Scalable and Efficient Self-configuring Networks

Changhoon Kim

chkim@cs.princeton.edu

Today's Networks Face New Challenges

- Networks growing rapidly in size
  - Up to tens of thousands to millions of hosts
- Networking environments getting highly dynamic
  - Mobility, traffic volatility, frequent re-adjustment
- Networks being increasingly deployed in non-IT industries and developing regions
  - Limited support for management

Networks must be scalable and yet easy to build and manage!

Pains In Large Dynamic Networks

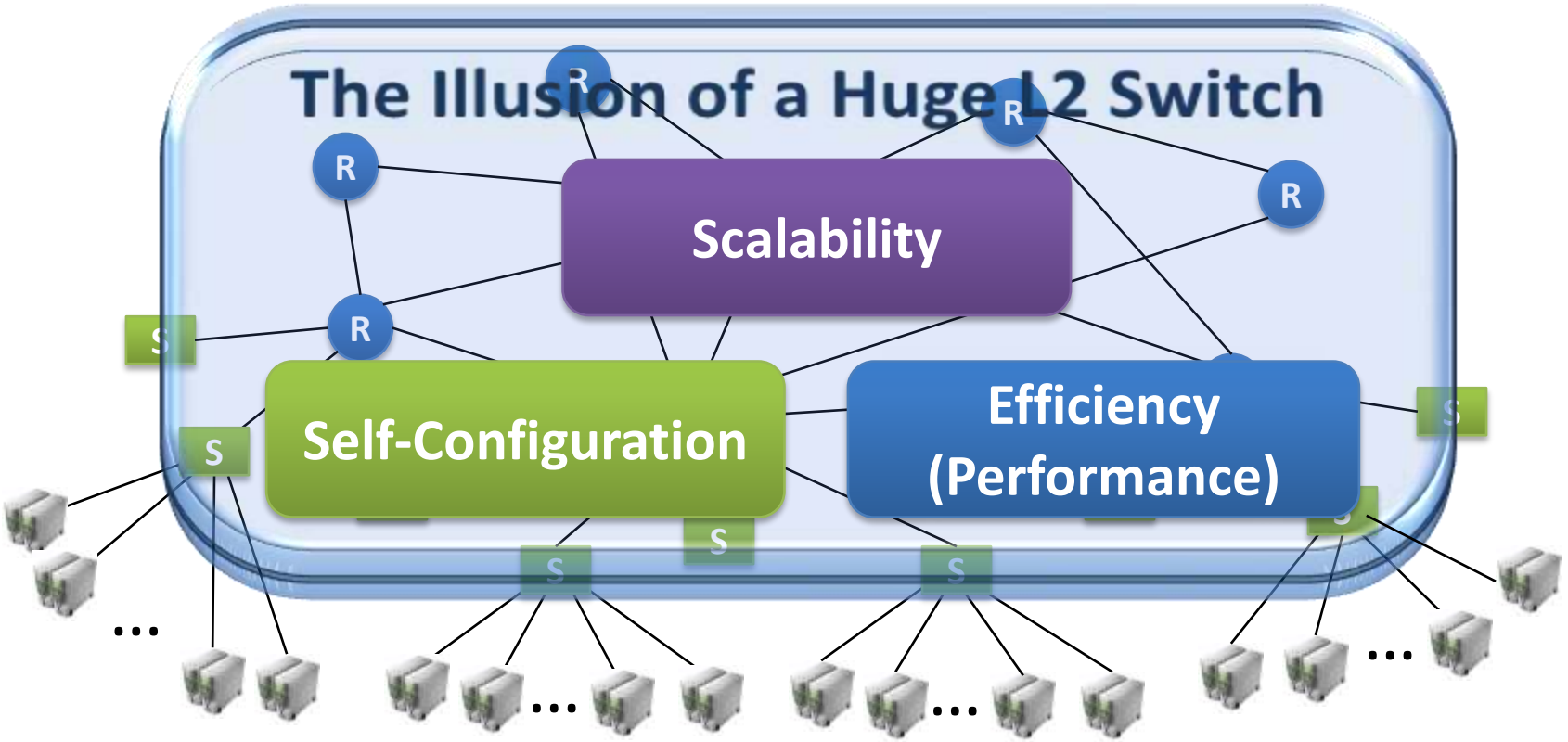
- Control-plane overhead to store and disseminate host state
  - Millions of host-info entries
    - Commodity switches/routers can store only ~32K/300K entries
  - Frequent state updates, each disseminated to thousands of switches/routers
- Status quo: Hierarchical addressing and routing
- Victimizes ease of management
  - Leads to complex, high-maintenance, fragile, hard-to-debug, and brittle network

More Pains …

- Limited data-plane capacity
  - Tbps of traffic workload on core links
    - Fastest links today are only 10 ~ 40 Gbps
  - Highly volatile traffic patterns
- Status quo: Over-subscription
- Victimizes efficiency (performance)
  - Lowers server utilization and end-to-end performance
  - Trying to mitigate this via traffic engineering can make management harder

All We Need Is Just A Huge L2 Switch, or An Abstraction of One!

- Host up to millions of hosts
- Obviate manual configuration
- Ensure high capacity and small end-to-end latency



Research Strategy and Goal

- Emphasis on architectural solutions
  - Redesign underlying networks
- Focus on edge networks
  - Enterprise, campus, and data-center networks
  - Virtual private networks (VPNs)

[Research Goal]

Design, build, and deploy architectural solutions enabling scalable, efficient, self-configuring edge networks

Why Is This Particularly Difficult?

- Scalability, self-configuration, and efficiency can conflict!
  - Self-configuration can significantly increase the amount of state and make traffic forwarding inflexible
  - Examples: Ethernet, VPN routing

Present solutions to resolve this impasse

Universal Hammers

Self-configuration

Flat Addressing

- Utilize location-independent names to identify hosts
- Let network self-learn hosts' info

Scalability

Traffic Indirection

- Deliver traffic through intermediaries (chosen systematically or randomly)

Efficiency

Usage-driven Optimization

- Populate routing and host state only when and where needed

Solutions For Various Edge Networks

Enterprise, Campus Network	Data-Center Network	Virtual Private Network
Config-free addressing and routing	Config-free addressing, routing, and traffic engineering	Config-free site-level addressing and routing
Huge overhead to store and disseminate individual hosts' info	Limited server-to-server capacity	Expensive router memory to store customers' info
Partition hosts' info over switches	Spread traffic randomly over multiple paths	Keep routing info only when it's needed
SEATTLE [SIGCOMM'08]	**VL2** [SIGCOMM'09]	**Relaying** [SIGMETRICS'08]

Strategic Approach for Practical Solutions

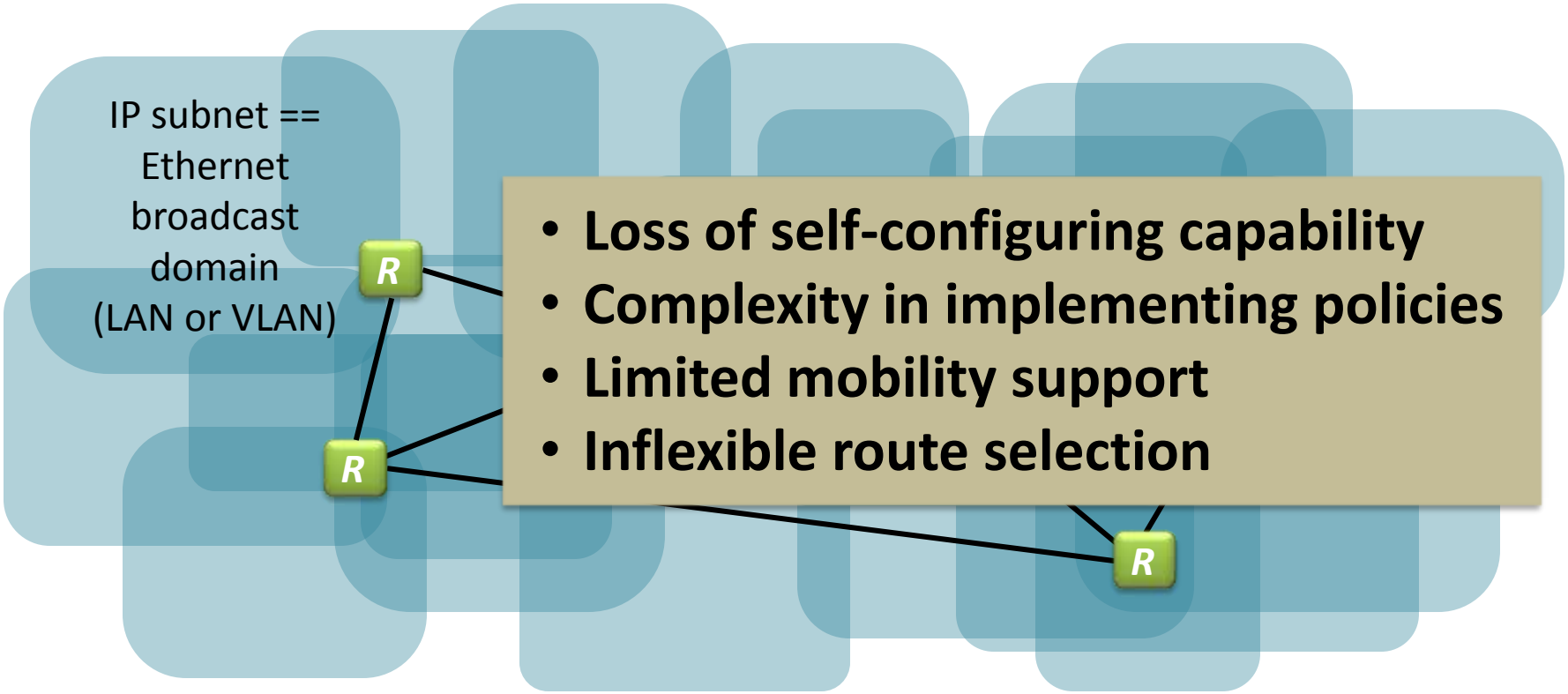
Enterprise, Campus Network	Data-Center Network	Virtual Private Network
• Heterogeneity of end-host environments	• Lack of programmability at switches and routers	• Immediate deployment • Transparency to customers
Clean slate on network	Clean slate on end hosts	Backwards compatible
Several independent prototypes	**Real-world Deployment**	**Pre-deployment Evaluation**

SEATTLE:
A Scalable Ethernet Architecture for Large Enterprises

Work with Matthew Caesar and Jennifer Rexford

Current Practice in Enterprises

A hybrid architecture comprised of several small Ethernet-based IP subnets interconnected by routers



Need a protocol that combines only the best of IP and Ethernet

Objectives and Solutions

Objective	Approach	Solution
1. Avoid flooding	Resolve host info via unicast	Network-layer one-hop DHT
2. Restrain broadcasting	Bootstrap hosts via unicast	
3. Reduce routing state	Populate host info only when and where needed	Traffic-driven resolution with caching
4. Enable shortest-path forwarding	Allow switches to learn topology	L2 link-state routing maintaining only switch-level topology

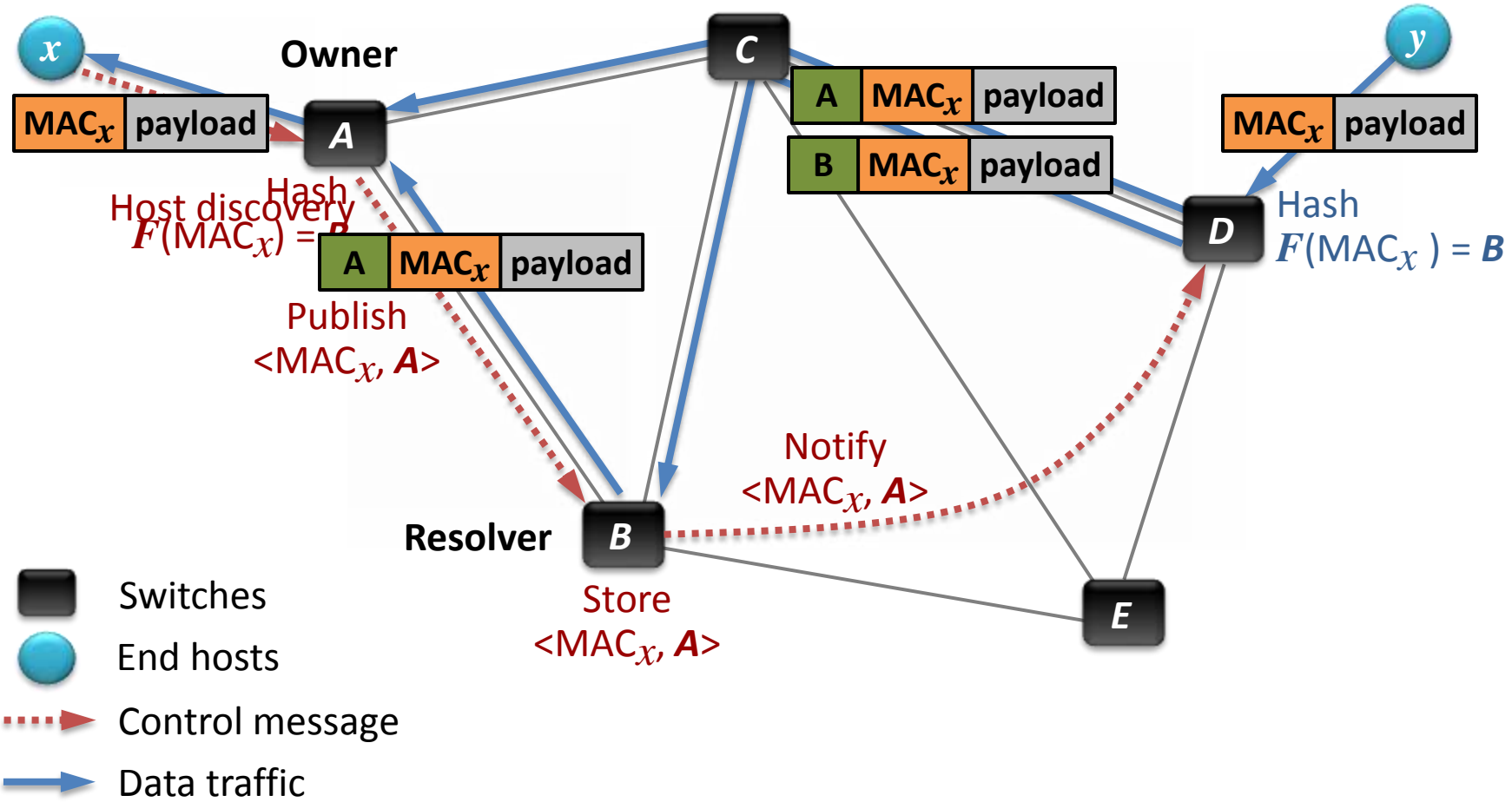
*** Meanwhile, avoid modifying end hosts**

Network-layer One-hop DHT

- Switches maintain <*key, value*> pairs by commonly using a hash function F
  - F: Consistent hash mapping a key to a switch
  - LS routing ensures each switch knows about all the other live switches, enabling one-hop DHT operations
- Unique benefits
  - Fast, efficient, and accurate reaction to churns
  - Reliability and capacity naturally growing with network size

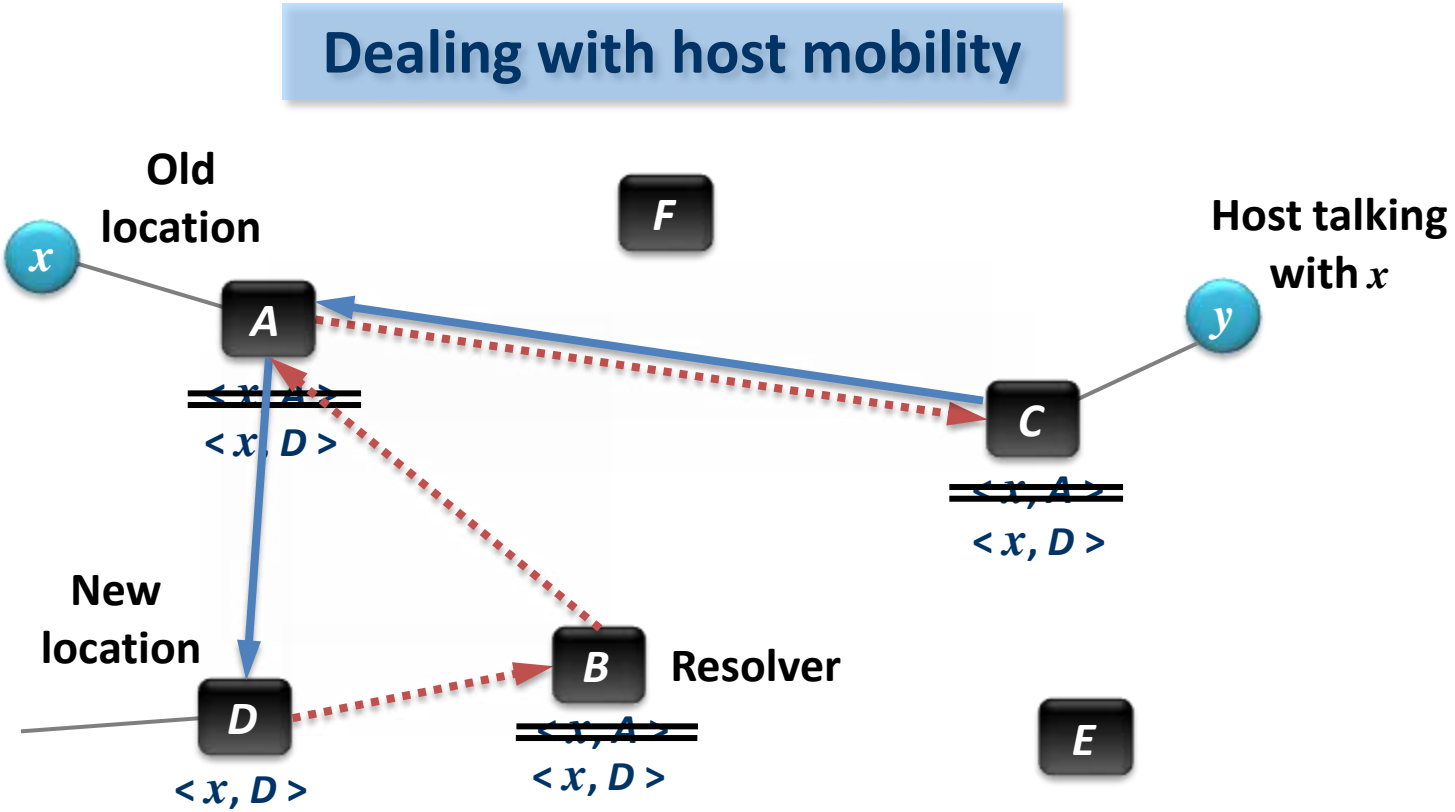
Location Resolution

<key, val> = <MAC addr, location>



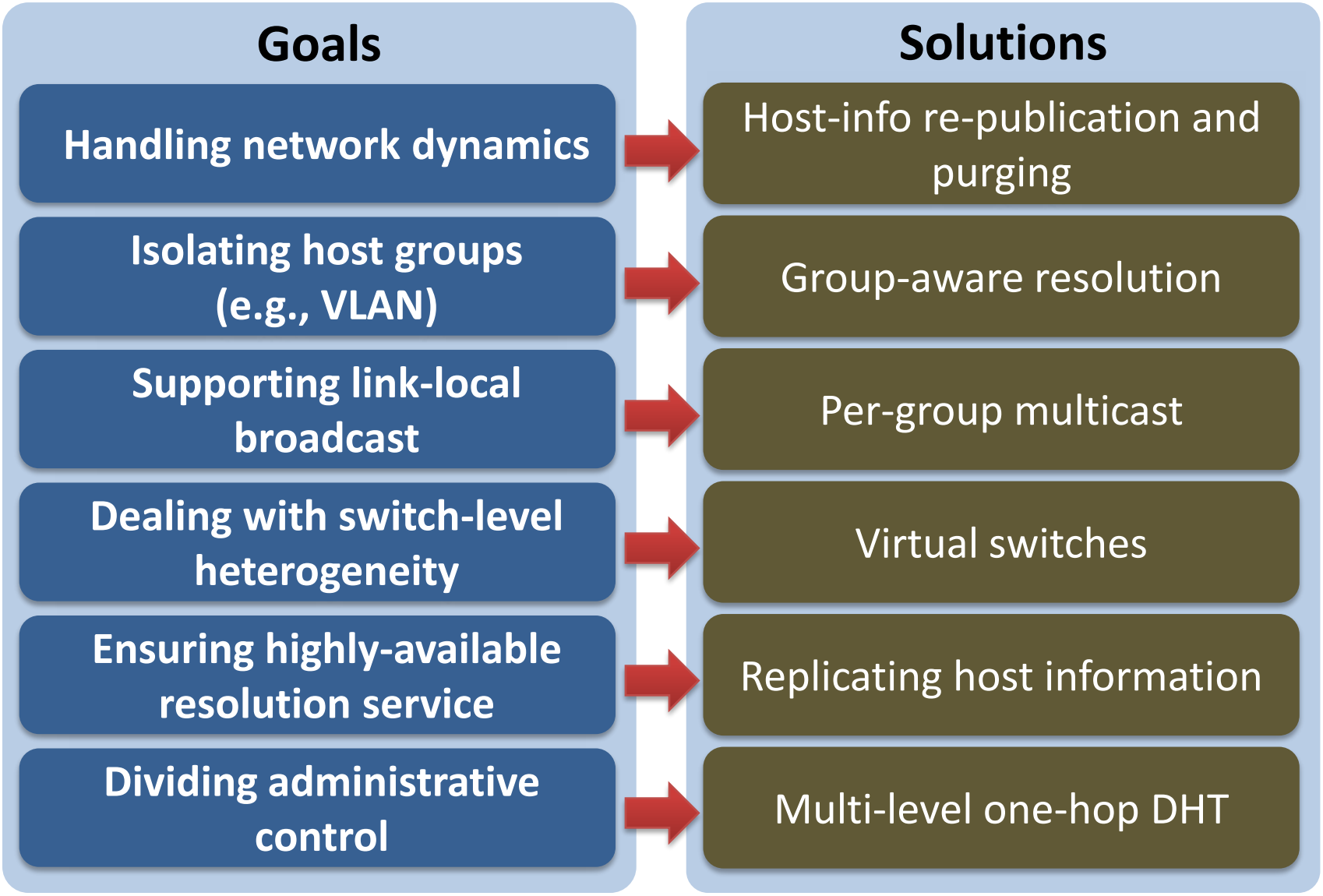
Handling Host Dynamics

- Host location, MAC-addr, or IP-addr can change



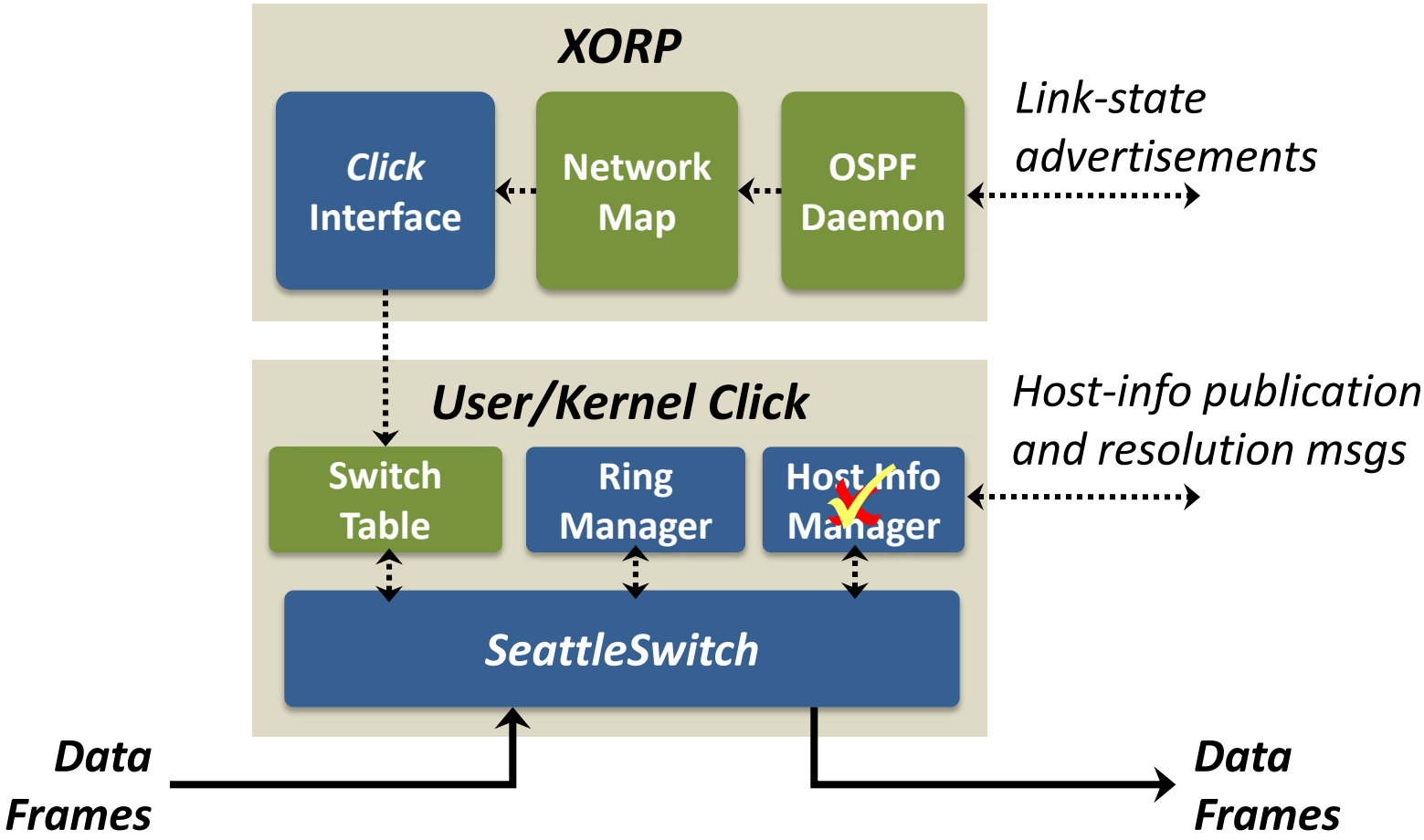
MAC- or IP-address change can be handled similarly

Further Enhancements Implemented

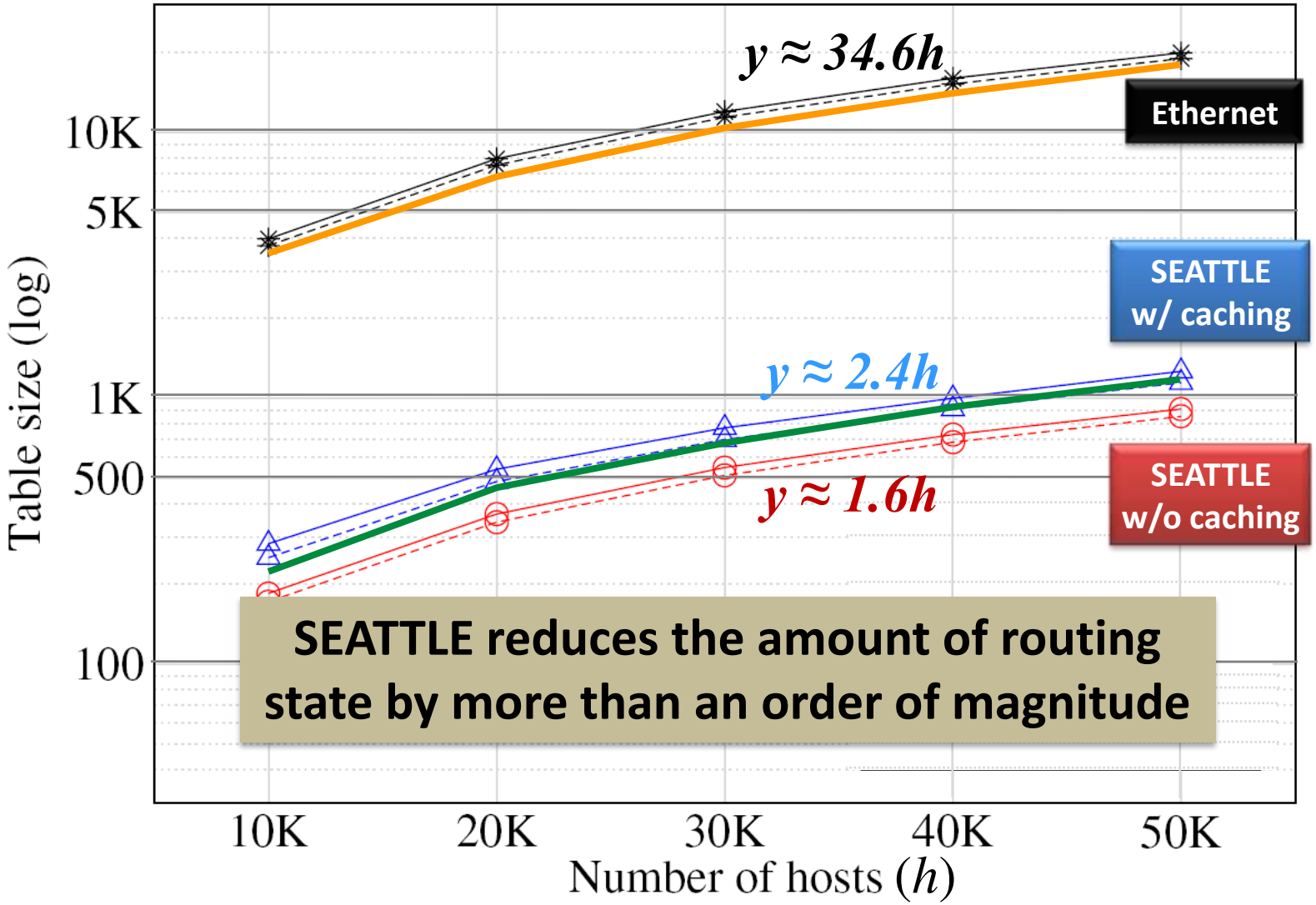


Prototype Implementation

- Link-state routing: ***XORP OSPFD***
- Host-info management and traffic forwarding: ***Click***



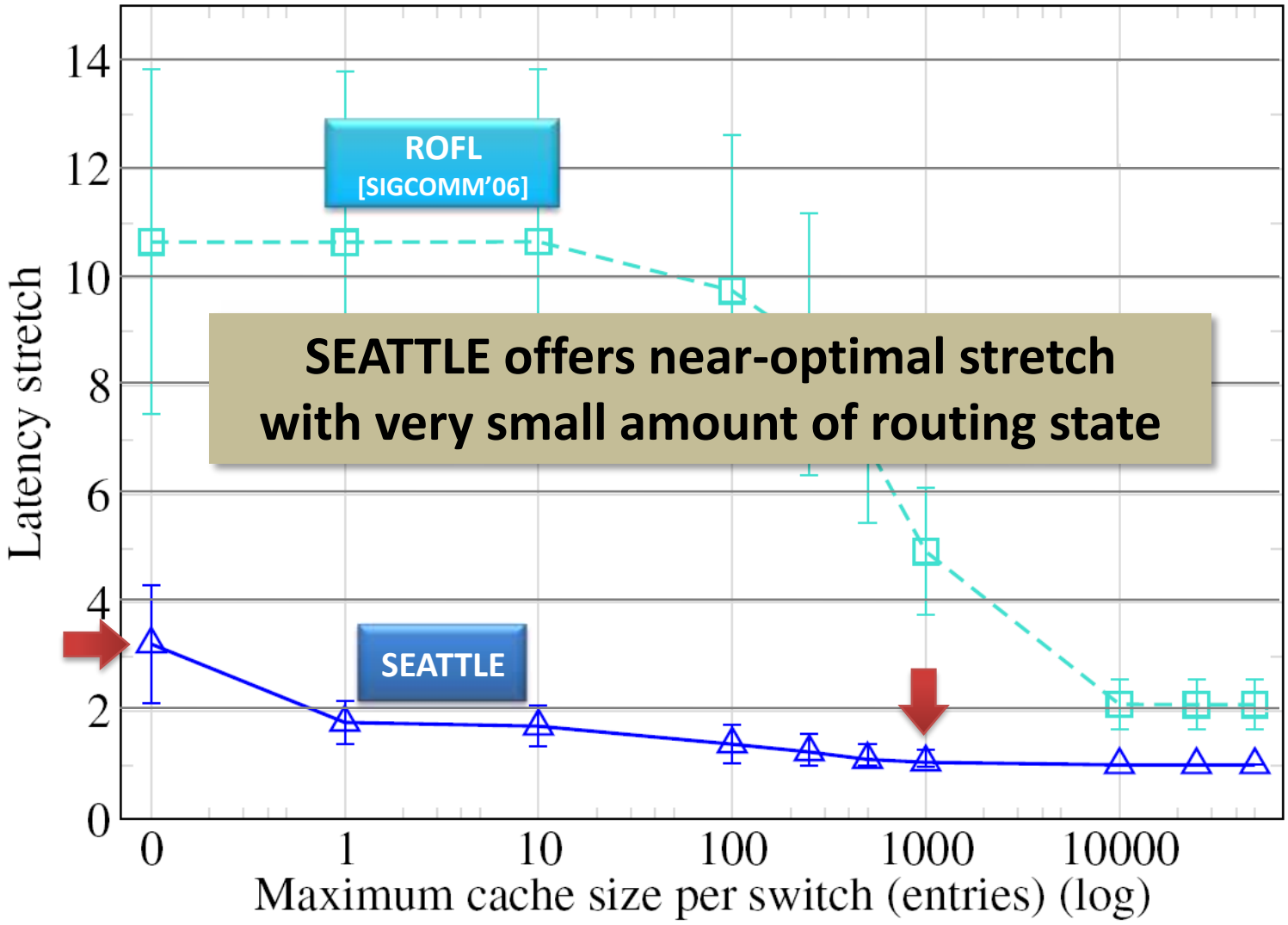
Amount of Routing State



SEATTLE reduces the amount of routing state by more than an order of magnitude

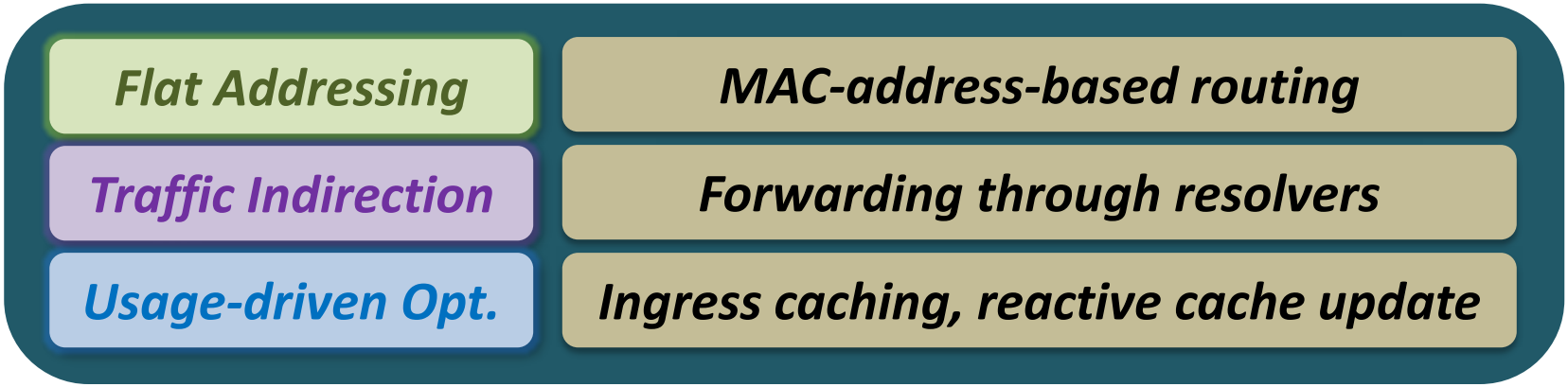
Cache Size vs. Stretch

Stretch = actual path length / shortest path length (in latency)



SEATTLE Conclusion

- Enterprises need a huge L2 switch
  - Config-free addressing and routing, support for mobility, and efficient use of links
- Key lessons
  - Coupling DHT with LS routing offers huge benefits
  - Reactive resolution and caching ensures scalability



Further Questions

- What other kinds of networks need SEATTLE?
- What about other configuration tasks?
- What if we were allowed to modify hosts?

These motivate my next work for data centers

VL2: A Scalable and Flexible Data-Center Network

Work with Albert Greenberg,
Navendu Jain, Srikanth Kandula,
Dave Maltz, Parveen Patel,
and Sudipta Sengupta

Data Centers

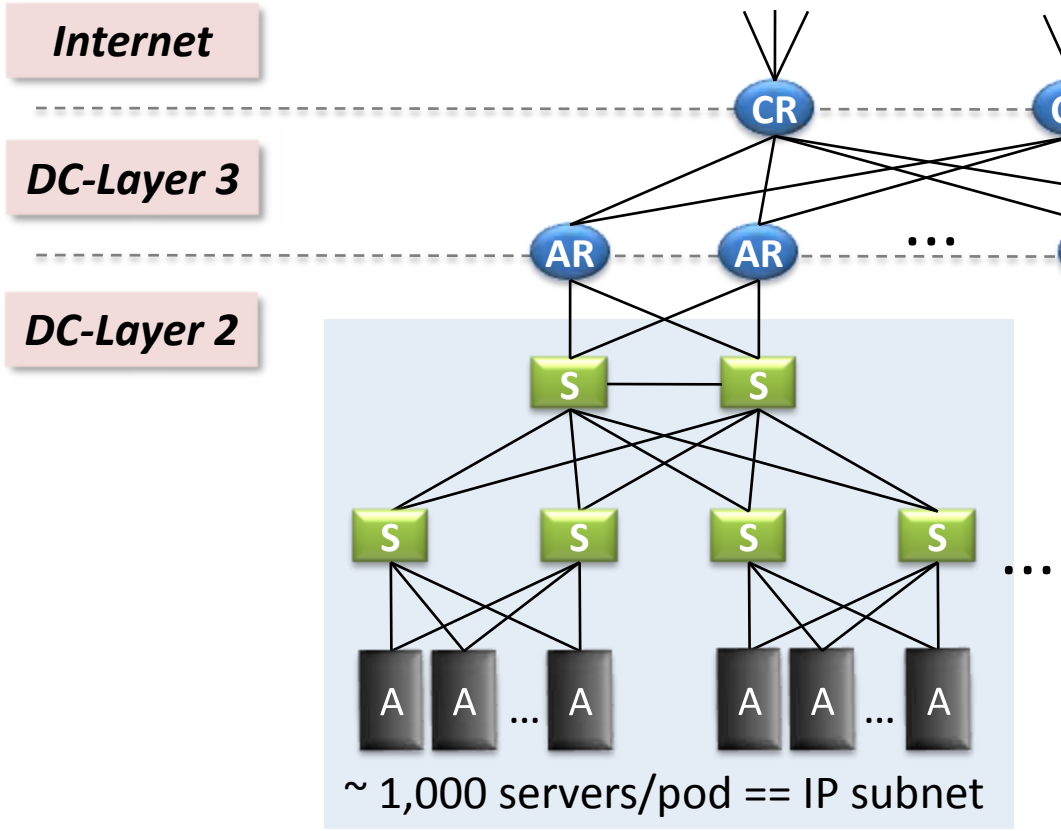
- Increasingly used for non-customer-facing decision-support computations
- Many of them will soon be outsourced to cloud-service providers
- Demand for large-scale, high-performance, cost-efficient DCs growing very fast

Tenets of Cloud-Service Data Center

- **Scaling out**: Use large pools of commodity resources
  - Achieves reliability, performance, low cost
- **Agility**: Assign any resources to any services
  - Increases efficiency (statistical multiplexing gain)
- **Low Management Complexity**: Self-configuring
  - Reduces operational expenses, avoids errors

Conventional DC network ensures none

Status Quo: Conventional DC Network



Reference – "Data Center: Load balancing Data Center Services", Cisco 2004

Key

- **CR** = Core Router (L3)
- **AR** = Access Router (L3)
- **S** = Switch (L2)
- **LB** = Load Balancer
- **A** = Rack of app. servers

Poor utilization and reliability

Status Quo: Traffic Patterns in a Cloud

- Instrumented a cluster used for data mining, then computed representative traffic matrices
- Traffic patterns are highly divergent
  - A large number (100+) of representative TMs needed to cover a day's traffic
- Traffic patterns are unpredictable
  - No representative TM accounts for more than a few hundred seconds of traffic patterns

Optimization approaches might cause more trouble than benefits

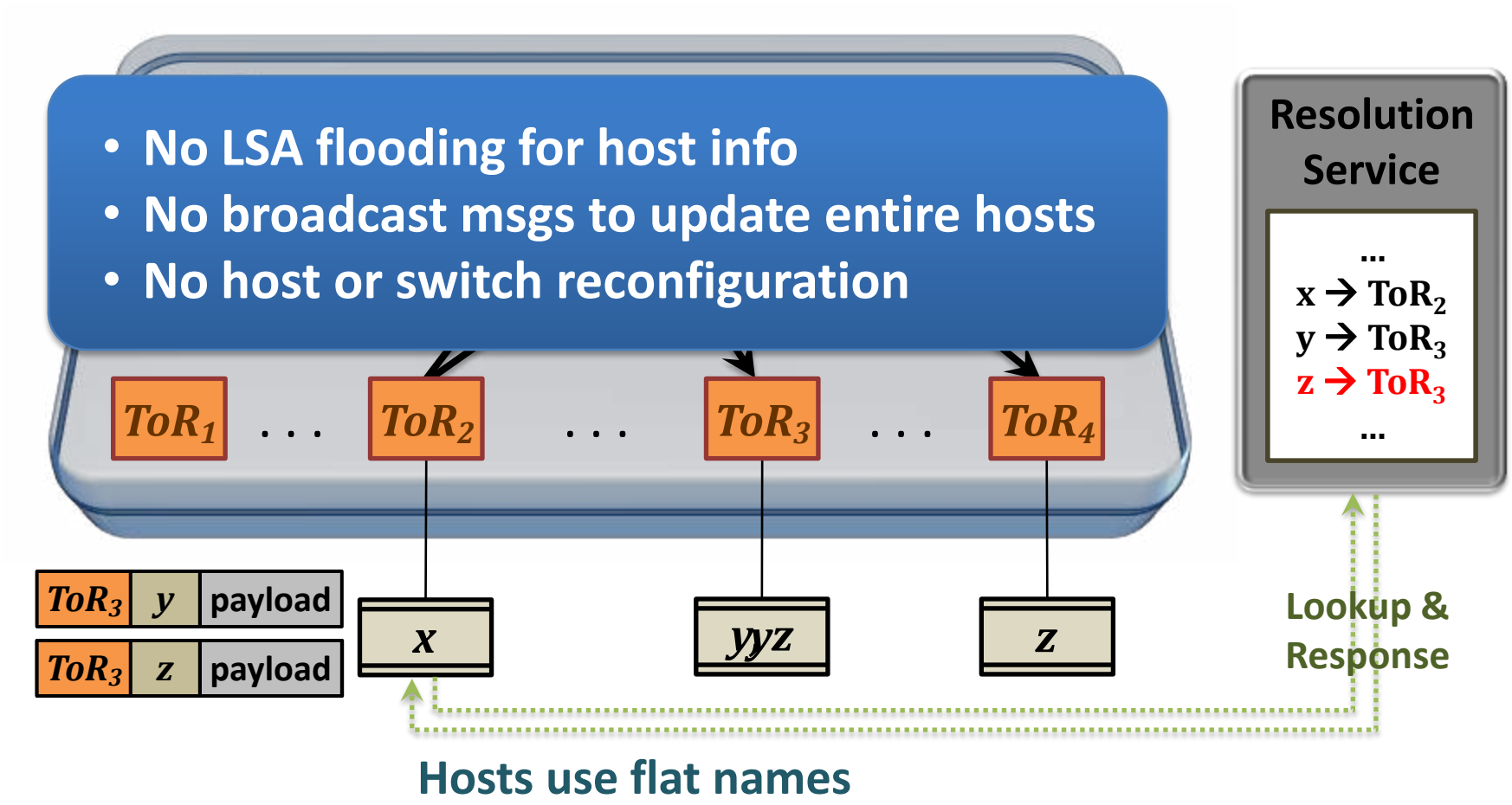
Objectives and Solutions

Objective	Approach	Solution
1. Ensure layer-2 semantics	Employ flat addressing	Name-location separation & resolution service
2. Offer uniform high capacity between servers	Guarantee bandwidth for hose-model traffic	Random traffic indirection (VLB) over a topology with huge aggregate capacity
3. Avoid hot spots w/o frequent reconfiguration	Use randomization to cope with volatility	

*** Embrace end systems!**

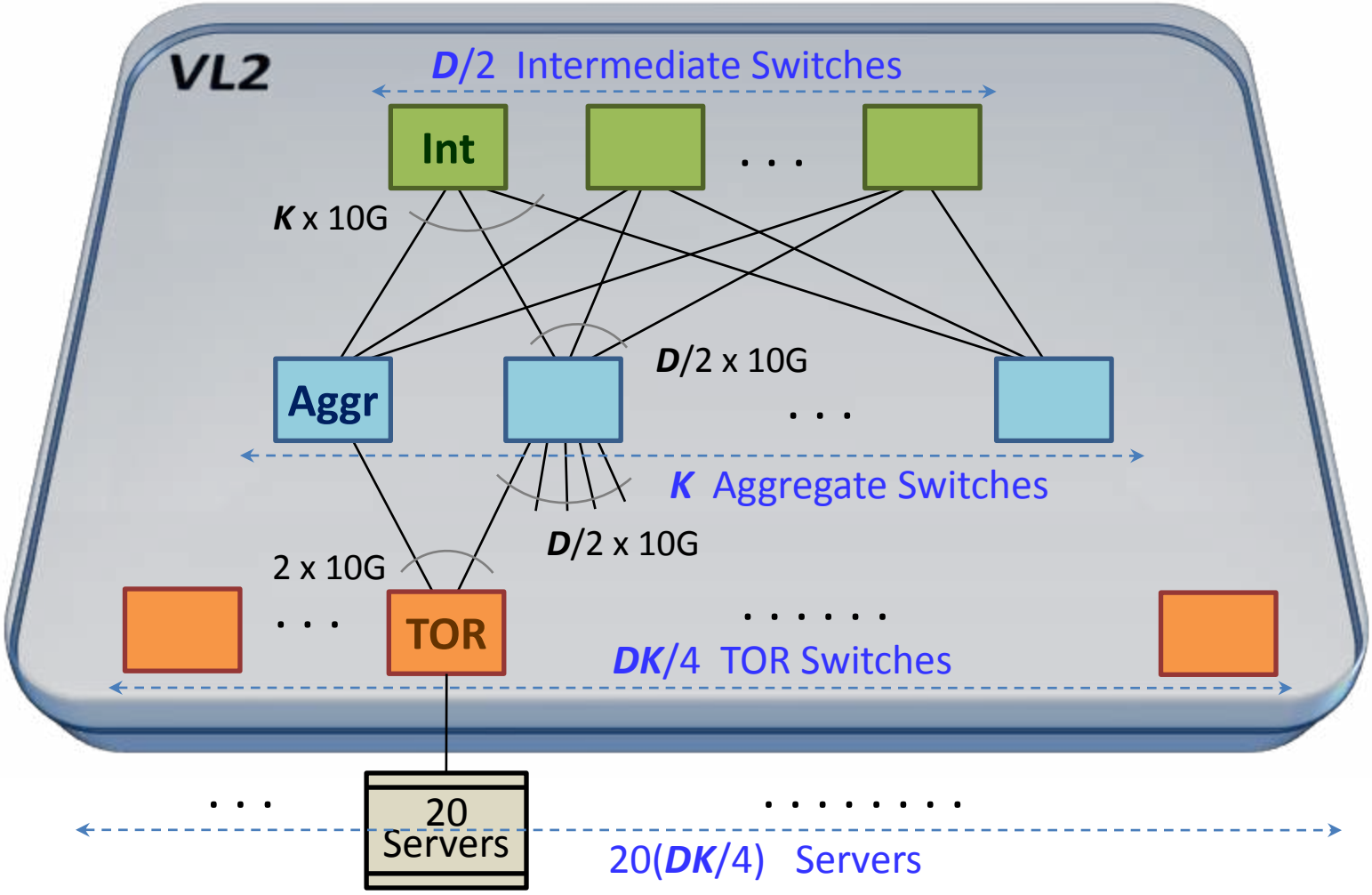
Addressing and Routing: Name-Location Separation

Cope with host churns with very little overhead



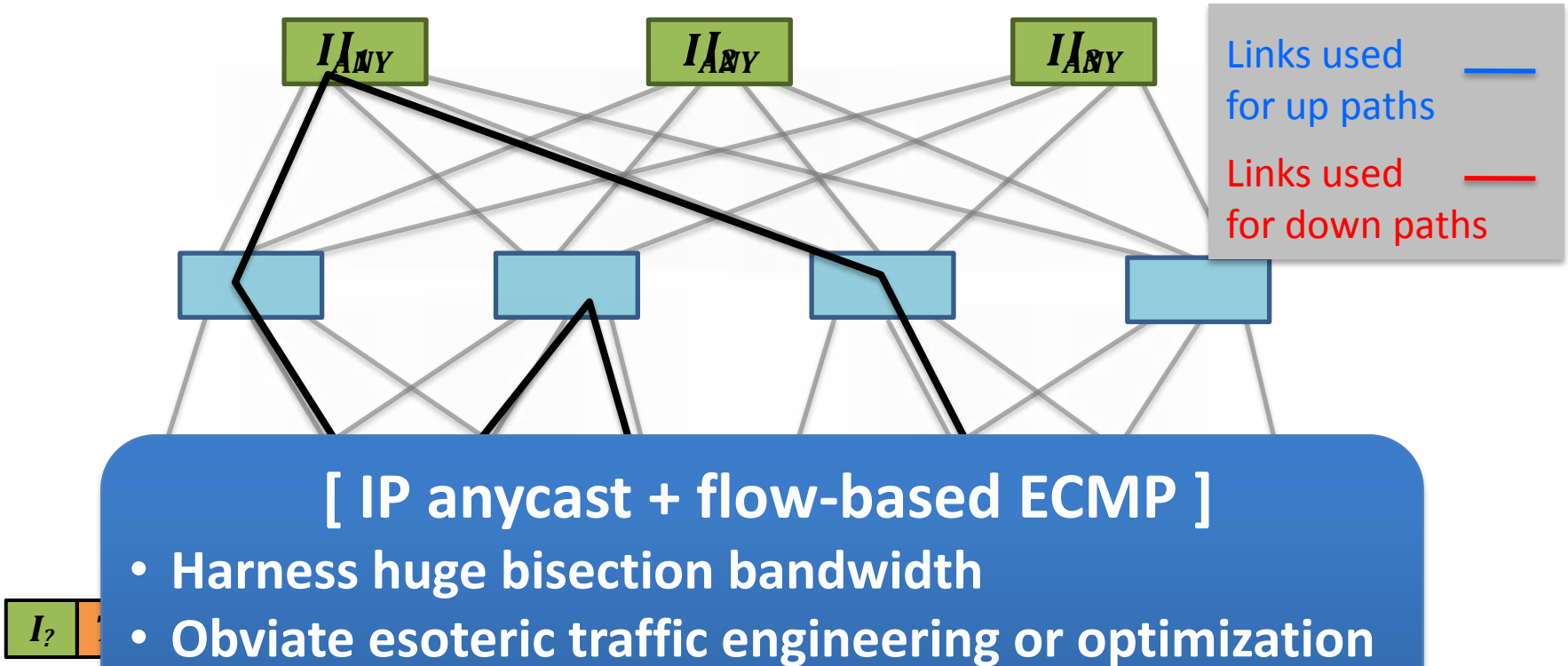
Example Topology: Clos Net

Offer huge <u>aggregate</u> capacity at modest cost



Traffic Forwarding: Random Indirection

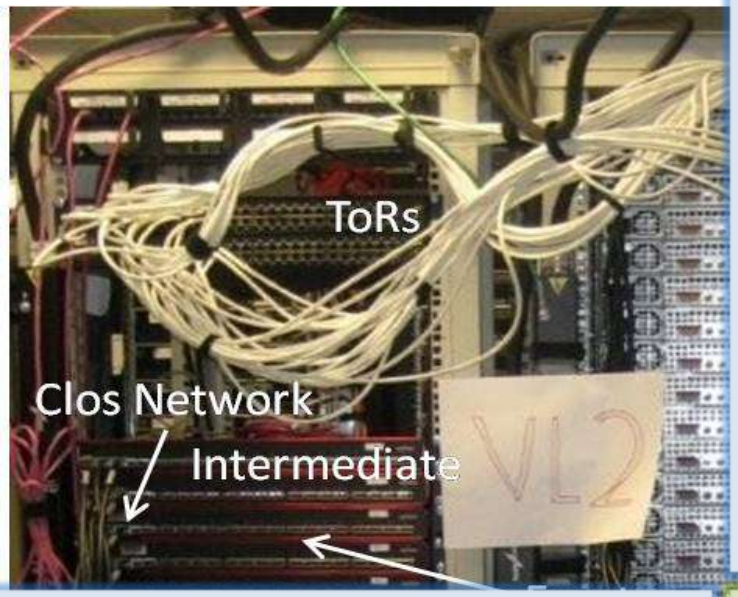
Cope with arbitrary TMs with very little overhead



[IP anycast + flow-based ECMP]

- Harness huge bisection bandwidth
- Obviate esoteric traffic engineering or optimization
- Ensure robustness to failures
- Work with switch mechanisms available today

Implementation



Switches

- Commodity Ethernet ASICs
- Custom settings for line-rate decapsulation
- Default buffer-size settings
- No QoS or flow control

Directory service

- Replicated state-machine (RSM) servers, and lookup proxies
- Various distributed-systems techniques

App servers

- Custom Windows kernel for encapsulation & directory-service access

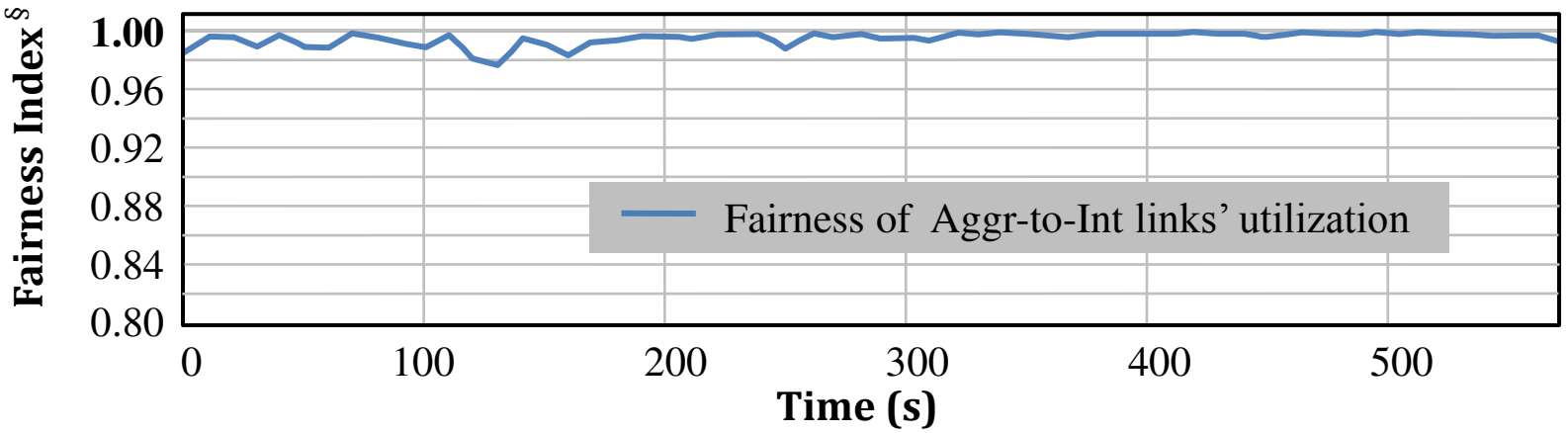
Data-Plane Evaluation

- Ensures uniform high capacity
  - Offered various TCP traffic patterns, then measured overall and per-flow goodput

	VL2	Fat Tree	Dcell
Goodput efficiency	**93+%**	75+% (w/o opt) 95+% (with opt)	40 ~ 60%
Fairness between flows	**0.995**[§]	n/a	n/a

- Works nicely with real traffic as well

[§] Jain's fairness index defined as $(\sum x_i)^2/(n\cdot\sum x_i^2)$



VL2 Conclusion

- Cloud-service DC needs a huge L2 switch
  - Uniform high capacity, oblivious TE, L2 semantics
- Key lessons
  - Hard to outsmart haphazardness; tame it with dice
  - Recipe for success: Intelligent hosts + Rock-solid network built with proven technologies

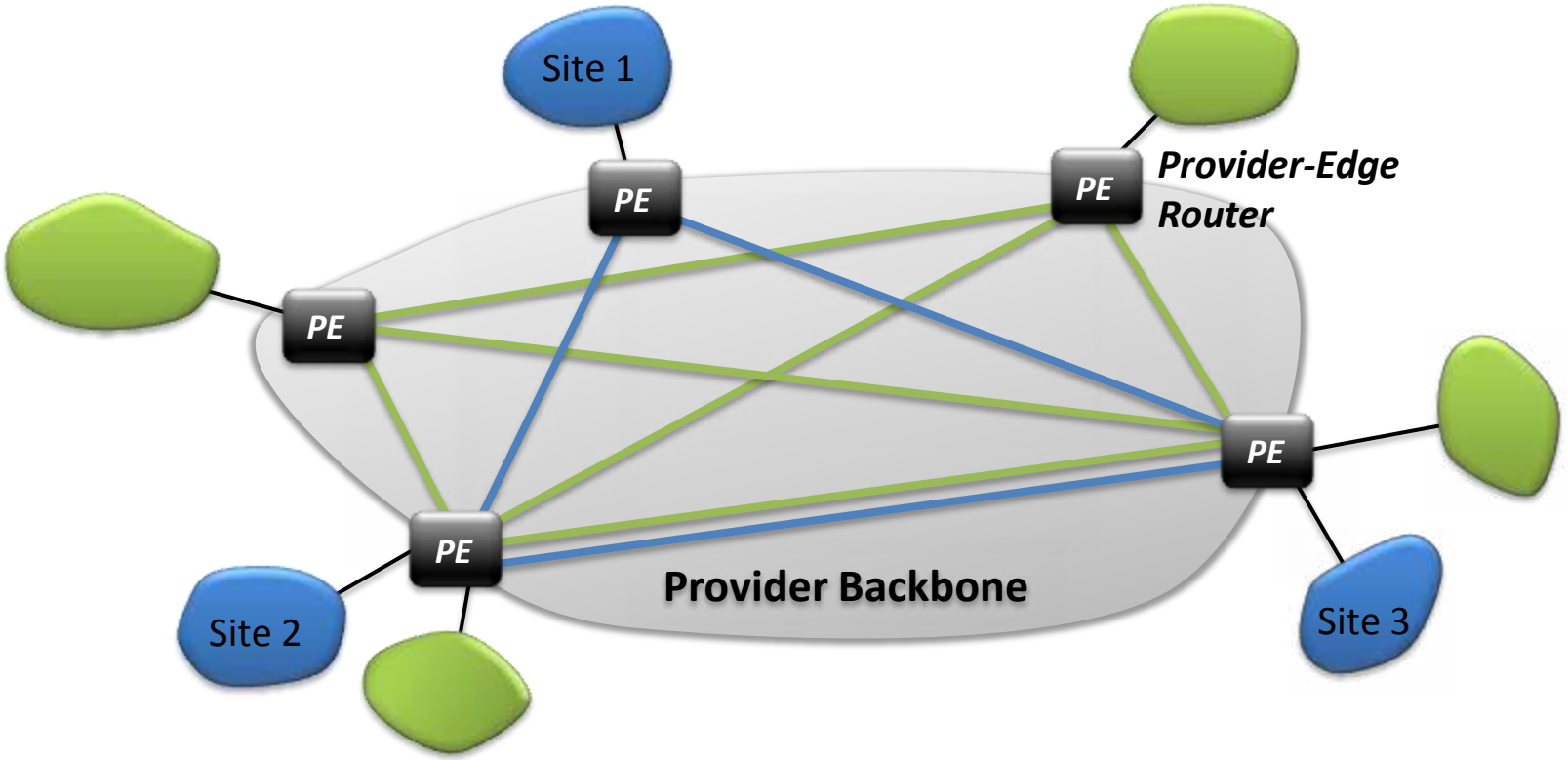
Flat Addressing	***Name-location separation***
Traffic Indirection	***Random traffic spreading (VLB + ECMP)***
Usage-driven Opt.	***Utilizing ARP, reactive cache update***

Relaying: A Scalable VPN Routing Architecture

Work with Alex Gerber, Shubho Sen,
Dan Pei, and Carsten Lund

Virtual Private Network

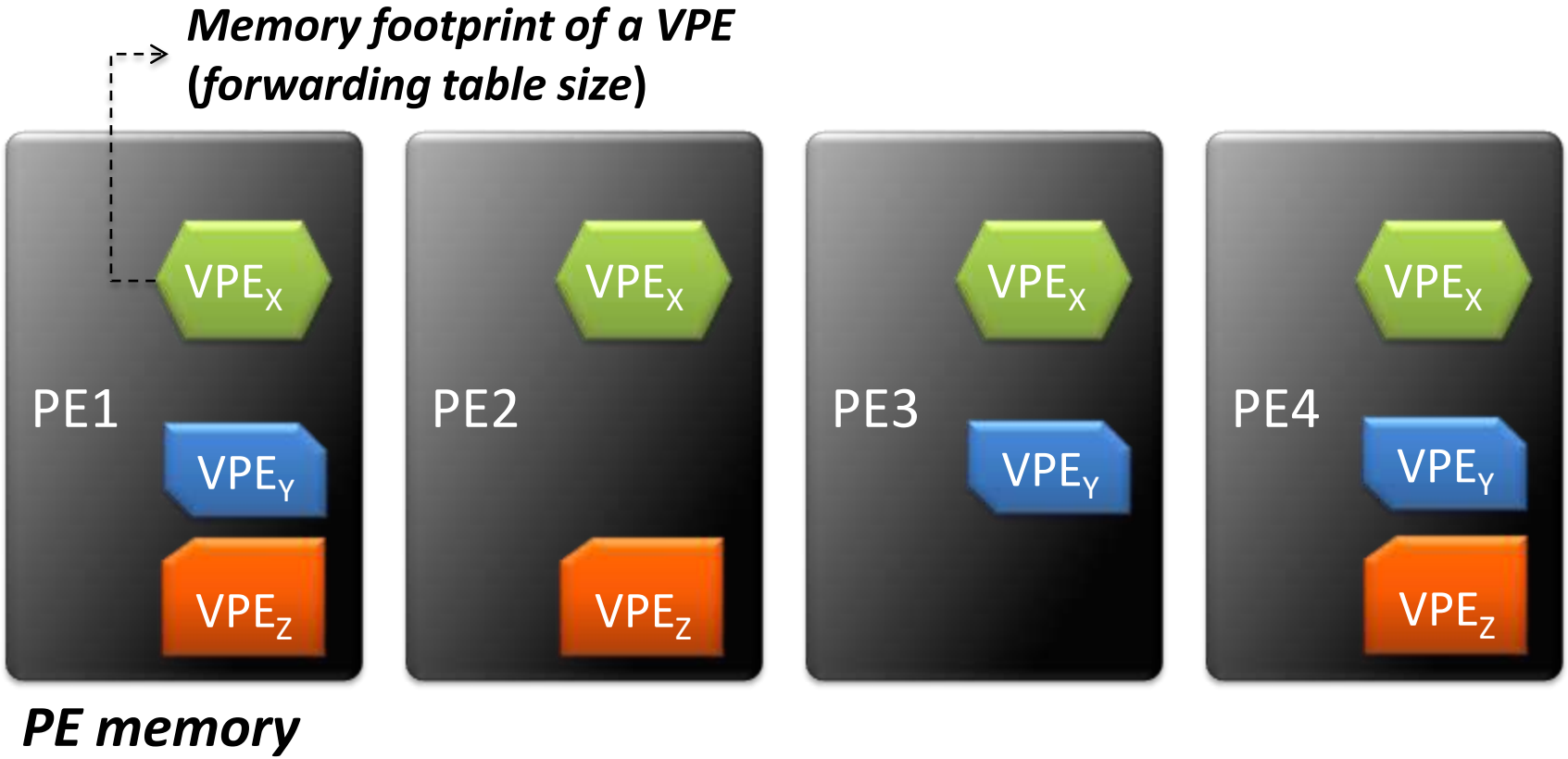
- Logically-isolated communication channels for corporate customers, overlayed over provider backbone
  - Direct any-to-any reachability among sites
  - Customers can avoid full-meshing via outsourcing routing



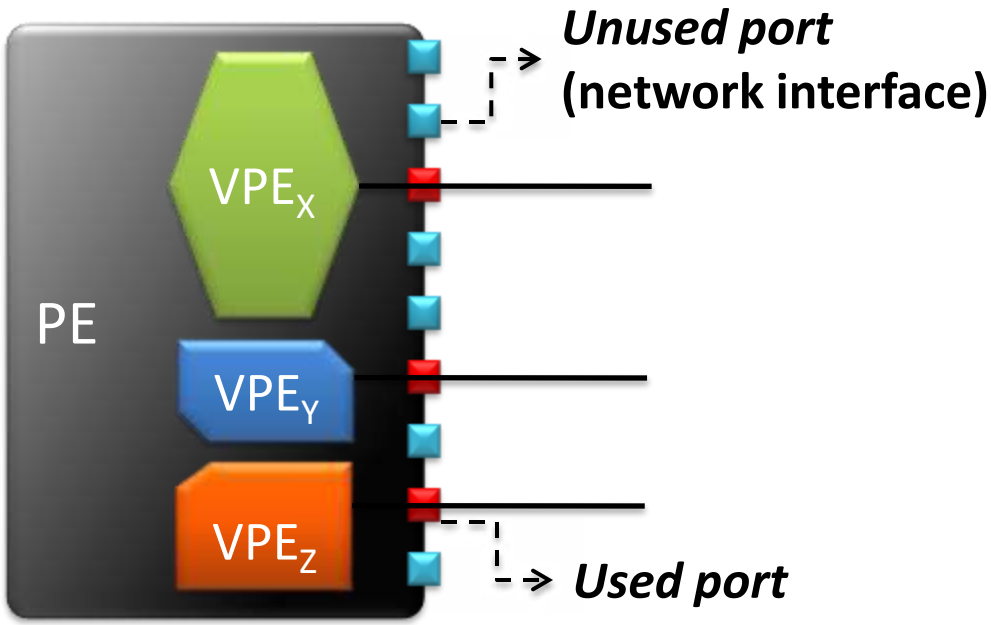
VPN Routing and Its Consequence

Site-level Flat Addressing:

Virtual PEs (VPEs) self-learn and maintain full routing state in the VPN (i.e., routes to every address block used in each site)



Mismatch in Usage of Router Resources



- Memory is full, whereas lots of ports still unused
- Revenue is proportional to provisioned bandwidth
- Large VPN with a thin connection per site is the worst case
- Unfortunately, there are many such worst cases
- Providers are seriously pinched

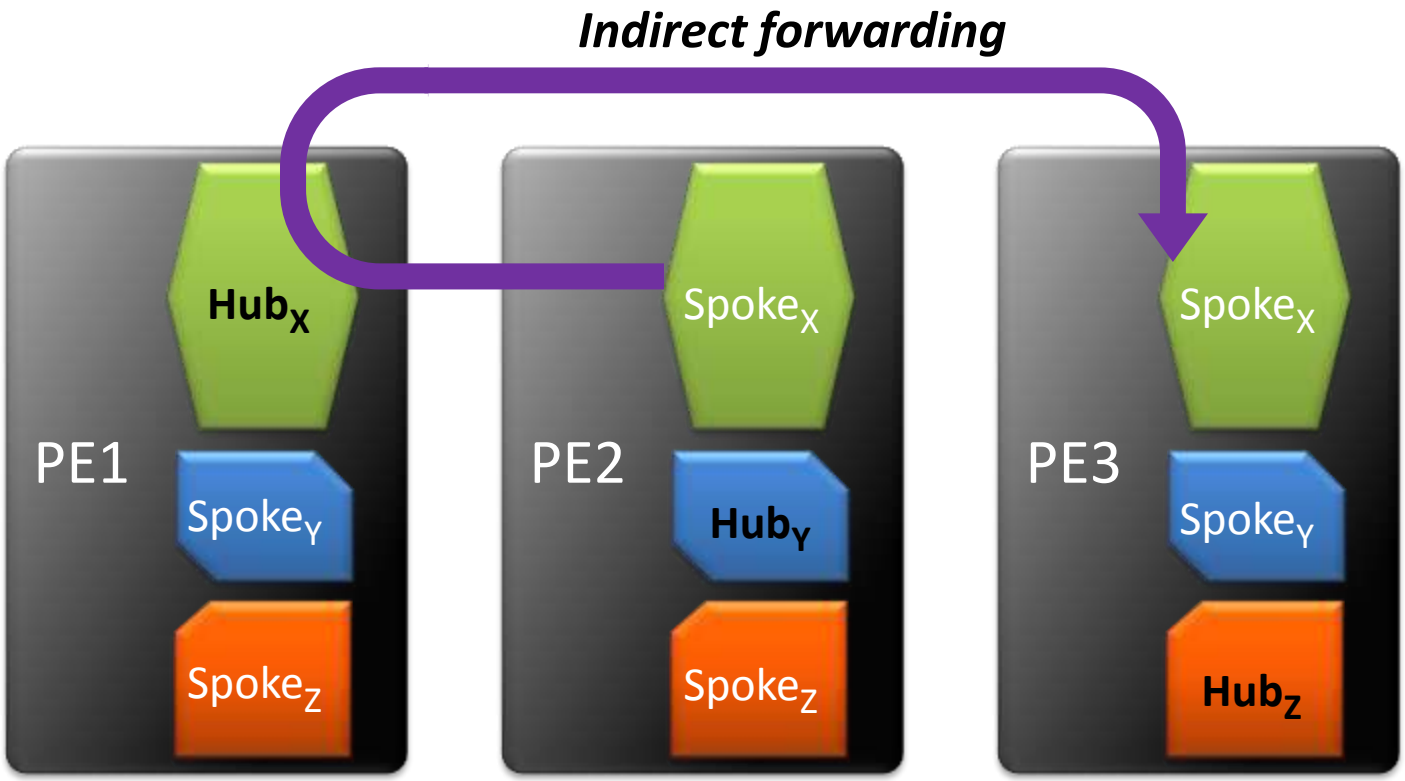
Key Questions

- What can we do better with existing resources and capabilities only, while maintaining complete transparency to customers?

- Do we really need to provide direct reachability for every pair of sites?
  - Even when most (84%) PEs communicate only with a small number (~10%) or popular PEs …

Relaying Saves Memory

- Each VPN has two different types of PEs
  - **Hubs**: Keep full routing state of a VPN
  - **Spokes**: Keep local routes and a single default route to a hub
- A spoke uses a hub consistently for all non-local traffic



Real-World Deployment Requires A Mgmt-Support Tool

- Two operational problems to solve
  - Hub selection: Which PEs should be hubs?
  - Hub assignment: Which hub should a given spoke use?
- Constraint
  - Stretch penalty must be bounded to keep SLAs
- Solve the problems individually for each VPN
  - Hub selection and assignment decision for a VPN is totally independent of that of other VPNs
  - Ensures both simplicity and flexibility

Latency-Constrained Relaying (LCR)

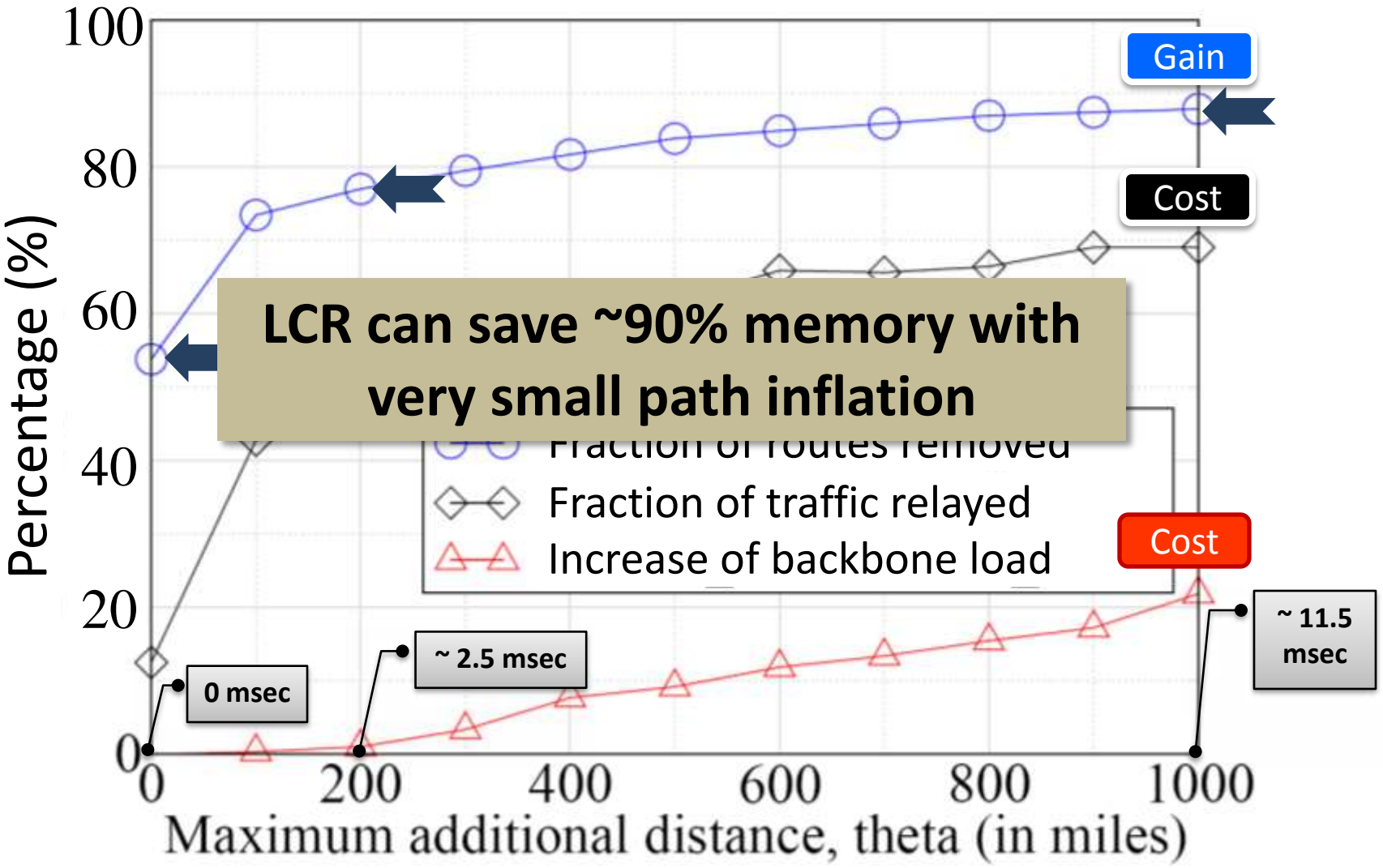
- Notation
  - PE set: $P = \{1, 2, \cdots, n\}$
  - Hub set: $H \subseteq P$
  - The hub of PE$_i$: $hub(i) \in H$
  - Usage-based conversation matrix: $C = (c_{i,j})$
  - Latency matrix: $L = (l_{i,j})$
- Formulation
  - Choose as few hubs as possible, while limiting additional distance due to Relaying

$$\begin{aligned} \min \quad & |H| \\ s.t. \quad & \forall s,d \in P \ \ whose \ \ c_{s,d} = 1, \\ & l_{s,hub(s)} + l_{hub(s),d} - l_{s,d} \le \theta \end{aligned}$$

→ Parameter

Memory Saving and Cost of LCR

Based on entire traffic in 100+ VPNs for a week in May, 2007

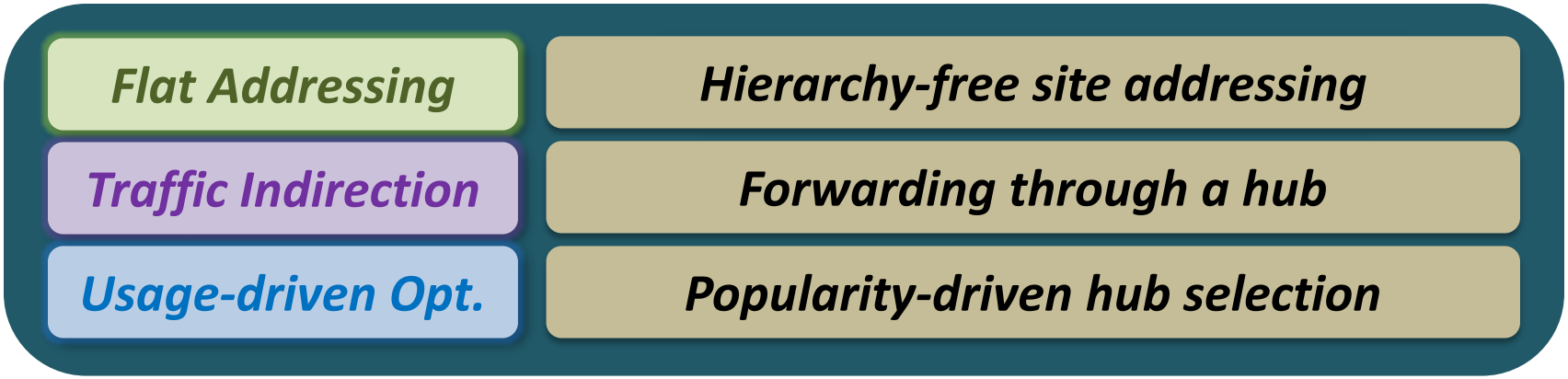


Deployment and Operation

- Oblivious optimization also leads to significant benefits
- Can implement this via minor routing protocol configuration change at PEs
- Performance degrades very little over time
  - Cost curves are fairly robust
  - Weekly/monthly adjustment: 94/91% of hubs remain as hubs
- Can ensure high availability
  - Need more than one hub located at different cities
  - 98.3% of VPNs spanning 10+ PEs have at least 2 hubs anyway
  - Enforcing "|H| > 1" reduces memory saving by only 0.5%

Relaying Conclusion

- VPN providers need a huge L2-like switch
  - Site-level PNP networking, any-to-any reachability, and scalability
- Key lessons
  - Traffic locality is our good friend
  - Presenting an immediately-deployable solution requires more than just designing an architecture



Goals Attained

Self-config + **Scalability** + **Efficiency**

SEATTLE	VL2	Relaying
Eliminates addressing and routing configuration	Additionally eliminates configuration for traffic engineering	Retains self-configuring semantics for VPN customers
Significantly reduces control overhead and memory consumption	Allows a DC to host over 100K servers w/o oversubscription	Allows existing routers to serve an order of mag. more VPNs
Improves link utilization and convergence	Boosts DC-server utilization by enabling agility	Only slightly increases end-to-end latency and traffic workload

Summary and Future Work

- Designed, built, and deployed **huge L2-like switches** for various networks
- The **universal hammers** are applicable for various situations
- Future work
  - Other universal hammers?
  - Self-configuration on the Internet scale?
  - Architecture for distributed mini data centers?