Towards In-Switch Reinforcement Learning

Kyle A. Simpson
k.simpson.1@research.gla.ac.uk
FelixMcFelix https://mcfelix.me

2nd December, 2020

University of Glasgow

Data-driven networking: Automate control, optimisation, configuration of the network.

- Flow performance optimisation.
- Resource allocation.
- Adaptive response to load, intrusions, etc.
- Feedback loop-like.

Why programmable data-planes?

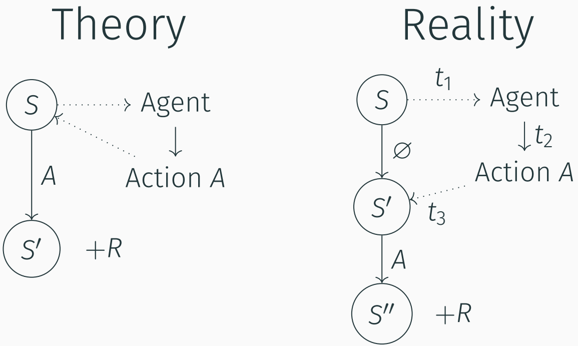
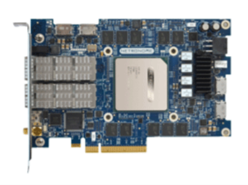
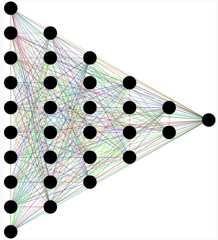


Figure 1: *Asynchronous RL delays and state slippage (policy updates omitted).*

- In data-driven, want to minimise time to act.
- RL assumes that action & policy update are zero-cost.
  - Not so in real deployments!
  - State drift, etc.
- Controller contact time, serialisation, …
- In other ML, often need line rate inference.
- Programmable network hardware fills this niche.

Recent Programmable Trends in Data-Driven Networks





- ML acceleration, line-rate packet classification.
- How? Train model off-NIC, convert to binary neural network[1], or decision tree[2].
- Limits? No online training, cost of backprop algo (expensive!), vast data needs.
- What if we need online learning?

[1] Siracusano *et al.*, 'Running Neural Networks on the NIC'.
[2] Xiong and Zilberman, 'Do Switches Dream of Machine Learning?: Toward In-Network Classification'.

Case study: DDoS Prevention

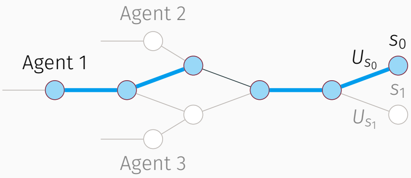


Figure 2: *'Global' state for any flow (**bold** means used by policy).*

- Prior work[3].
- Monitor flow statistics at ingress/egress points.
- Use load measurements from flow paths as global state.
- OUTPUT: throttle individual flows.
- Classical RL.
  - Learning time on order of minutes.

[3] Simpson, Rogers and Pezaros, 'Per-Host DDoS Mitigation by Direct-Control Reinforcement Learning'.

Case Study: DDoS Prevention (Architecture)

- Reward measurements come from network.
- Input state mixes local flow data with global load data.
- Flow measurements combined, decisions scheduled to prevent overload.

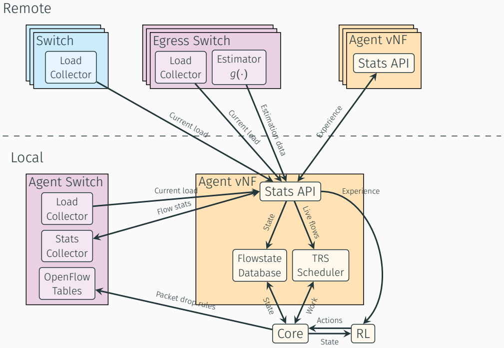


Figure 3: *(Non-PDP) system architecture for RL-driven DDoS defence.*

Goals… and Reality

Goal	Solution	Drawback
• On-NIC action compute	Quantisation	Platform Model
• On-NIC learning	Classical RL	Info capacity
• Direct action installation	Custom actions	No intrinsic support
• (Partly) External State, Rewards, and Task Independence	Independent Module	—
• Runtime reconfiguration	P4 Parser + RTE	—

Device classes

SmartNICs (e.g., Netronome)

- Low port density.
- Easy to have asynchrony:
- SOC-based → extra cores off the datapath.
- NetFPGA → can define extra functional units and interconnect.

We'll mainly focus on SmartNICs.

Programmable Switches (e.g., Tofino)

- High port density.
- No additional compute units. Very close to P4 PSA.
- But still more powerful–e.g., Tofino supports MUL operations.
- Might be doable... deadline-aware.

Architecting On-NIC RL

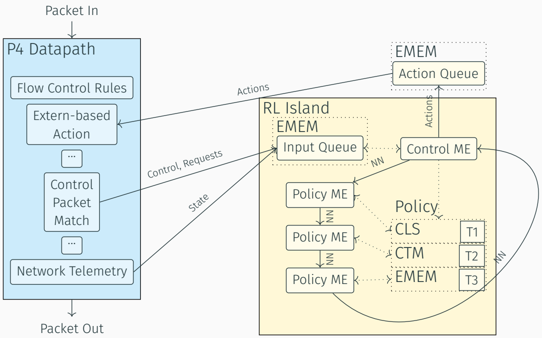


Figure 4: *Architecture for generalised RL agent on Netronome hardware.*

- Classical RL built on tile-coding—online.
- Async wrt. datapath.
- Dynamic selection of last reward, trace info.
- Runtime configurable (policy, size, application) from data/control-plane.
- Task independent.

How could this (hypothetically) fit the case study?

- P4 table-action to gather state on collected flows.
  - Flow telemetry in P4 well-documented.
  - Pass in/schedule state vectors every δt.
- Directly pass reward measurements to RL core.
- Base policy installed by controller, updated live.
- Matched packets poll actions from RL core.
  - Map RL actions to state machine, maintain throttling hash table.
  - Every $\delta t'$, batch actions to controller.

Timing: Why not offload to the controller?

For SmartNICs, the attached host *is the (closest) controller*.

- PCIe access times $\mathcal{O}(\mu s)$.[4]
- MATs recompiled $\mathcal{O}(s)$. Many rules $\implies$ batching.
- Thrift serde time $\mathcal{O}(ms)$.

Meanwhile core-to-core on chip around 100 ns @ 1.2 GHz.

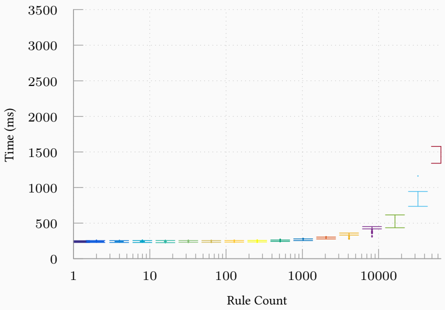


Figure 5: *Netronome rule installation cost (1 table, 1–65 536 rules).*

[4] Neugebauer *et al.*, 'Understanding PCIe performance for end host networking'.

Quantisation

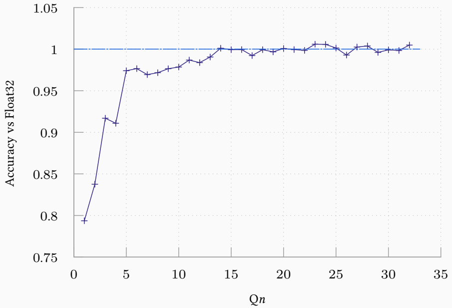


Figure 6: *DDoS throttle accuracy wrt. fixed-point Qn.*

No FPU... so use Qn fixed-point.

- For RL updates: need only shift, multiply, add.
- Policy exec only needs add.
- 16 bit fraction has negligible loss.

Dynamic Action Installation

- Netronome w/ P4?
  - MAT structure (DCFL) computed on host...
  - So can't directly interop with P4.
  - ...But we can have other actions/externs maintain their own lookup tables.
- Tofino has features to make this more feasible.
  - Action Profiles/Selects

Execution costs (Q15.16)

- Single-threaded—still to be accelerated.
- Small? One tiling per memory tier.
- Max? 20-element state, 17 mixed-dim tilings, 8 per set.
- Versus use-case: 330 µs on commodity i7 (4.2 GHz).

Policy Size	Action (µs)	Plus Update (µs)
Small	10.66	14.60
Max (DDoS)	512.67	612.55

Takeaways:

Online in-NIC RL is possible!

In-switch? Less so.

Platform-specific, but similar design for SmartNIC hardware class.

Work-in-progress: end-to-end timing, training accuracy, other use-cases (AQM?), optimisation.

Questions?