NetVRM: Virtual Register Memory for Programmable Networks

Hang Zhu

Tao Wang, Yi Hong, Dan R. K. Ports, Anirudh Sivaraman, Xin Jin

Johns Hopkins University · NYU · Microsoft Research · Peking University

Data plane objects

Stateless (lifespan <= 1 packet)

- Metadata, packet headers

Stateful (lifespan > 1 packet)

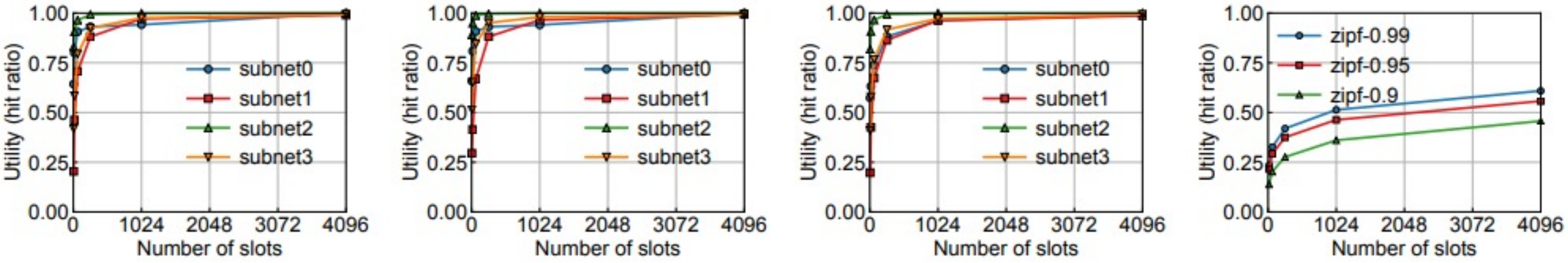
- Tables, counters, meters, registers

Registers enable a new class of **reg-stateful applications**

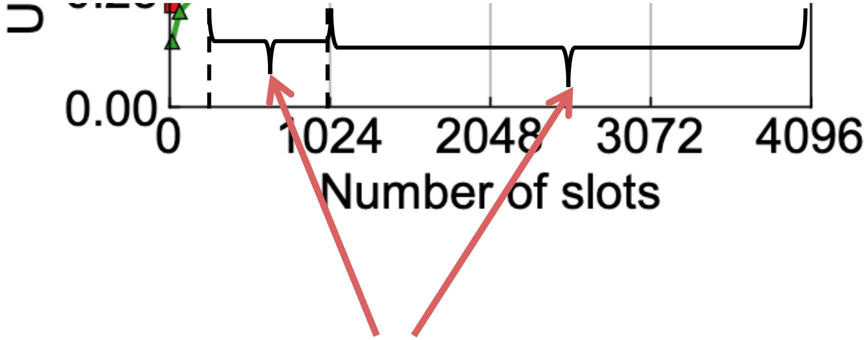
The case of dynamic allocation for register memory

- **Necessity**
  - Limited register memory (e.g., a few Mb/stage)
  - Concurrent reg-stateful applications
- **Potential benefits**
  - Diminishing return

Diminishing return



(a) Heavy hitter detection. (b) Newly opened TCP connections. (c) Superspreader detection. (d) NetCache.



- 0.28->0.36: 768 register slots
- 0.36->0.46: 3072 register slots

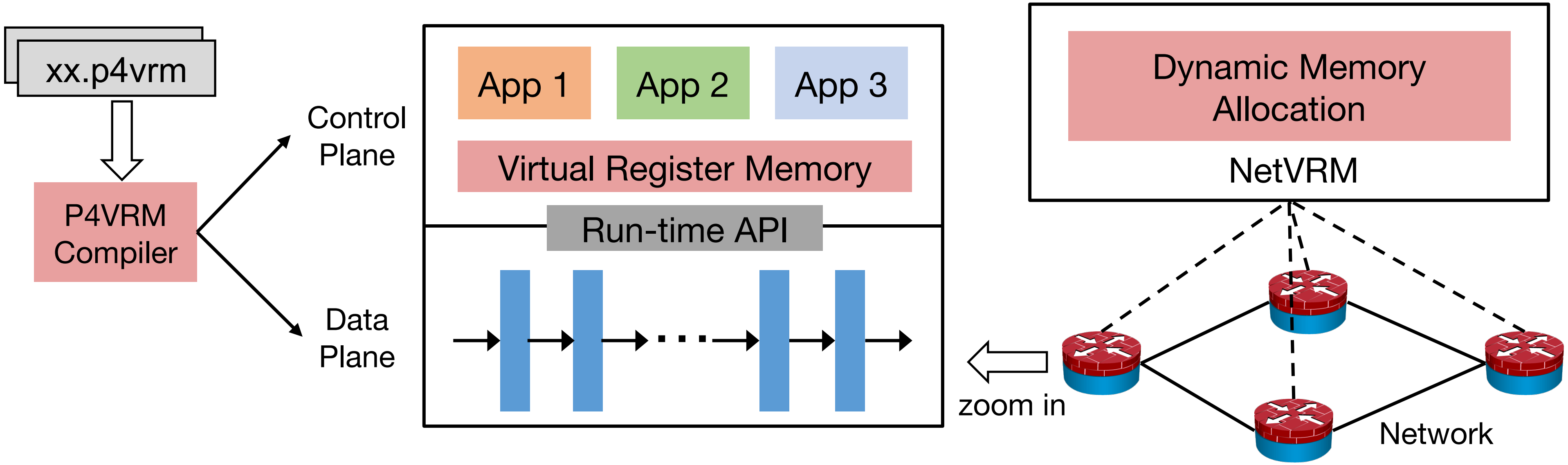
Existing solutions and limitations

Static binding of register memory	Ignore hardware constraints	No network-wide dynamic allocation
➢ Merged in compilation time ➢ P4Visor [CoNEXT'18]	➢ DPDK, BMv2 ➢ Hyper4 [CoNEXT'16]	➢ Allocation in a single switch

Realizing dynamic register memory allocation

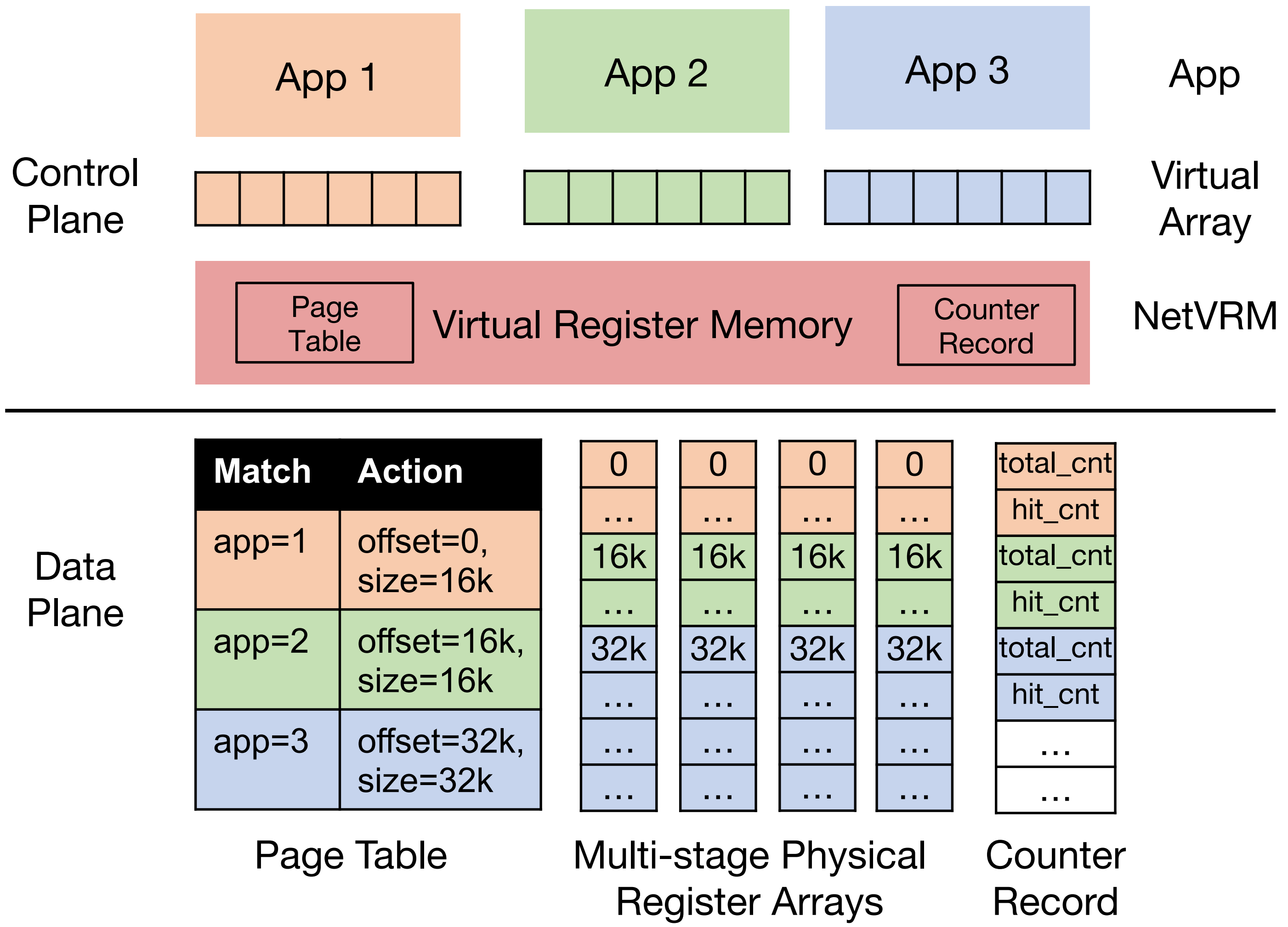
How to enable online allocation	→ Virtual register memory
How to modify P4 programs	→ P4VRM compiler
How to maximize multiplexing benefits	→ Dynamic allocation

NetVRM architecture



Virtual register memory

- ➢Page tables
- ➢Counter record



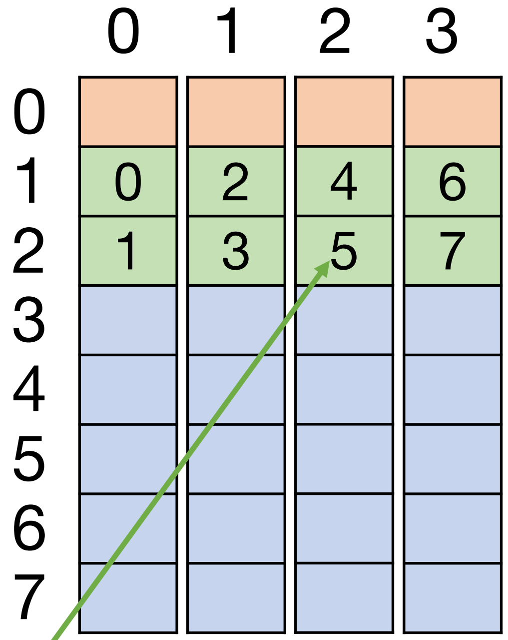
Address translation

Translation formula

$$PA = \left(\frac{VA}{size}, VA\%size + offset\right)$$

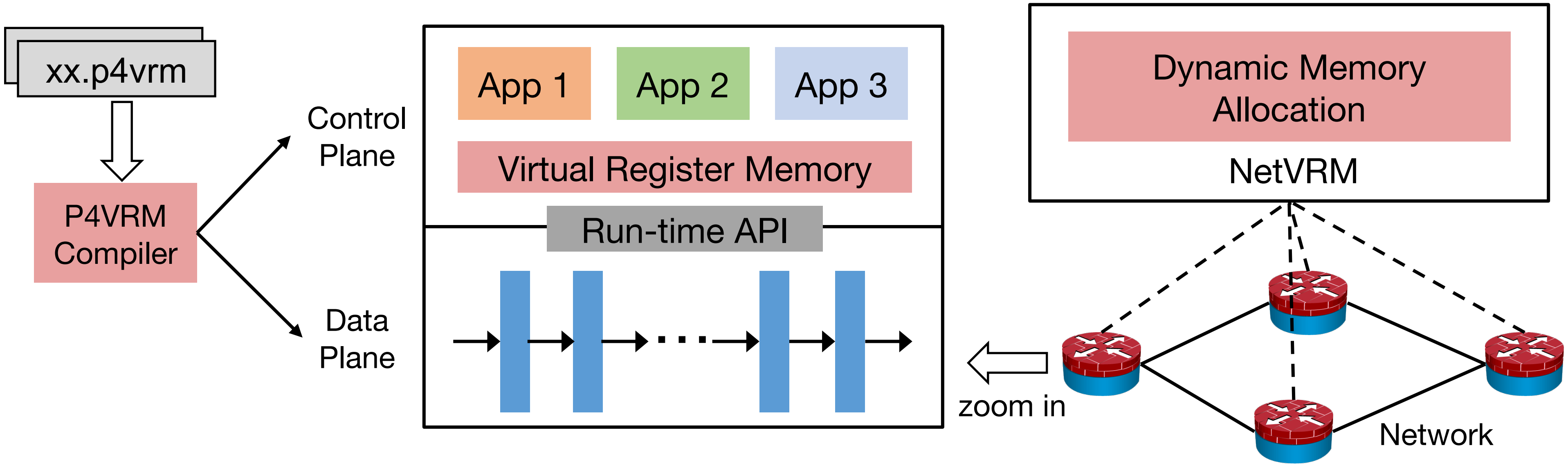
physical array index

physical slot index



$$size = 2, offset = 1$$
$$VA = 5 \rightarrow PA = (2, 2)$$

NetVRM architecture



Problem formulation

$$\max \sum_{i=1}^{l} \mathbf{1}(i.utility(i.m_1, \ldots, i.m_c, i.T) \geq i.target)$$

$$\text{s.t.} \sum_{i=1}^{l} i.m_j \leq M_j, \forall j = 1, \ldots, c$$

Objective

- Maximize number of applications with satisfied utility target

Constraints

- Register memory constraints on each switch

Scope of dynamic resource allocation

Elastic applications

- Work with a variable amount of register memory
- Overcome insufficient register memory with a fallback mechanism

Inelastic applications

- Require a fixed amount of register memory
- Cannot work with less

benefit

support

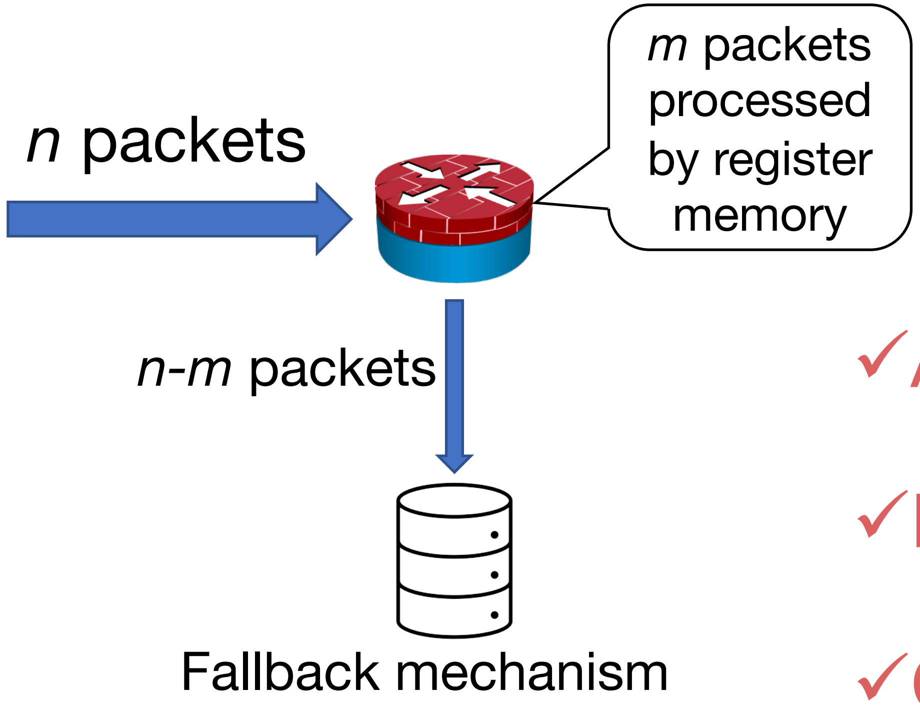
Challenges for dynamic resource allocation

Definition of application utility → Memory hit ratio by default

Unknown and dynamic utility functions

Multiple paths of an application

Memory hit ratio by default



$$hit\ ratio = \frac{m}{n}$$

✓Application-agnostic

✓Reflect application-level performance

✓Computed online

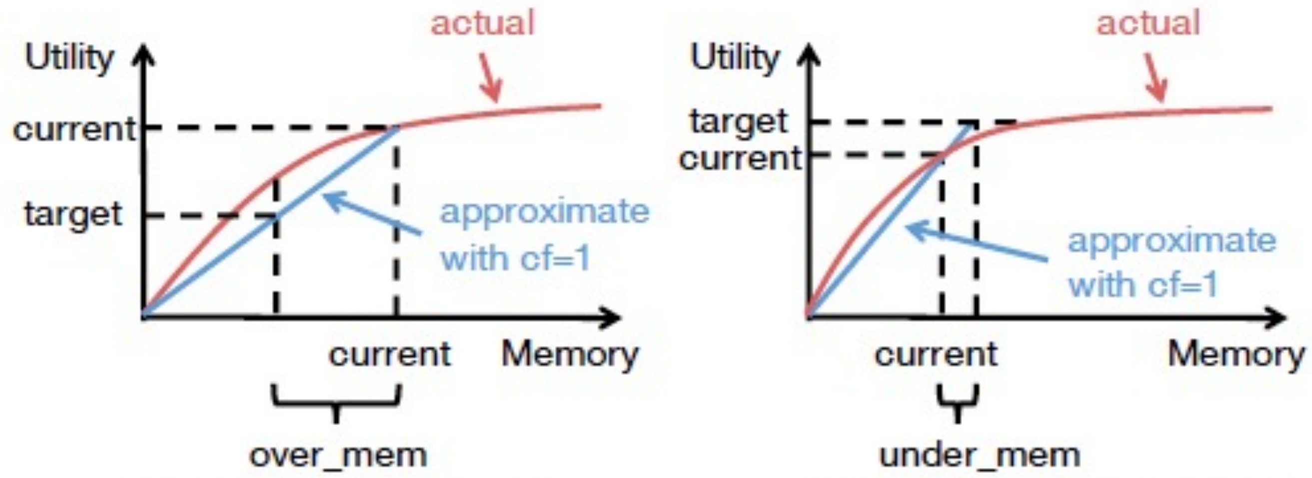
Challenges for dynamic resource allocation

Definition of application utility → Memory hit ratio by default

Unknown and dynamic utility functions → Online utility curve estimation

Multiple paths of an application

Online utility curve estimation



(a) Estimate over_mem.

$$i.over\_mem \leftarrow i.mem - \left(\frac{i.target}{i.util}\right)^{cf} * i.mem$$

(b) Estimate under_mem.

$$i.under\_mem \leftarrow \left(\frac{i.target}{i.util}\right)^{cf} * i.mem - i.mem$$

cf: compensate for the diminishing return

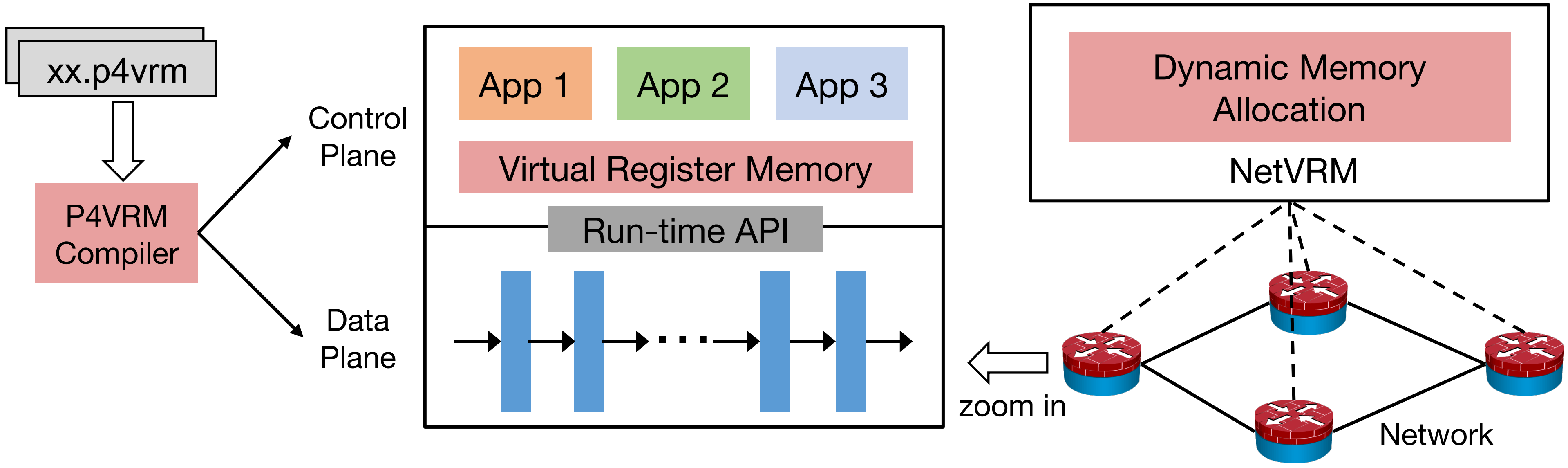
Dynamic resource allocation

Definition of application utility → Memory hit ratio by default

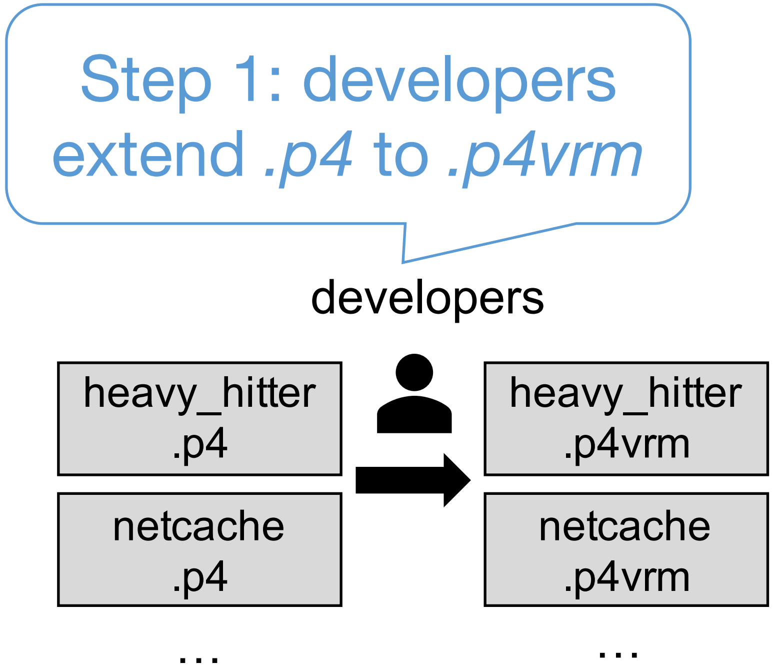
- Step 1: decompose *over_mem* and *under_mem*
- Step 2: *over_mem -> under_mem*

Multiple paths of an application → Network-wide register memory allocation

NetVRM architecture

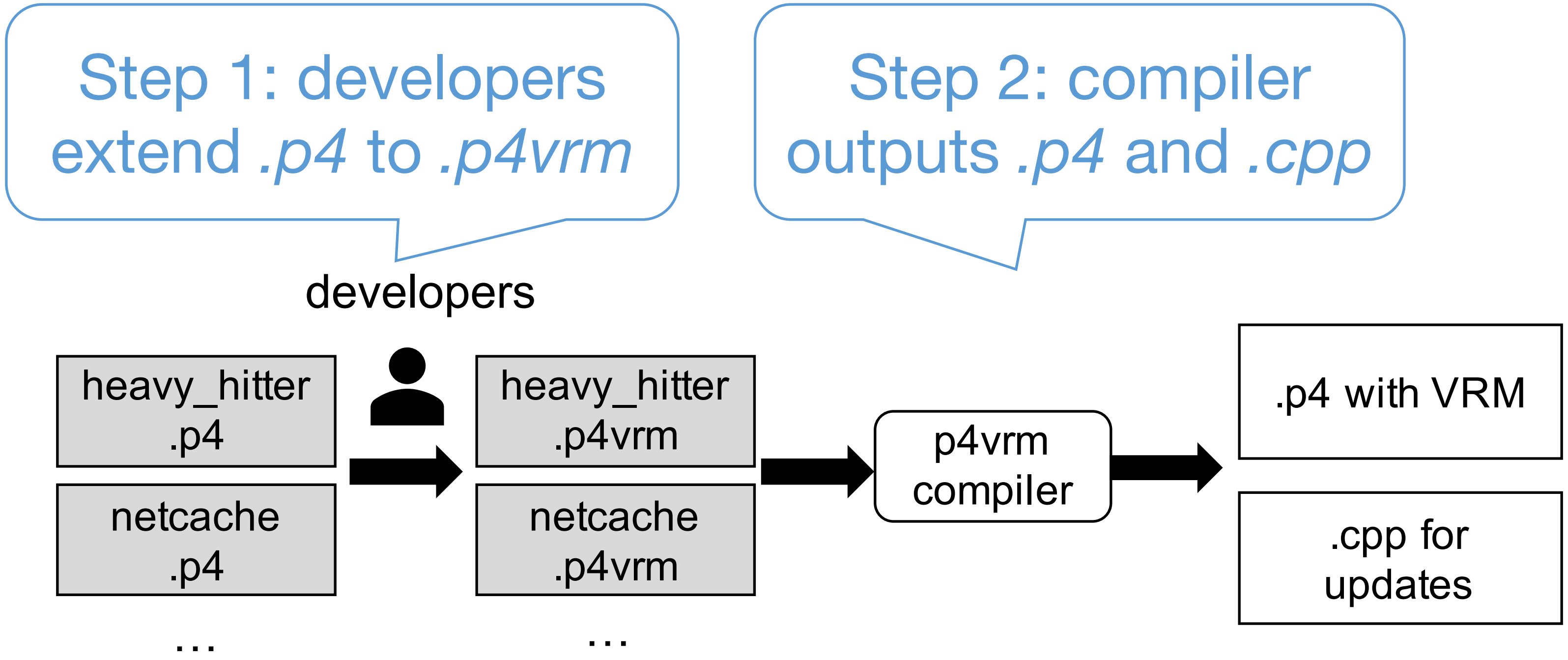


P4VRM compiler



➢ Mark the register arrays and related declarations as virtualized

P4VRM compiler



Implementation

- 6.5 Tbps Intel Tofino switch
- Four emulated switches with four independent pipelines
- P4VRM compiler
  - built on Flex/Bison

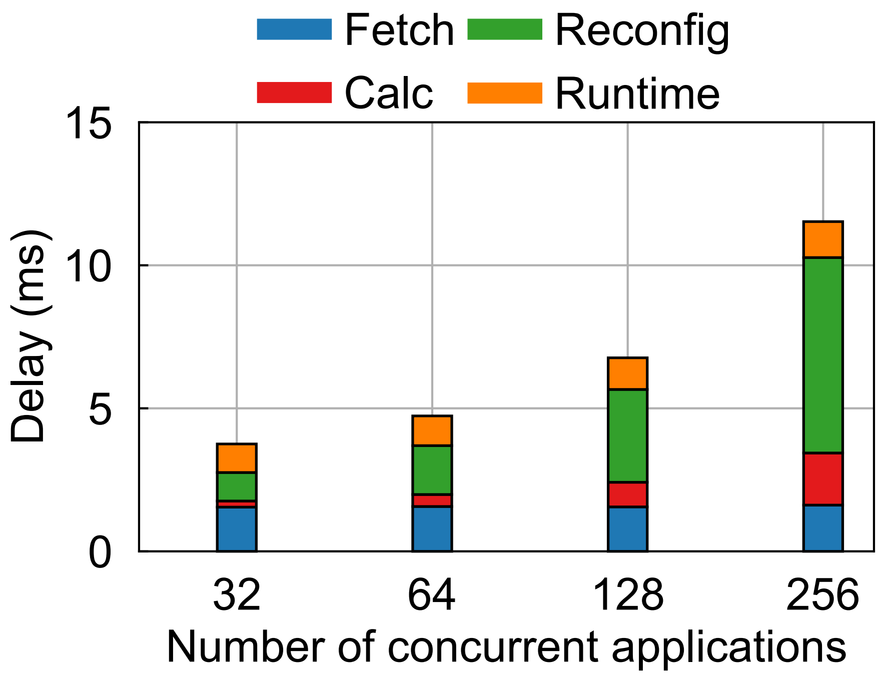
Evaluation

- Microbenchmark
  - Control loop delay
  - Stability and fast convergence of NetVRM
- Macrobenchmark
  - Generality
  - Impact of allocation epochs
  - Impact of workload parameters
  - NetVRM in datacenter network

Evaluation

- Microbenchmark
  - Control loop delay
  - Stability and fast convergence of NetVRM
- Macrobenchmark
  - Generality
  - Impact of allocation epochs
  - Impact of workload parameters
  - NetVRM in datacenter network

Control loop delay

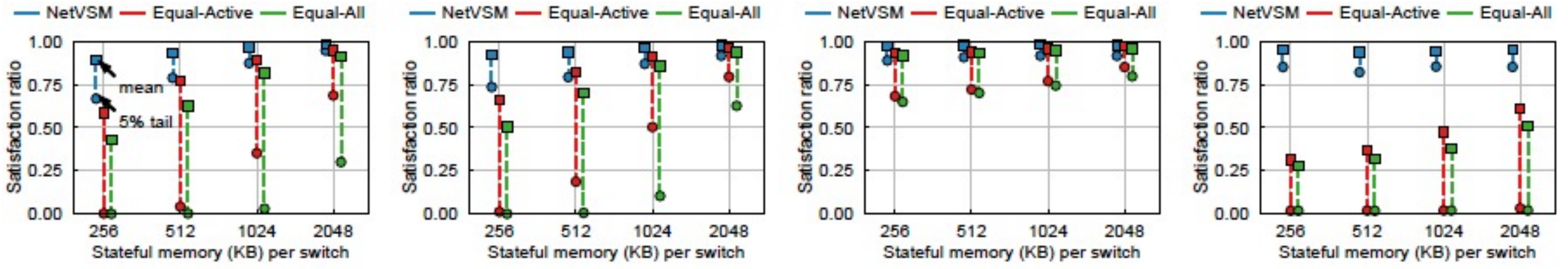


✓ **One reallocation can be done in ~10 ms**

✓ ***Reconfig* dominates the control loop**

Generality

- Each application has traffic from four switches independently
- Satisfaction ratio as the performance metric
- Alternatives: Equal-All, Equal-Active

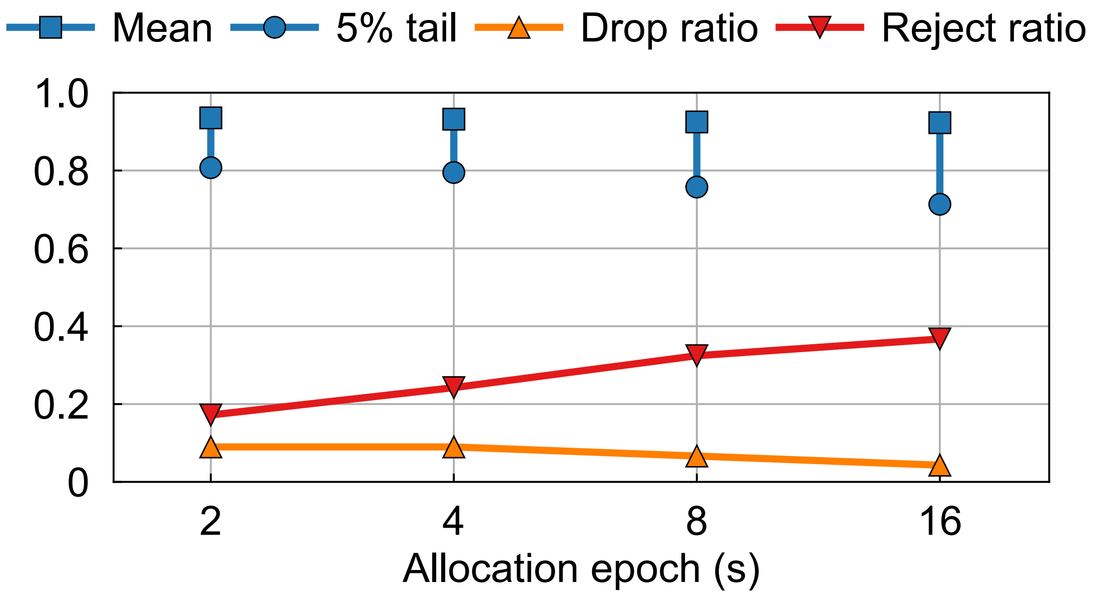


(a) Heavy hitter detection (HH).
(b) Newly opened TCP (NO).
(c) Superspreader detection (SS).
(d) Sketch-based heavy hitter detection (SHH).

✓ **NetVRM outperforms alternatives on both the mean and the tail**

✓ **NetVRM is general to different network application types**

Impact of allocation epochs



✓ **A shorter allocation epoch leads to a better performance**

Conclusion

- NetVRM supports dynamic register memory sharing between multiple concurrent applications on a programmable network
  - Virtual register memory: enable online register memory sharing
  - Dynamically allocate memory for better resource efficiency
  - P4VRM: easily equip the programs with virtual register memory

Thank you!

E-mail address: hzhu@jhu.edu