Elastic RSS

Co-Scheduling Packets and Cores Using Programmable NICs

Alexander Rucker
Tushar Swamy, Muhammad Shahbaz, and Kunle Olukotun
Stanford University

August 17, 2019

How do we meet

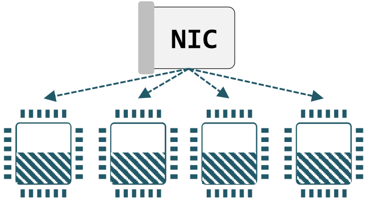
tail latency

constraints?

Existing systems have several limitations.

Random Hashing

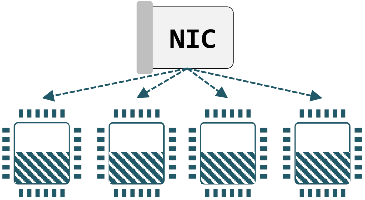
- Load imbalance
- Over provisioned



Existing systems have several limitations.

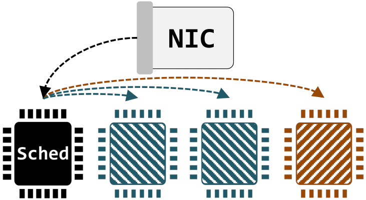
Random Hashing

- Load imbalance
- Over provisioned



Centralized Scheduling

- Dedicated core
- Limited throughput

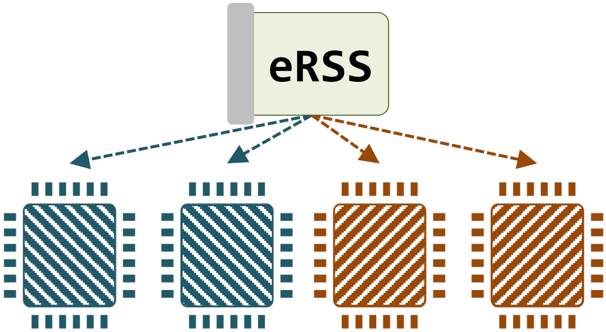


How do we

scalably & **CPU-efficiently**

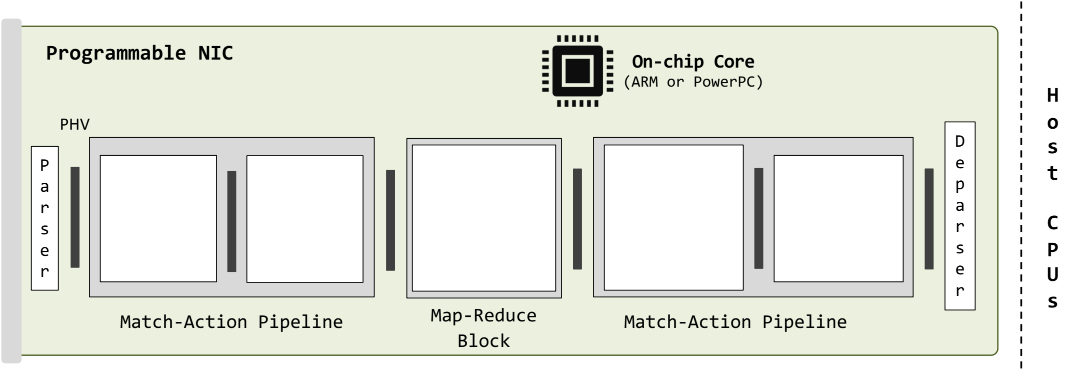
meet tail latency constraints?

eRSS uses all cores for useful work and runs at line rate.

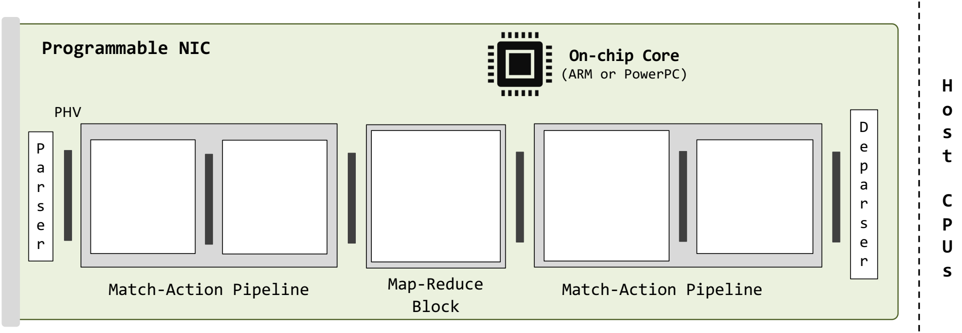


Design

eRSS's packet processing maps to a PISA NIC with map-reduce extensions.

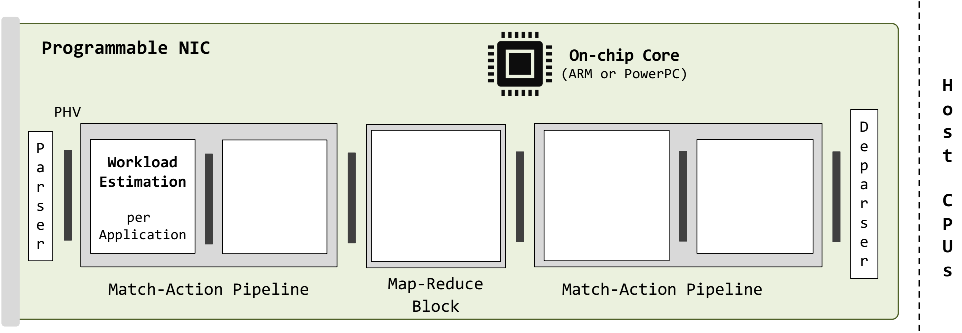


1. Assign each packet to an application.



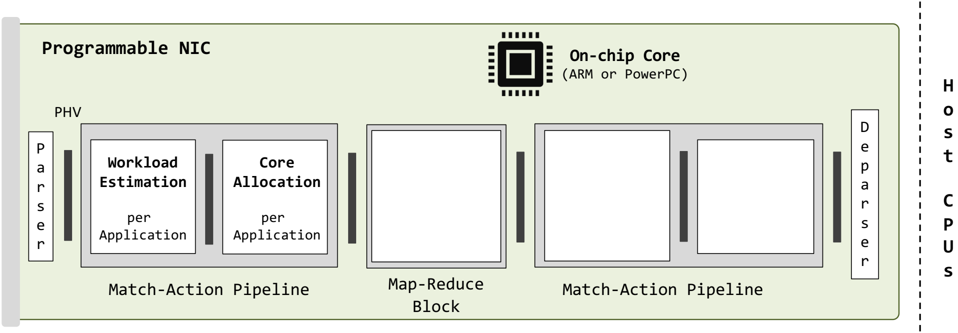
- For example, use **IP address** or **port** number.

2. Estimate the per-packet workload.



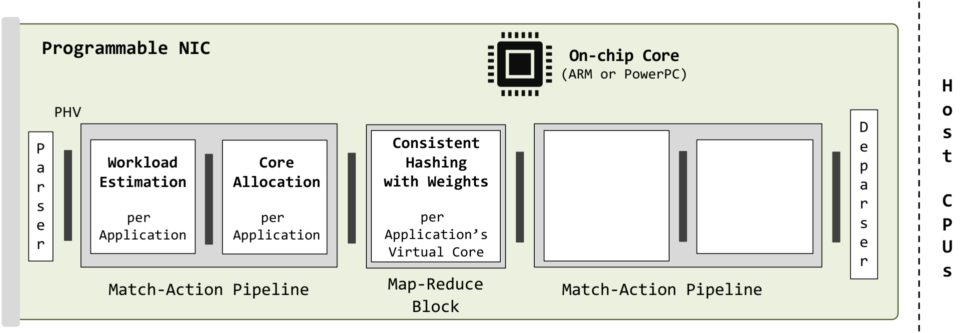
- Can use any set of packet header fields (currently, only packet size).
- Model is periodically **trained by the CPU.**

3. Determine core count for the application.



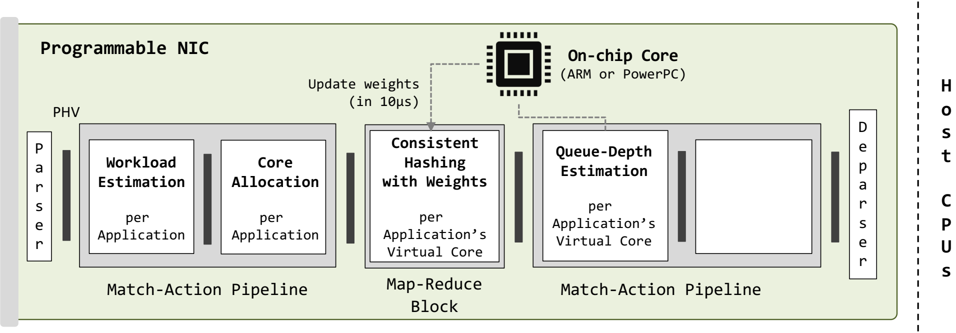
- Compare allocated cores to exponential moving **average of workload.**
- Use **heuristics and hysteresis** to avoid ringing.

4. Select a virtual core.



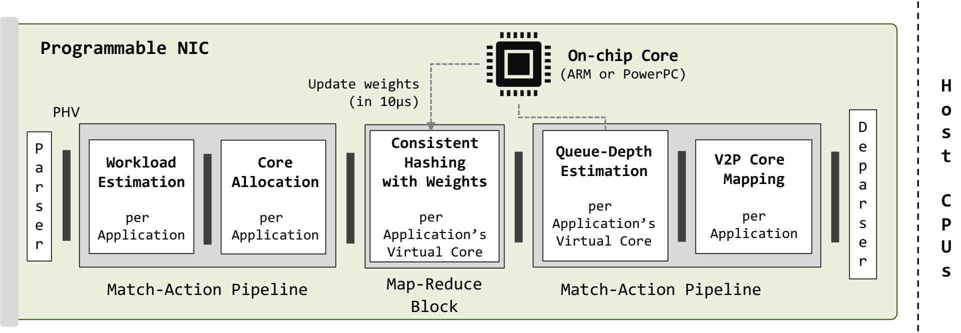
- Virtual cores within each application are allocated densely, starting at 0.
- Packets are hashed & the **best allocated core** is chosen.

5. Estimate queue depths.



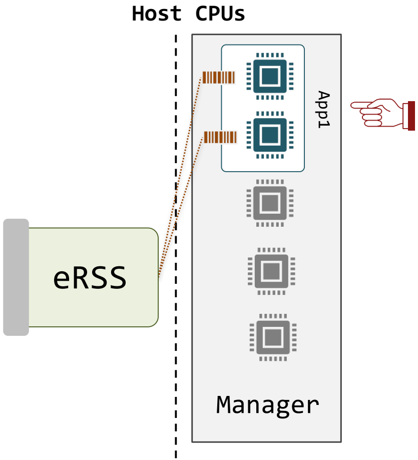
- Queues are **estimated** per-virtual core.
- Estimates are used to **adjust** consistent hashing weights.

6. Map the virtual core to a physical core.



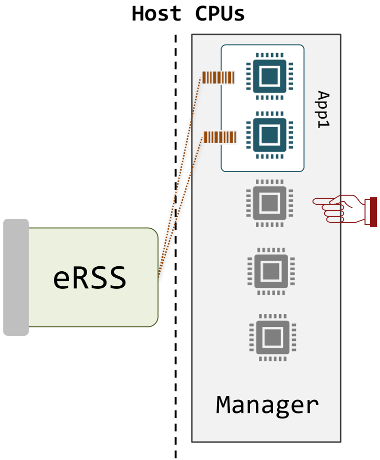
- CPU assigns each physical core to an application as an active/slack core.
- Look up ⟨**Application, Virtual Core**⟩ → **Physical Core** in match-action table.

1. An application needs additional headroom.

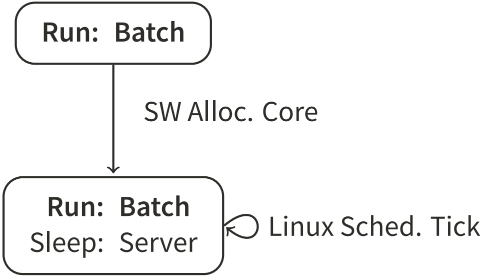


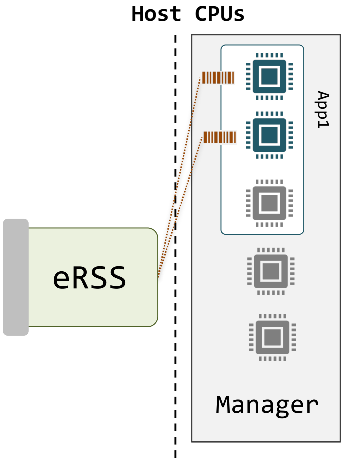
2. The core is initially running a batch job.

Run: Batch

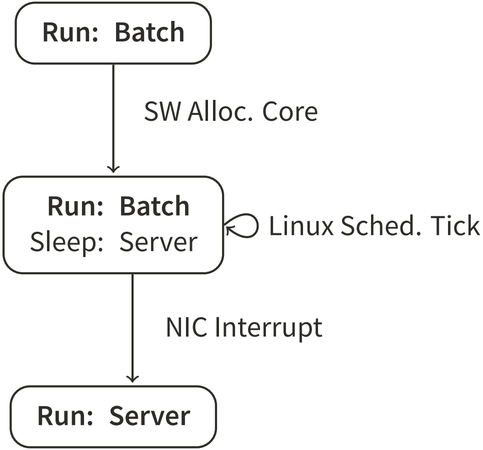


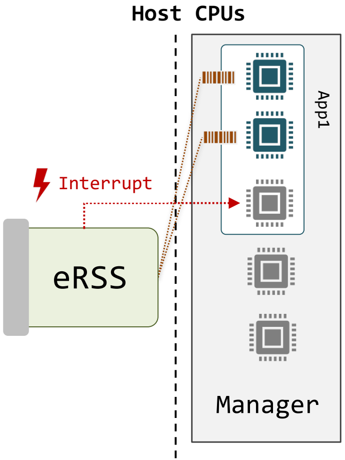
3. The software manager starts and pins a sleeping thread to the core.



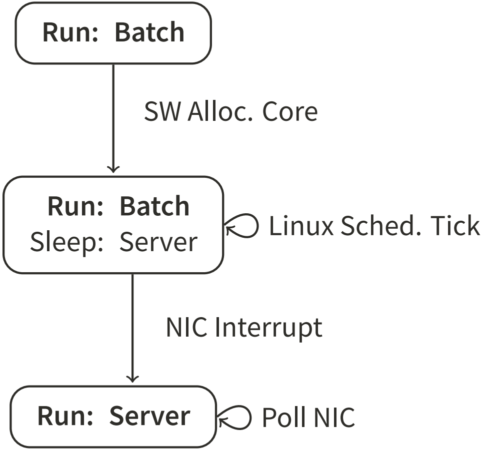


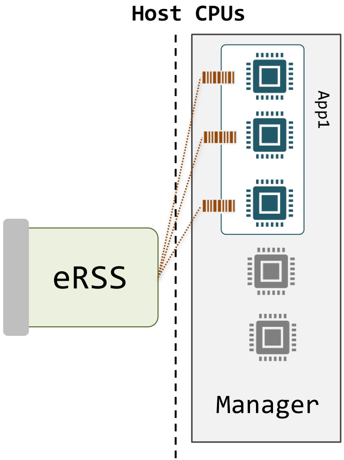
4. When the NIC allocates a core, it wakes up the resident thread.



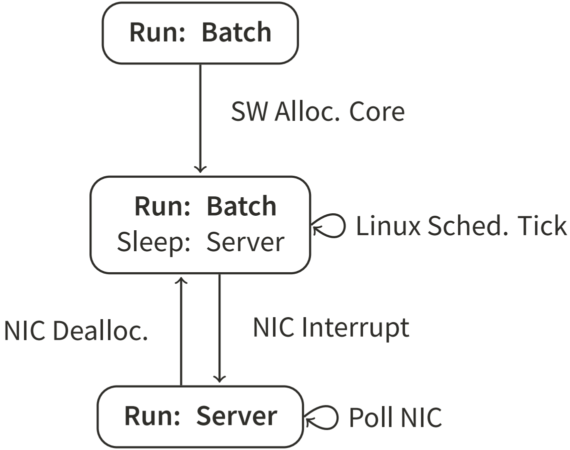


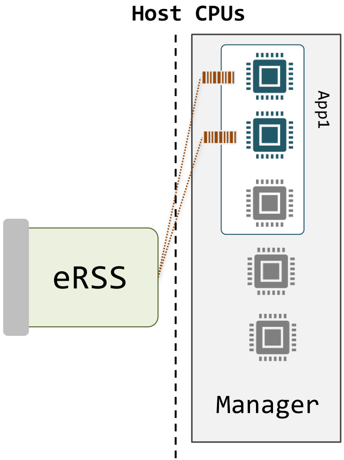
5. Cores can run any server software, incl. distributed work stealing or preemption.





6. Upon deallocation, the packet thread sleeps and the OS schedules a batch job.



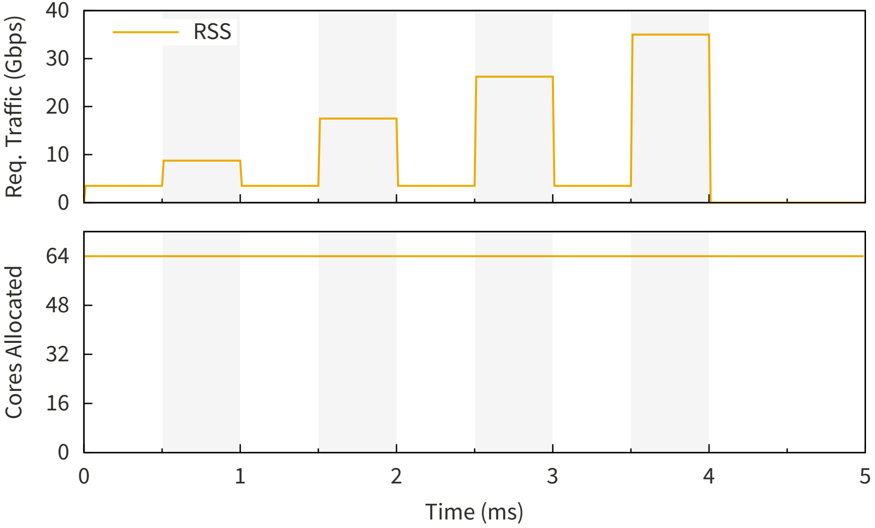


Preliminary Evaluation

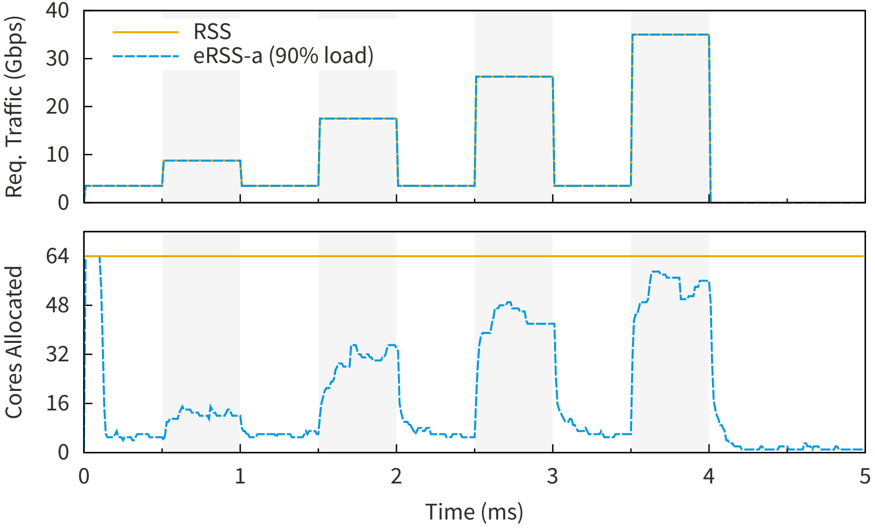
We simulate eRSS's performance on a synthetic model.

- Packets have **Poisson-distributed** inter-arrival times.
- Packet sizes are representative of **Internet traffic.**
- Packet processing time correlates with size and added noise.

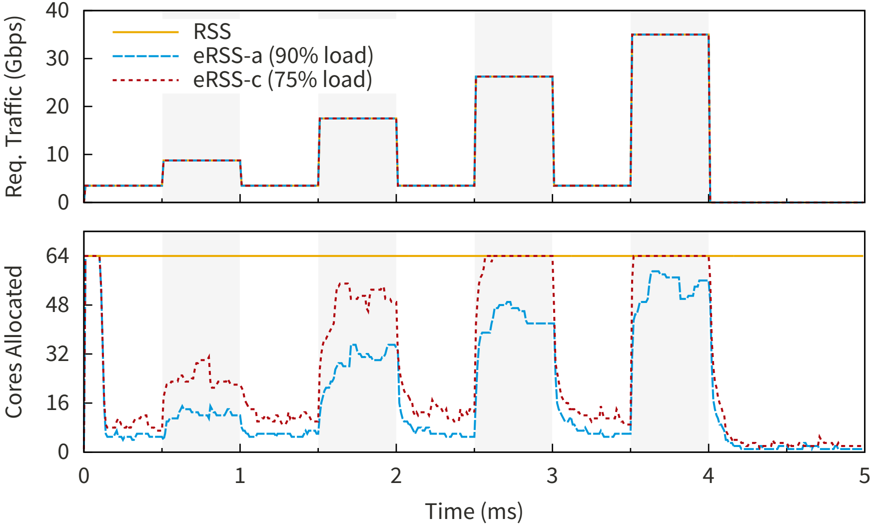
eRSS responds quickly to load variations.



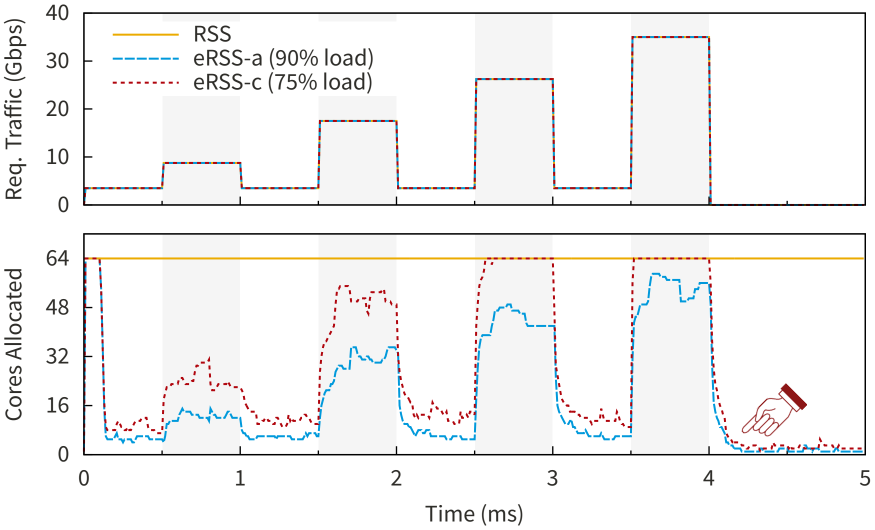
eRSS responds quickly to load variations.



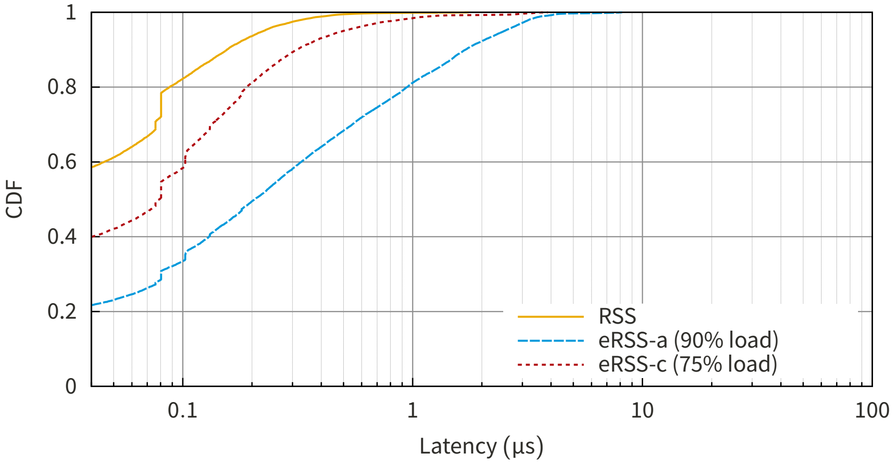
eRSS responds quickly to load variations.



eRSS deallocates slowly to ensure queues are drained.



eRSS adds controllable tail latency.



SLO

Future Work & Summary

eRSS will be extended with ML.

- Workload estimation
  - Efficient core scheduling requires **accurate workload estimates.**
  - Use packet **header** fields and deep **packet inspection** to gather statistics.

eRSS will be extended with ML.

- Workload estimation
  - Efficient core scheduling requires **accurate workload estimates.**
  - Use packet **header** fields and deep **packet inspection** to gather statistics.
- Core scheduling with **Reinforcement Learning (RL)**
  - Replace heuristics for **adding/removing** cores to an application.
  - Replace consistent hashing for **distributing packets** between cores.

eRSS meets tail latency constraints while saving cores.

- Parameters control **trade-off** between core use and tail latency.
- eRSS runs at **line rate** using slight extensions to existing NICs.
- eRSS is **compatible** with a variety of software solutions.
- eRSS can be extended with **ML** for automatic operation.

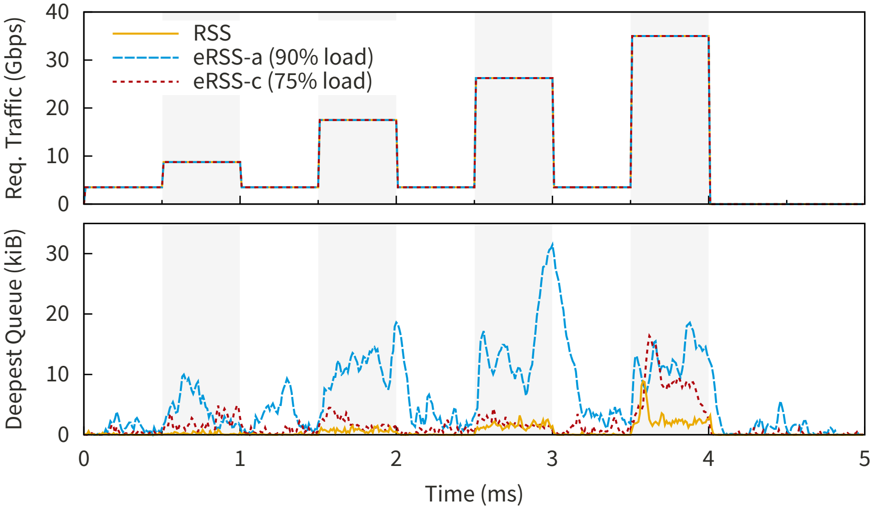
eRSS

scalably & **CPU-efficiently**

meets tail latency constraints.

Questions?

eRSS adds a controllable amount of additional queue depth.



eRSS minimizes breaking flows.