Flexible Enterprise Network Management on Commodity Switches

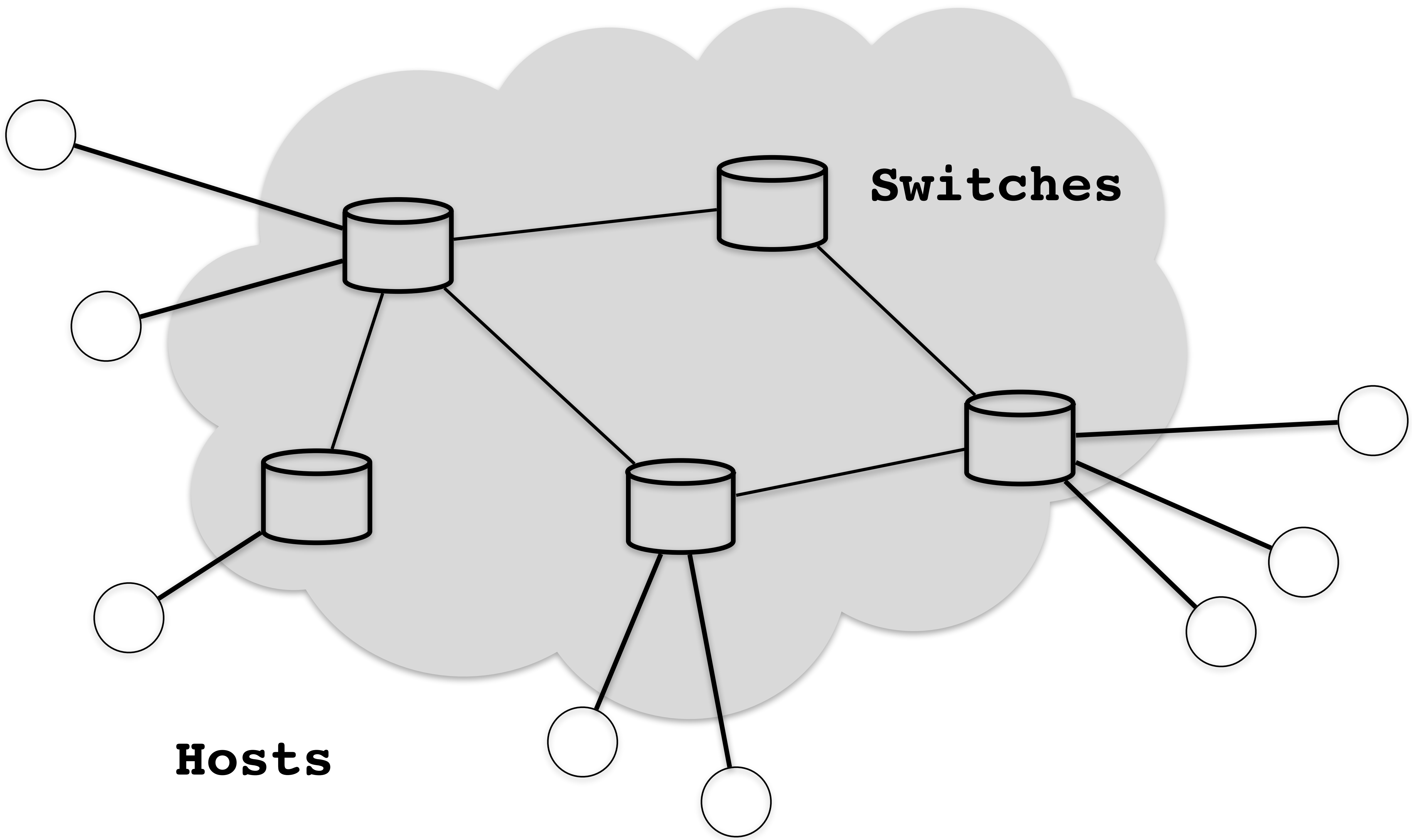
Nanxi Kang

Committee: Jennifer Rexford (advisor),

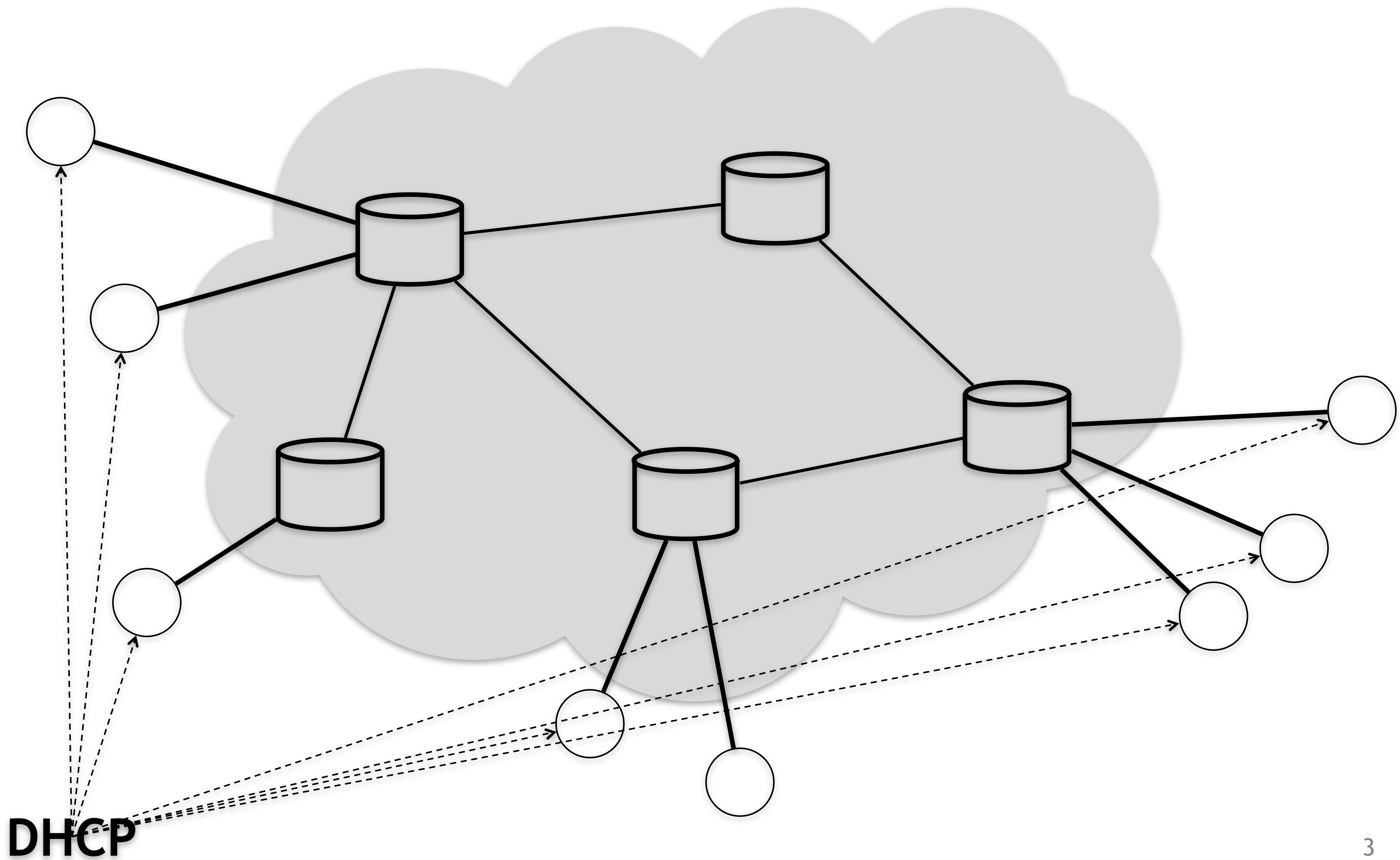
Nick Feamster, Sanjay Rao,

David Walker and Mike Freedman

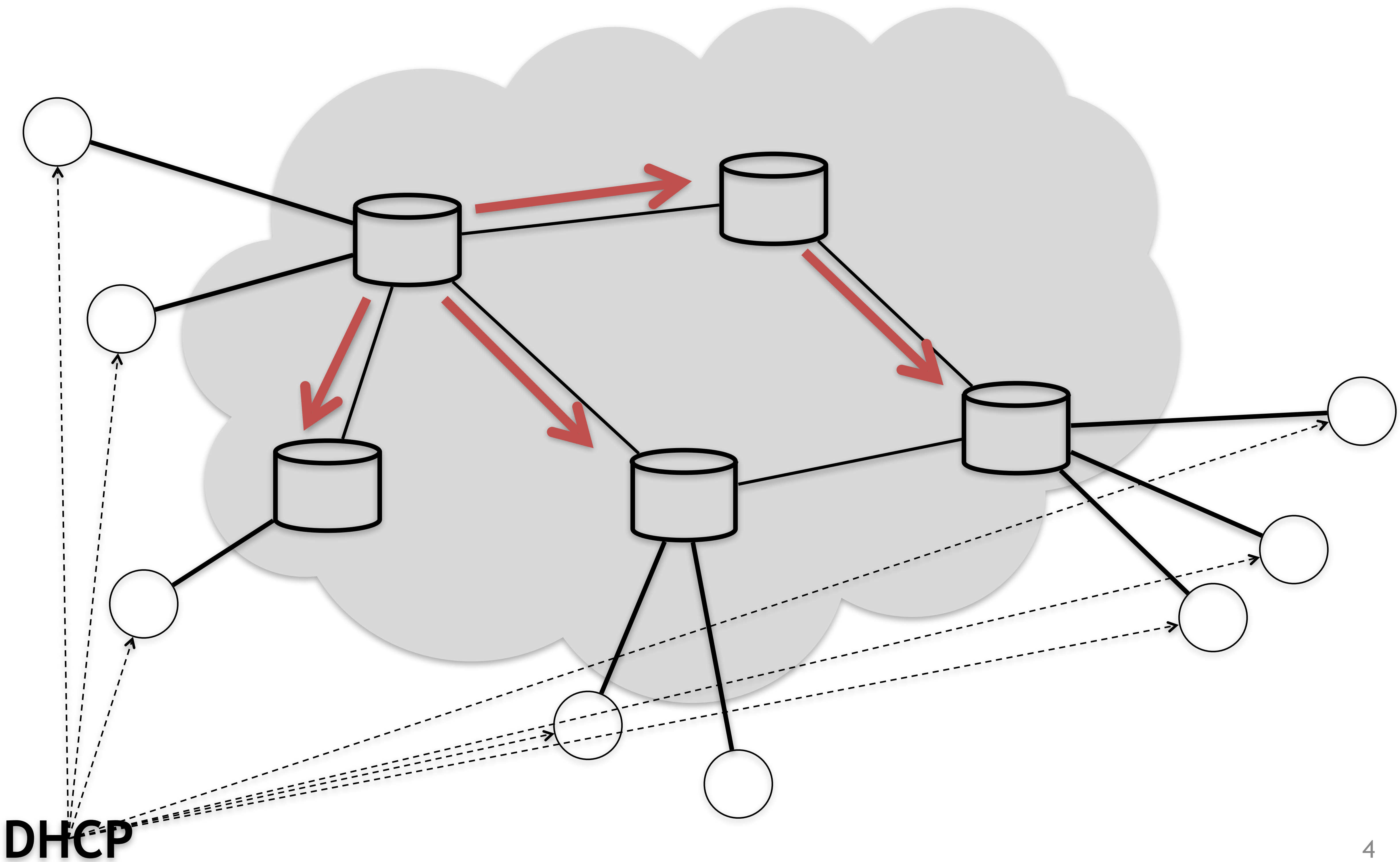
Manage a Network



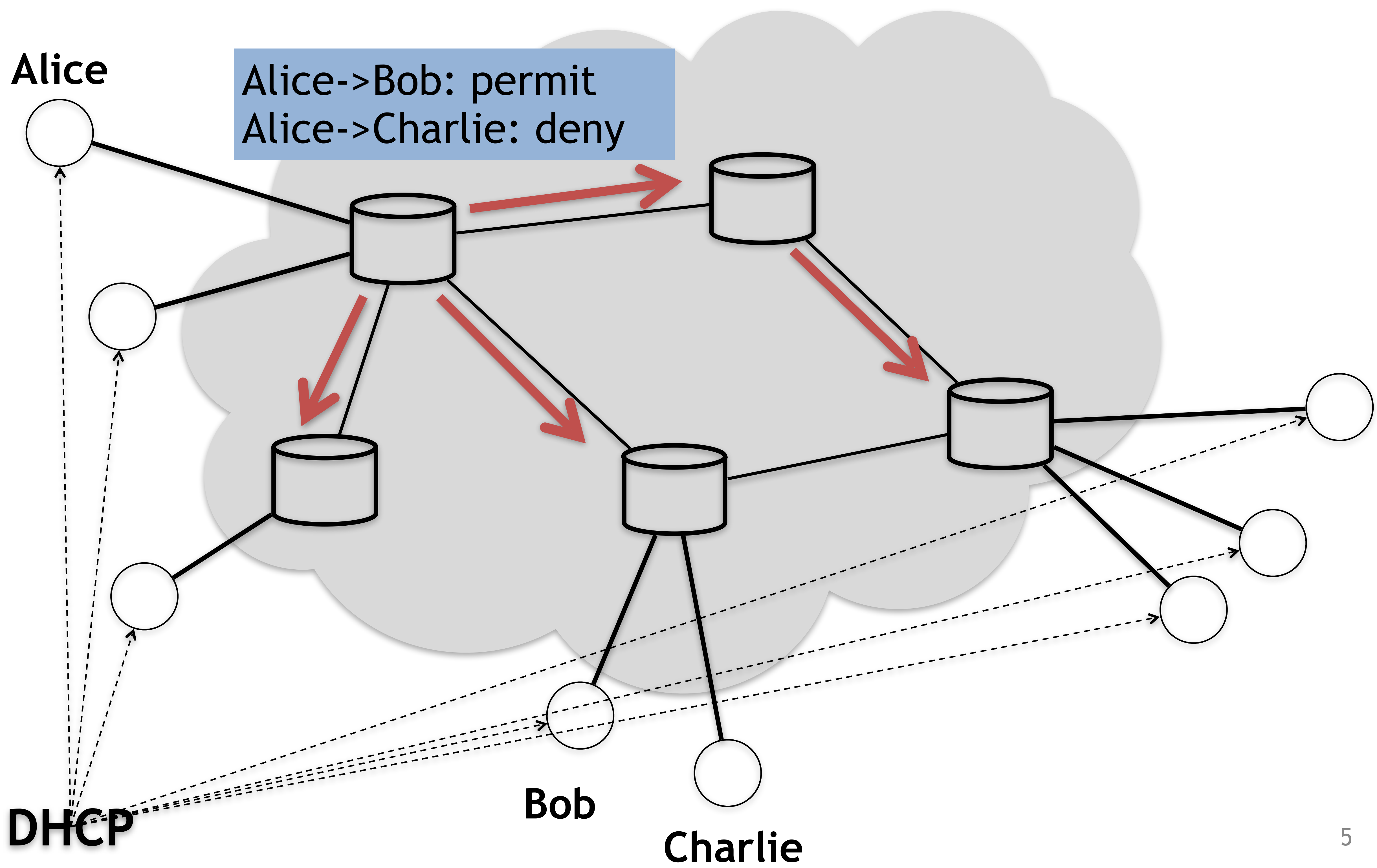
Address Assignment



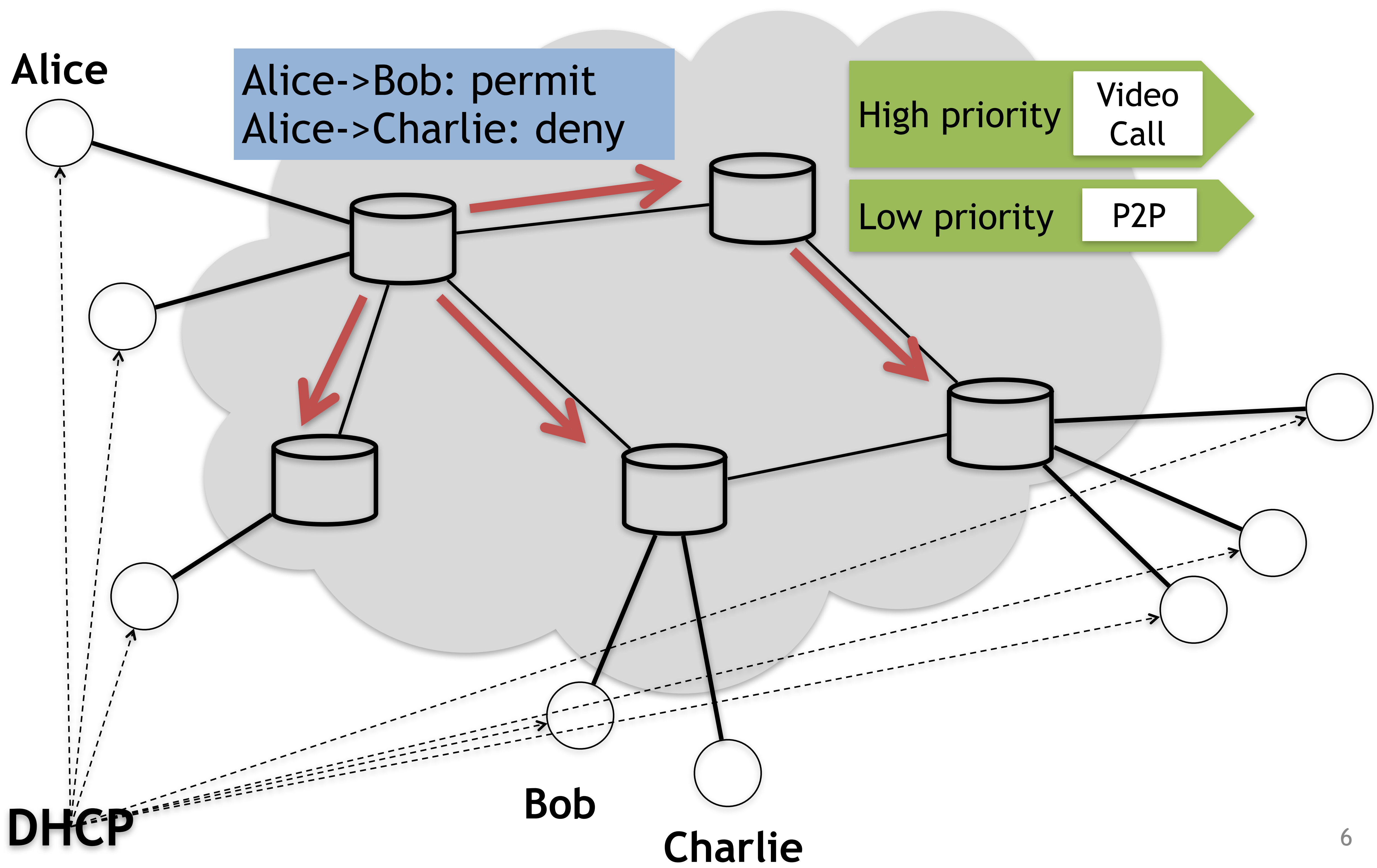
Routing



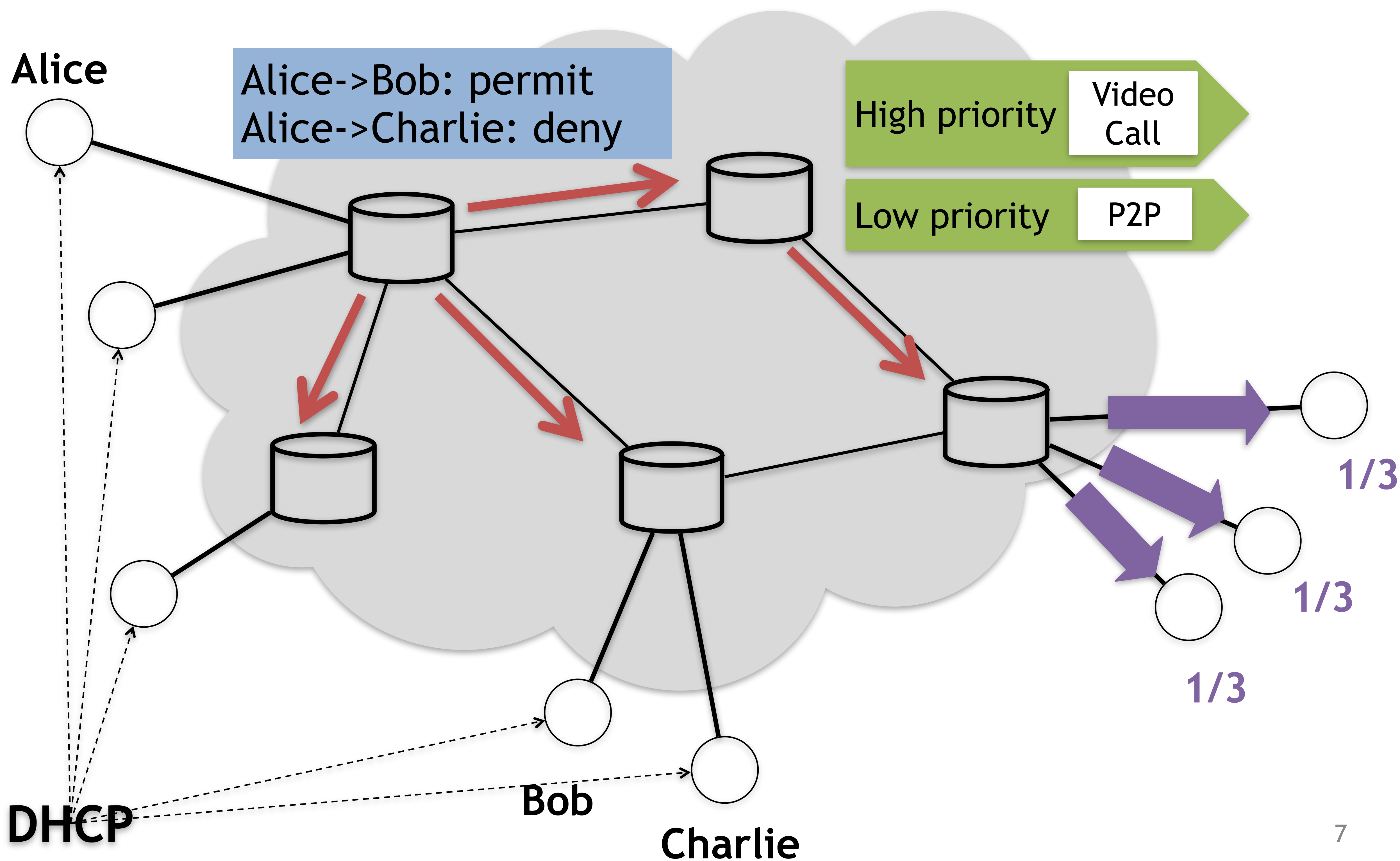
Access Control



Quality-of-Service



Load Balancing

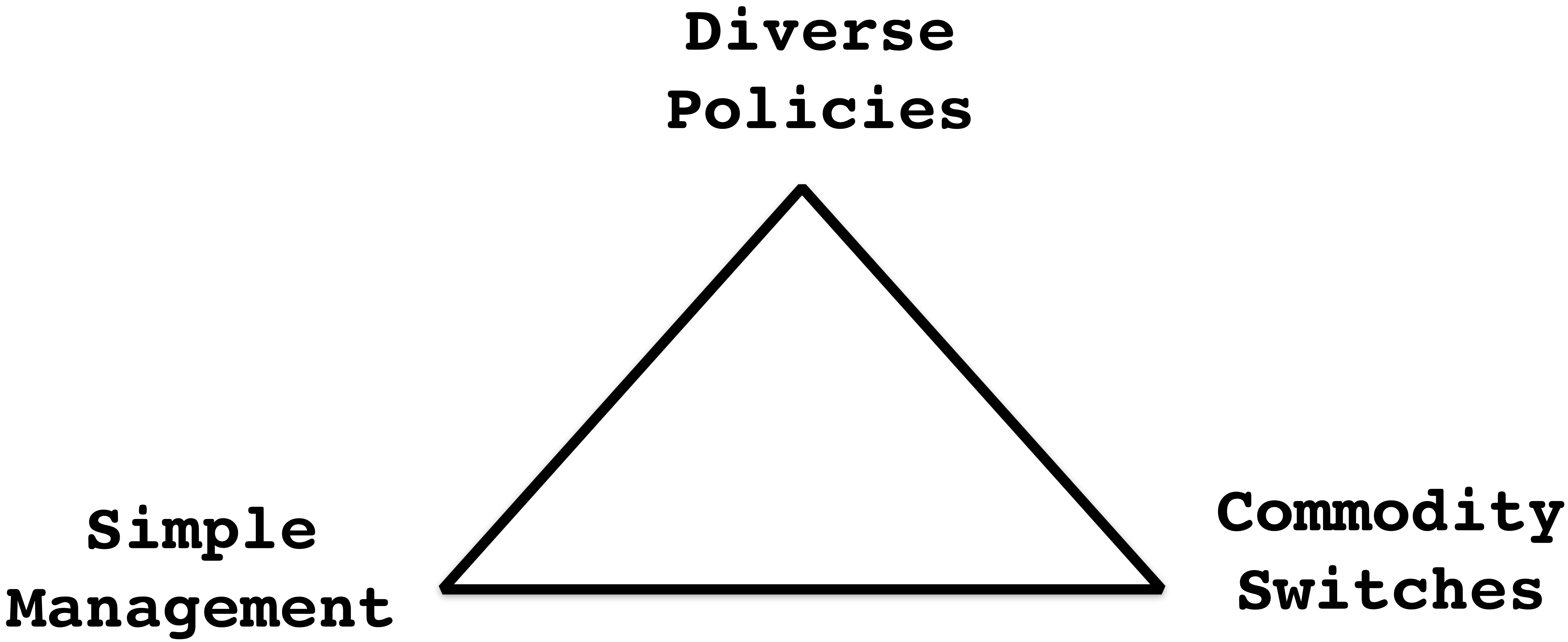


Today’s Network

- Need diverse policies for different purposes

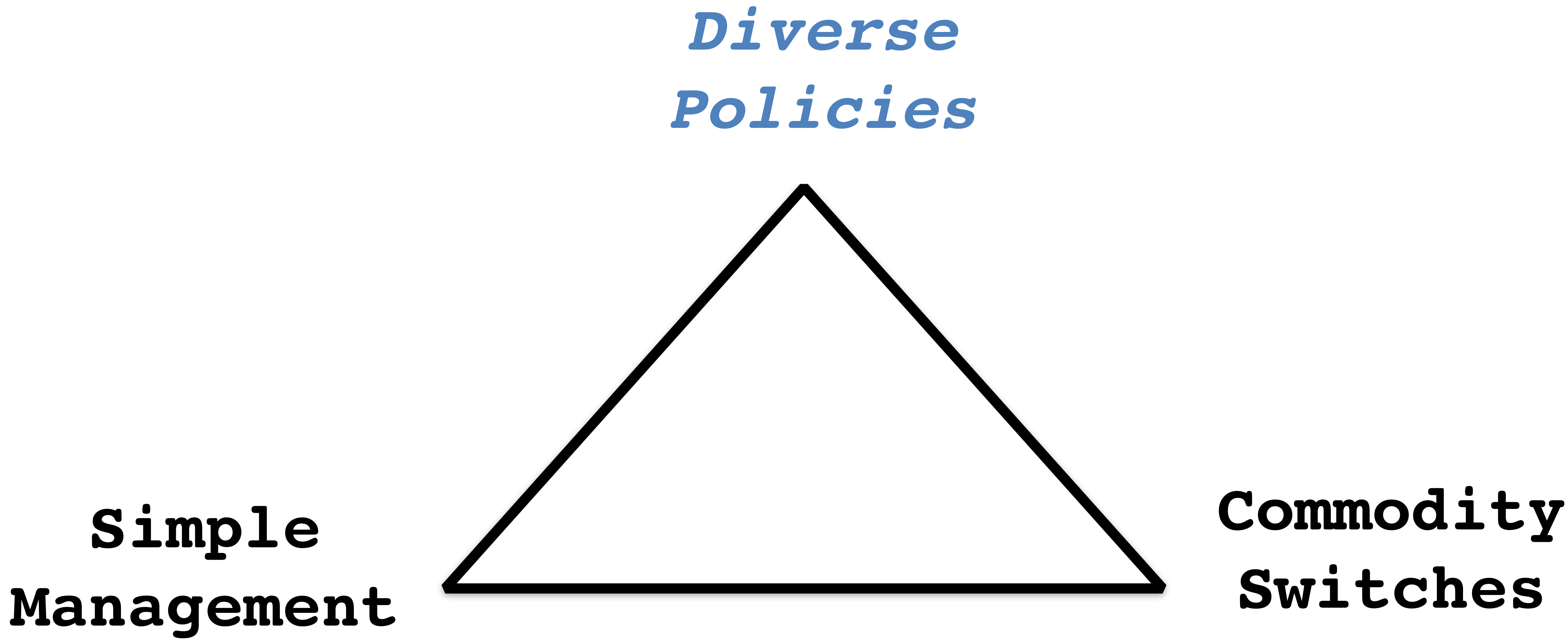
- However…
- *Per-device* configuration
- *Limited* policy support
- *Expensive* devices
    - An F5 Load balancer costs $50K

Our Goals



Support diverse policies with simple management on commodity switches

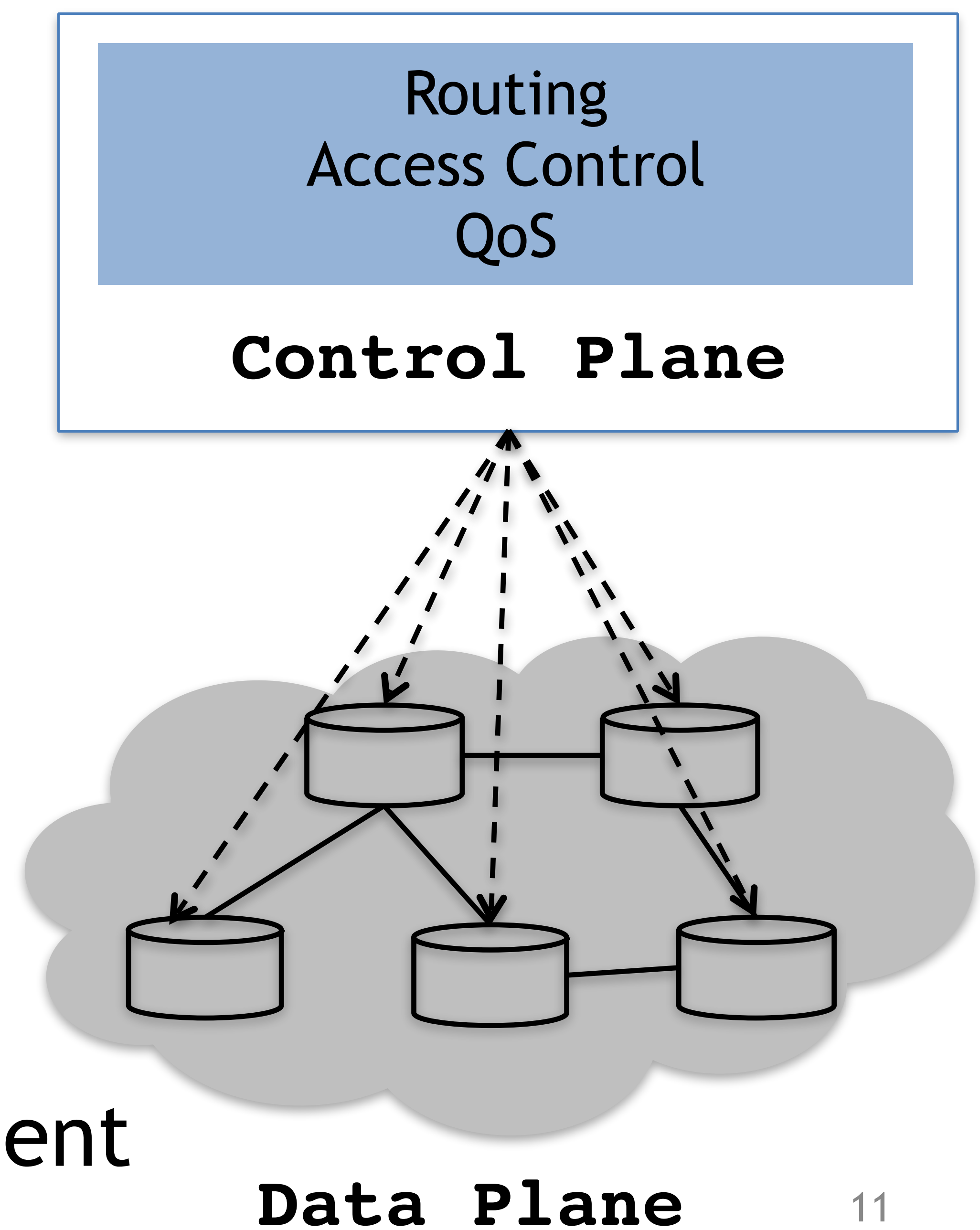
Our Goals



Support diverse policies with simple management on commodity switches

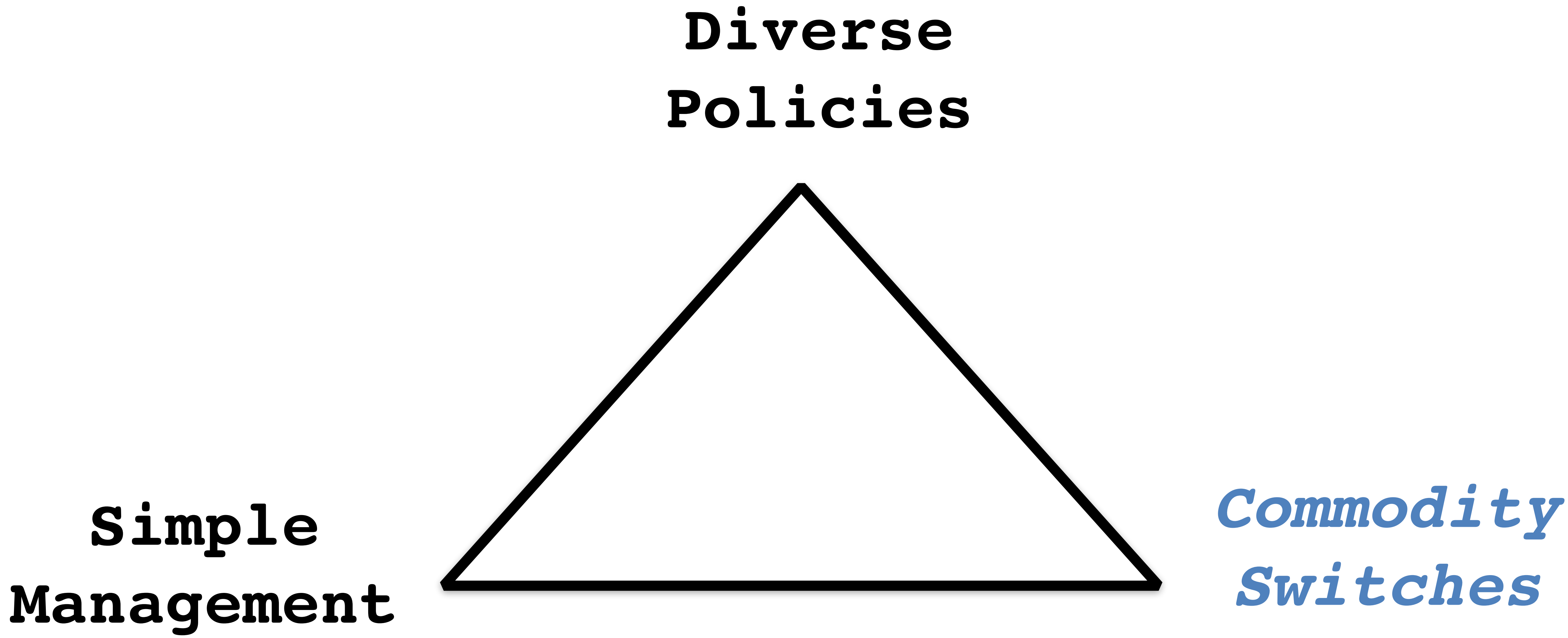
Software-Defined Networks

- Decoupled control and data plane
  - Use standard protocols to program switch rule-tables
- Centralized control
  - network-wide view
- Flexible switch rules
  - diverse policies



Redesign enterprise network management

Our Goals



Support diverse policies with simple management on commodity switches

Commodity Switches in SDN

- Unified open interfaces introduce competition to the market
  - 90% off the market price of vendor switches[1]
- Commodity switches require the controller to directly deal with hardware constraints

[1] Byan Larish, "Software-Defined Networking at the National Security Agency"

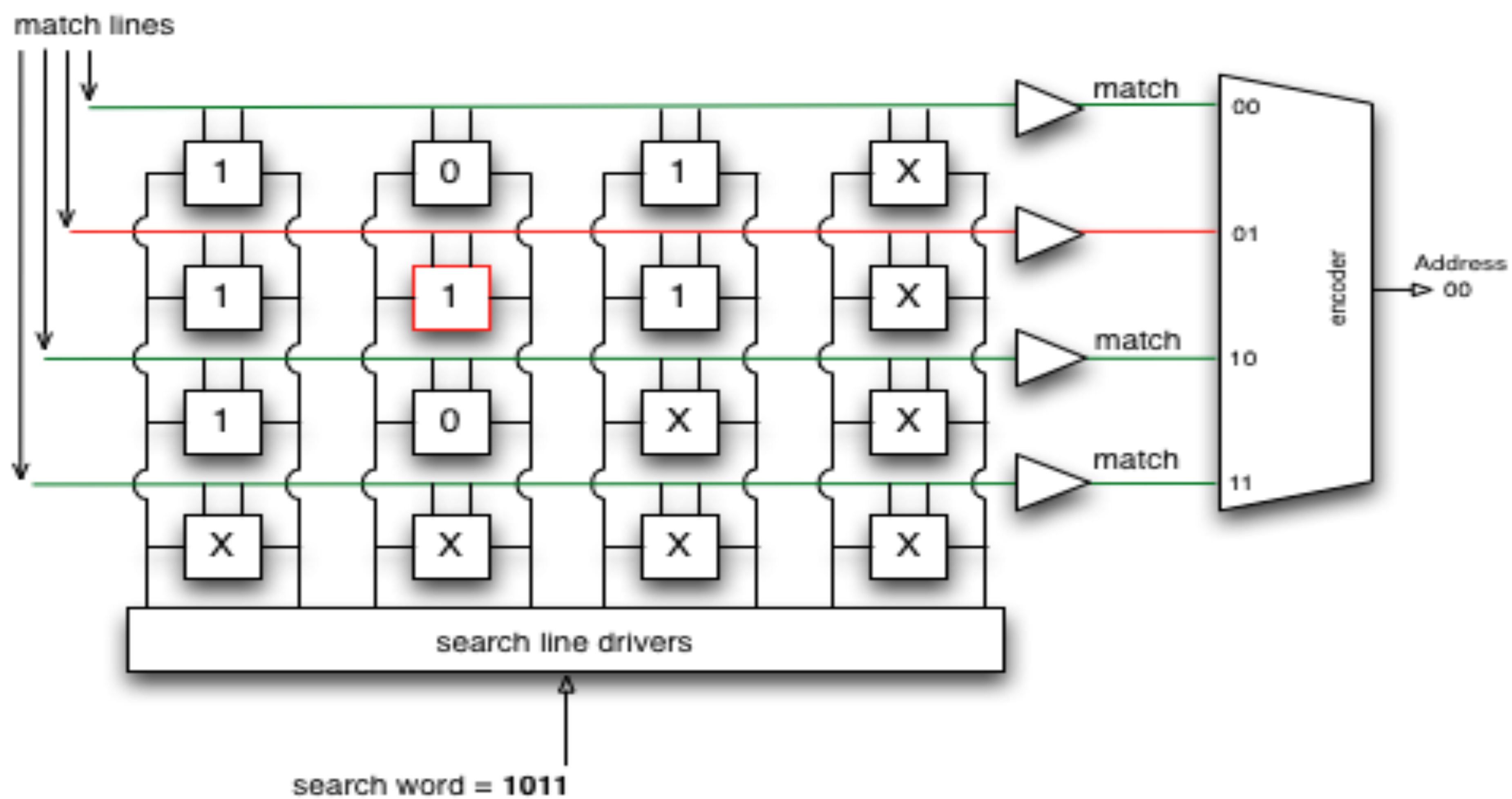
Switch Rule-table

- Each rule contains a match and an action
  – Match
    - e.g., exact, prefix or wildcard
  – Action
    - e.g., forward, drop, rewrite headers
  – E.g., (src_ip = *2, dst_ip = 1.1.1.1): fwd to port 2

- Packets are processed by the 1st matching rule

TCAM

- Wildcard matching on multiple header fields
  - Used for QoS, ACL and routing[1]



[1] Cisco Catalyst 3750 Series Switches. http://www.cisco.com/c/en/us/support/docs/switches/catalyst-3750-series-switches/44921-swdatabase-3750ss-44921.html

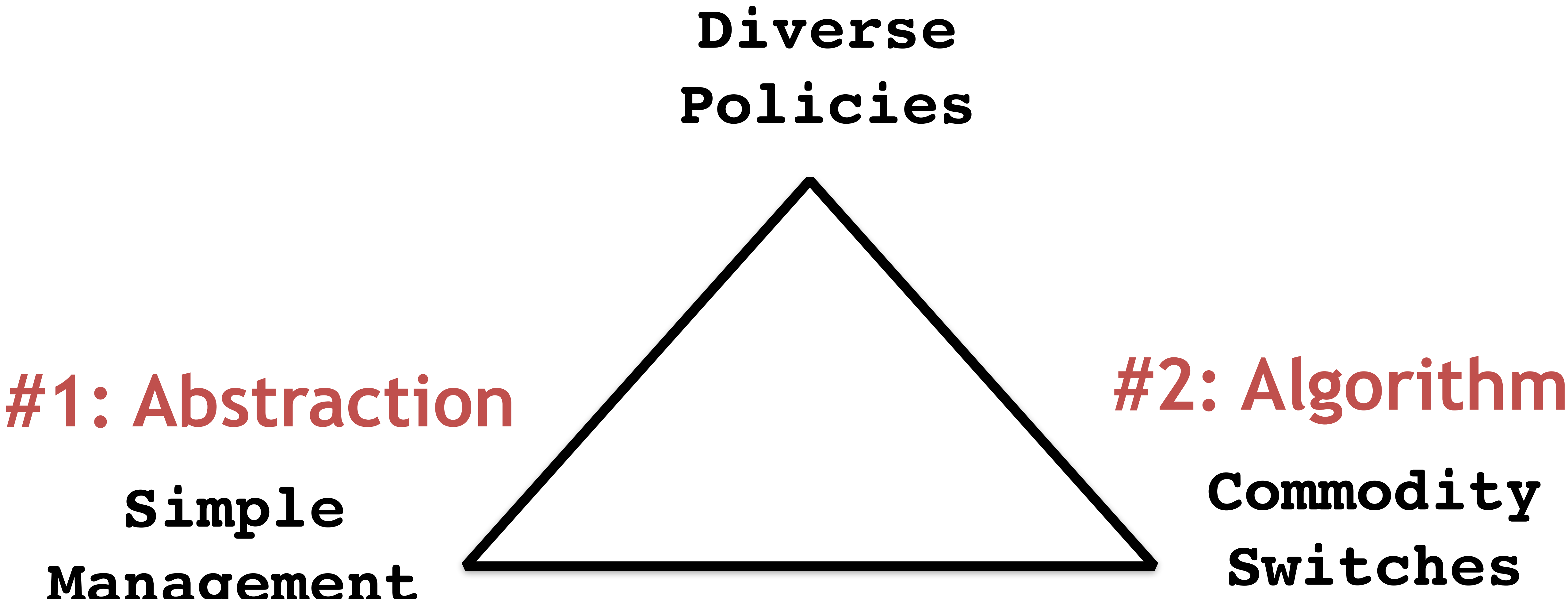
Small Rule-table

- A typical TCAM can hold 500 – 4000 rules[1]

- Power-hungry
- Limited throughput
  - Need parallel TCAM for greater throughput
  - Greater throughput means smaller table

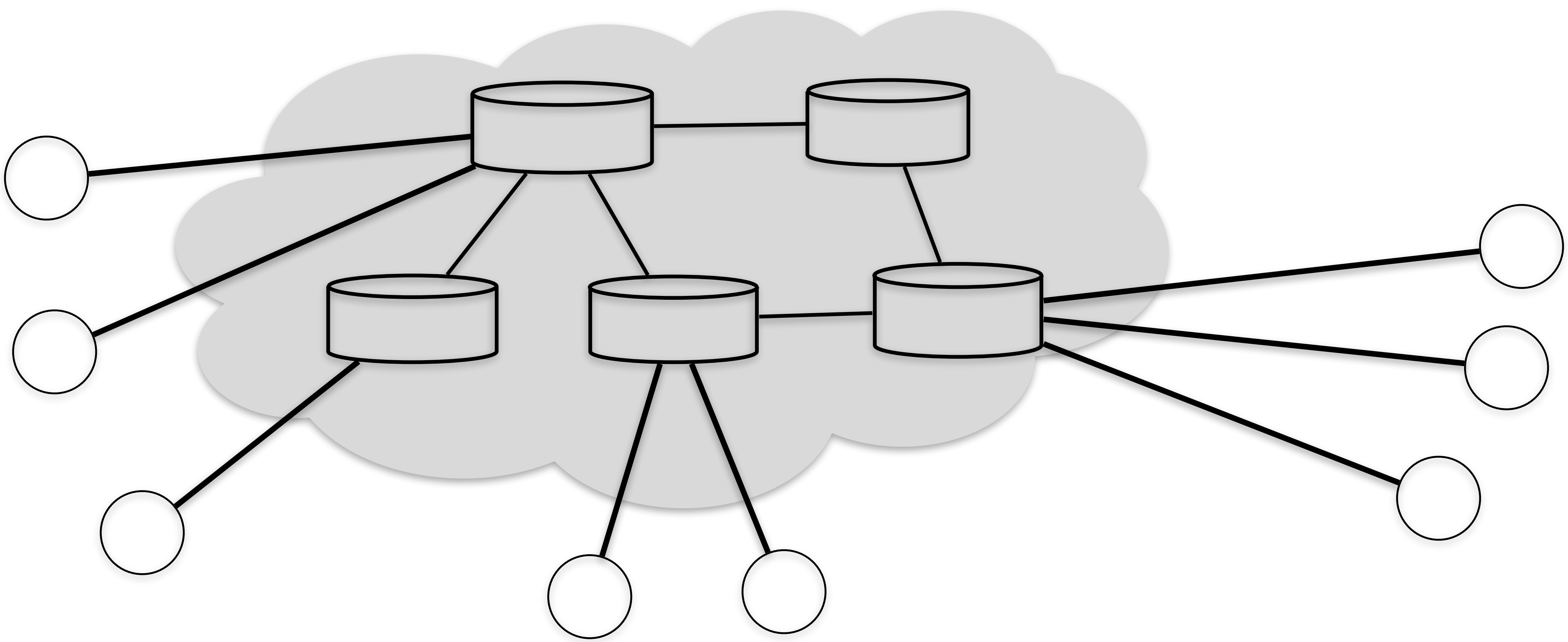
[1] OpenFlow Switches in GENI. https://www.youtube.com/watch?v=RRiOcjAvIsg

Contributions

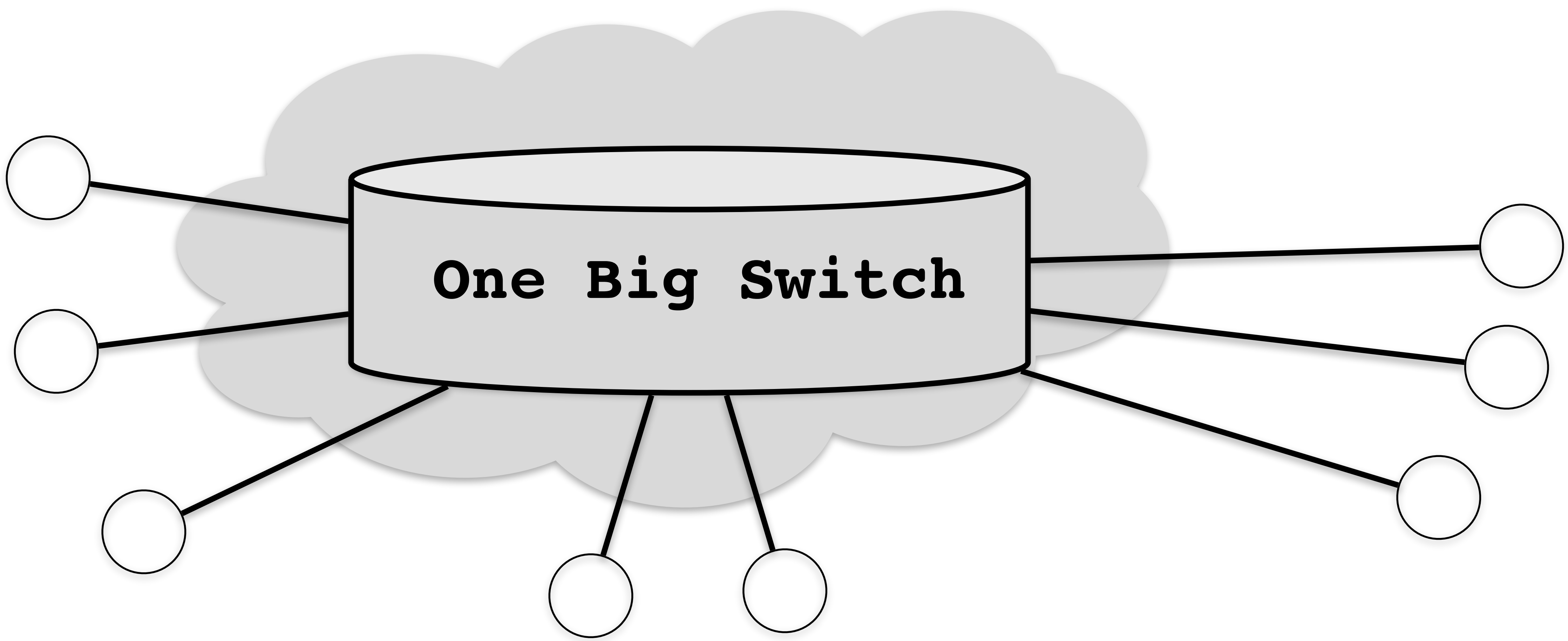


Support diverse policies with simple management on commodity switches

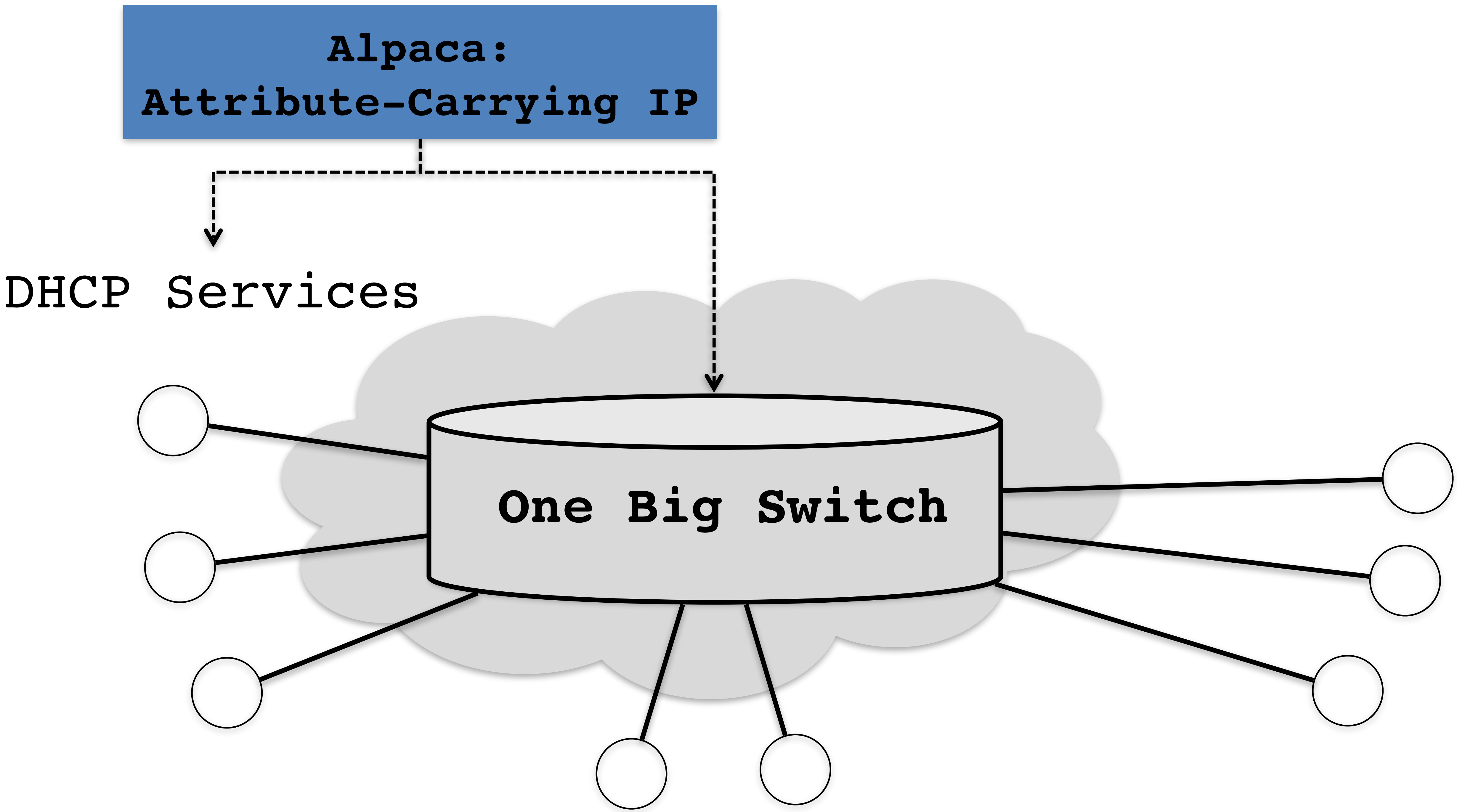
My proposal (One-Big-Switch)



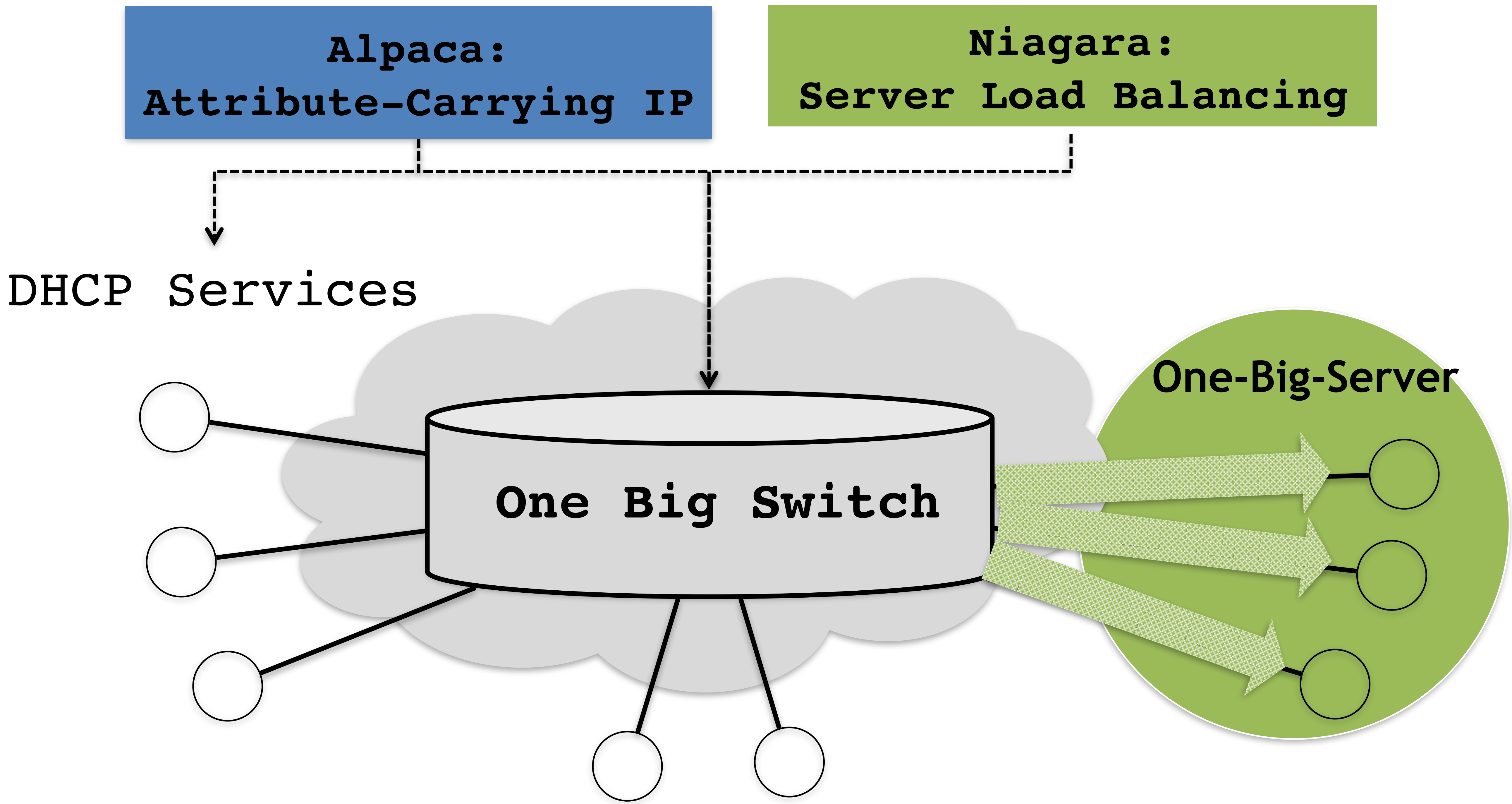
My proposal (One-Big-Switch)



My proposal (Attribute-Carrying IP)



My proposal (One-Big-Server)



Thesis Overview

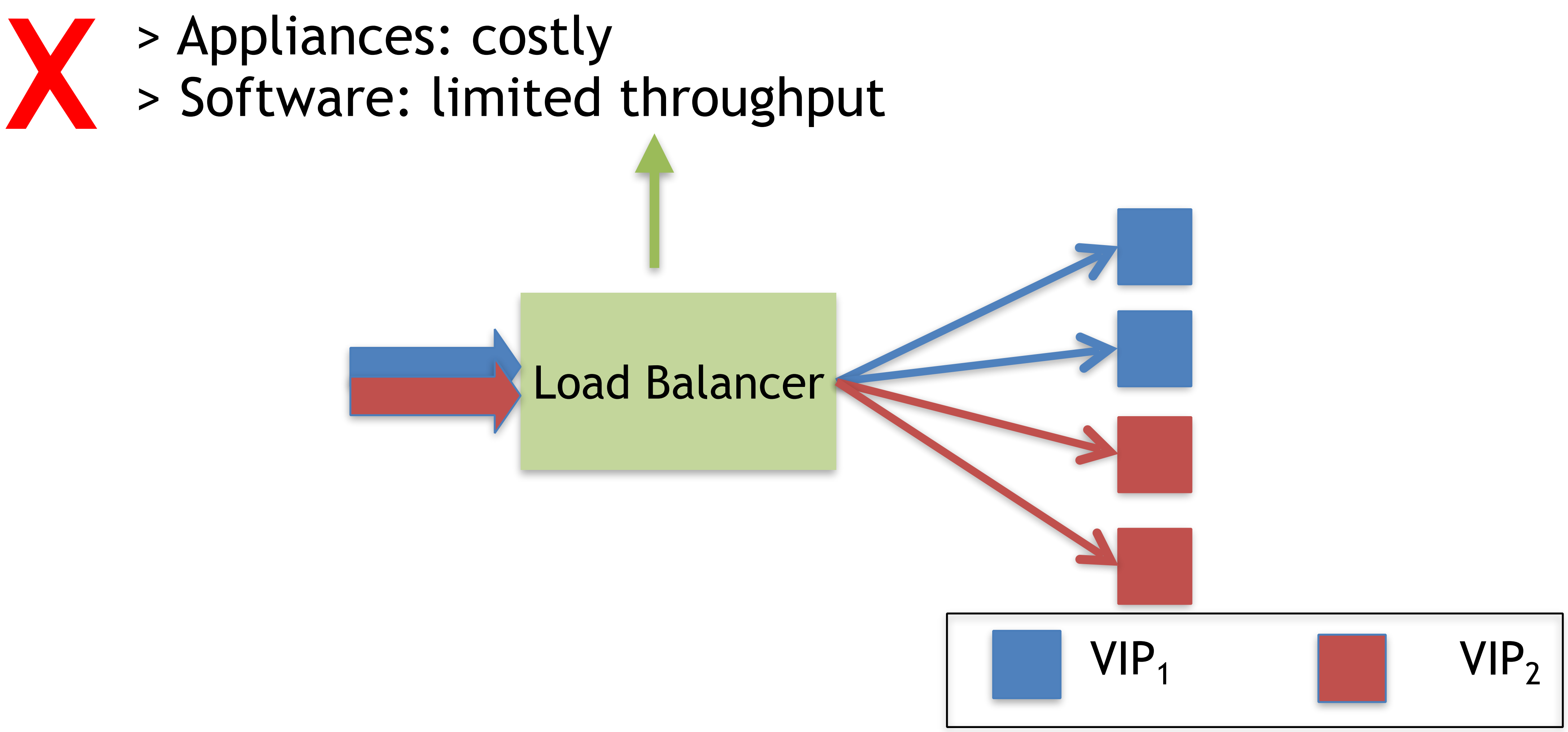
Name	Abstraction	Publication
One-Big-Switch	Configure One-Big-Switch	CoNEXT'13
Niagara	Configure One-Big-Server	CoNEXT'15
Alpaca	Enforce attribute-based network policies	CoNEXT'15

Niagara: Efficient Traffic Splitting on Commodity Switches

Nanxi Kang, Monia Ghobadi,
John Reumann, Alexander Shraer,
Jennifer Rexford

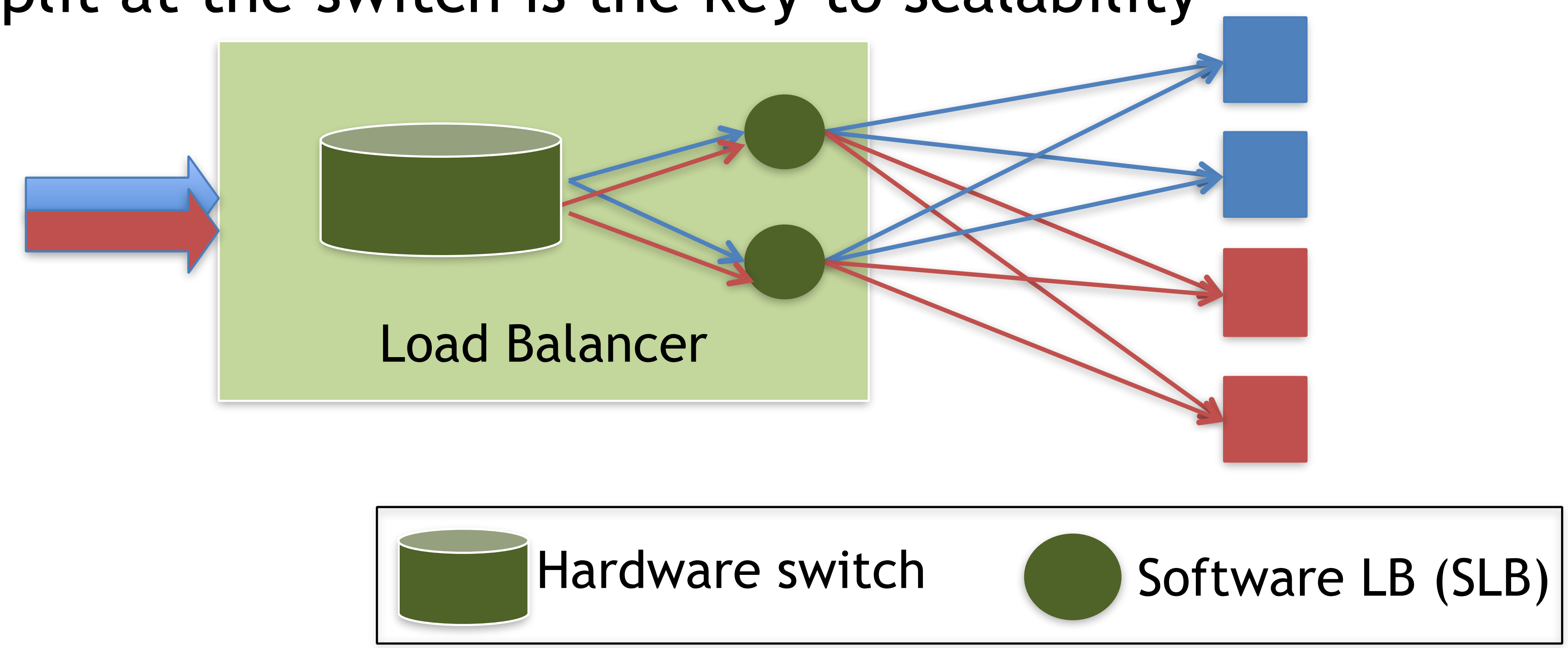
Service load balancing

- A network hosts many services (Virtual-IPs)
- Each service is replicated for greater throughput
- A load balancer spreads traffic over service instances



Hierarchical Load Balancer

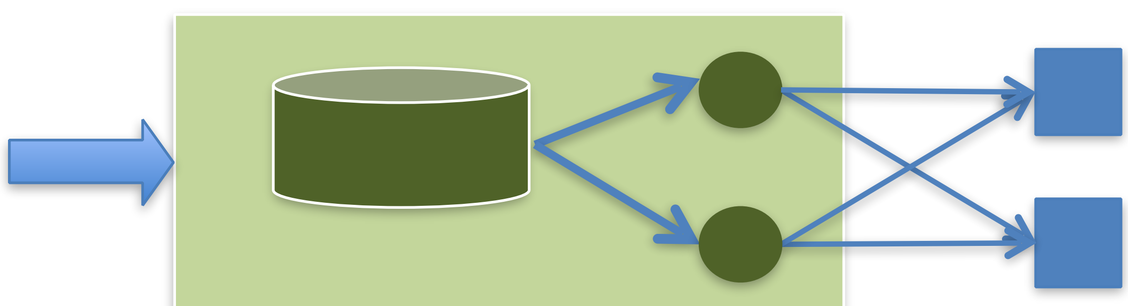
- Modern LB scales out with a hierarchy[1][2]
  - A hardware switch split traffic over SLBs
  - SLBs direct requests to servers
    - SLBs track connections and monitor health of servers
- Traffic split at the switch is the key to scalability



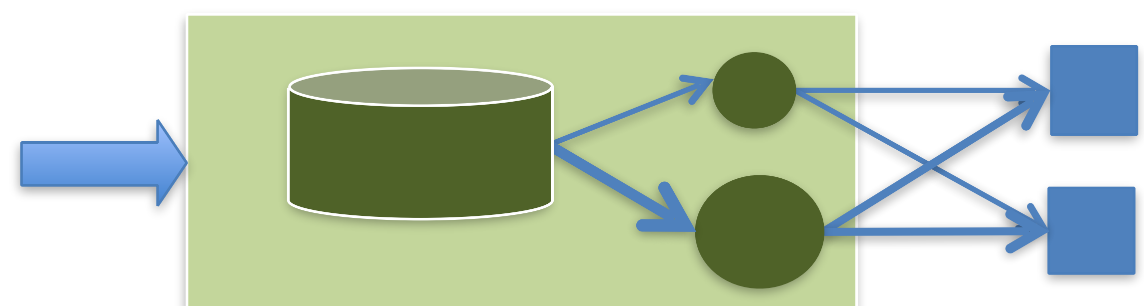
[1]: Duet (SIGCOMM'14)
[2]: Ananta (SIGCOMM'13)

Accurate Weighted Split

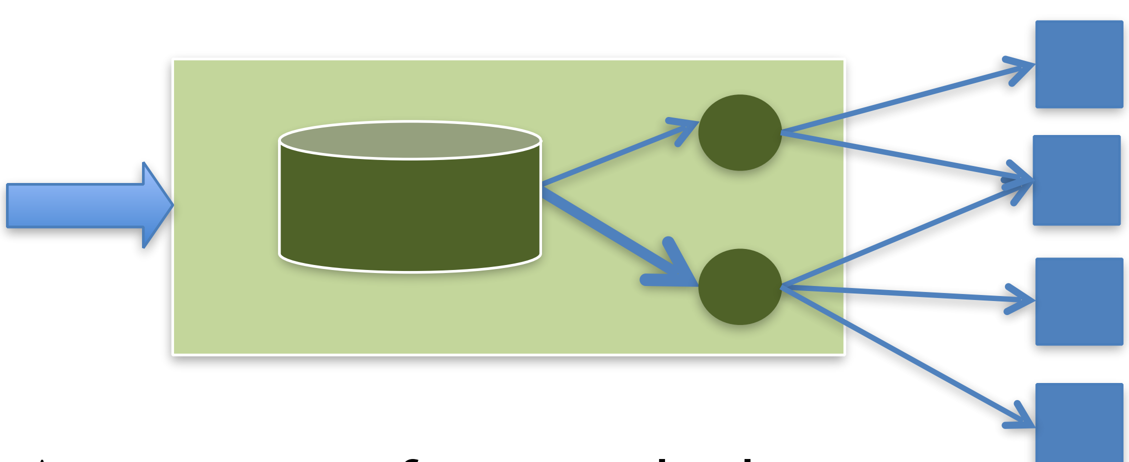
- SLBs are weighted in the traffic split
  - Throughput of SLB
  - Deployment of VIP
  - Failures, or recovery



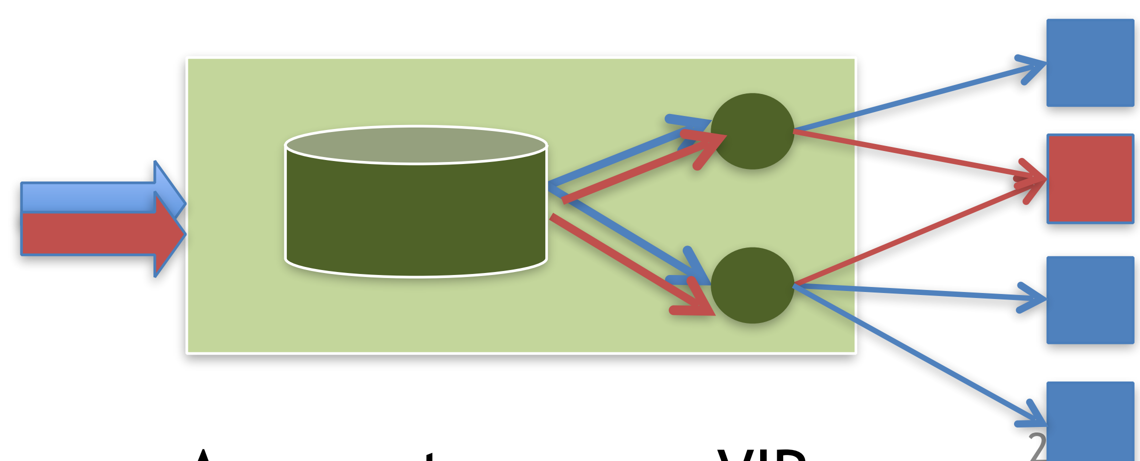
Symmetry



Asymmetry of LB



Asymmetry of server deployment



Asymmetry across VIPs

Existing hash-based split

- Hash-based ECMP
  - Hash 5-tuple header fields of packets
  - Dst_SLB = Hash_value mod #SLBs

DstIP	*Action*
1.1.1.1	*Hash, ECMP Group 1*
…	…

ECMP	*Mod*	*Action*
1	*0*	*Forward to 1*
1	*1*	*Forward to 2*
…	…	…

Equal split over two SLBs

Existing hash-based split

- Hash-based ECMP
  - Hash 5-tuple header fields of packets
  - Dst_SLB = Hash_value mod #SLBs
- WCMP gives unequal split by repeating

DstIP	*Action*
1.1.1.1	*Hash, ECMP Group 1*
…	…

ECMP	*Mod*	*Action*
1	*0*	*Forward to 1*
1	*1*	*Forward to 2*
1	*2*	*Forward to 2*
…	…	…

(1/3, 2/3) is achieved by adding the second SLB twice

Existing hash-based split

- ECMP and WCMP only split the *flowspace* equally
  - WCMP cannot scale to many VIPs, due to the rule-table constraint
    - e.g., (1/8, 7/8) takes 8 rules

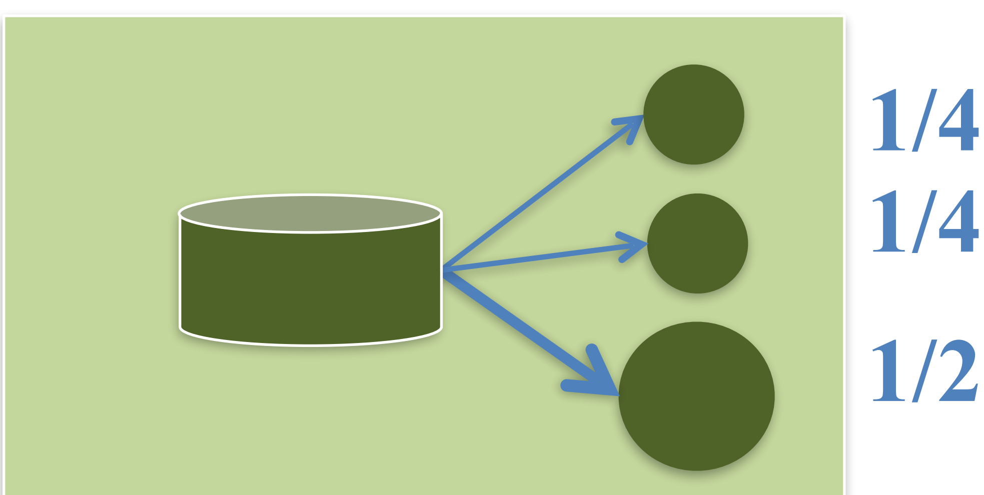
DstIP	*Action*
1.1.1.1	*Hash, ECMP Group 1*
…	…

ECMP	*Mod*	*Action*
1	*0*	*Forward to 1*
1	*1*	*Forward to 2*
1	*2*	*Forward to 2*
…	…	…

Limited rule-table size (TCAM)

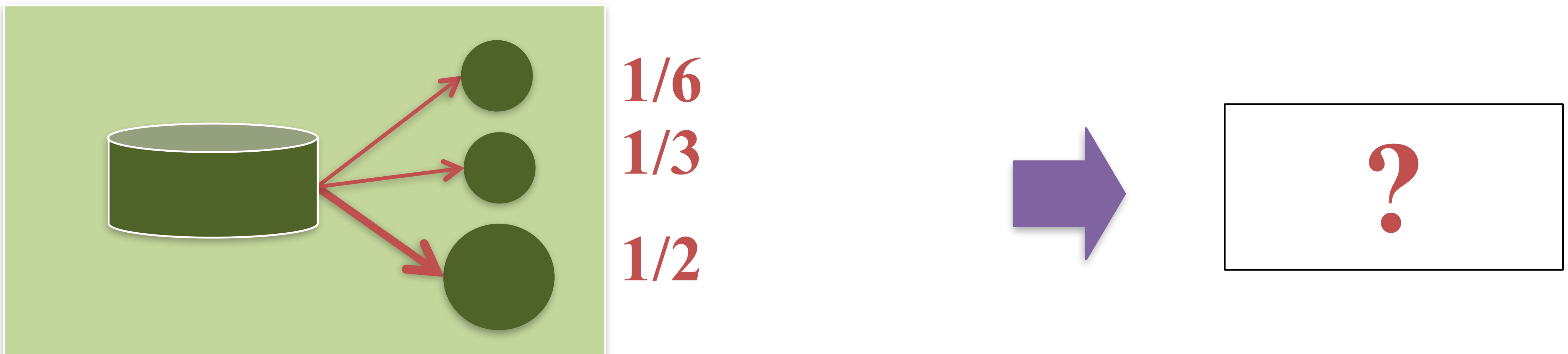
A wildcard-matching approach

- OpenFlow + TCAM
  - OpenFlow : program rules at switches
  - TCAM : support wildcard matching on packet headers
- A starting example
  - Single service : VIP = 1.1.1.1
  - Weight vector: (1/4, 1/4, 1/2)



Match (dst_ip, src_ip)		*Action*
1.1.1.1	**00*	*Forward to 1*
1.1.1.1	**01*	*Forward to 2*
1.1.1.1	***	*Forward to 3*

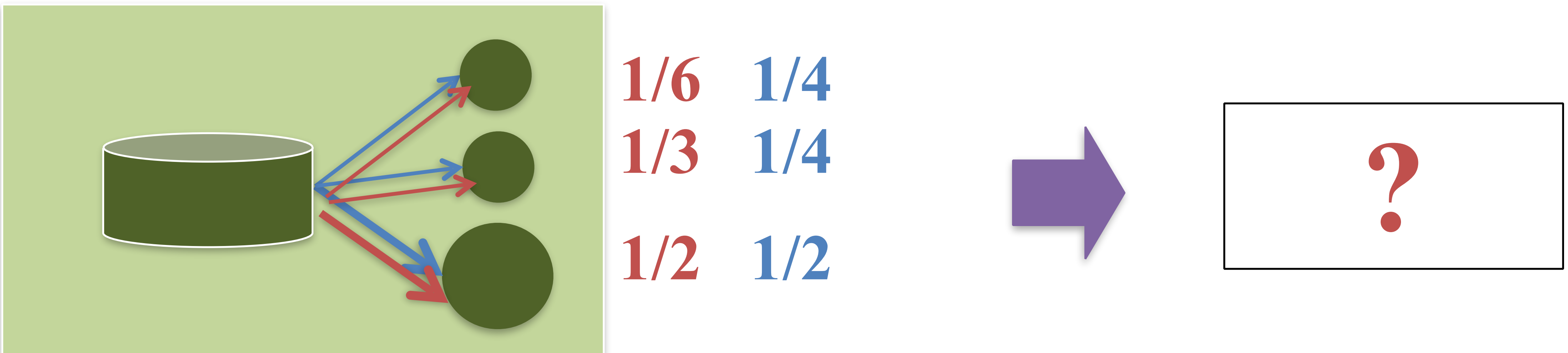
Challenges: Accuracy



- How rules achieve the weight vector of a VIP?
  - Arbitrary weights
  - Non-uniform traffic distribution over flowspace

#bytes or #connections

Challenges: Accuracy

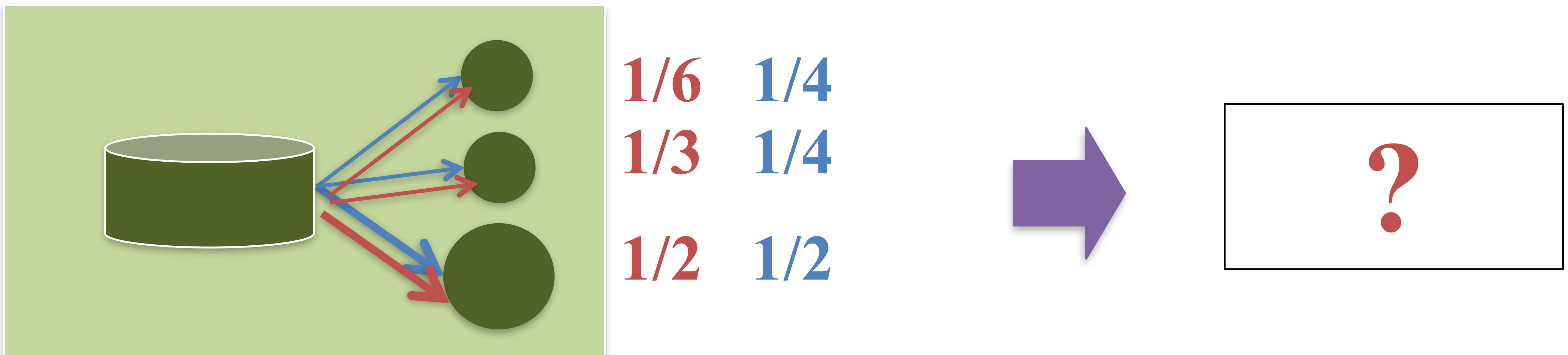


- How rules achieve the weight vector of a VIP?
  - Arbitrary weights
  - Non-uniform traffic distribution over flowspace
- How VIPs (100 -10k) share a rule table (~4,000)?

1. Approximate weights with rules
2. Packing rules for multiple VIPs
3. Sharing default rules
4. Grouping similar VIPs

Niagara: rule generation algorithms!

Challenges: Accuracy

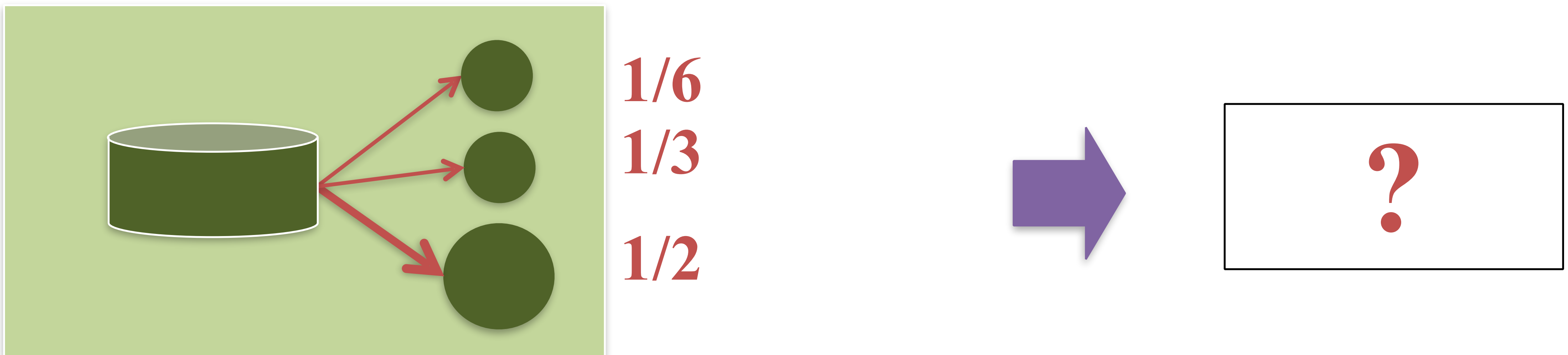


- How rules achieve the weight vector of a VIP?
  - Arbitrary weights **1. Approximate weights with rules**
  - Non-uniform traffic distribution over flowspace
- How VIPs (100 -10k) share a rule table (~4,000)?

2. Packing rules for multiple VIPs
3. Sharing default rules
4. Grouping similar VIPs

Niagara: rule generation algorithms!

Basic ideas

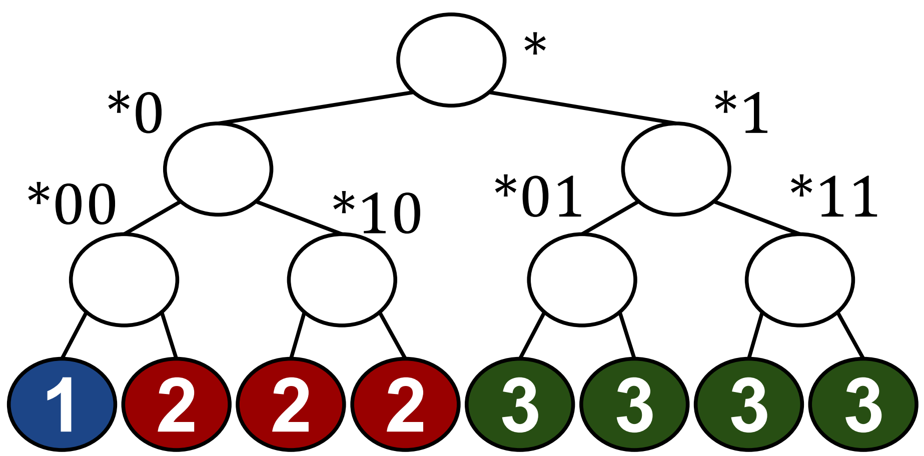


- Uniform traffic distribution
  - e.g., *000 represents 1/8 traffic
- "*Approximation*" of the weight vector?
  - Header matching discretizes portions of traffic
  - Use error bound to quantify approximations
- $1/3 \approx 1/8 + 1/4$

Match	*Action*
**100*	*Forward to 1*
**10*	*Forward to 1*

Naïve solution

- Bin pack suffixes
  - Round weights to multiples of $1/2^k$
  - When k = 3, (1/6, 1/3, 1/2) ≈ (1/8, 3/8, 4/8)



*000	Fwd to 1
*100	Fwd to 2
*10	Fwd to 2
*1	Fwd to 3

*000	Fwd to 1
*0	Fwd to 2
*	Fwd to 3

- Observation
  - 1/3 ≈ 3/8 = **1/2 - 1/8** saves one rule
  - Use ***subtraction*** and ***rule priority***

Approximation with $1/2^k$

- Approximate a weight with powers-of-two terms
  - 1/2, 1/4, 1/8, …
- Start with

#	Weight w	Approx v	Error v - w
1	1/6	0	-1/6
2	1/3	0	-1/3
3	1/2	1	1/2

Under-approximated

Over-approximated

Approximation with $1/2^k$

- Reduce errors *iteratively*
- In each round, move $1/2^k$ from an over-approximated weight to an under-approximation weight

#	Weight w	Approx v	Error $v - w$	
1	1/6	0	-1/6	
2	1/3	0	-1/3	Under-approximated
3	1/2	1	1/2	Over-approximated

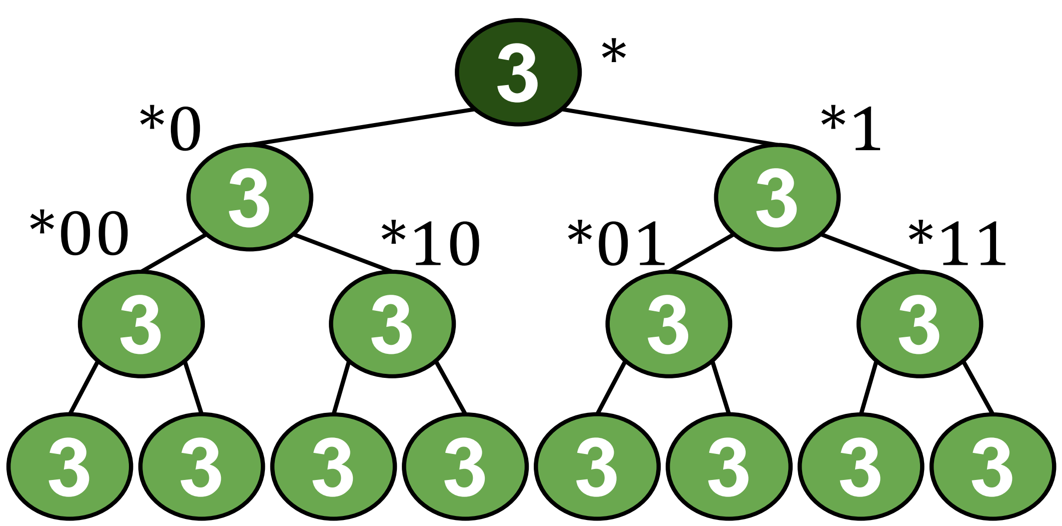
move 1/2

1	1/6	0	-1/6
2	1/3	1/2	-1/3 + 1/2 = 1/6
3	1/2	1 - 1/2	1/2 - 1/2 = 0

Initial approximation

#	Weight	Approx	Error
1	1/6	0	-1/6
2	1/3	0	-1/3
3	1/2	1	1/2

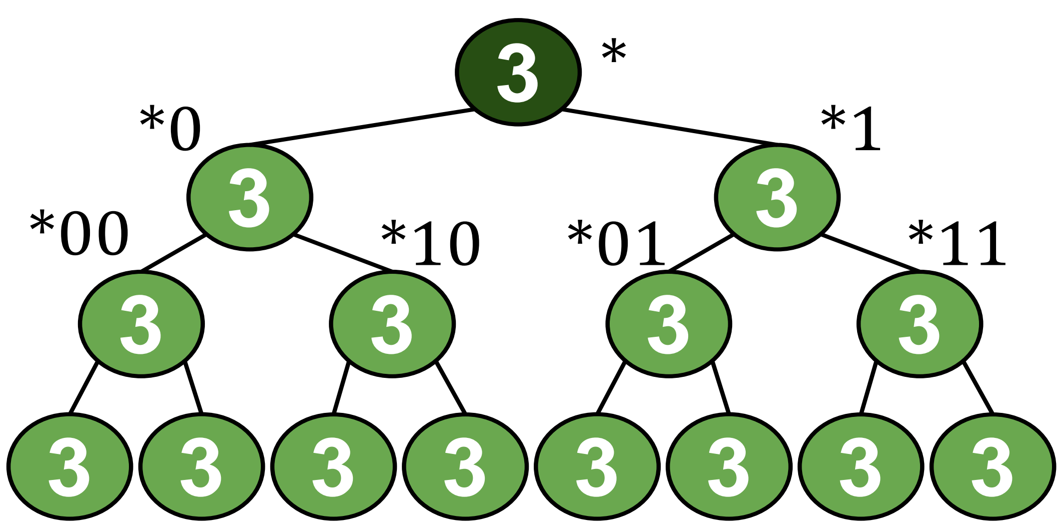
*	Fwd to 3

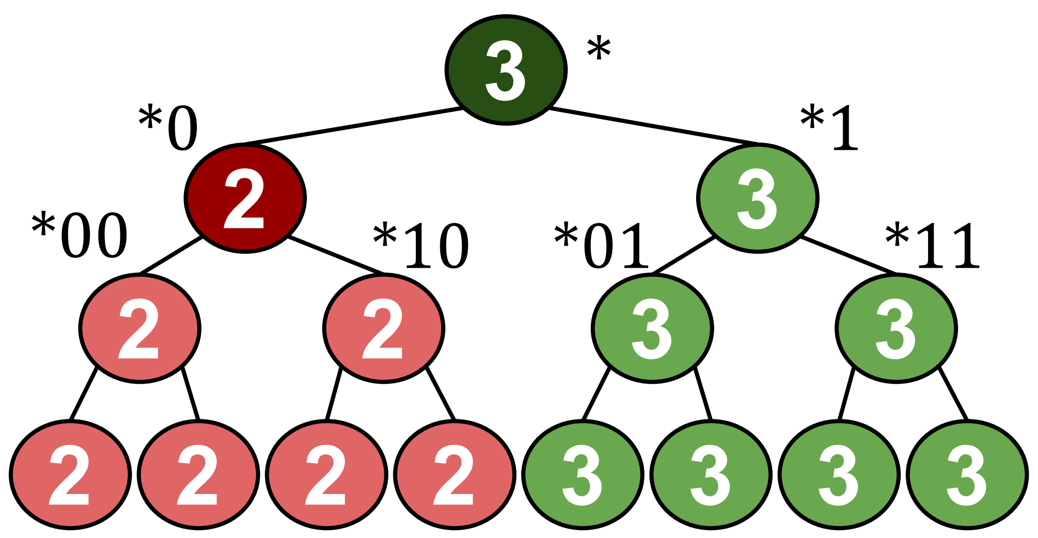


Move 1/2 from W_3 to W_2

#	Weight	Approx	Error
1	*1/6*	*0*	*-1/6*
2	*1/3*	*1/2*	*1/6*
3	*1/2*	*1 -1/2*	*0*

**0*	*Fwd to 2*
***	*Fwd to 3*



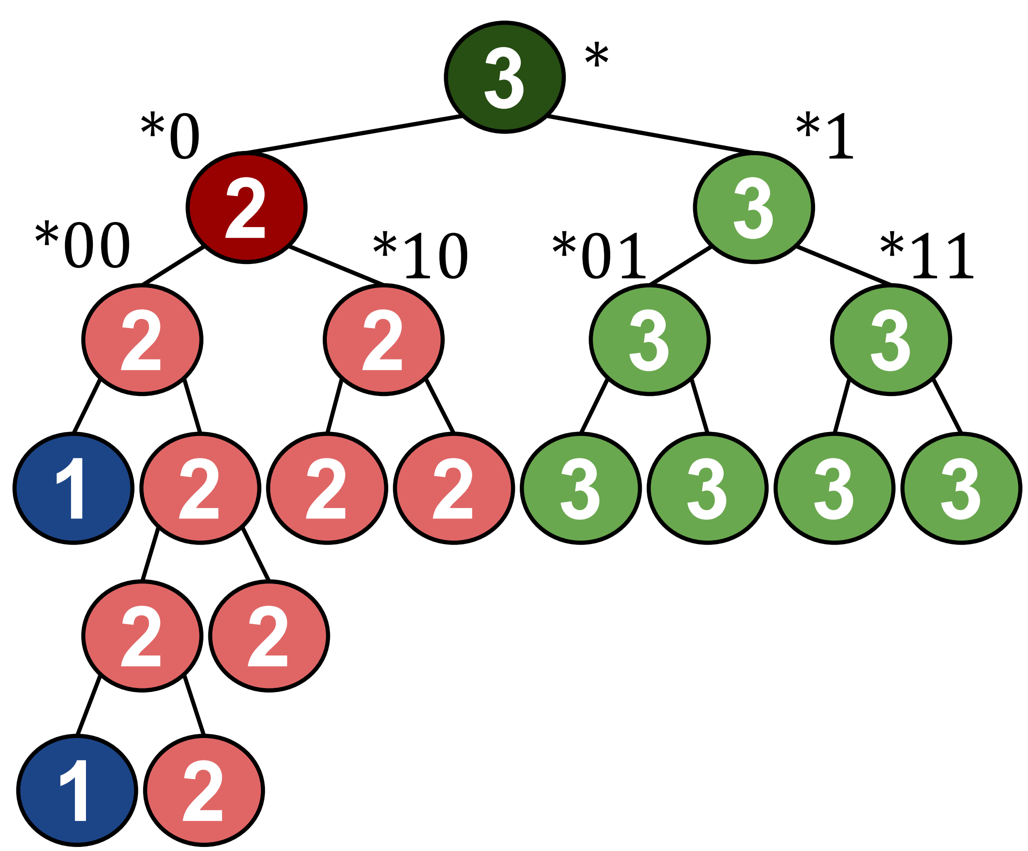


Final result

#	Weight	Approx
1	1/6	1/8 +1/32
2	1/3	1/2 -1/8 -1/32
3	1/2	1 -1/2

*00100	Fwd to 1
*000	Fwd to 1
*0	Fwd to 2
*	Fwd to 3

Reduce errors exponentially!



Truncation

- Limited rule-table size?
  - Truncation, i.e., stop iterations earlier
- Imbalance: $\Sigma\,|error_i|\;/\;2$
  - Total over-approximation

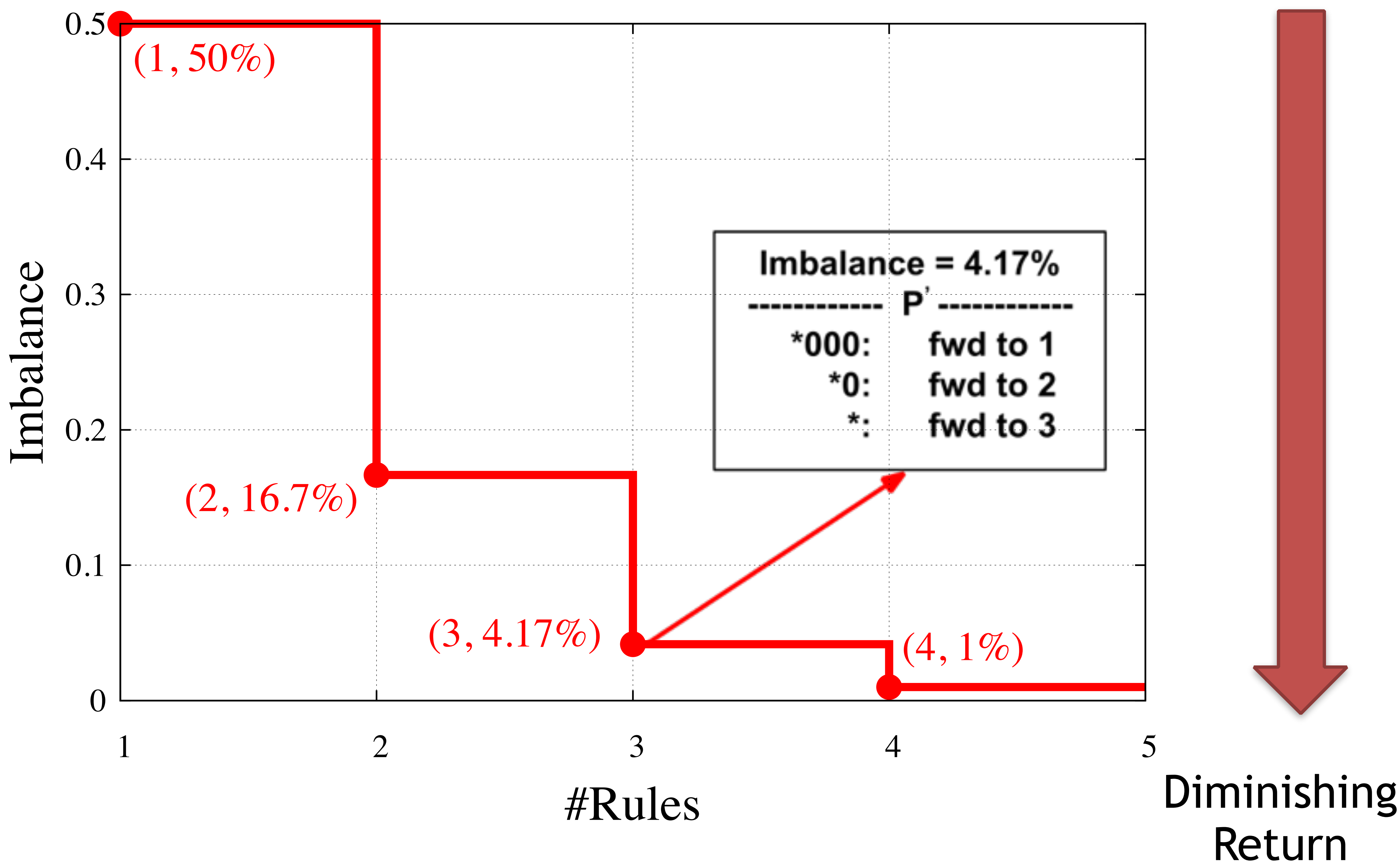
**00100*	*Fwd to 1*
**000*	*Fwd to 1*
**0*	*Fwd to 2*
***	*Fwd to 3*

Full rules
Imbalance = 1%

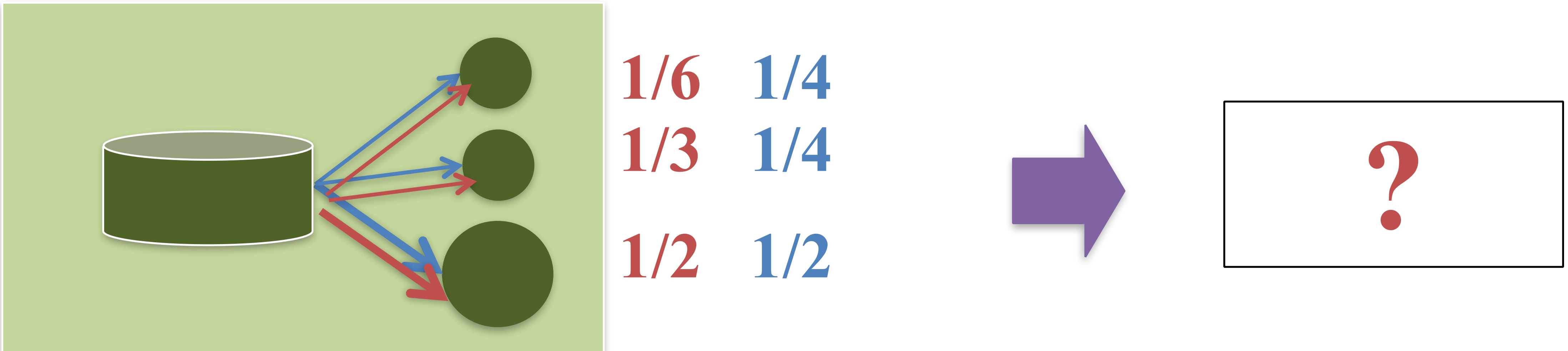
**000*	*Fwd to 1*
**0*	*Fwd to 2*
***	*Fwd to 3*

Rules after truncation
Imbalance = 4%

Stairstep: #Rules v.s. Imbalance



Multiple VIPs

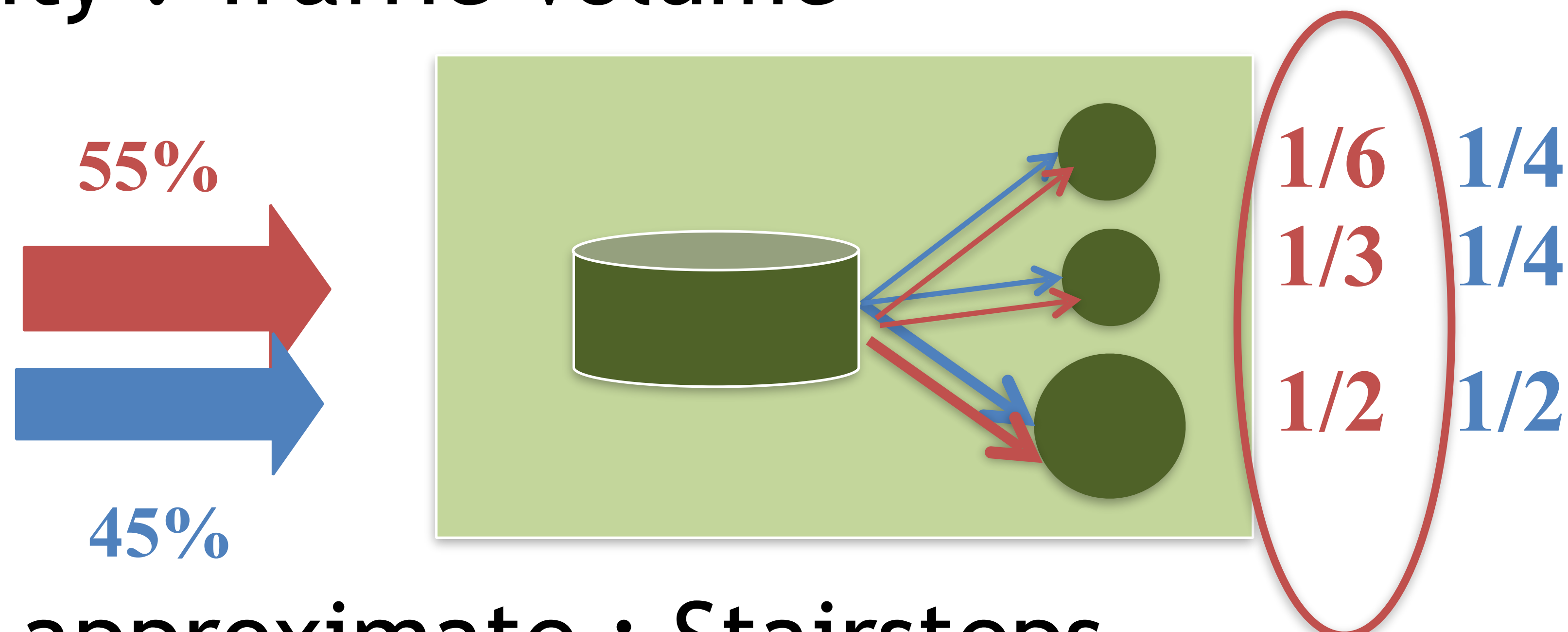


- How rules achieve the weight vector of a VIP?
  - Arbitrary weights
  - Non-uniform traffic distribution over flowspace
- How VIPs (100-10k) share a rule table (~4,000)?

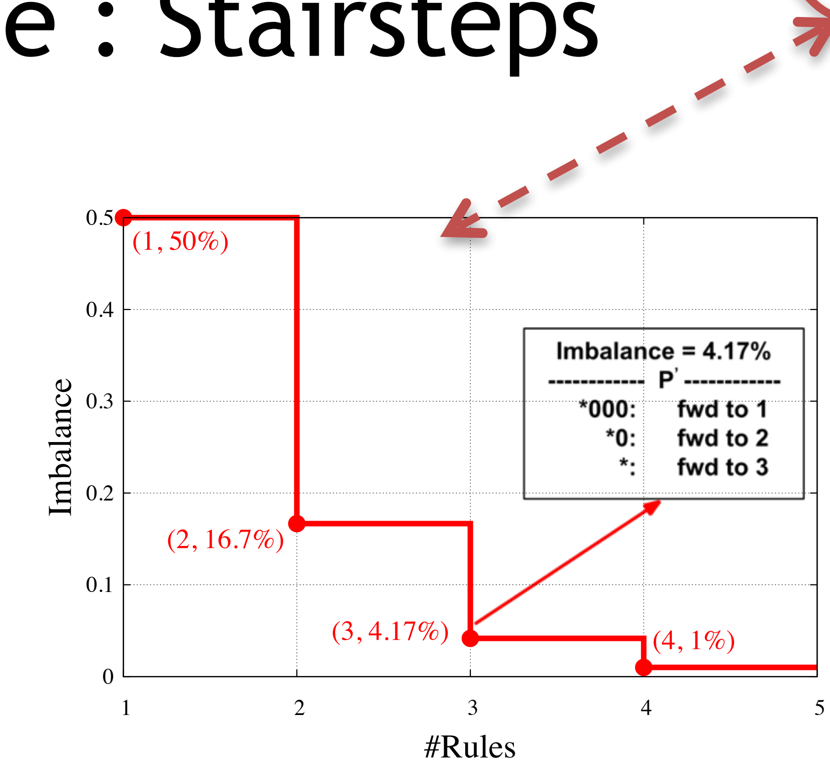
Minimize $\Sigma\, traffic\_volume_j \; x \; \Sigma\, |error_{ij}| / 2$

Characteristics of VIPs

- Popularity : Traffic Volume



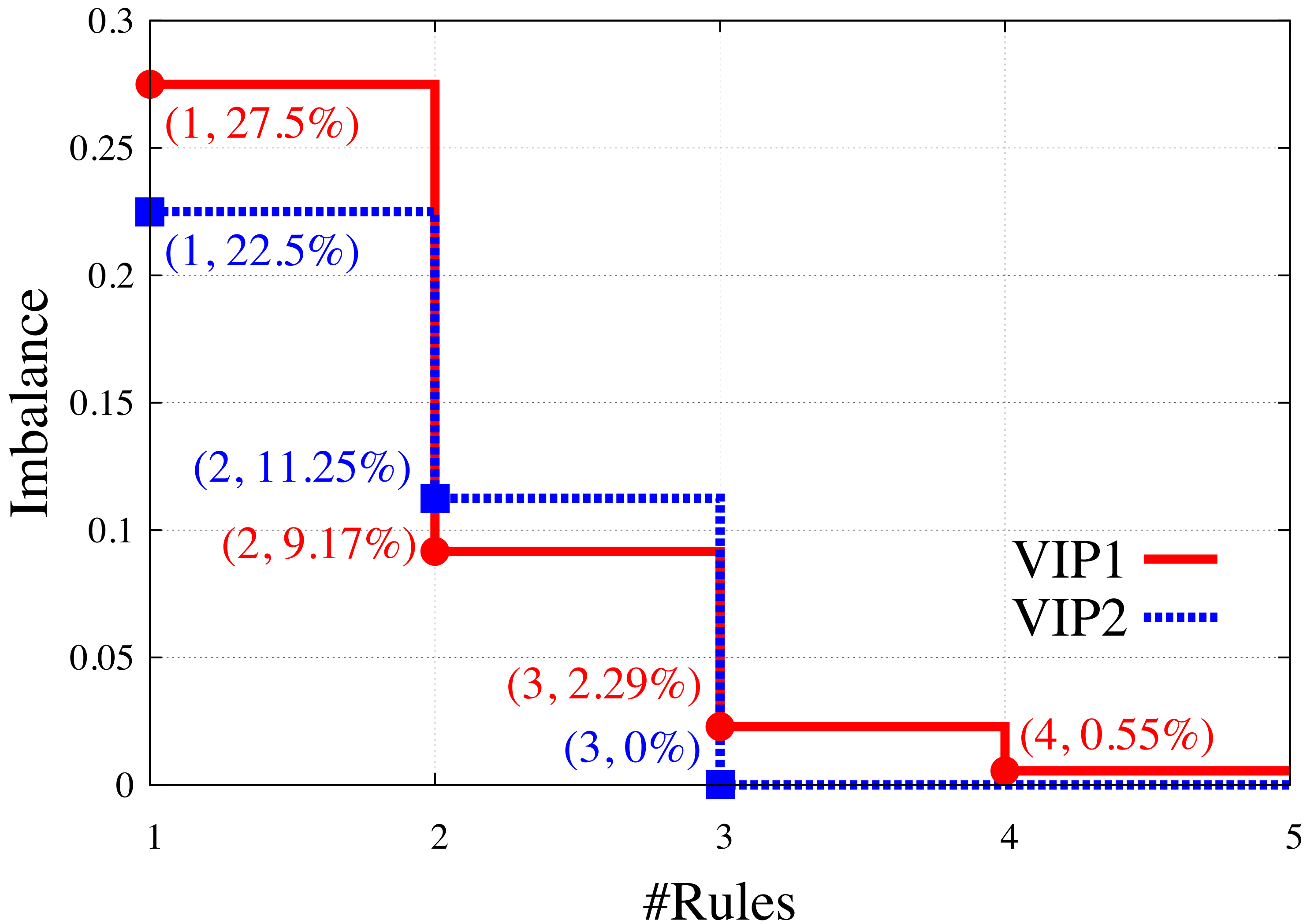
- Easy-to-approximate : Stairsteps



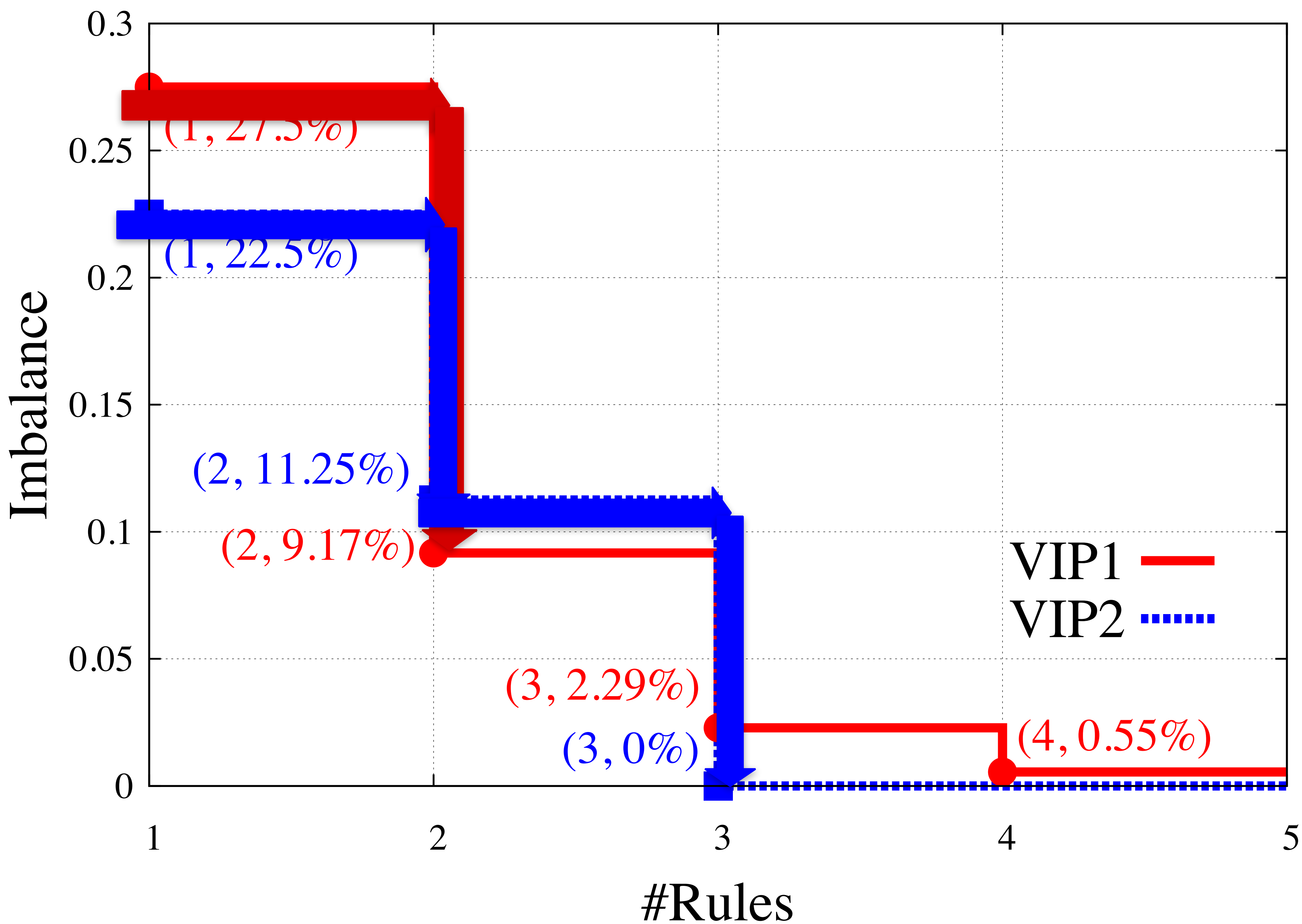
Stairsteps

- Each stairstep is scaled by its traffic volume

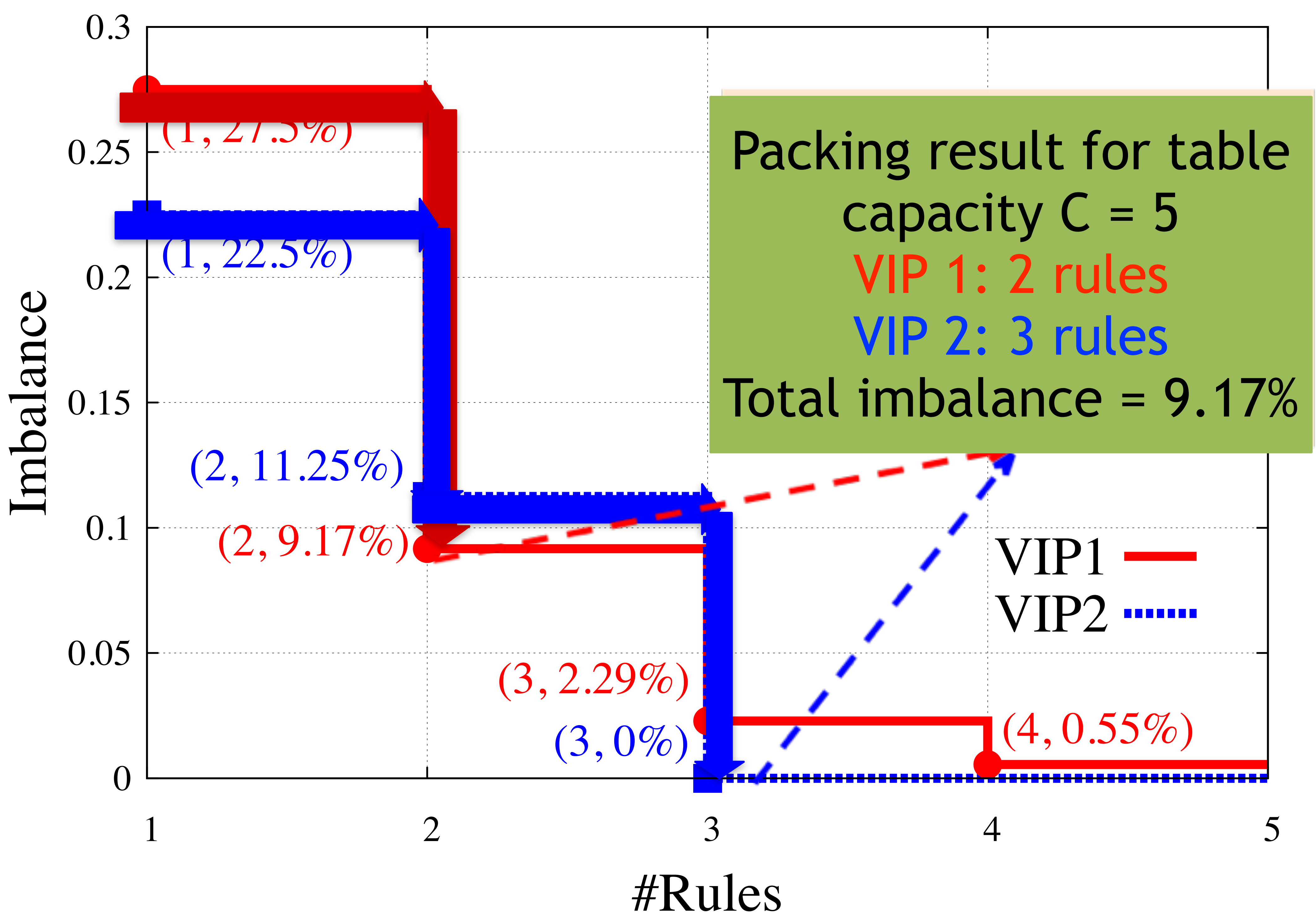
VIP_1	***55%***	***(1/6, 1/3, 1/2)***
VIP_2	***45%***	***(1/4, 1/4, 1/2)***



Rule allocation



Rule allocation



Pack Result

Packing result for table capacity C = 5
VIP 1: 2 rules
VIP 2: 3 rules
Total imbalance = 9.17%

Match (dst, src)	*Action*
*VIP 1, *0*	*Fwd to 2*
*VIP 1, **	*Fwd to 3*
*VIP 2, *00*	*Fwd to 1*
*VIP 2, *01*	*Fwd to 2*
*VIP 2, **	*Fwd to 3*

Sharing default rules

- Build default split for ALL VIPs

Weights		=	Default	+	Delta	
1/6	1/4		0		1/6	1/4
1/3	1/4		1/2		-1/6	-1/4
1/2	1/2		1/2		0	0

*VIP 1, *0*	*Fwd to 2*
*VIP 1, **	*Fwd to 3*
*VIP 2, *00*	*Fwd to 1*
*VIP 2, *01*	*Fwd to 2*
*VIP 2, **	*Fwd to 3*

Imbalance = 9.17%

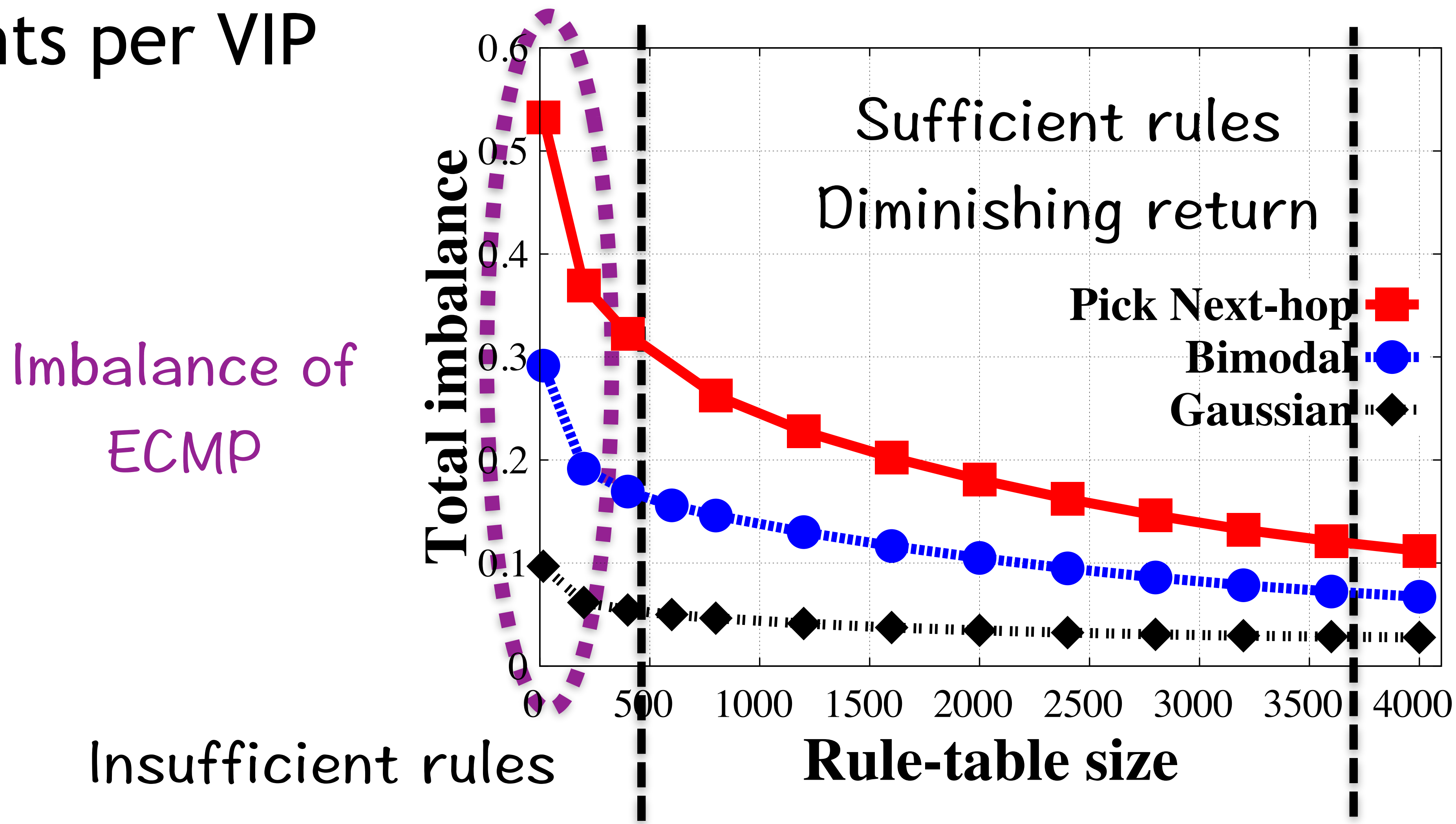
V.S.

*VIP 1, *00100*	*Fwd to 1*
*VIP 1, *000*	*Fwd to 1*
*VIP 2, *00*	*Fwd to 1*
**0*	*Fwd to 2*
***	*Fwd to 3*

Imbalance = 0.55%

Load Balance 10,000 VIPs

- Weights
  - Gaussian: equal weights
  - Bimodal: big (4x) and small weights
  - Pick_Next-hop: big(4x), small and zero-value weights
  - 16 weights per VIP



Niagara Summary

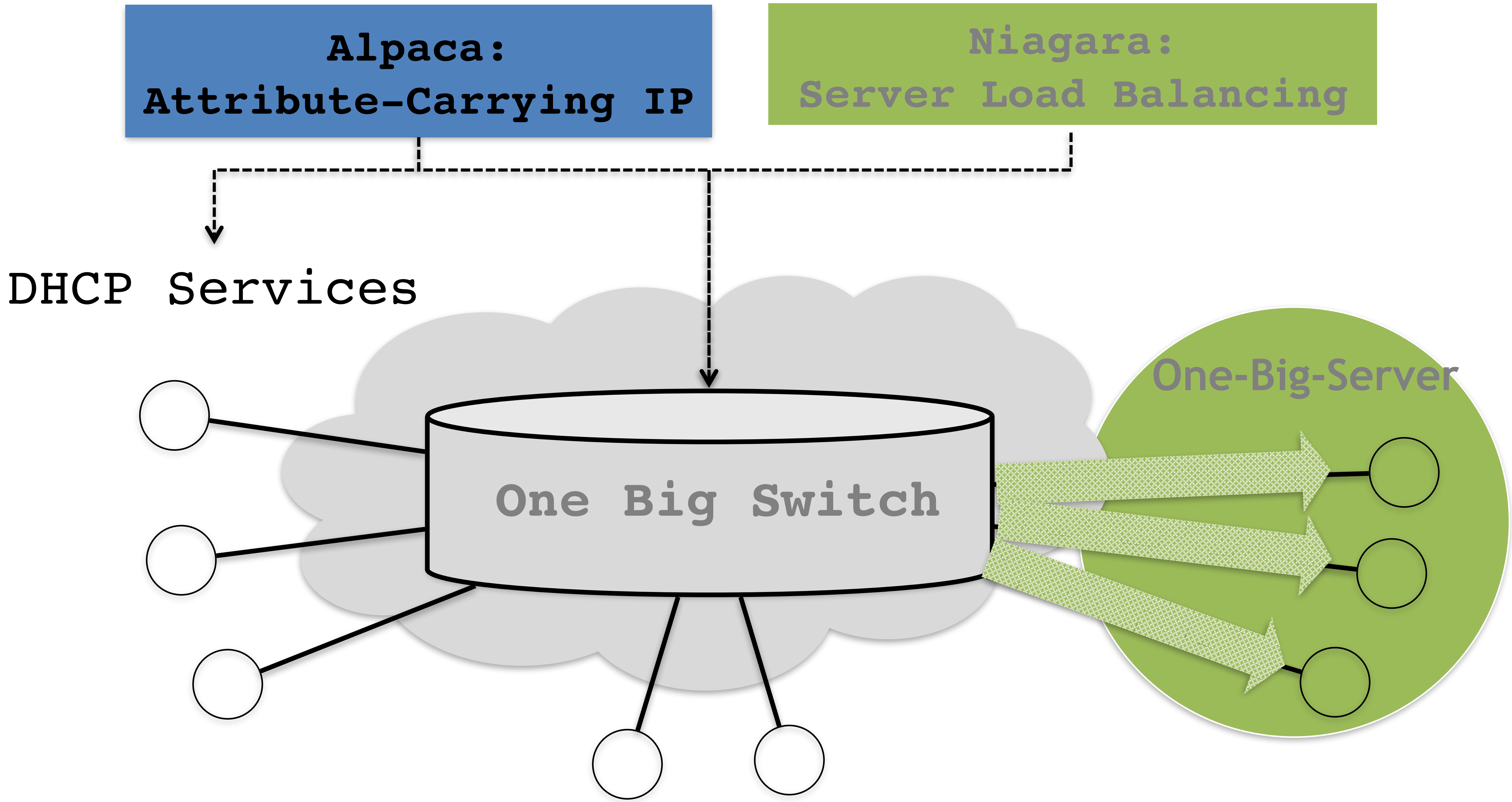
- Wildcard matches approximate weights well
  - Exponential drop in errors
- Prioritized packing reduces imbalance sharply
- Default rules serve as a good starting point

- Full algorithms
  - Multiple VIP Grouping
  - Incremental update
    - reduce “churn”, multi-stage update, flow consistency
  - Niagara for multi-pathing

Alpaca: Compact Network Policies with Attribute-Carrying Addresses

Nanxi Kang, Ori Rottenstreich,
Sanjay Rao, Jennifer Rexford

Attribute-Carrying IP



Attribute-based Network Policies

- Policies are defined based on host attributes
  - Permit *CS hosts* to a database
  - Rate limit *student hosts*’ traffic to 50Mbps

- We surveyed policies in 22 campus networks
  - ACL and QoS consider *Departments* and *Roles*
  - ACL may ban particular *OS*
  - QoS may give different priorities based on *Usage*

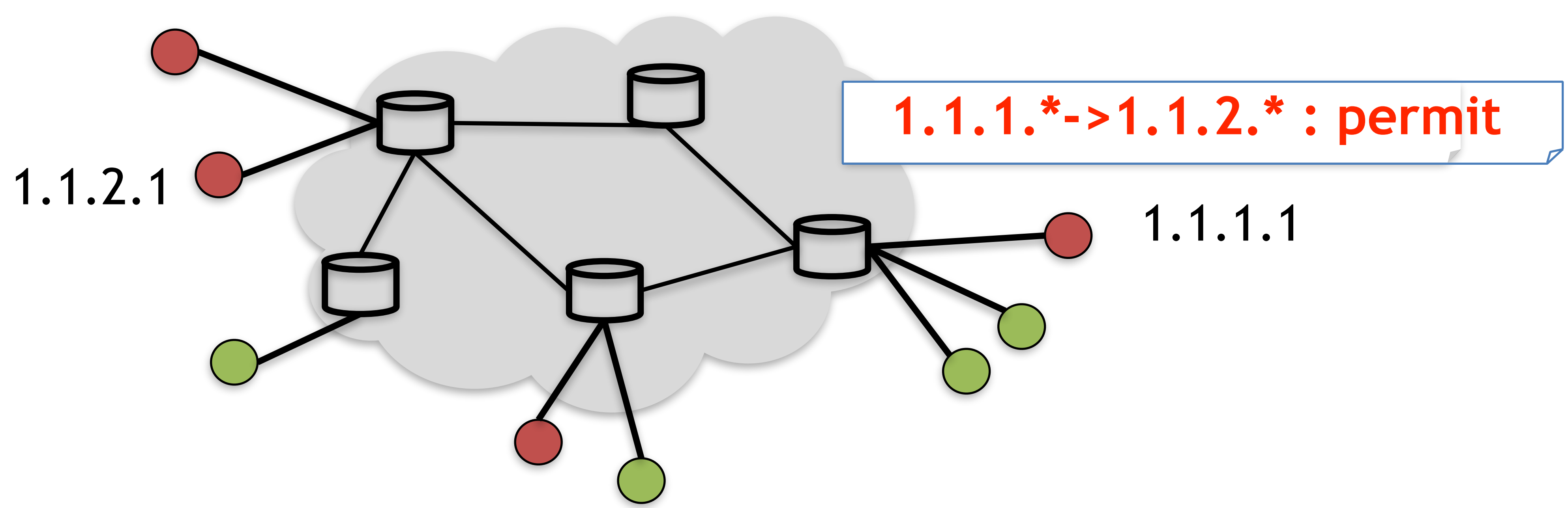
Dimensions and Attributes

- Dimensions: orthogonal categorization
- Attributes: values in a dimension

Dimension	Example Attributes
Department	CS, EE
Role	Faculty, Students
Security Level	Deny all, Permit web (80), Permit SSH
Status	In service, In testing
Location	-
Usage	Research, Teaching, Infrastructure
CS_owned	Yes, No
OS	MacOS, Windows

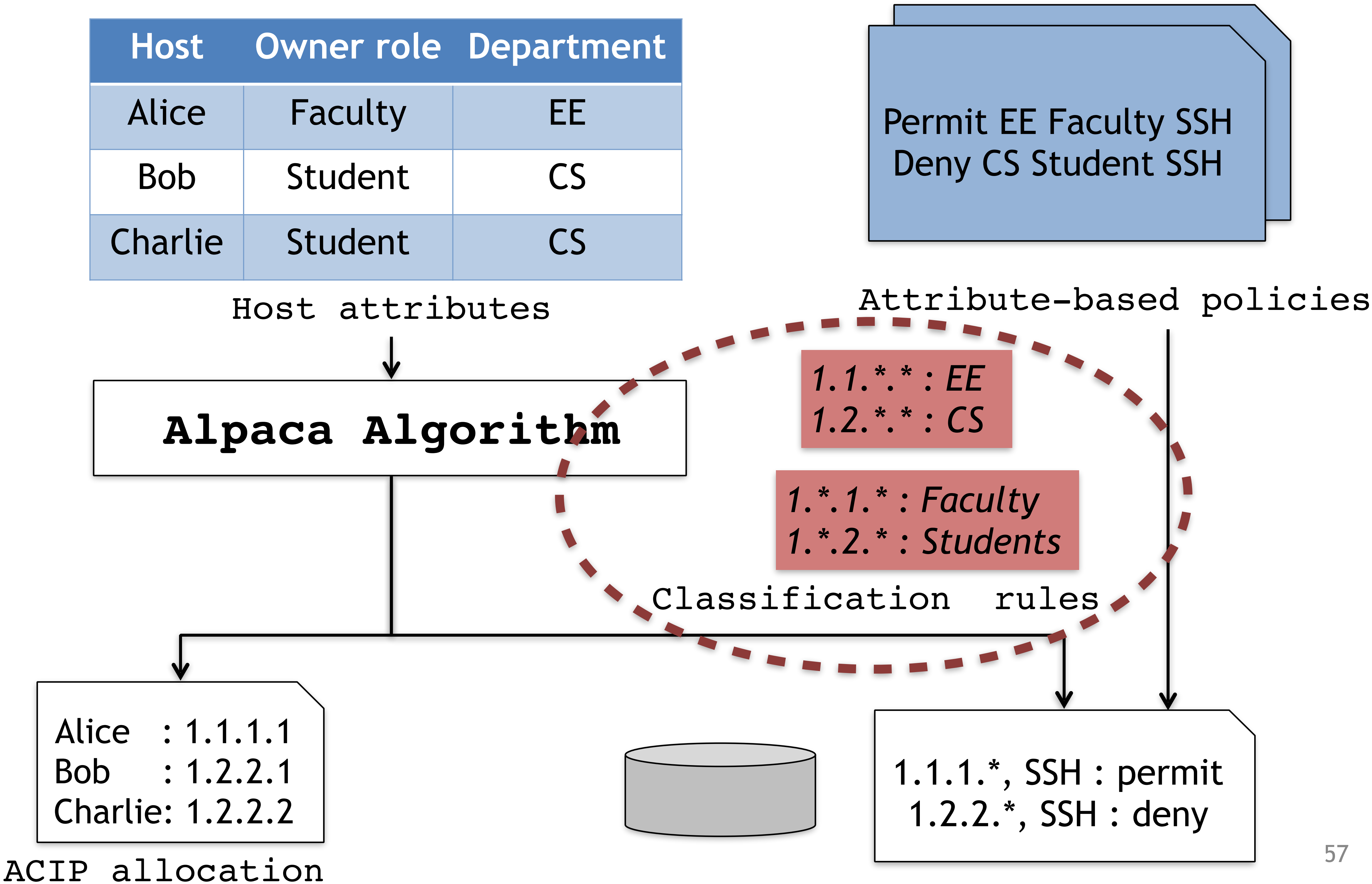
Attribute-Carrying IP (ACIP)

- Embed attribute information
  - Do *once* when hosts join the network
- Reduce rule space usage
  - *Aggregate* addresses



ACIP Allocation

Host	Owner role	Department
Alice	Faculty	EE
Bob	Student	CS
Charlie	Student	CS



Solutions: Use 2^k

- An address pattern with k *s represent 2^k hosts
  - e.g., 00** represents $2^2 = 4$ hosts
- Use 2^k to represent group sizes

	CS	EE
Faculty	***5***	*3*
Students	*2*	*6*

	CS	EE
Faculty	***1 + 4***	*1 + 2*
Students	*2*	*2 + 4*

(CS, Faculty, 1)
(CS, Faculty, 4)

Solutions: Use 2^k

- An address pattern with k *s represent 2^k hosts
  – e.g., 00** represents $2^2 = 4$ hosts
- Use 2^k to represent group sizes

	CS	EE
Faculty	***5***	*3*
Students	*2*	*6*

↓

	CS	EE
Faculty	***1 + 4***	*1 + 2*
Students	*2*	*2 + 4*

(CS, Faculty, 1)
(CS, Faculty, 4)
(EE, Faculty, 1)
(EE, Faculty, 2)
(CS, Students, 2)
(EE, Students, 2)
(EE, Students, 4)

Representation of Attributes

- 8 Faculty hosts
  - (CS, F, 1), (CS, F, 4), (EE, F, 1), (EE, F, 2)

Worst case: 4 patterns

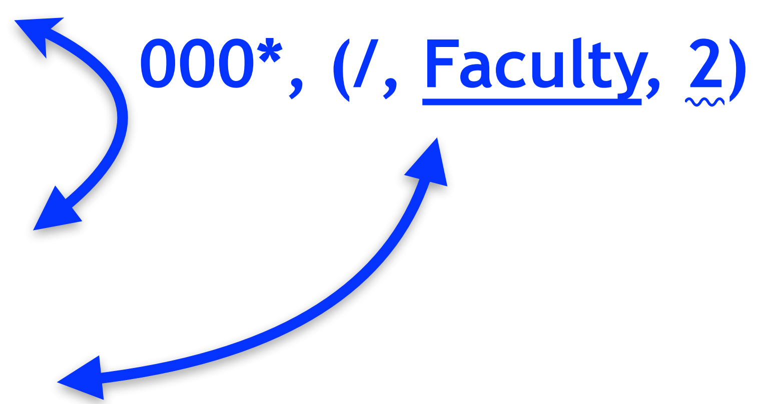
Can we do better?

(CS, Faculty, 1)
(CS, Faculty, 4)
(EE, Faculty, 1)
(EE, Faculty, 2)
(CS, Students, 2)
(EE, Students, 2)
(EE, Students, 4)

Flip bits

- Flip one bit for two terms with
  - at least one attribute in common
  - equal values

(CS, Faculty, 1)	0000
(CS, Faculty, 4)	
(EE, Faculty, 1)	0001
(EE, Faculty, 2)	
(CS, Students, 2)	
(EE, Students, 2)	
(EE, Students, 4)	



Flip bits

- Flip one bit for two terms with
  - at least one attribute in common
  - equal values

(CS, Faculty, 1)	0000
(CS, Faculty, 4)	01**
(EE, Faculty, 1)	0001
(EE, Faculty, 2)	001*
(CS, Students, 2)	100*
(EE, Students, 2)	101*
(EE, Students, 4)	11**

000*, (/, Faculty, 2)

Classification rules

- Role
  - Faculty: 0***
  - Students: 1***

(CS, Faculty, 1)	0000
(CS, Faculty, 4)	01**
(EE, Faculty, 1)	0001
(EE, Faculty, 2)	001*
(CS, Students, 2)	100*
(EE, Students, 2)	101*
(EE, Students, 4)	11**

Classification rules

- Role
- Department
  - CS: 0000, 100*, 01**
  - EE: 0001, *01*, 11**

Configure Alpaca to compute prefix or wildcard patterns

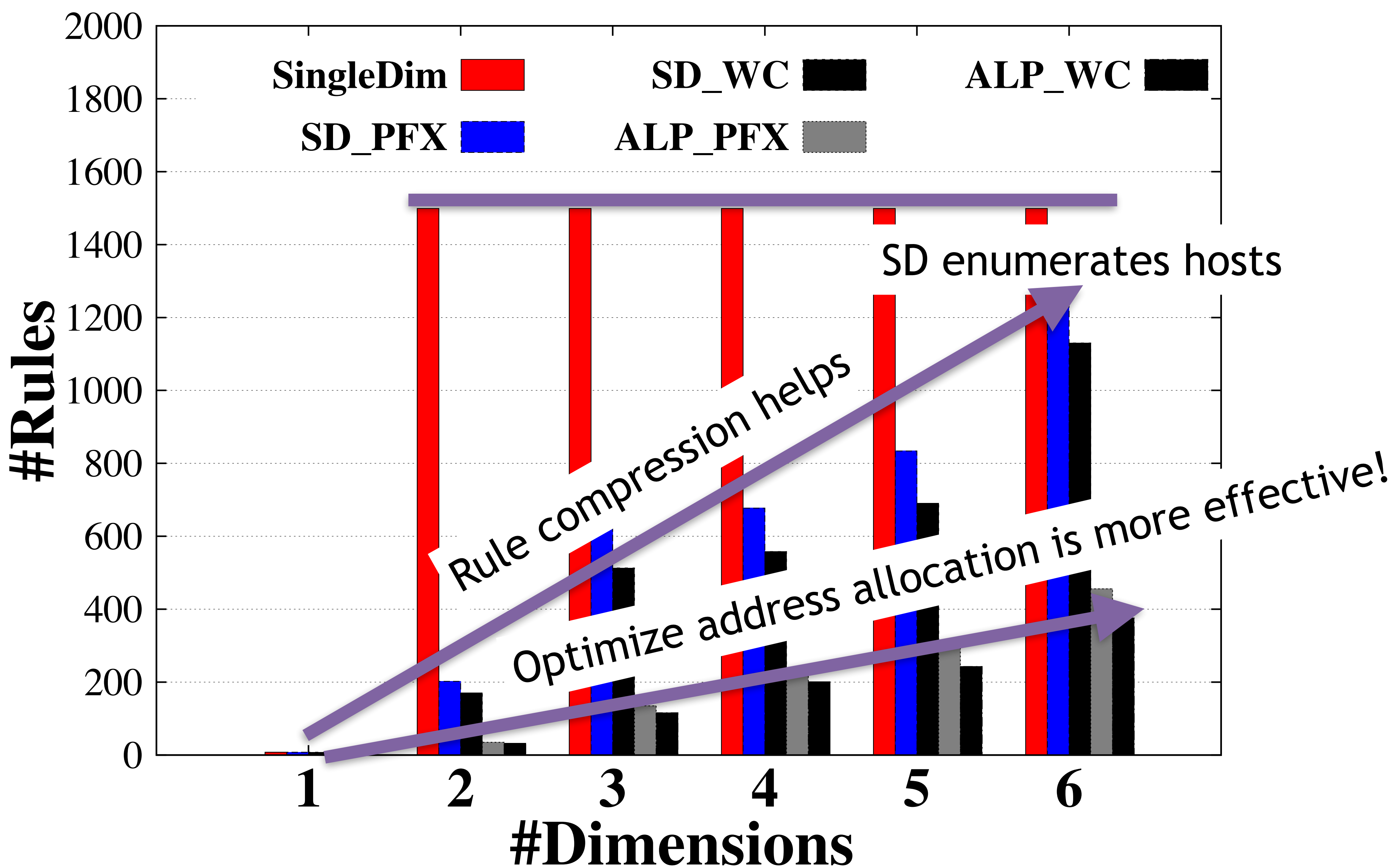
(CS, Faculty, 1)	0000
(CS, Faculty, 4)	01**
(EE, Faculty, 1)	0001
(EE, Faculty, 2)	001*
(CS, Students, 2)	100*
(EE, Students, 2)	101*
(EE, Students, 4)	11**

Evaluation

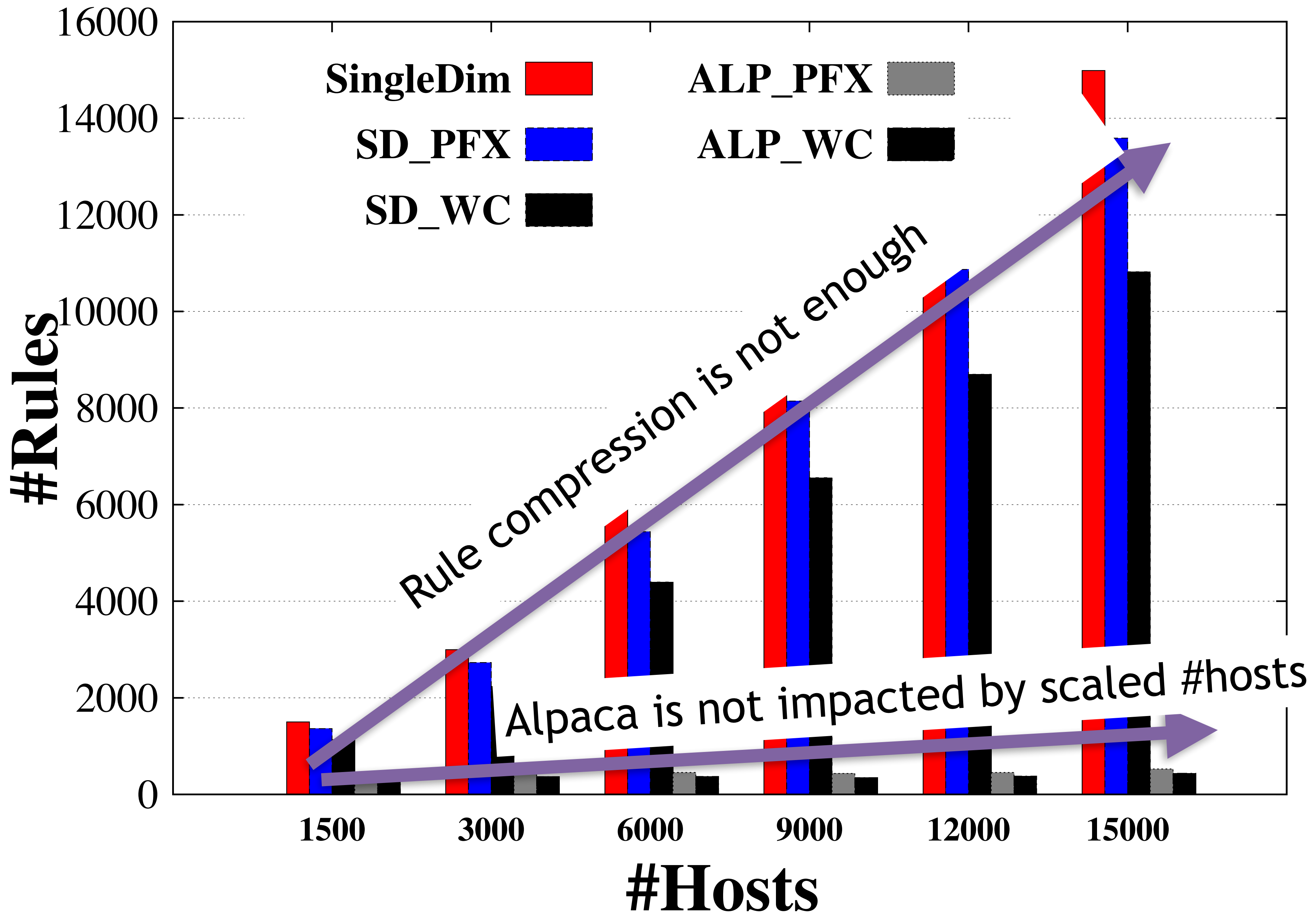
- Princeton CS data: 6 dimensions, ~1500 hosts
- Metric: Σ |classification rules for a dimension|
- Compared with

SingleDim	Classify hosts along "Department", e.g., VLAN
SD_PFX	"Department": SingleDim Others : Optimal prefix compression
SD_WC	"Department": SingleDim Others : Wildcard compression heuristics

Increased #dimensions



Increase #hosts



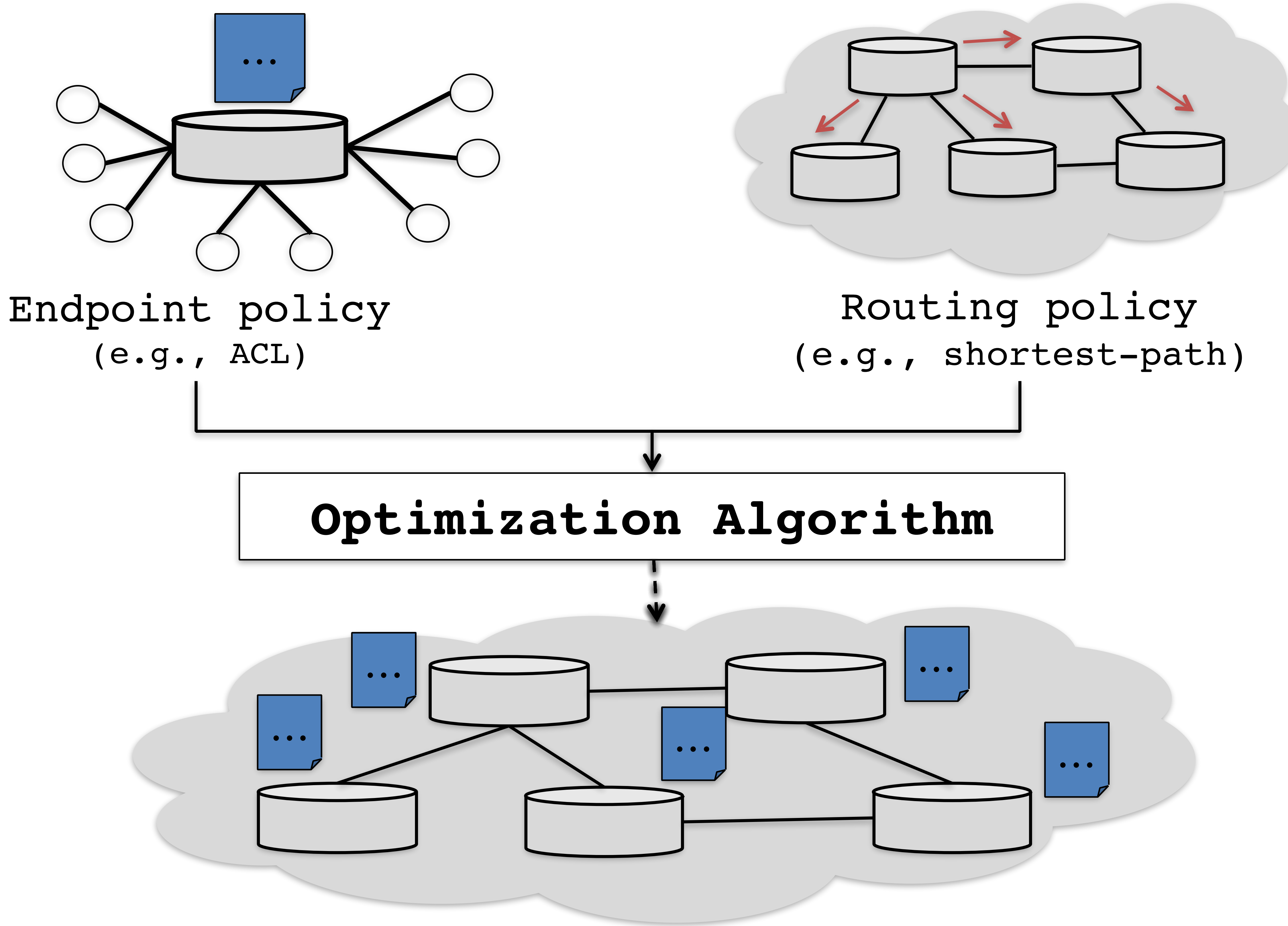
Alpaca Summary

- Flip bits to allocate ACIPs to host groups
- Optimize address allocation is more effective than compression on fixed address allocation
- Full algorithm:
  - Incremental update of ACIP allocation

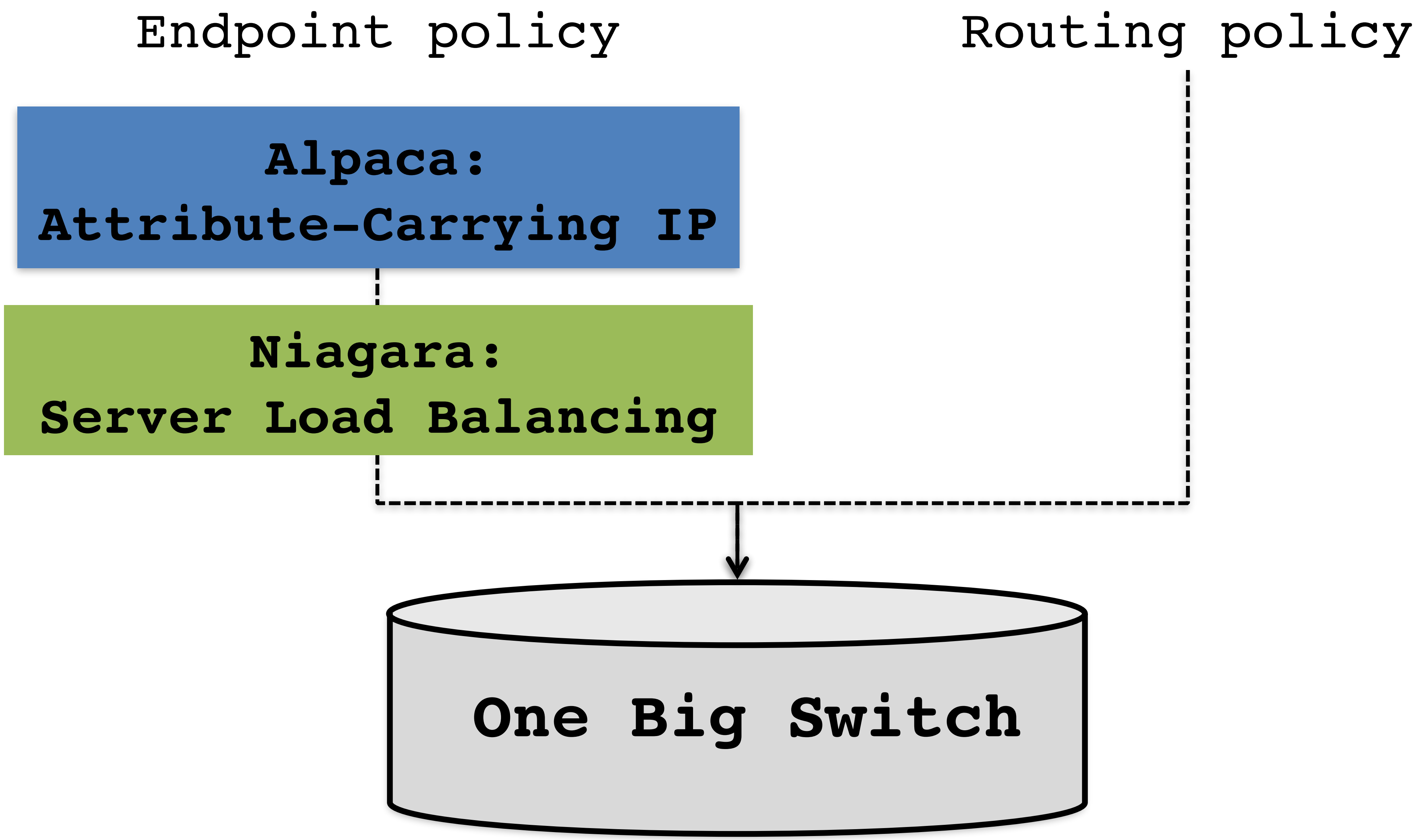
Optimizing the One-Big-Switch Abstraction in Software-Defined Networks

Nanxi Kang, Zhenming Liu,
Jennifer Rexford, David Walker

Optimize One-Big-Switch Abstraction



Put Everthing All Together

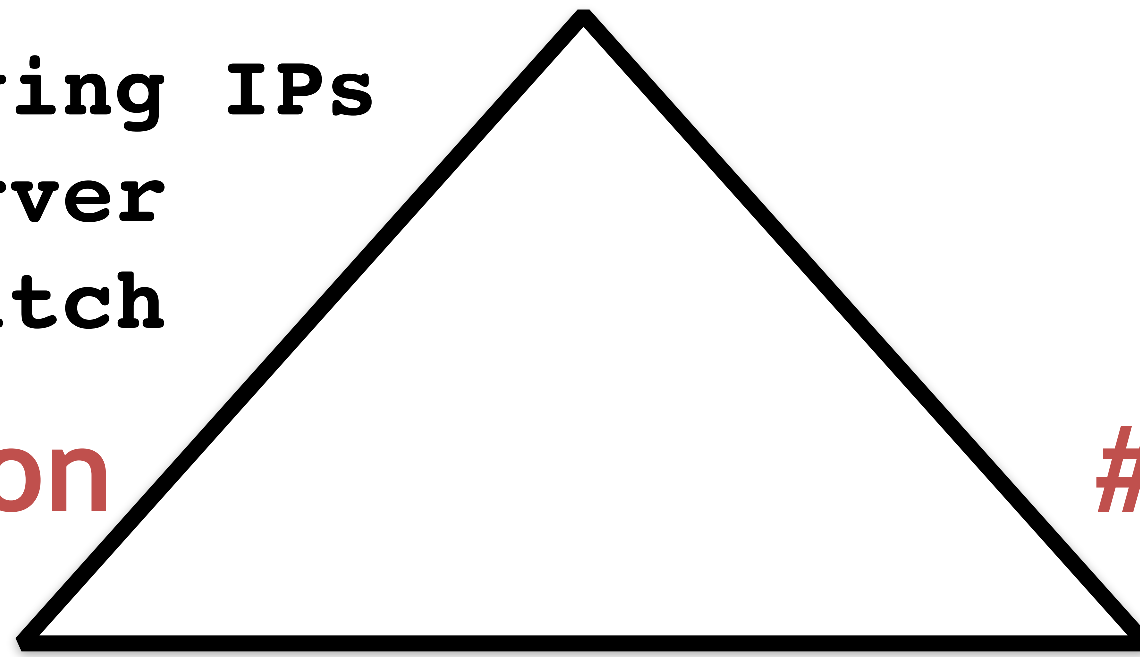


Contributions

Diverse
Policies

Attribute-Carrying IPs
One-Big-Server
One-Big-Switch

#1: Abstraction



Alpaca
Niagara
Optimize OBS

#2: Algorithm

<u>Smart algorithms realize simple abstractions!</u>

Thanks!