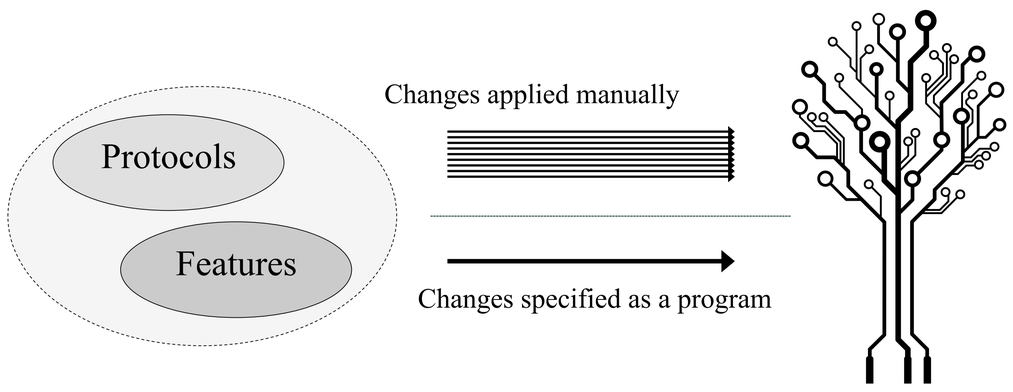
Customizing Open vSwitch using P4

Muhammad Shahbaz, Sean Choi, Ben Pfaff, Chaitanya Kodeboyina, Changhoon Kim,
Nick McKeown, Nick Feamster, and Jennifer Rexford

1. Problem Statement

- Adding protocols and features requires
  a. manual changes throughout the source tree
  b. maintaining changes across newer versions



- Instead, describe these changes as P4 programs. This has many benefits:
  a. Rapid addition of new protocols and features
  b. Automated testing and debugging
  c. Backward compatibility

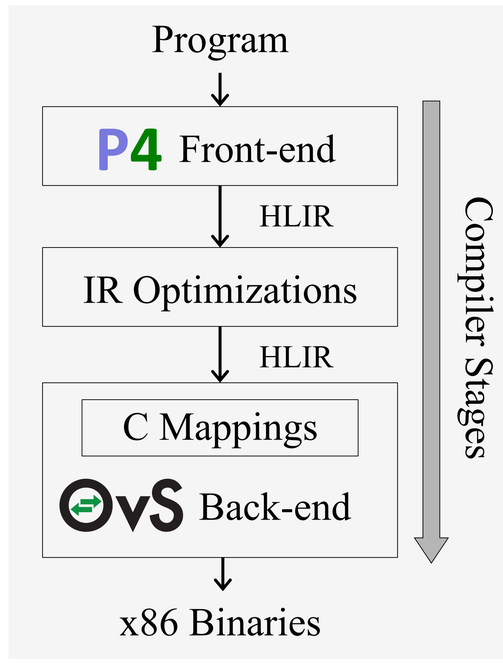
What's the Cost of Programmability on Performance?

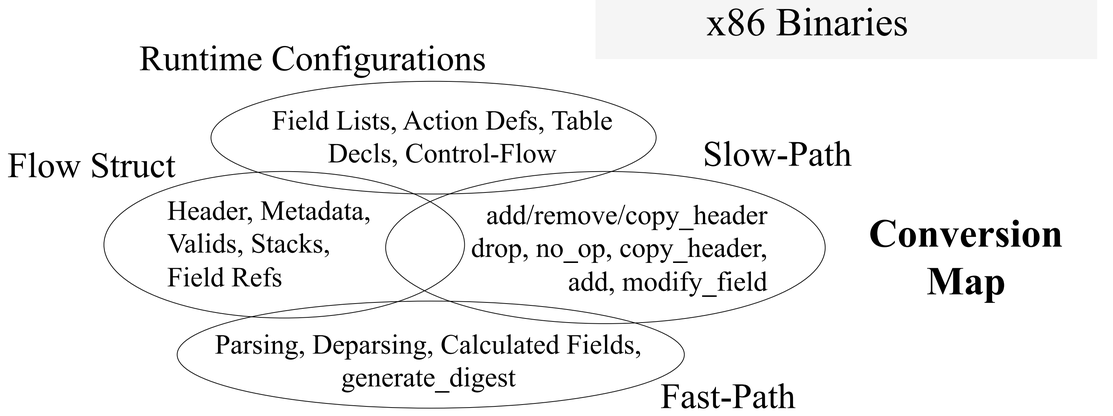
2. Approach

- Build a compiler for P4 to Open vSwitch (OvS).

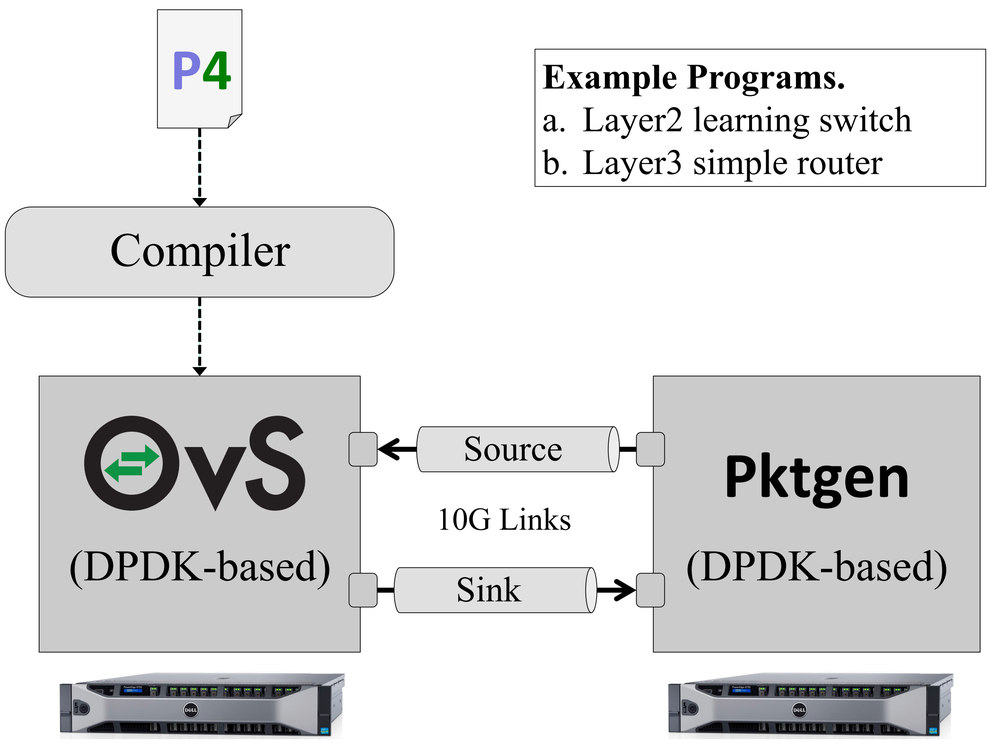
Challenges:

- Mapping P4 to OvS
  - Headers, Parser, and Action Primitives etc.
- Efficient Compilation
  - Program Analysis
  - Optimizations





3. Demo Setup



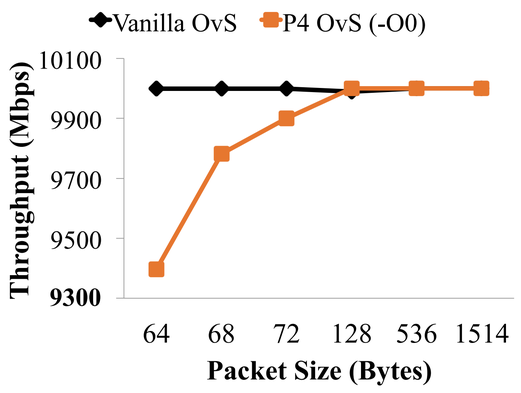
Example Programs.
a. Layer2 learning switch
b. Layer3 simple router

Server Specifications.

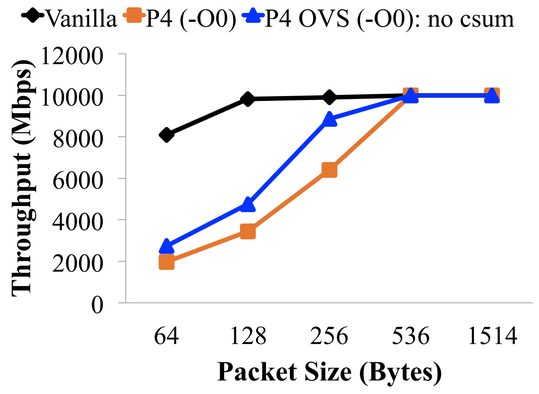
CPU: Intel Xeon E5-2640 v3 2.6GHz
Memory: 32GB RDIMM, 2133 MT/s, Dual Rank
Harddisk: 1TB 7.2K RPM NLSAS 6Gbps
NICs: Intel X710 DP/QP DA SFP+ Cards

4. Performance Results

a. Layer2 learning switch



b. Layer3 simple router



-O0: no optimizations

- Packet parsing is the primary bottleneck
  - Performance degrades with increasing no. of protocols
    - e.g., 19% decrease in Vanilla OvS with IP protocol parsing
  - More significant in P4 OvS as fast-path maintains its own copy of the parsed representation (i.e., extra overhead on parsing)
- **Future Goals:**
  - Implement optimizations to avoid extra copies of the parsed representation in the fast-path (based on dead-code elimination and liveness analysis etc.)

NOISe Lab
http://noise-lab.net

OvS Open vSwitch

Source Code
https://goo.gl/jlRHih

Acknowledgments:
The authors wish to thank Mihai Buidu, Ramkumar Krishnamoorthy, and Ed Doe for helpful early discussions and valuable feedback.