Runtime Programmable Switches

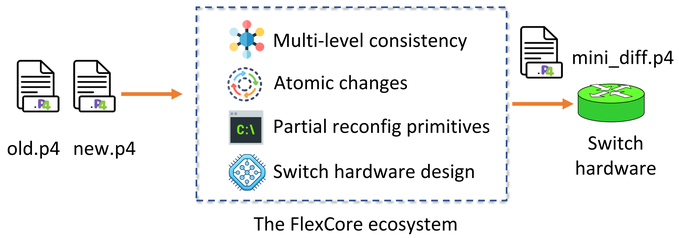
Jiarong Xing *(jxing@rice.edu), Kuo-Feng Hsu, Matty Kadosh†, Alan Lo†, Yonatan Piasetzky†, Arvind Krishnamurthy‡, Ang Chen*

Rice University, †Nvidia, ‡University of Washington

NSDI Paper | Code | Vision paper

1. Runtime programmability

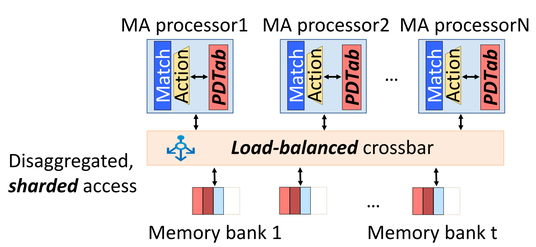
- ***Motivation:*** Today's programmable switches are only programmable at compile time. The switch becomes fixed once the program is deployed.
- ***FlexCore*** is a programmable switch that supports **live** program upgrades with **strong consistency** and **no downtime**. It enables a **new paradigm** for network programmability. Example use cases include:
  - Real-time attack mitigation
  - Just-in-time network optimization
  - Tenant-specific network extension



The FlexCore ecosystem

2. FlexCore hardware

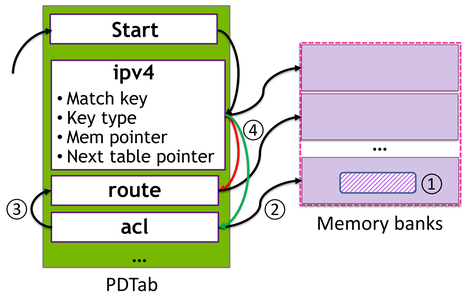
- **RMT is inflexible for runtime reprogramming:**
  - Compute and memory are tightly coupled in stages.
  - Inserting a table may require device-wide table shuffling.
  - Removing a table will leave "holes" in the memory.
- FlexCore uses an enhanced **dRMT architecture:**
  - Compute and memory are disaggregated.
  - Memory is sharded, and accesses are load-balanced.
  - MA processors use an indirection mechanism.



The enhanced disaggregated RMT architecture in FlexCore

3. Partial reconfiguration

- FlexCore reprograms the switch at runtime efficiently using ***partial*** reconfiguration that reuses the existing program as much as possible.
- FlexCore achieves this by using ***a pointer-based indirection*** mechanism. The key data structure is a program description table (PDTab), generated from a given P4 program:
  - PDTab entries are chained together by "next table pointers".
  - Pointers can be changed at runtime atomically.
  - FlexCore provides atomic partial reconfig primitives for P4 elements.



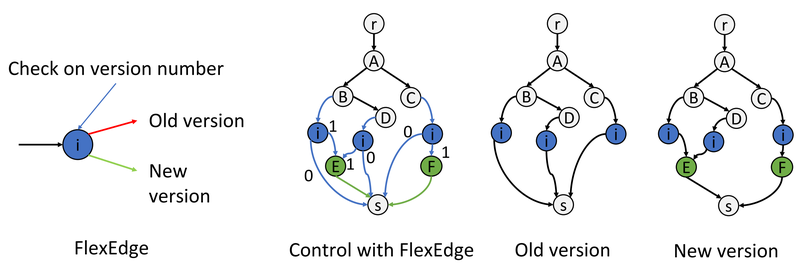

```
control ingress {
   ipv4.apply();
+  acl.apply();
   route.apply();
}
```

P4 program

Steps to insert a table at runtime with indirection

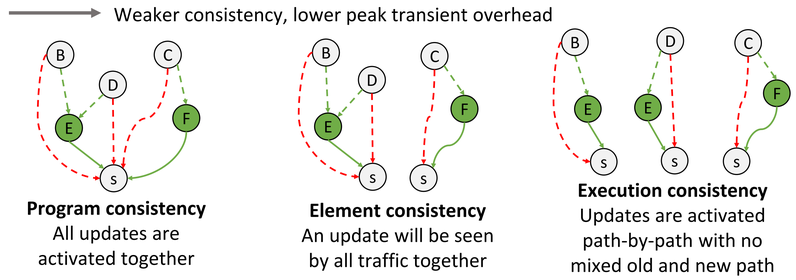
4. Atomic changes

- The hardware guarantees the atomicity of a single partial reconfig primitive, but a program might have multiple changes. FlexCore achieves program-level atomicity by using ***FlexEdge*** that controls the version based on a global version number.



5. Multi-level consistency

- ***Multi-level consistency:*** Not all upgrades require program level consistency that has high peak transient overhead and could fail when the switch has insufficient headroom.
- FlexCore supports three different levels of consistency.



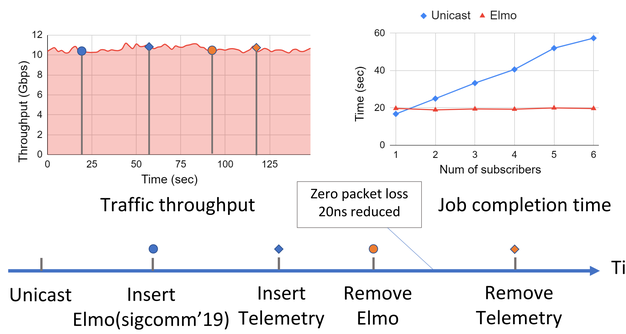
Program consistency
All updates are activated together

Element consistency
An update will be seen by all traffic together

Execution consistency
Updates are activated path-by-path with no mixed old and new path

6. Case studies

- **Accelerated multicast:**
  - T0: Use unicast to transmit data to a few subscribers.
  - T1: Insert Elmo, a source routed multicast program.
  - T2: Insert a telemetry program.
  - T3: Uninstall Elmo after the task is completed.
  - T4: Uninstall the telemetry program.
  - Runtime switch function upgrade with FlexCore has **no downtime.**
  - Runtime network optimization greatly **improves performance.**
  - See more case studies in the paper!



- **Code:** https://github.com/jiarong0907/FlexCore
- **Come to our talk at 10:50am Tuesday, April 5 (Track 2)**
- Also see our vision paper at HotNets'21

This work is partially supported by the National Science Foundation through the grant CNS-1801884.