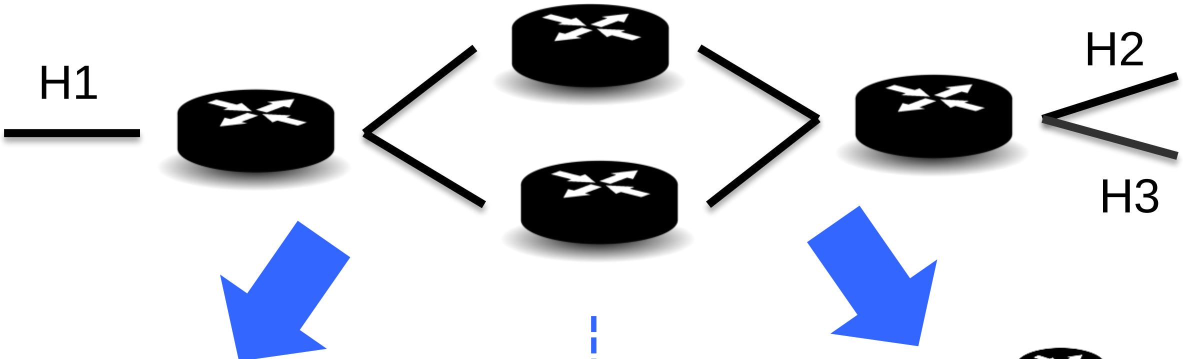
An Efficient Distributed Implementation of One Big Switch

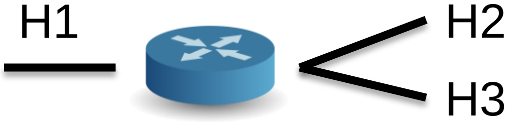
Nanxi Kang
Princeton University

in collaboration with Zhenming Liu, Jennifer Rexford, David Walker

One Big Switch Abstraction

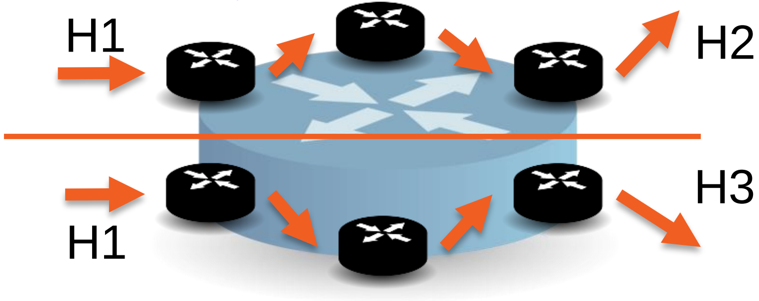


From H1, dstIP = 0* => go to H2
From H1, dstIP = 1* => go to H3



Endpoint policy

e.g, Firewall, Load Balancer

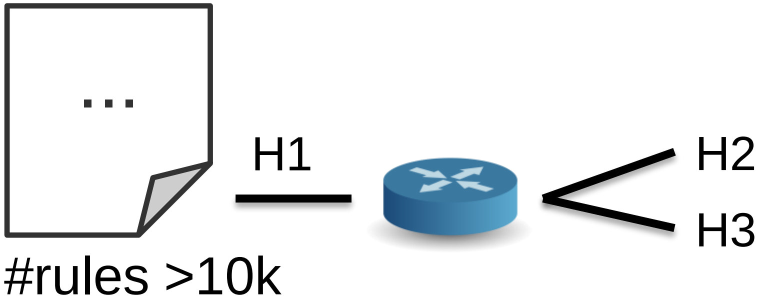


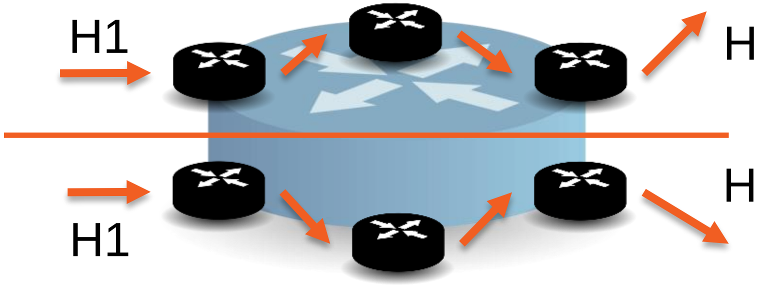
Routing policy

e.g, Shortest path routing

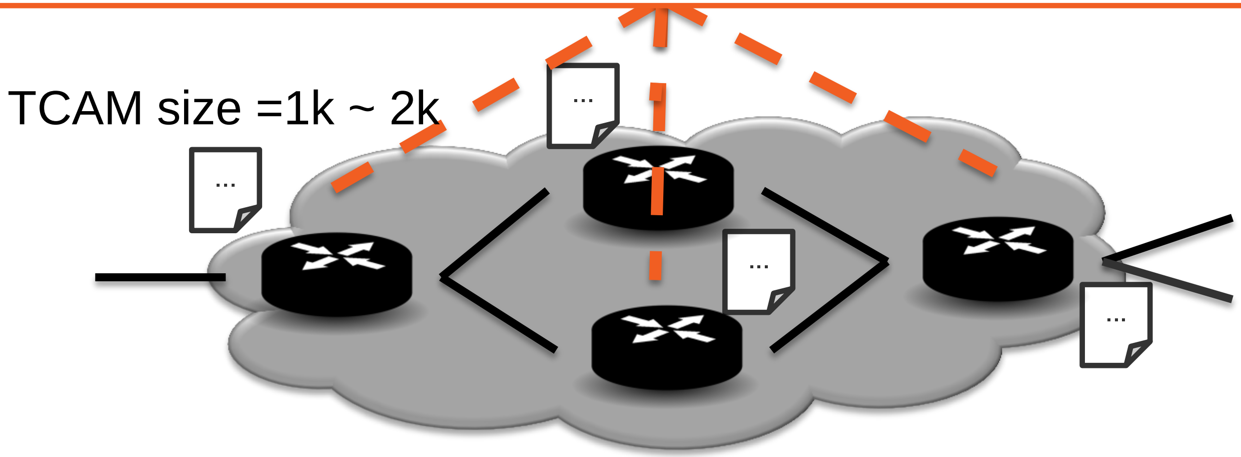
Automatic Rule Placement

Challenges of Rule Placement





Automatic Rule Placement



Contributions

- Design a new rule placement algorithm
  - Stay within rule capacity of switches
  - Minimize the total number of installed rules
- Handle policy update incrementally
- Evaluation on real and synthetic data

Contribution

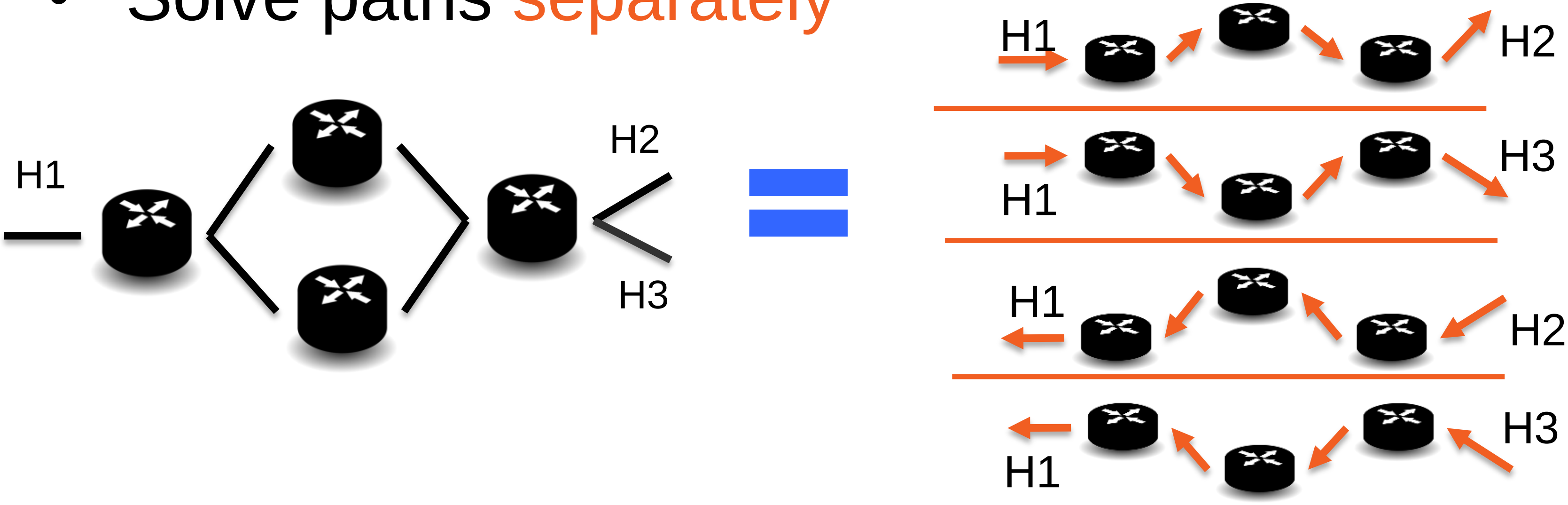
- Design a new rule placement algorithm
  - Stay within rule capacity of switches
  - Minimize the total number of installed rules

- Handle policy update incrementally

- Evaluation on real and synthetic data

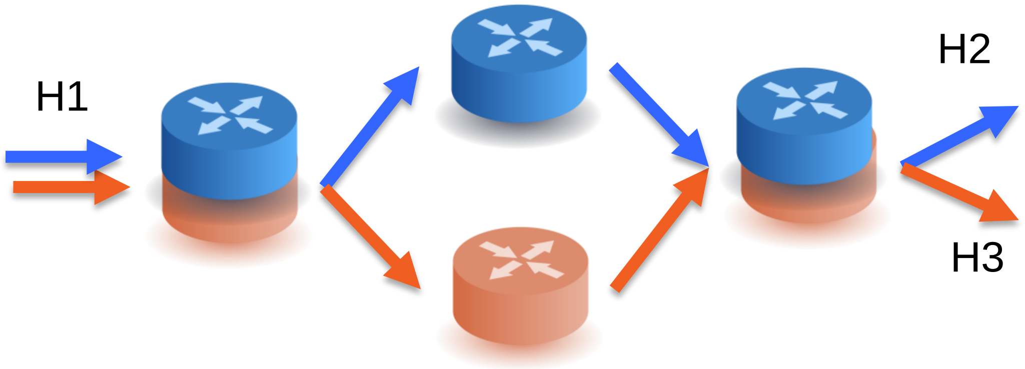
Topology = {Paths}

- Enforce routing policy
  - install rules on switches to forward packets
- Enforce endpoint policy
  - topology as multiple paths: an ordered list of switches
  - Solve paths separately



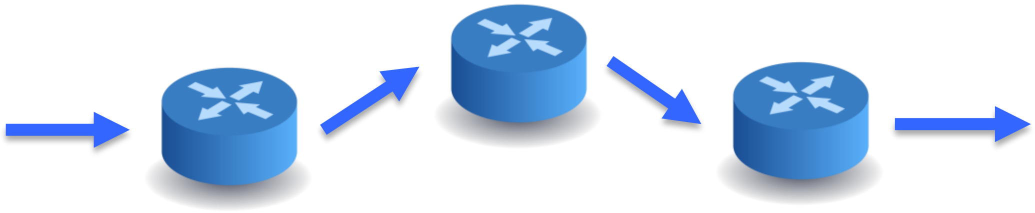
Model shared switches

- Multiple paths share the same switch
- Split shared rule capacity over paths
  - Paths have different demands for *total* rule capacities
  - Linear Programming



Place rules over a path

- How to place rules over a path ?



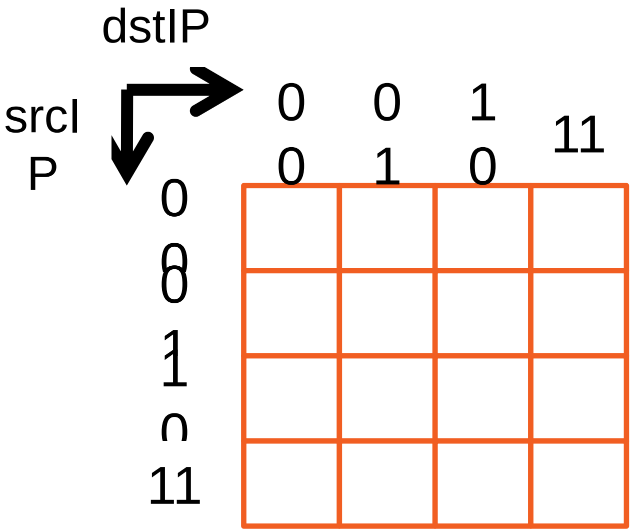
R1: (srcIP = 0*, dstIP = 00*), permit

R2: (srcIP = 01, dstIP = 1*), permit

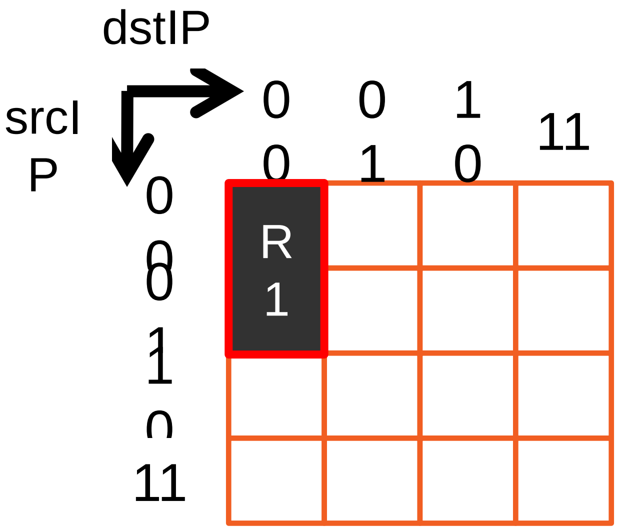
R3: (srcIP = *, dstIP = 11*), deny

R4: (srcIP = 11*, dstIP = *), permit

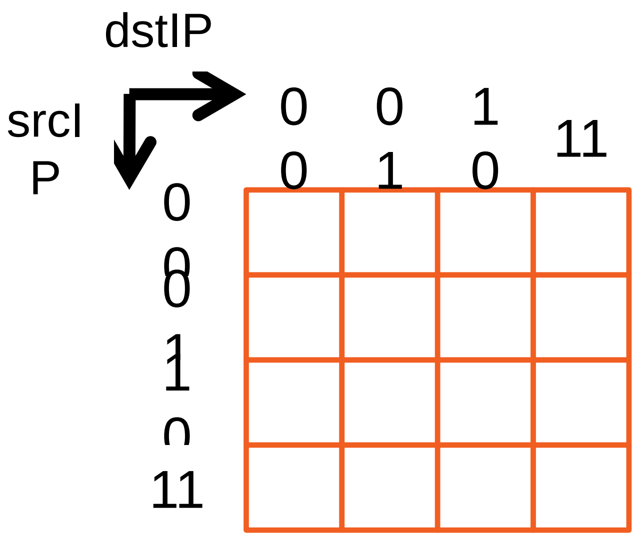
Map rule to rectangle



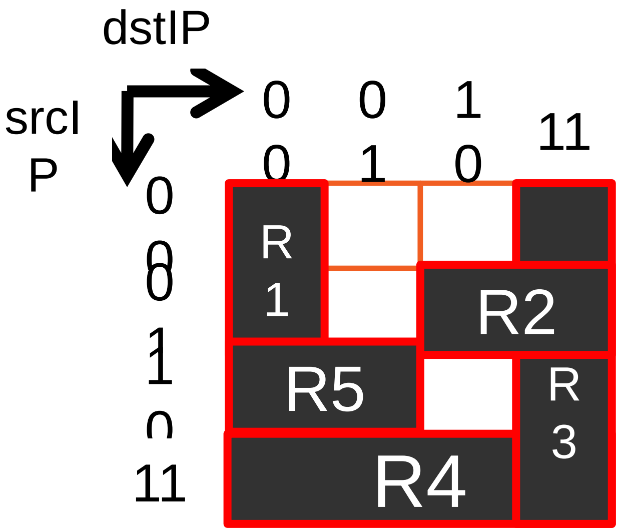
R1: (0*, 00),P
R2: (01, 1*),P
R3: (**, 11),D
R4: (11, **),P
R5: (10, 0*),P
R6: (**, **),D

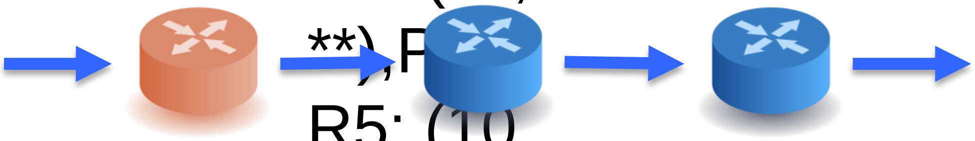


Map rule to rectangle

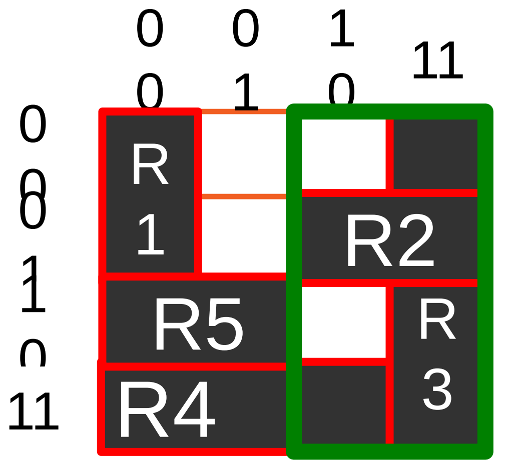


R1: (0*, 00),P
R2: (01, 1*),P
R3: (**, 11),D
R4: (11, **),P
R5: (10, 0*),P
R6: (**, **),D





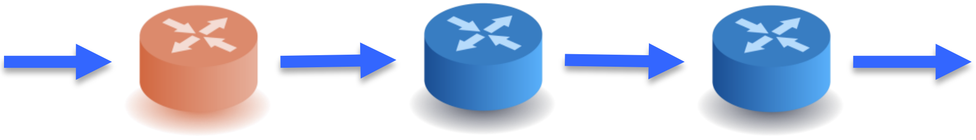
Cover a rectangle



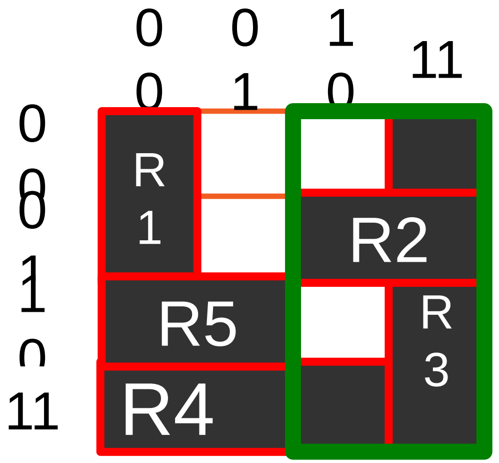
q: (**,1*)

R1: (0*, 00)
R2: (01, 1*)
R3: (**, 11)
R4: (11, **)
R5: (10, 0*)
R6: (**, **)

- Overlapped rules:
  R2, R3, R4, R6
- Internal rules:
  R2, R3

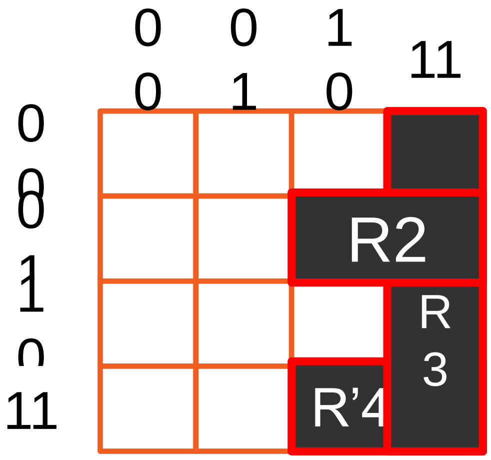


Install rules in first switch

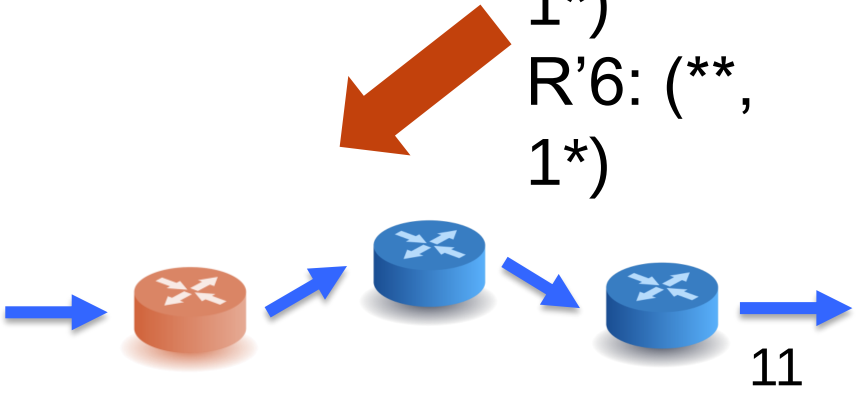


q: (**,1*)

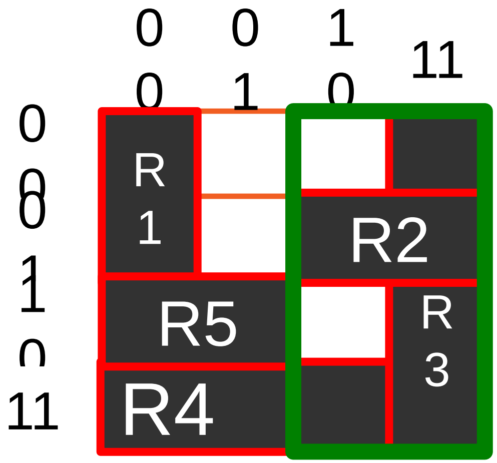
R1: (0*, 00)
R2: (01, 1*)
R3: (**, 11)
R4: (11, **)
R5: (10, 0*)
R6: (**, **)



R2: (01, 1*)
R3: (**, 11)
R'4: (11, 1*)
R'6: (**, 1*)



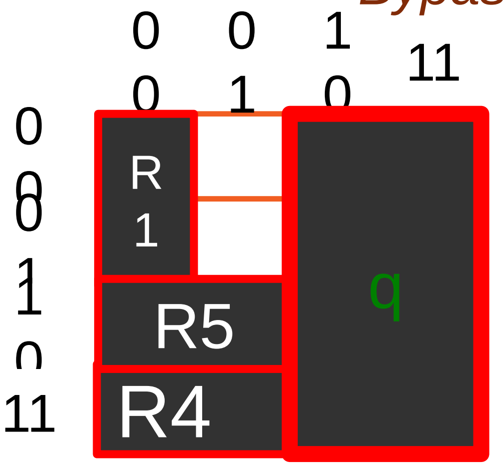
Rewrite policy



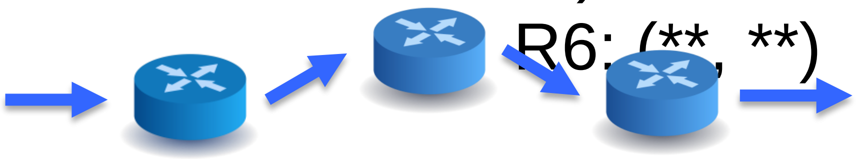
q: (**,1*)

R1: (0*, 00)
R2: (01, 1*)
R3: (**, 11)
R4: (11, **)
R5: (10, 0*)
R6: (**, **)

Bypass everything in q



q^{fwd}: (**, 1*)
R1: (0*, 00)
R4: (11, **)
R5: (10, 0*)
R6: (**, **)



Summary

- Contribution
  - An efficient rule placement algorithm
  - Support for incremental update
  - Evaluation on real and synthetic data
    - Path: 8-hop, 14k rules, <1.9k rules/switch
    - Graph:100 switches, 0.5s(LP) + 0.5s ~ 9s(Path)
- Future work
  - Integrate with real-time SDN systems
  - Combine with policy checking and verification

Thanks!

Q & A?

Related Work

- Single switch optimization
  - TCAM Razor
  - “Compressing Rectilinear Pictures and Minimizing Access Control Lists”
- Distributed switch optimization
  - vCRIB
    - Algorithm assumes control over routing
  - Palette
    - Enforce the whole network-wide policy on every path