Infinite CacheFlow in Software-Defined Networking

Naga Katta

Omid Alipourfard, Jennifer Rexford, David Walker

Princeton University

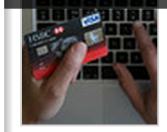
Recent News

THE WALL STREET JOURNAL. | TECH

LOG IN | SUBSCRIBE

TOP STORIES IN TECH

1 of 12



Data Breach Puts Focus on Beefed-Up Car...

2 of 12

What Happens When Police Wear Cameras?

3 of 12

Vintage Videogame Venues

TWC Complicates Comc Merger

TECHNOLOGY

Echoes of Y2K: Engineers Buzz That Internet Is Outgrowing Its Gear

Routers That Send Data Online Could Become Overloaded as Number of Internet Routes Hits '512K'

A A

By DREW FITZGERALD CONNECT

Updated Aug. 13, 2014 7:38 p.m. ET

Network engineers are buzzing this week as the Internet outgrows some of its gear.

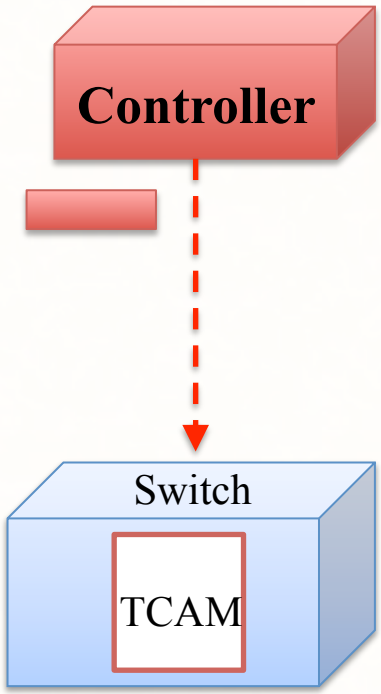
Internet providers, corporations and universities all rely on a common map of routes to send emails, videos and everything else on the Web where it's supposed to go.

Available to WSJ.com Subscribers

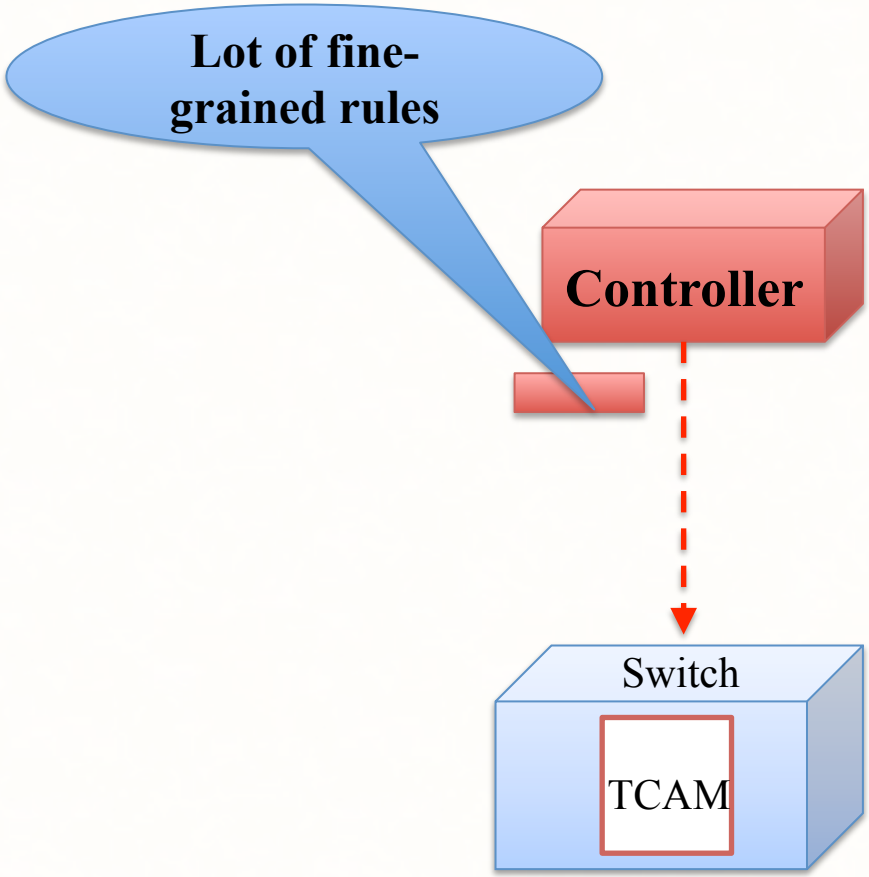
Fed Bets It Won't Fall Behind Curve on Rates

Kurds, With U.S. Aid,

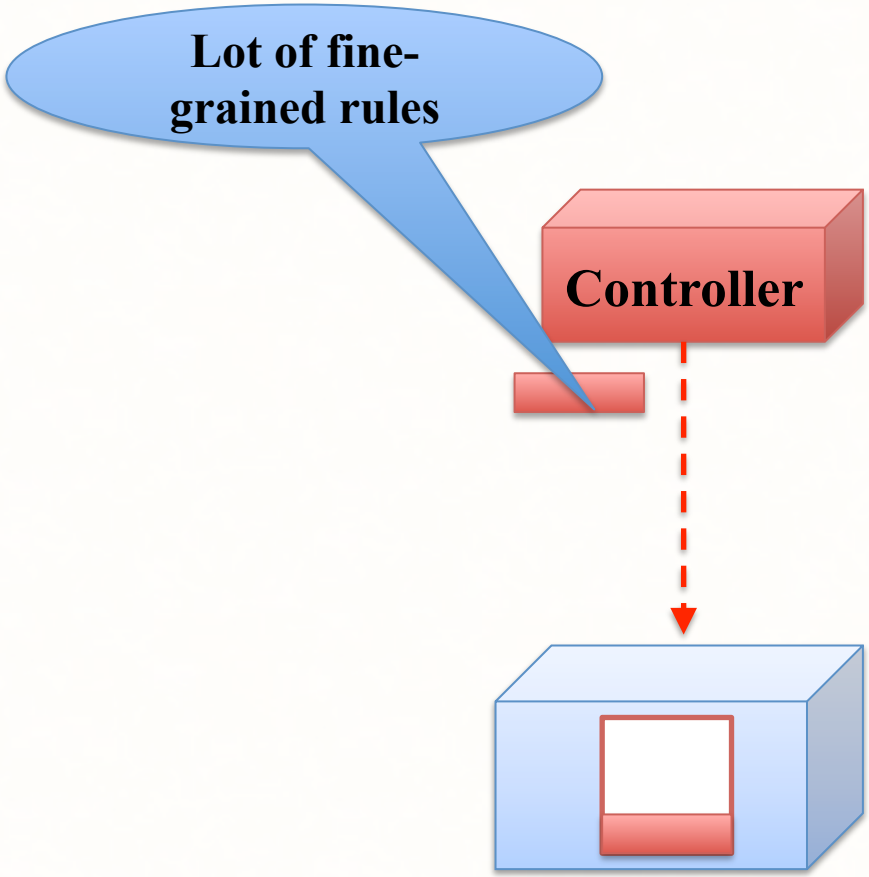
SDN Promises Flexible Policies



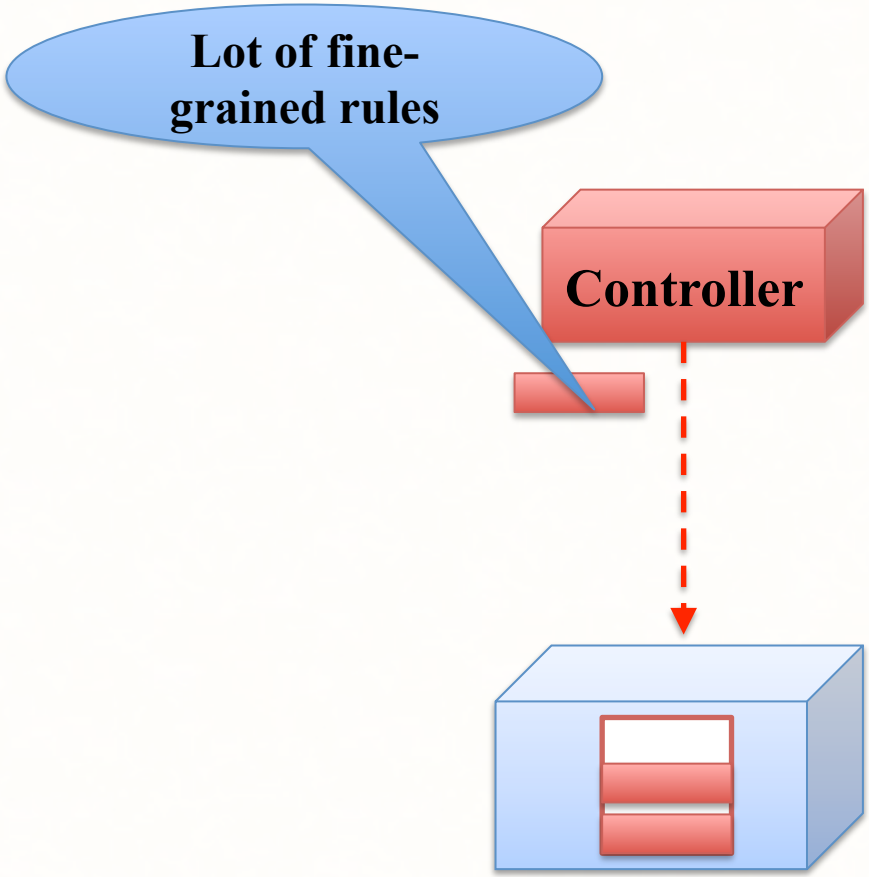
SDN Promises Flexible Policies



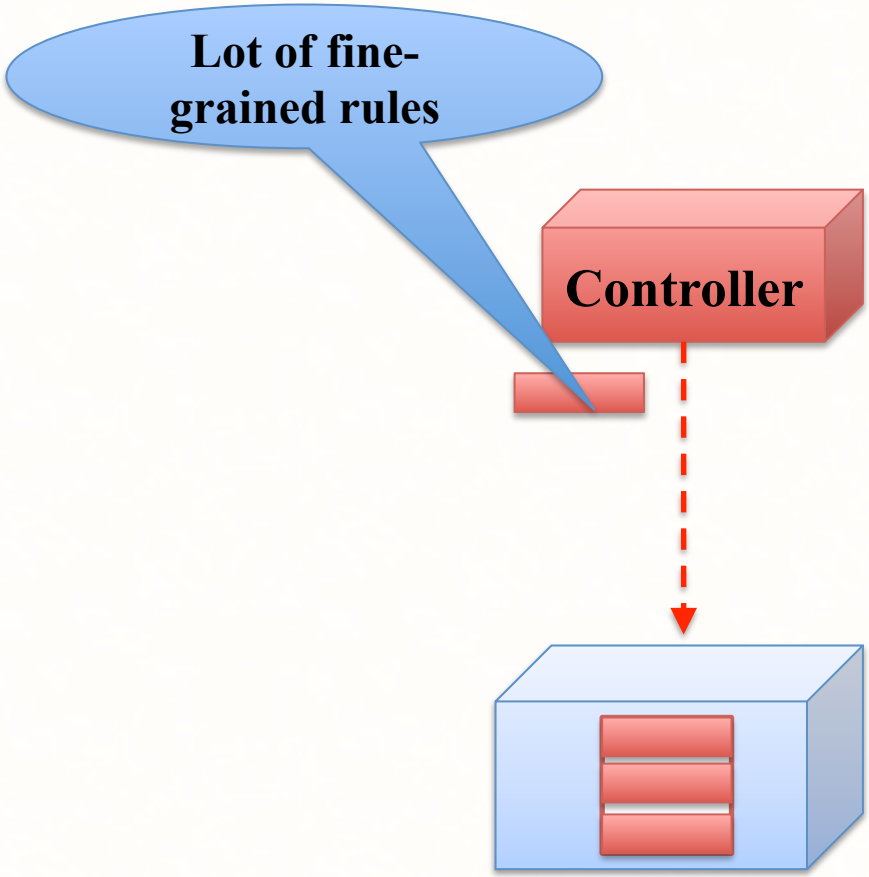
SDN Promises Flexible Policies



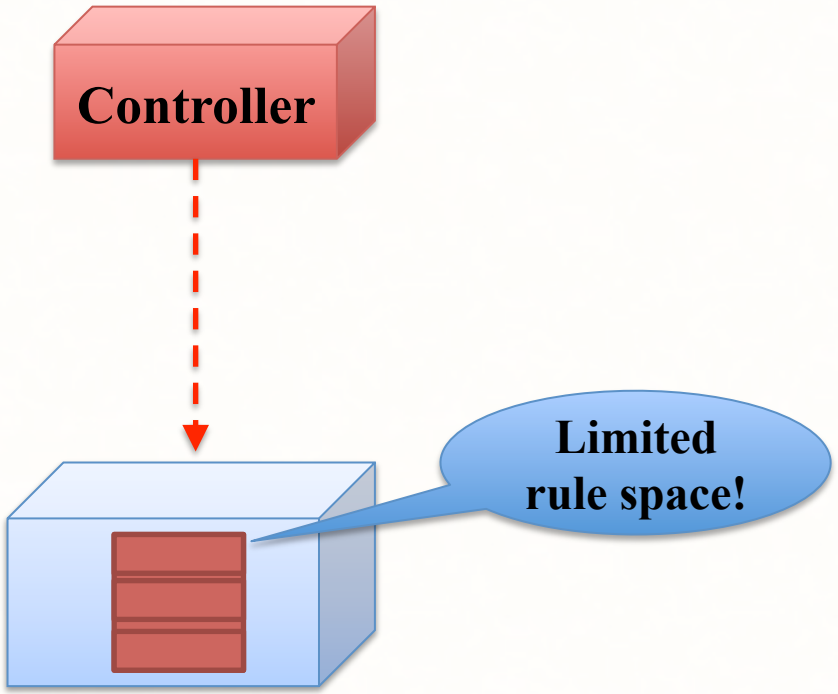
SDN Promises Flexible Policies



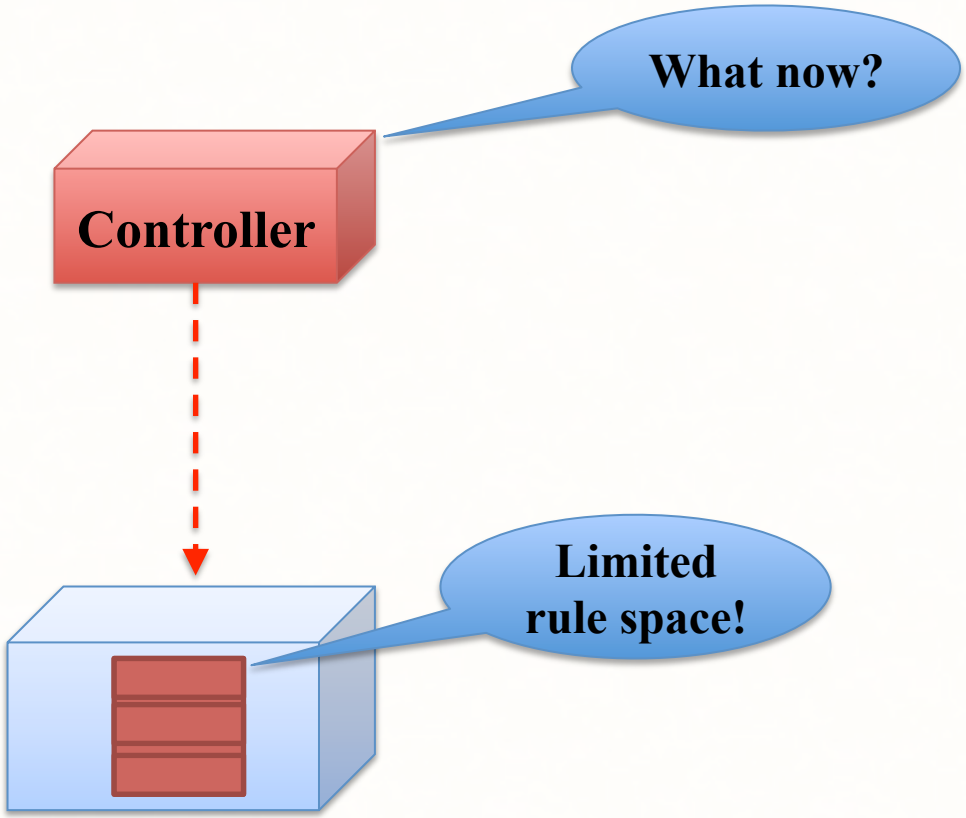
SDN Promises Flexible Policies



SDN Promises Flexible Policies



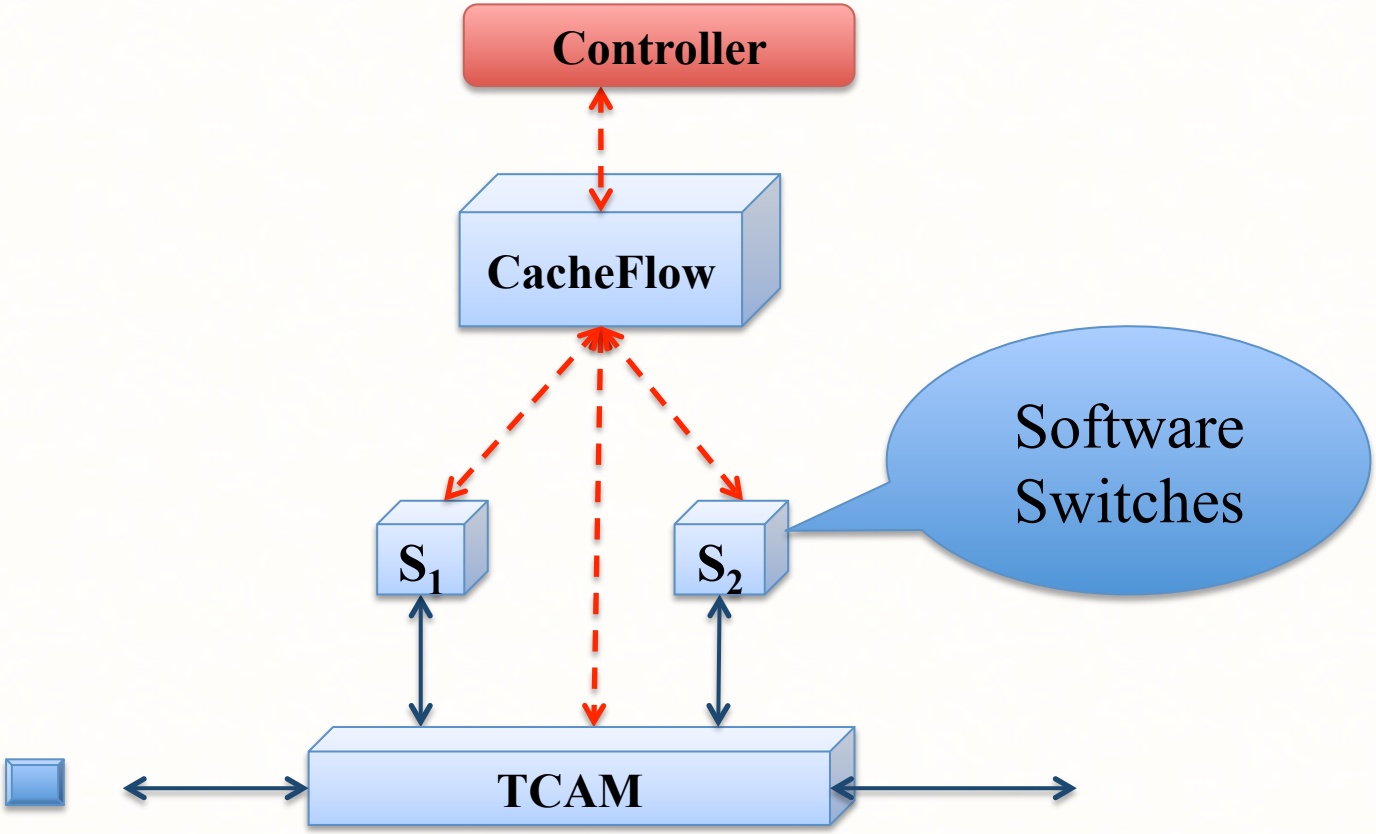
SDN Promises Flexible Policies



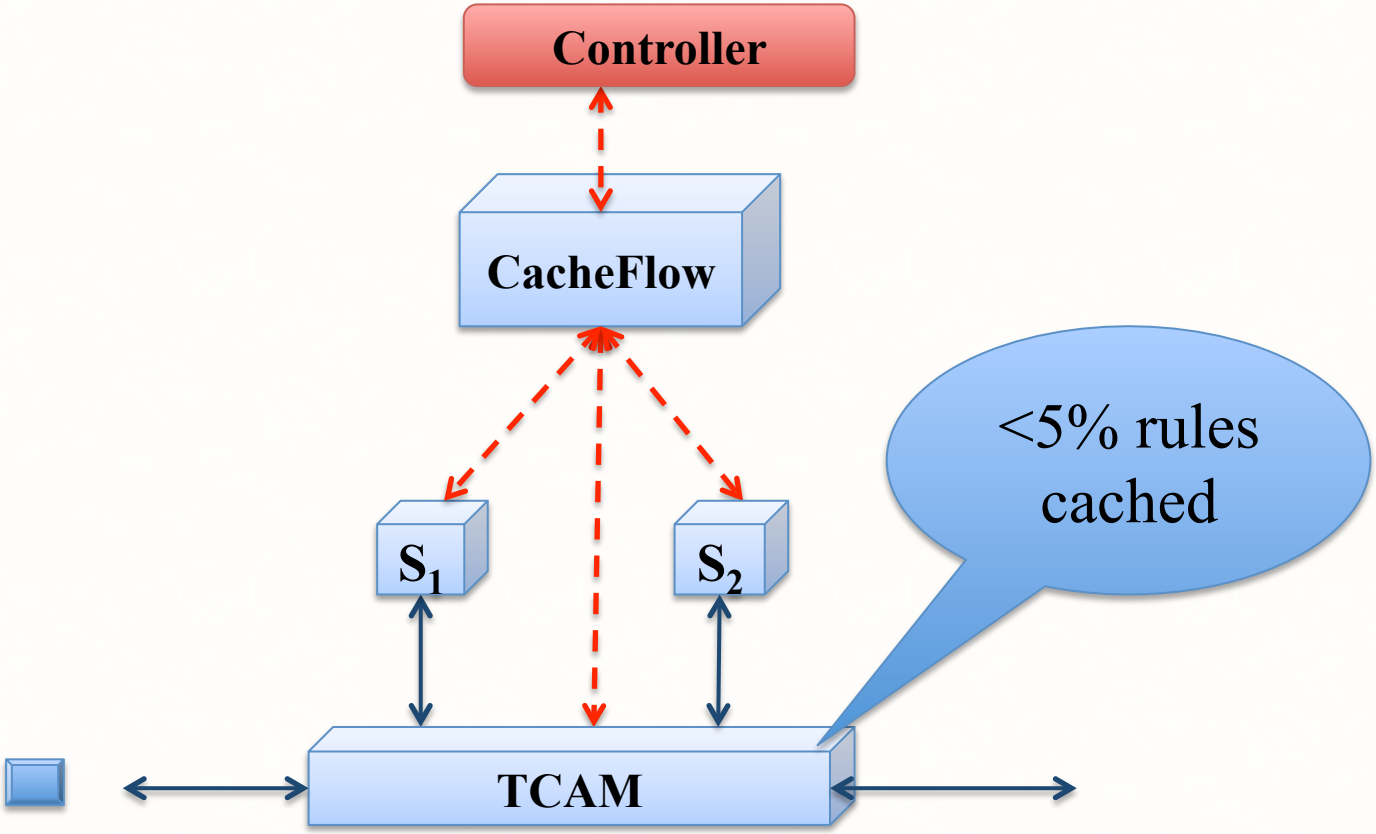
State of the Art

	Hardware Switch	Software Switch
Rule Capacity	Low (~2K-4K)	High
Lookup Throughput	High (>400Gbps)	Low (~40Gbps)
Port Density	High	Low
Cost	Expensive	Relatively cheap

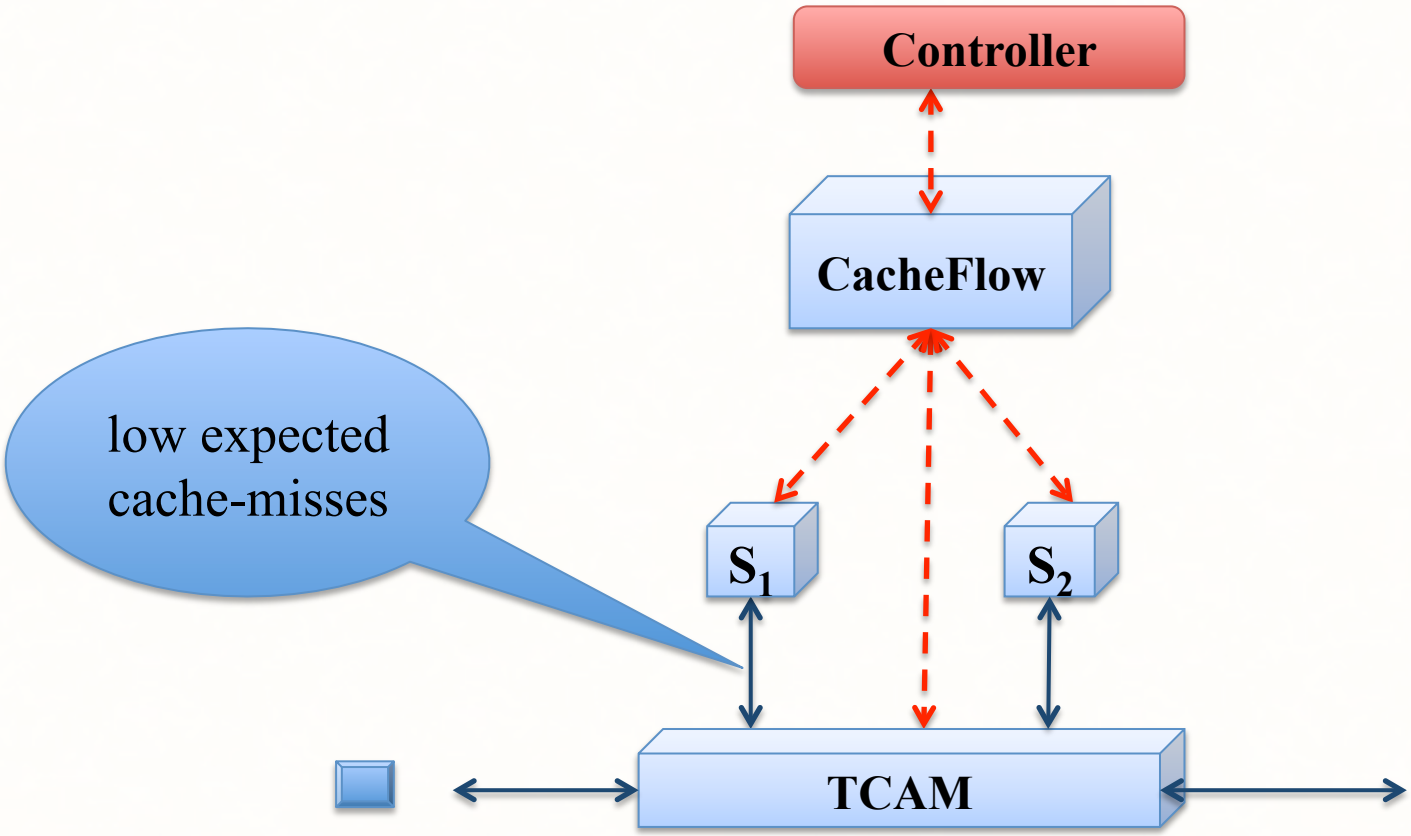
TCAM as cache



TCAM as cache

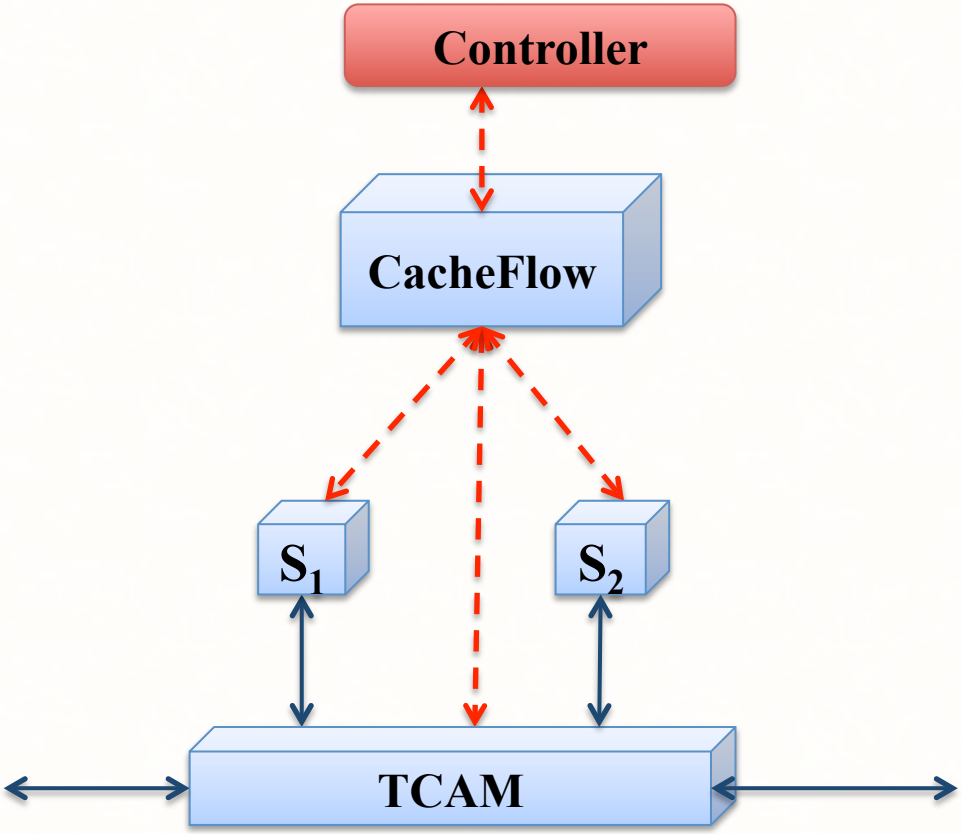


TCAM as cache



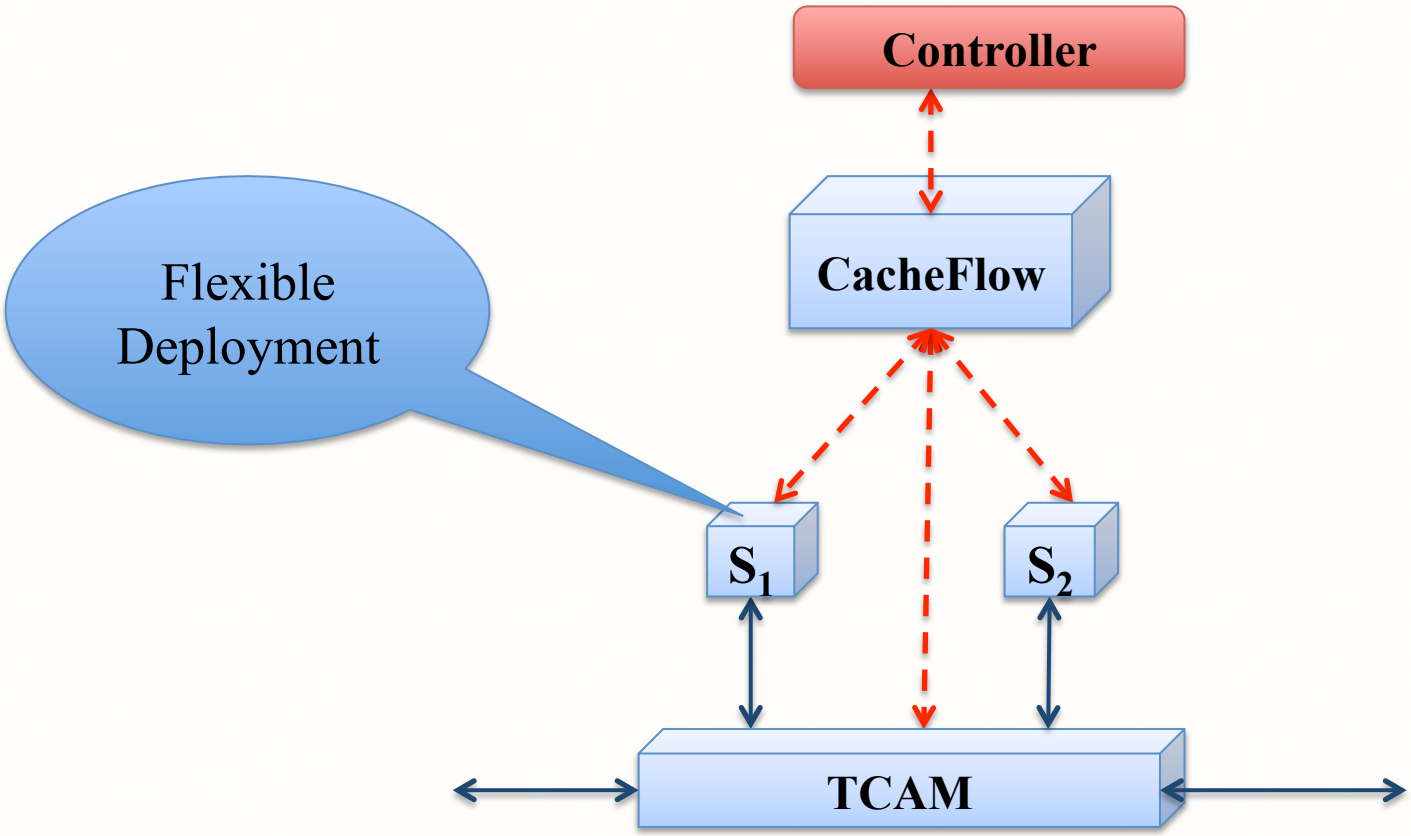
TCAM as cache

- High throughput + high rule space



TCAM as cache

- High throughput + high rule space



A ***Correct***, ***Efficient*** and ***Transparent*** Caching System

- Abstraction of an "infinite" switch
  - Correct: realizes the policy
  - Efficient: high throughput & large tables
  - Transparent: unmodified applications/switches

1. Correct Caching

Caching under constraints

Rule	Match	Action	Priority	Traffic
R1	110	Fwd 1	3	10
R2	100	Fwd 2	2	60
R3	101	Fwd 3	1	30

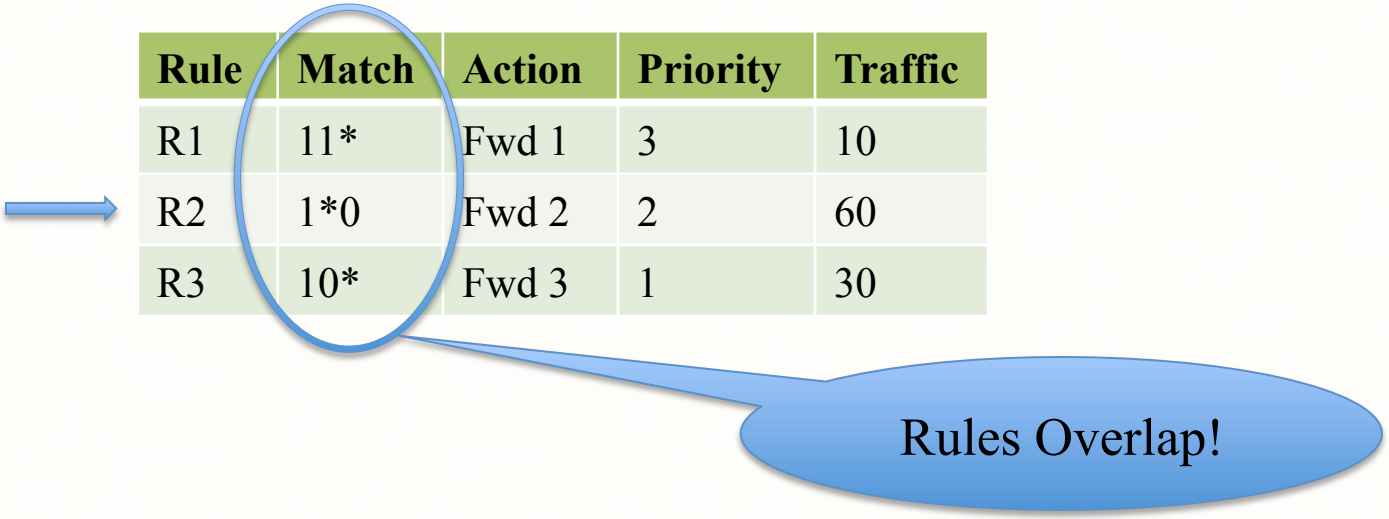
Easy: Cache rules greedily

Caching Ternary Rules

Rule	Match	Action	Priority	Traffic
R1	11*	Fwd 1	3	10
R2	1*0	Fwd 2	2	60
R3	10*	Fwd 3	1	30

- Greedy strategy breaks rule-table semantics

Caching Ternary Rules



- Greedy strategy breaks rule-table semantics

Caching Ternary Rules

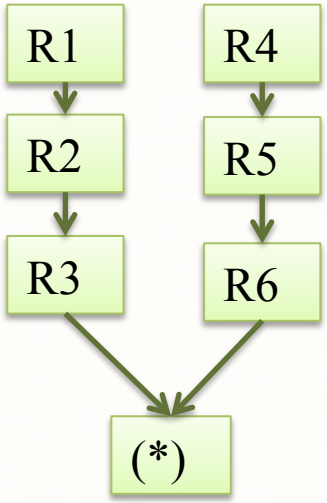
Rule	Match	Action	Priority	Traffic
R1	11*	Fwd 1	3	10
R2	1*0	Fwd 2	2	60
R3	10*	Fwd 3	1	30

Rules Overlap!

- Greedy strategy breaks rule-table semantics
- Beware of switches that claim large rule tables

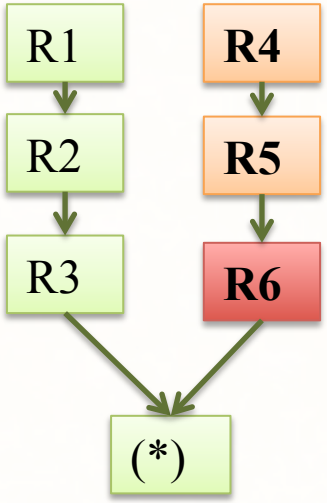
Dependency Graph

Rule	Match	Action	Priority	Traffic
R1	0000	Fwd 1	6	10
R2	000*	Fwd 2	5	20
R3	00**	Fwd 3	4	90
R4	111*	Fwd 4	3	5
R5	11**	Fwd 5	2	10
R6	1***	Fwd 6	1	120



Dependent-Set Caching

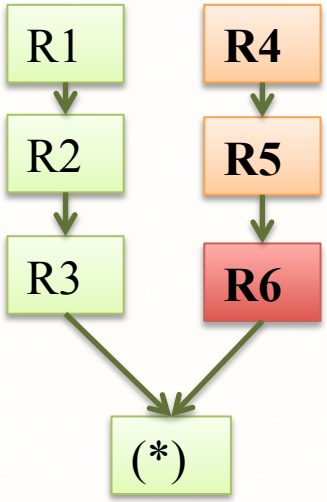
- All descendants in DAG are dependents
- Cache dependent rules for correctness



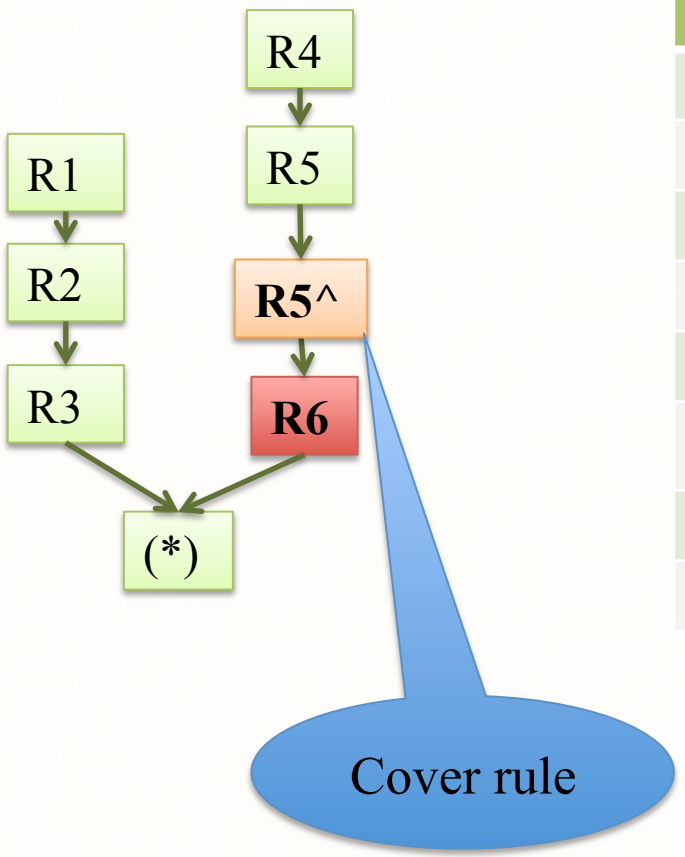
2. Efficient Caching

Dependent-Set Overhead

Too Costly?

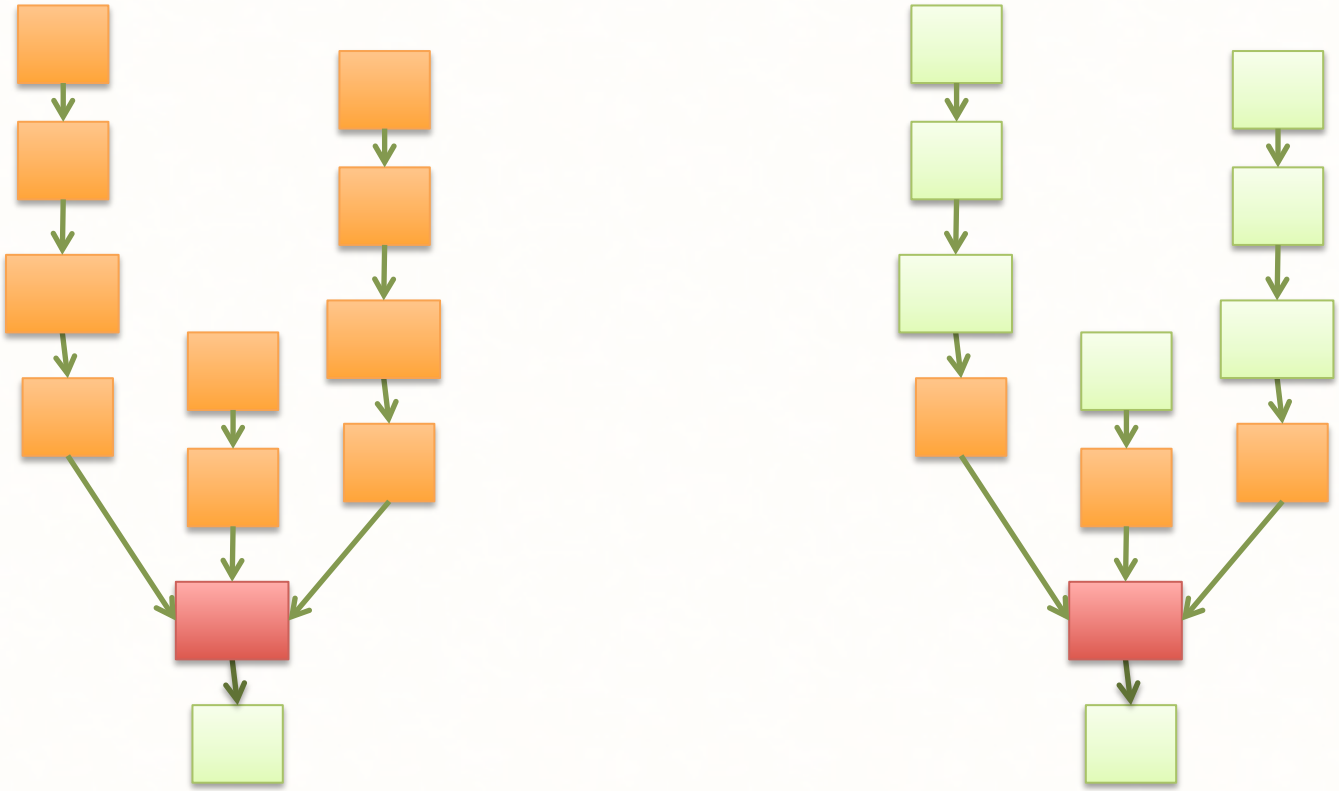


Cover-Set



Rule	Match	Action
R1	000	Fwd 1
R2	00*	Fwd 2
R3	0**	Fwd 3
R4	11*	Fwd 4
R5	1*0	Fwd 5
R5^	**1*0**	**To_SW**
R6	10*	Fwd 6
(*)	***	To_SW

Dependency **Splicing** reduces rule cost!



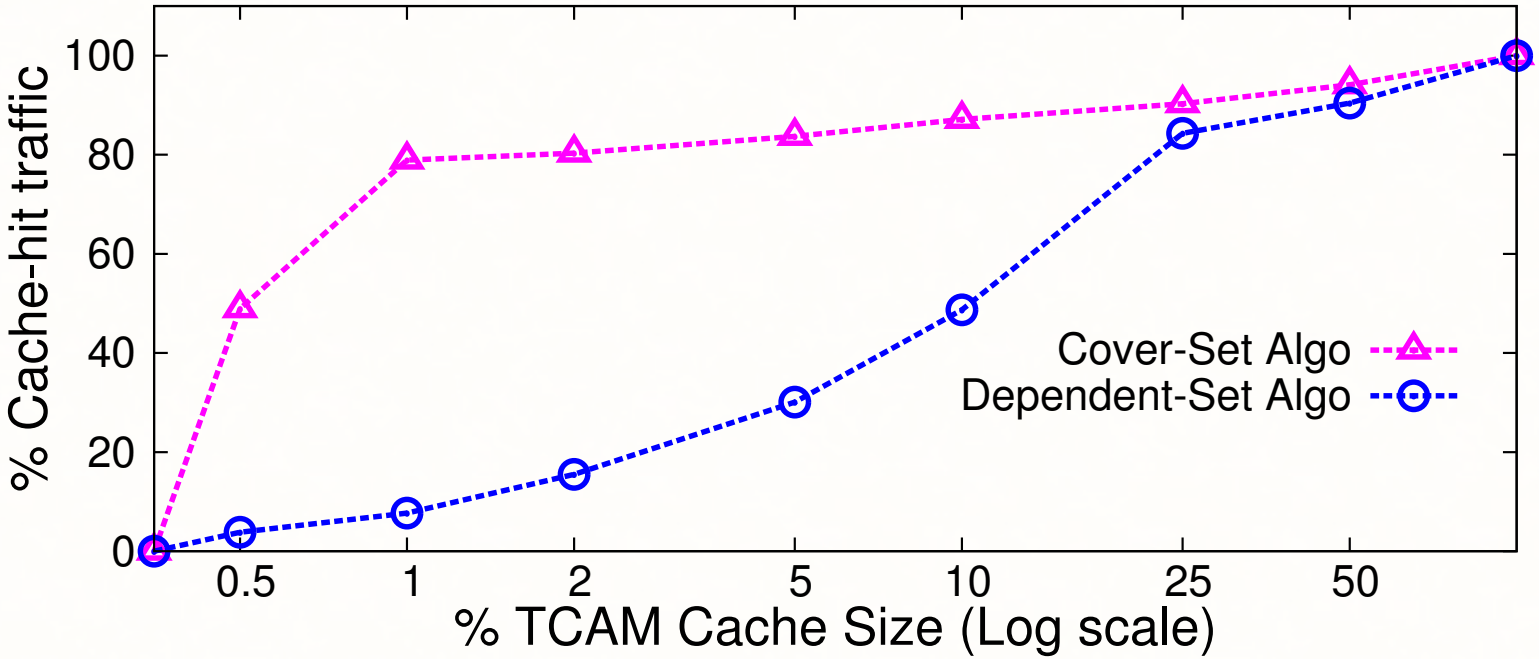
Dependent-Set

Cover-Set

Rule Space Cost

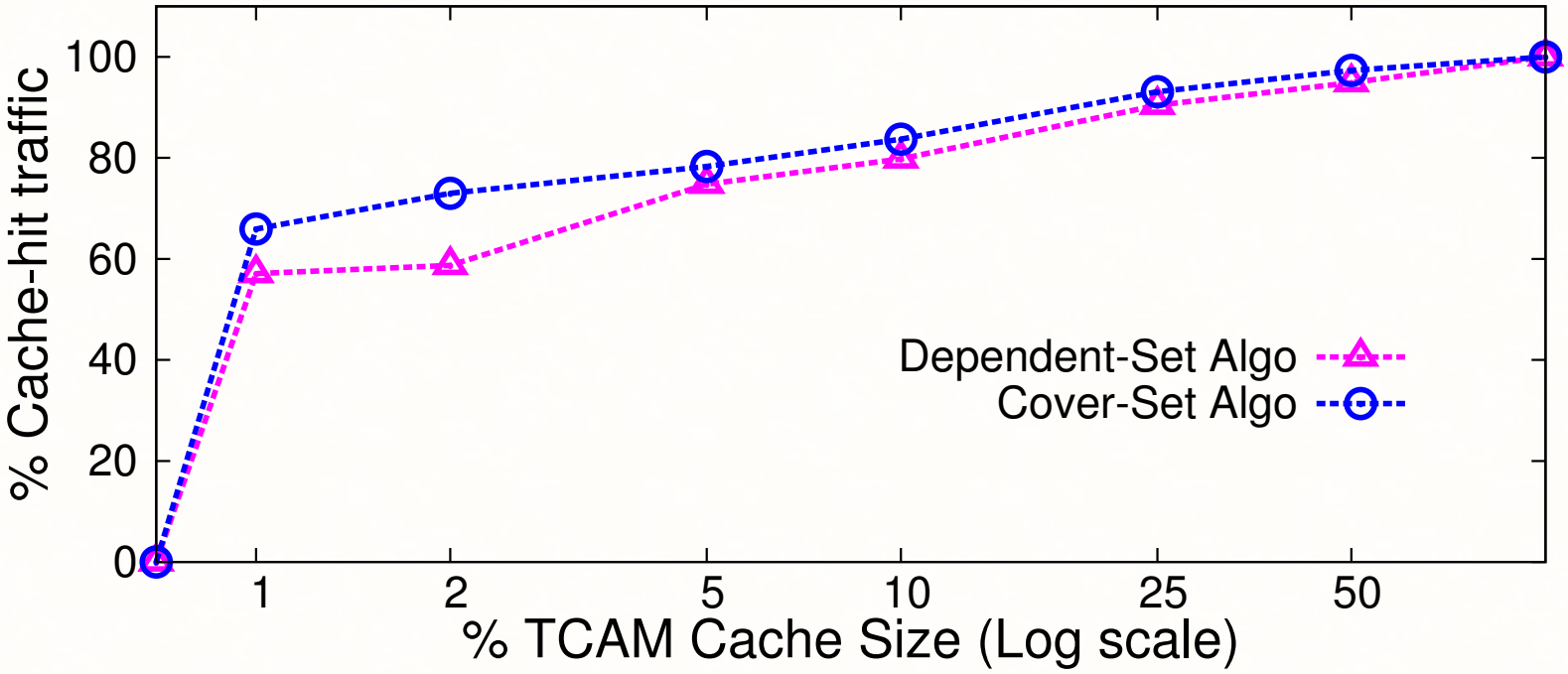
Deep Dependency Chains – Clear Gain

- ClassBench Generated ACL



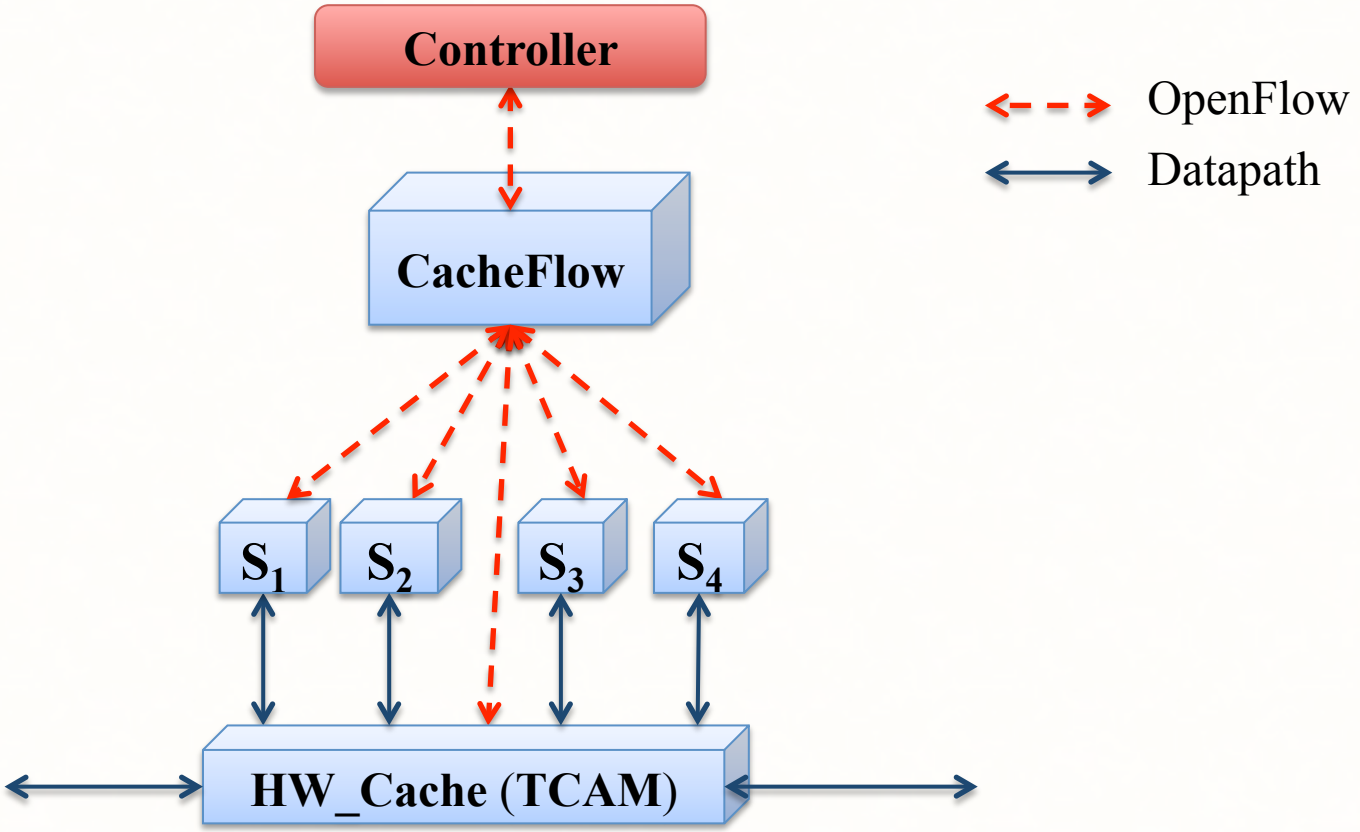
Shallow Dependency Chains – Marginal Gain

- Stanford Backbone Routing table

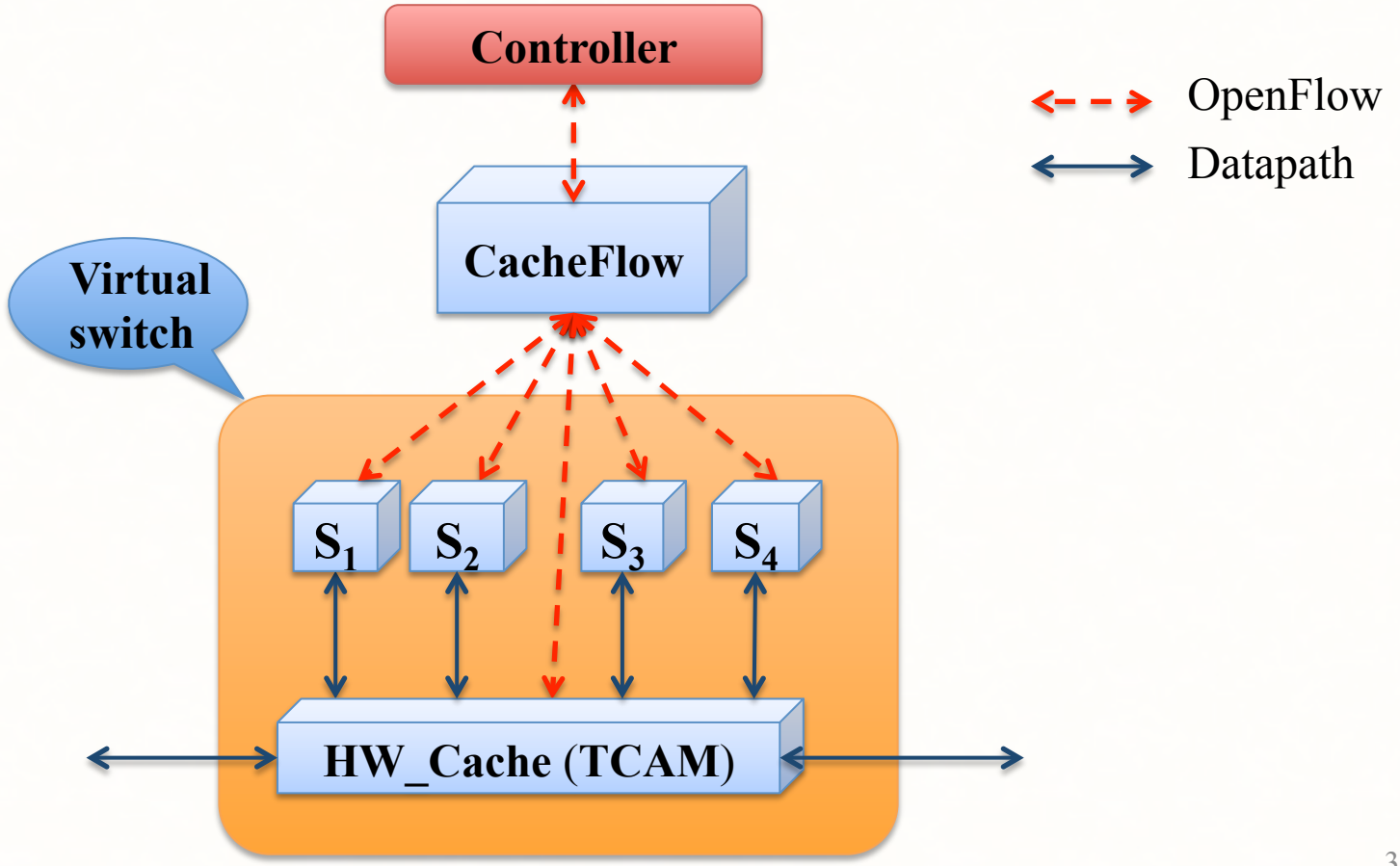


3. Transparent Caching

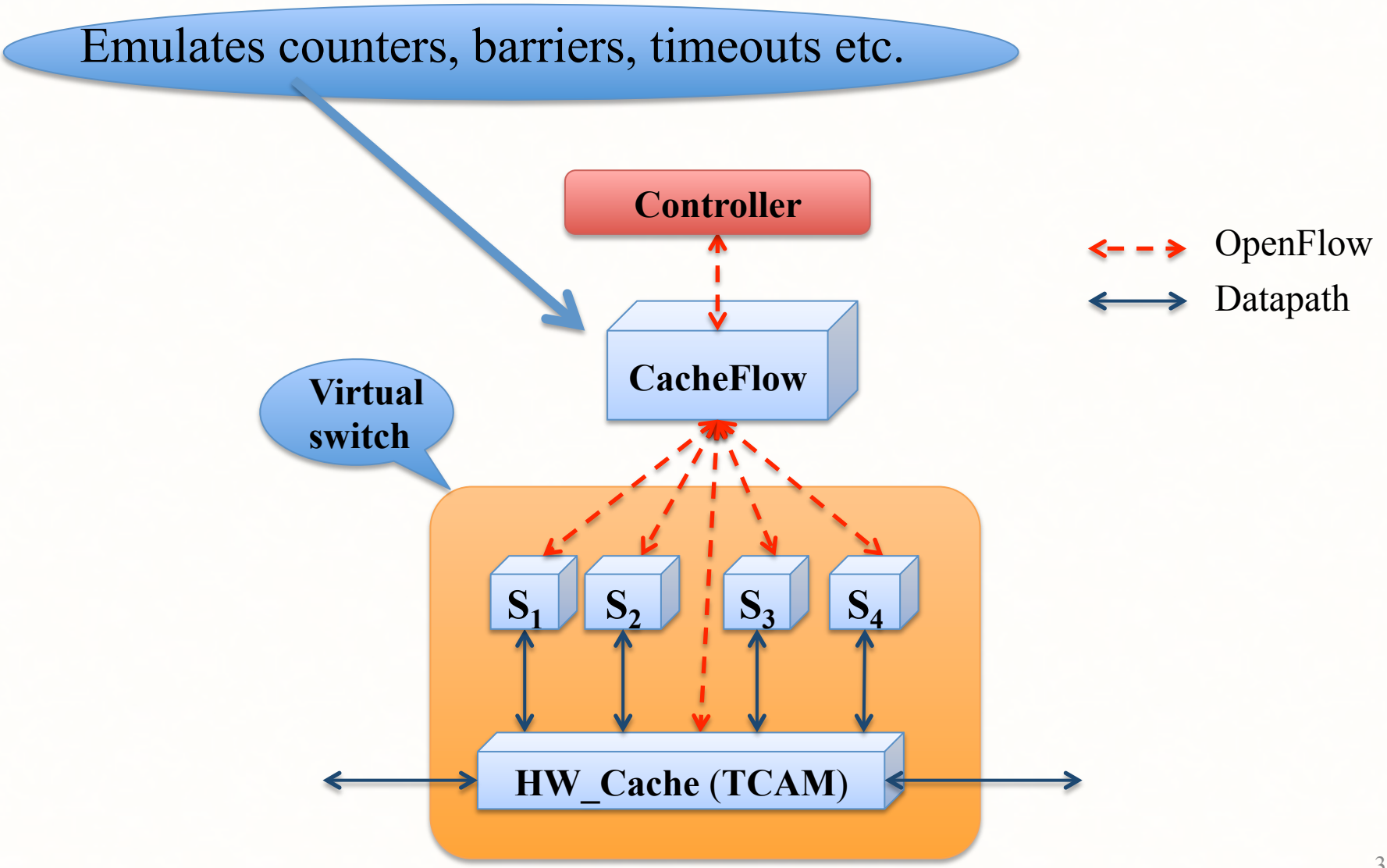
3. Transparent Design



3. Transparent Design



3. Transparent Design



Conclusion

- Rule caching for OpenFlow rules
  - Dependency analysis for ***correctness***
  - Splicing dependency chains for ***efficiency***
  - ***Transparent*** design

Infinite Ca$heFlow in SDN

Questions?

frenetic ≫