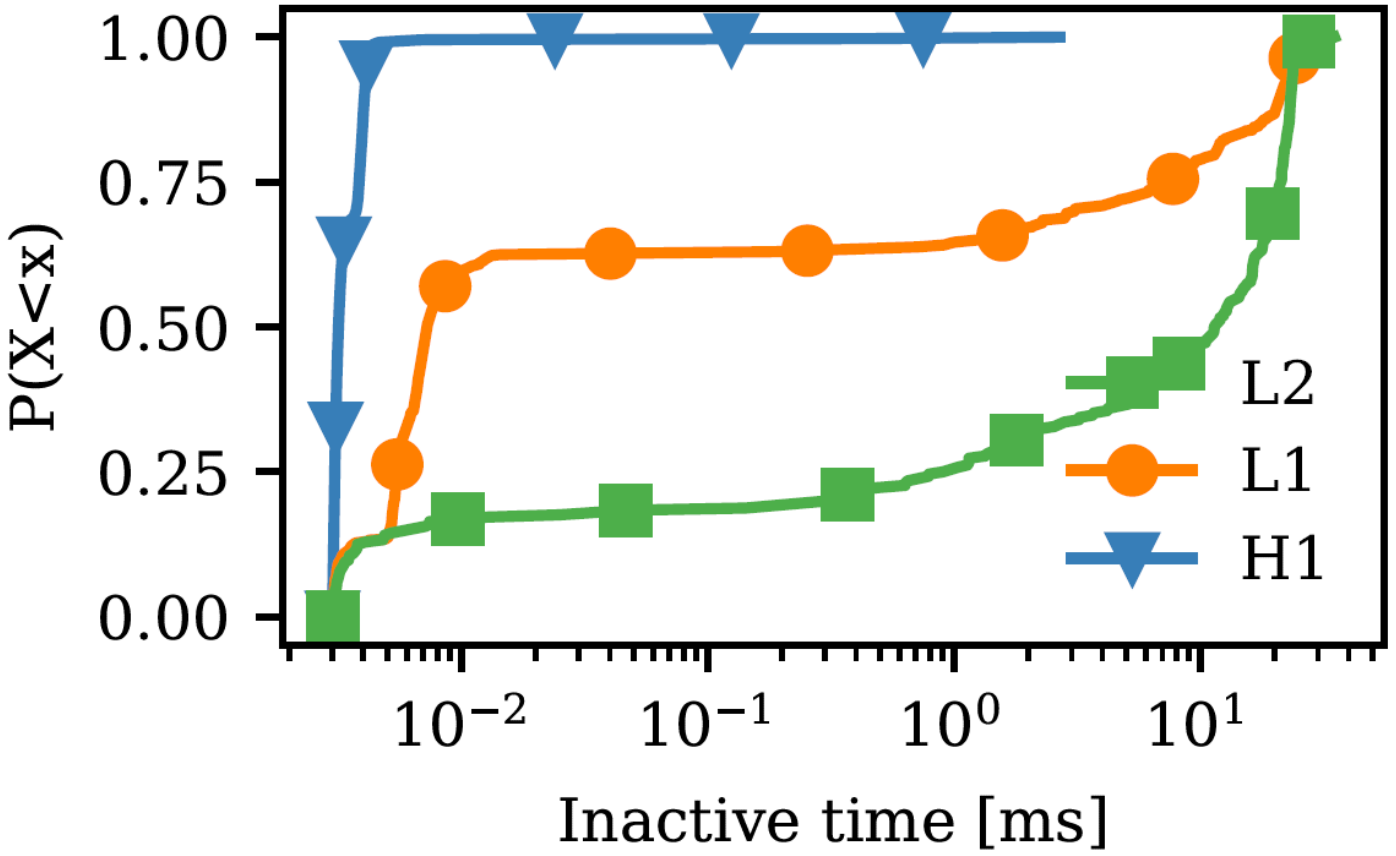
(b) RC- 1.5 Mpps.



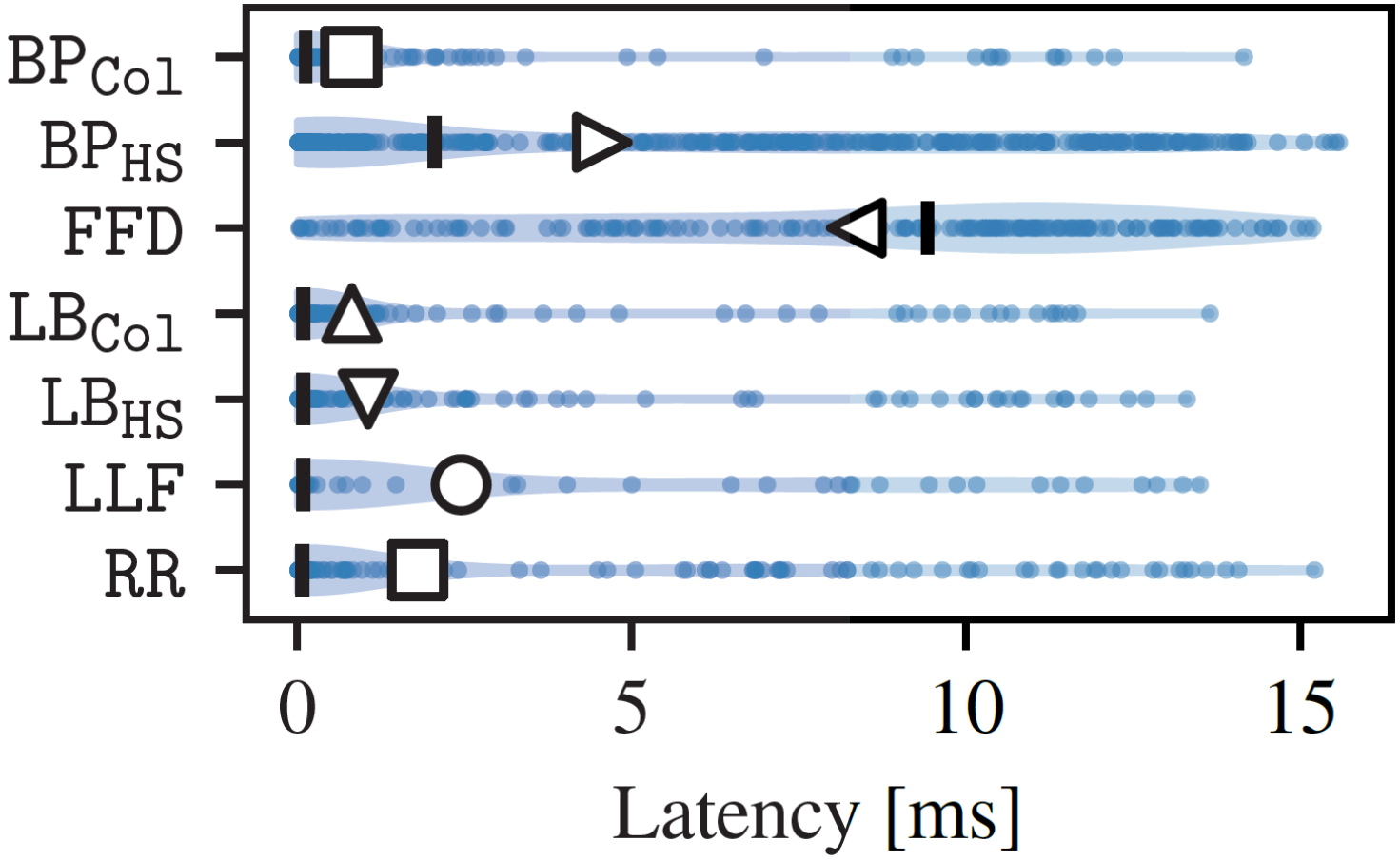
(c) RC- 2 Mpps.

S. G. Kulkarni et al., “NFVnice: Dynamic Backpressure and Scheduling for NFV Service Chains,” in SIGCOMM’17, Los Angeles, CA, USA, 2017, pp. 71–84. doi: 10.1145/3098822.3098828.

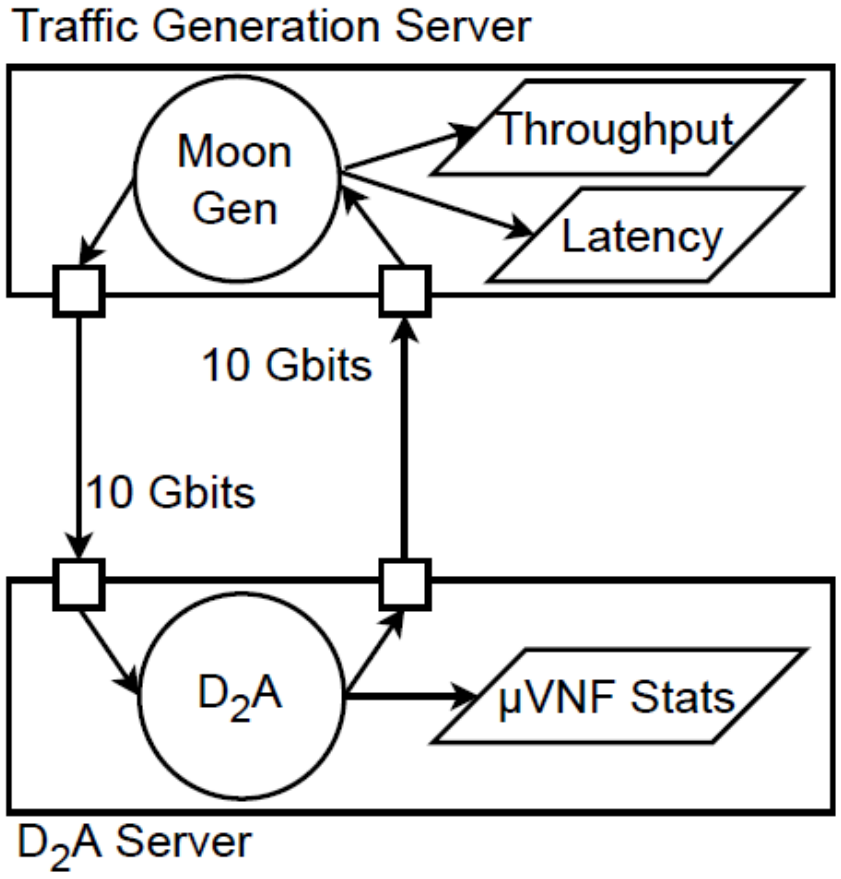
Overload leads to inactivities >20ms.

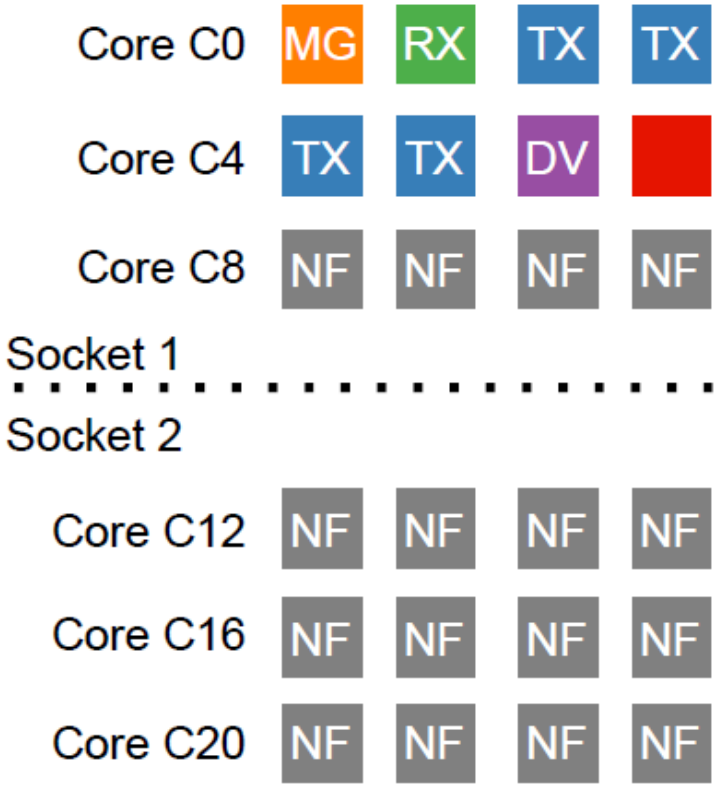


The learned algorithms have the smallest latency.



Testbed setup uses two servers and 16 CPU cores for VNF assignment.

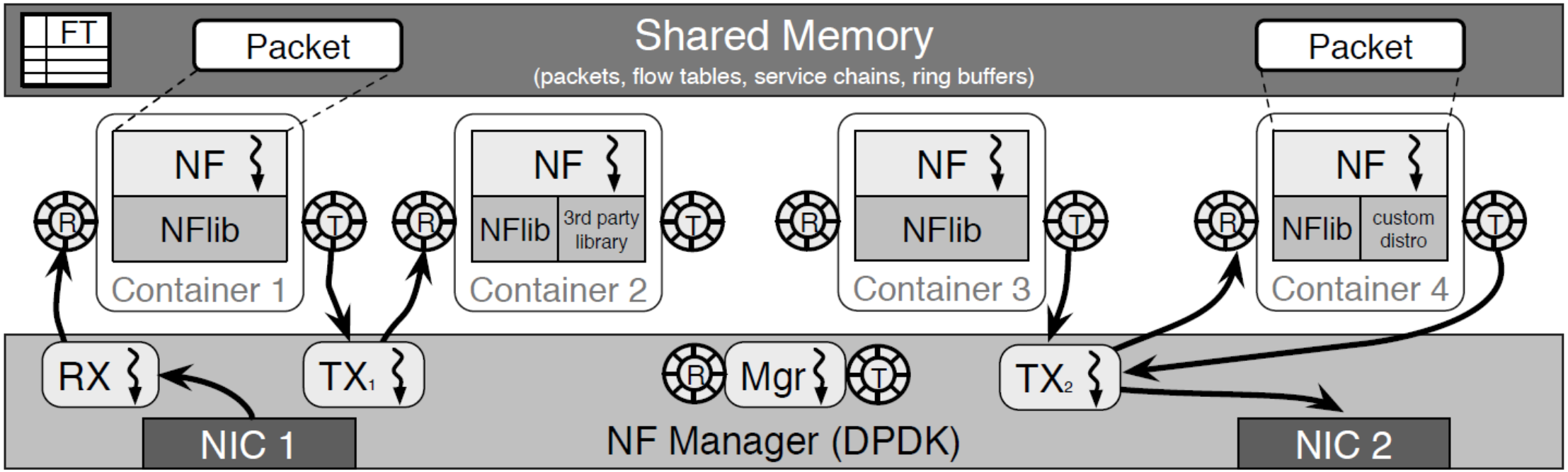




The reward function incentives the distribution of VNFS across cores and includes a penalty for overutilizing.

$$\text{Reward} \begin{cases} -10 & \text{If the chosen core is over-utilized after assignment} \\ -1 \dfrac{\text{Load on most loaded CPU}}{\text{Capacity of CPU}} & \text{else.} \end{cases}$$

OpenNetVM is a mature Network Function platform.



Generating challenging problems.

We generate problem instances that are neither trivial nor impossible to solve

$\#SFCs \sim U(\{1, \ldots, 8\})$

$\lambda_{SFC_1}, \ldots, \lambda_{SFC_{\#SFCs}} \sim \lambda \cdot Dir(\mathbf{1}_{\#SFCs} \cdot 5)$

```
while cpu available and free sfc exists \
        and sampled_vnfs < max_num_vnfs do
    sfc ← randomly_sampled_from_available_SFCs()
    compute ← randomly_sample_compute()
    do
        cpu = find_next_free_cpu(compute, λ_sfc)
        if not cpu:
            compute = reduce_compute(compute)
    while not cpu and is_reducible(compute);

    if cpu:
        add_vnf_to_sfc(sfc, compute)
        add_vnf_to_cpu(cpu, λ_sfc, compute)
    else:
        break
```