6.888:
Lecture 2
Data Center Network Architectures

Mohammad Alizadeh

Spring 2016

✧ Slides adapted from presentations by Albert Greenberg and Changhoon Kim (Microsoft)

Data Center Costs

Amortized Cost*	Component	Sub-Components
~45%	Servers	CPU, memory, disk
~25%	Power infrastructure	UPS, cooling, power distribution
~15%	Power draw	Electrical utility costs
~15%	Network	Switches, links, transit

The Cost of a Cloud: Research Problems in Data Center Networks. Sigcomm CCR 2009. Greenberg, Hamilton, Maltz, Patel.

*3 yr amortization for servers, 15 yr for infrastructure; 5% cost of money

Server Costs

Ugly secret: 30% utilization considered "good" in data centers

Uneven application fit

- Each server has CPU, memory, disk: most applications exhaust one resource, stranding the others

Long provisioning timescales

- New servers purchased quarterly at best

Uncertainty in demand

- Demand for a new service can spike quickly

Risk management

- Not having spare servers to meet demand brings failure just when success is at hand

Session state and storage constraints

- If the world were stateless servers, life would be good

Goal: Agility – Any service, Any Server

Turn the servers into a single large fungible pool

- Dynamically expand and contract service footprint as needed

Benefits

- Increase service developer productivity
- Lower cost
- Achieve high performance and reliability

The 3 motivators of most infrastructure projects

Achieving Agility

Workload management

- Means for rapidly installing a service's code on a server
- *Virtual machines, disk images, containers*

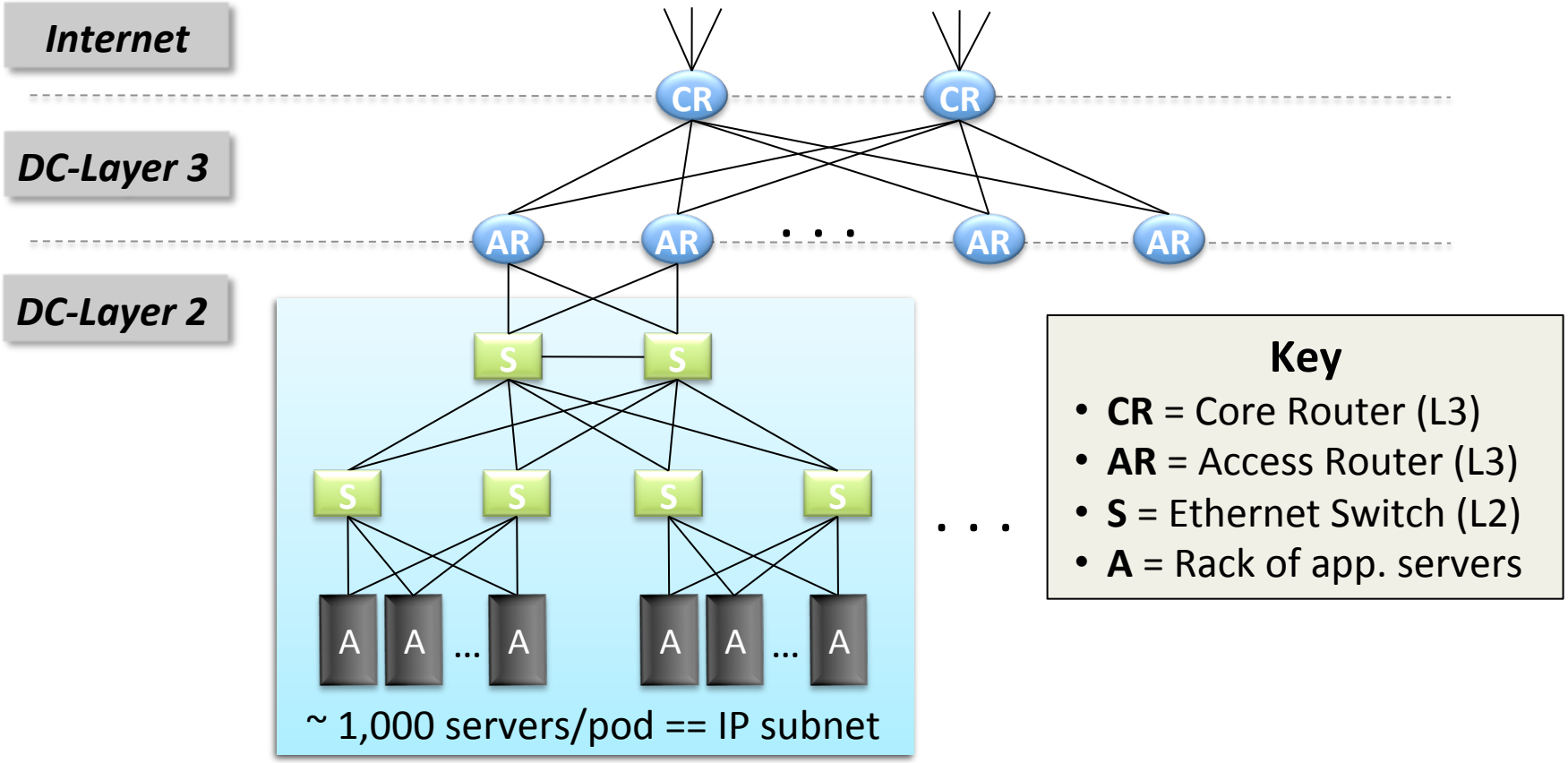
Storage Management

- Means for a server to access persistent data
- *Distributed filesystems (e.g., HDFS, blob stores)*

Network

- Means for communicating with other servers, regardless of where they are in the data center

Conventional DC Network



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

Layer 2 vs. Layer 3

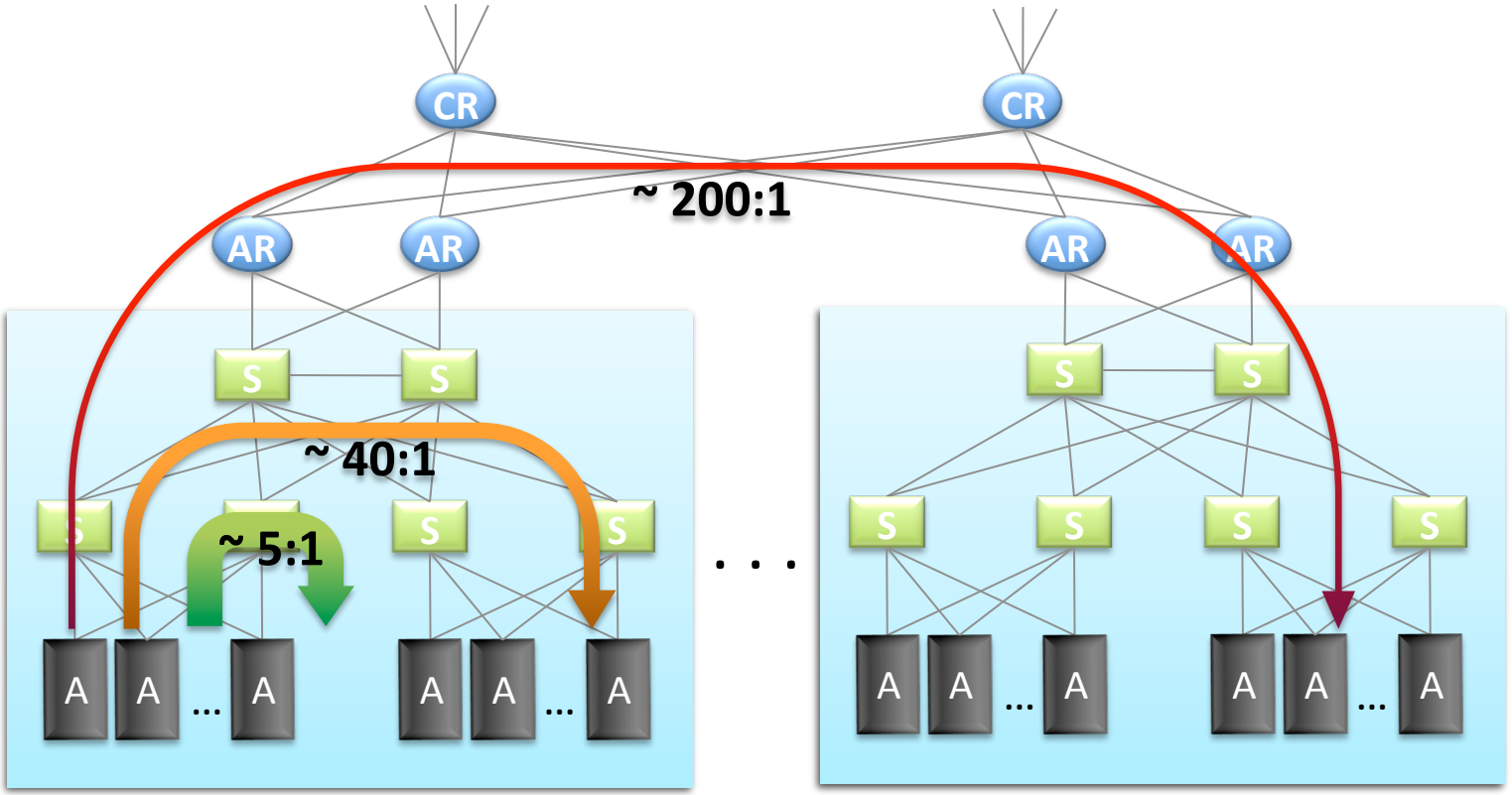
Ethernet switching (layer 2)

- ✓ Fixed IP addresses and auto-configuration (plug & play)
- ✓ Seamless mobility, migration, and failover
- x Broadcast limits scale (ARP)
- x Spanning Tree Protocol

IP routing (layer 3)

- ✓ Scalability through hierarchical addressing
- ✓ Multipath routing through equal-cost multipath
- x More complex configuration
- x Can’t migrate w/o changing IP address

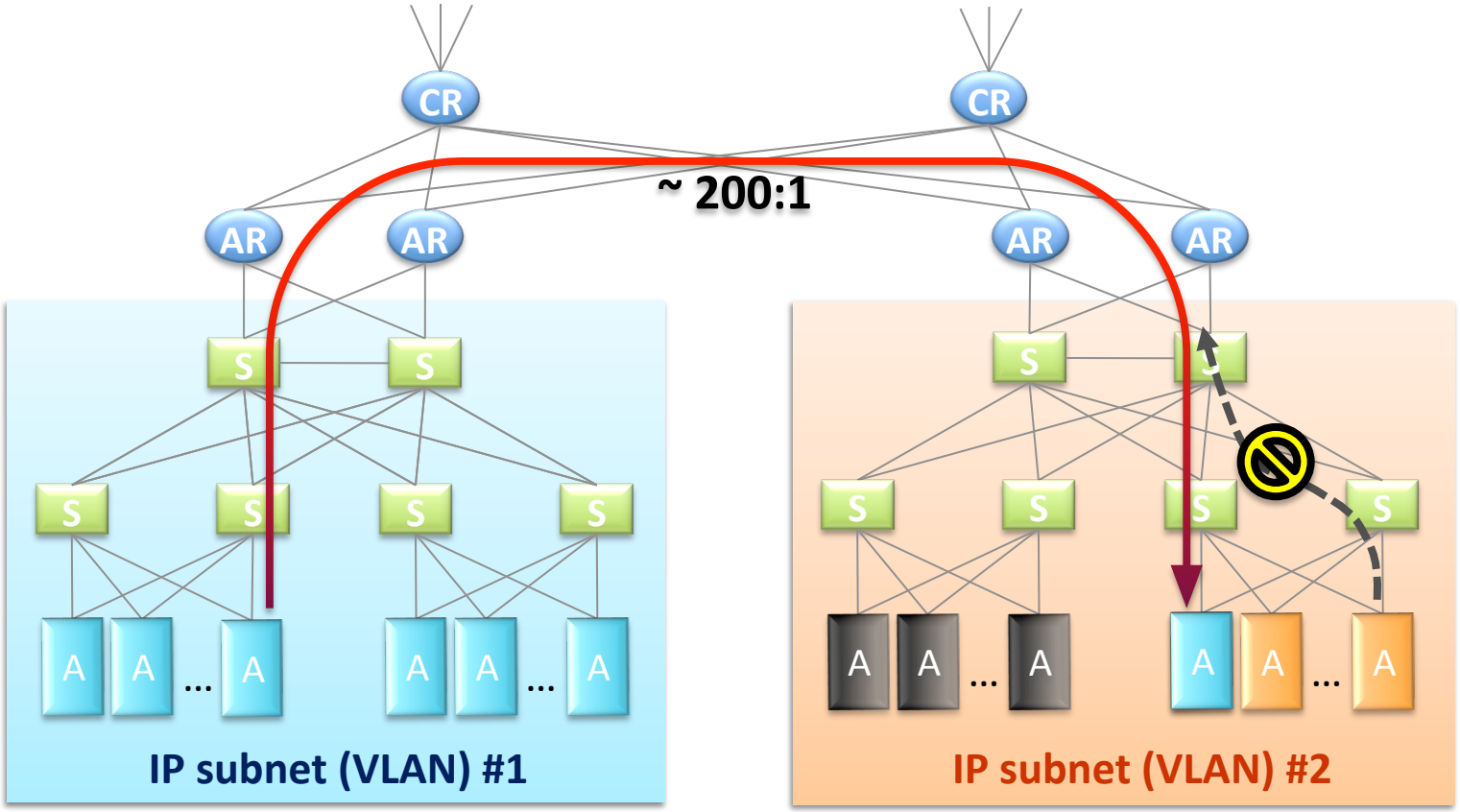
Conventional DC Network Problems



Dependence on high-cost proprietary routers

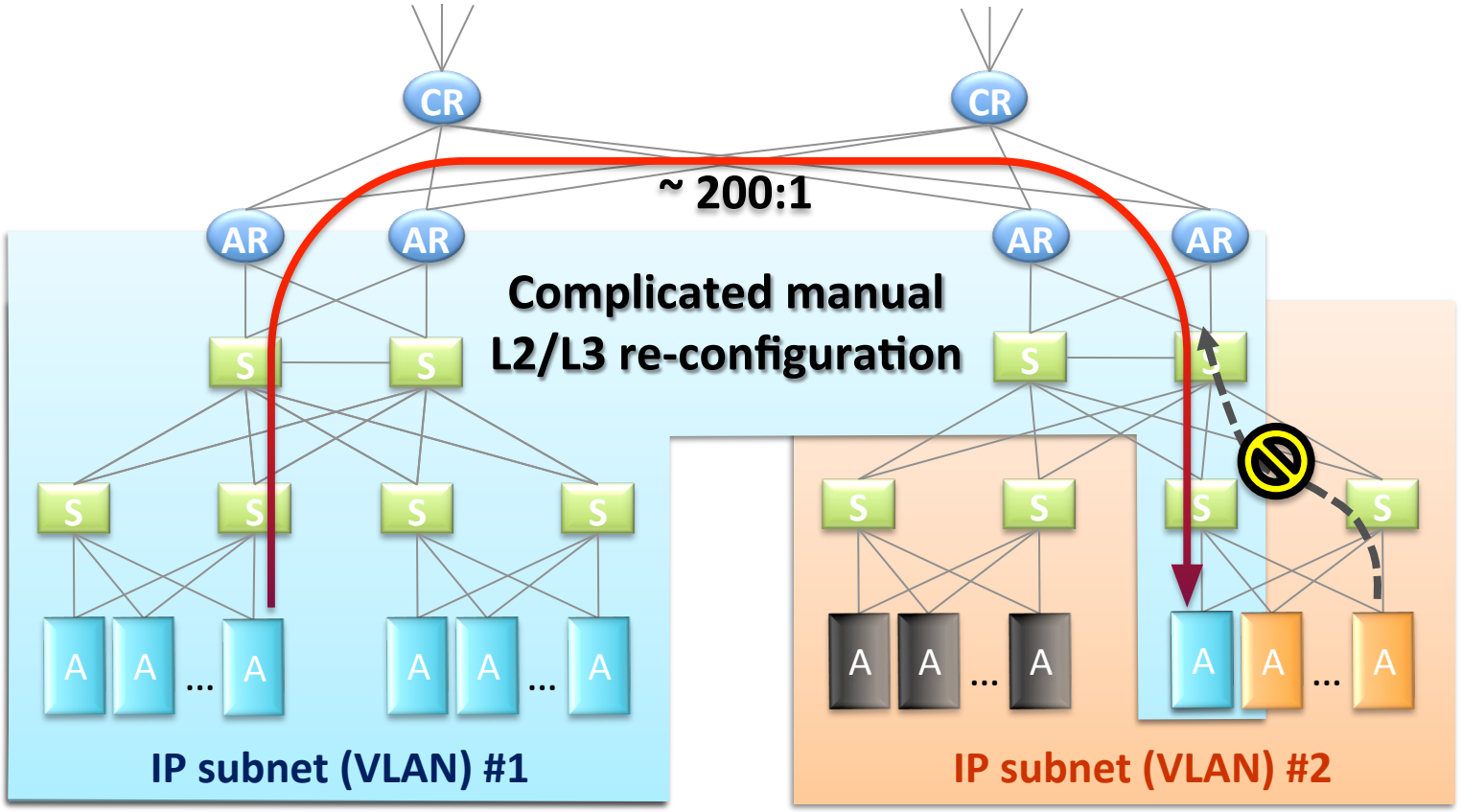
Extremely limited server-to-server capacity

And More Problems …



- Resource fragmentation, significantly lowering cloud utilization (and cost-efficiency)

And More Problems …



- Resource fragmentation, significantly lowering cloud utilization (and cost-efficiency)

Measurements

DC Traffic Characteristics

Instrumented a large cluster used for data mining and identified distinctive traffic patterns

Traffic patterns are **highly volatile**

- A large number of distinctive patterns even in a day

Traffic patterns are **unpredictable**

- Correlation between patterns very weak

Traffic-aware optimization needs to be done frequently and rapidly

DC Opportunities

DC controller knows **everything** about **hosts**

Host OS's are easily **customizable**

Probabilistic flow distribution would work well enough, because …

- Flows are numerous and not huge – no elephants
- Commodity switch-to-switch links are substantially thicker (~ 10x) than the maximum thickness of a flow

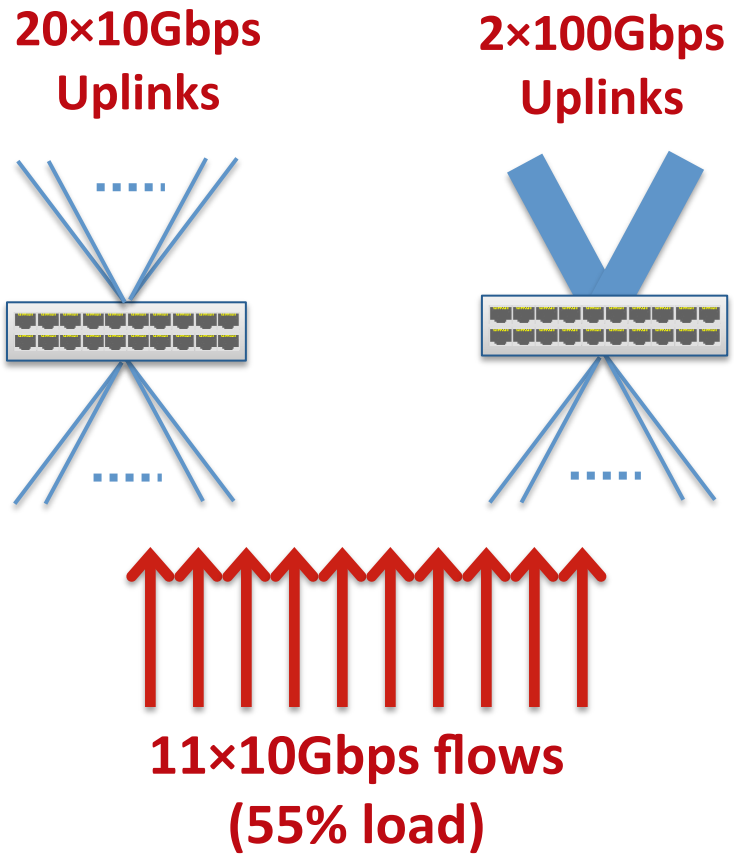
?

?

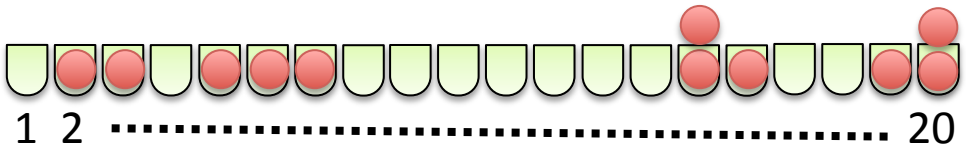
DC network can be made simple

Intuition

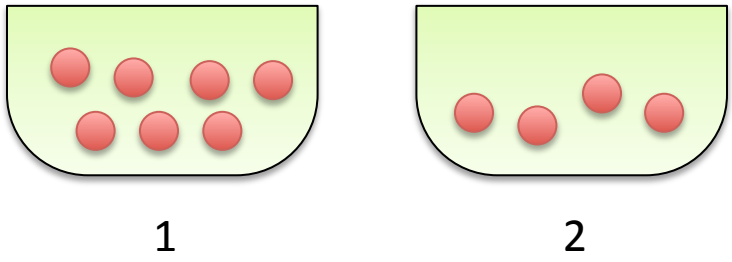
Higher speed links improve *flow-level* load balancing (ECMP)



Prob of 100% throughput = 3.27%



Prob of 100% throughput = 99.95%

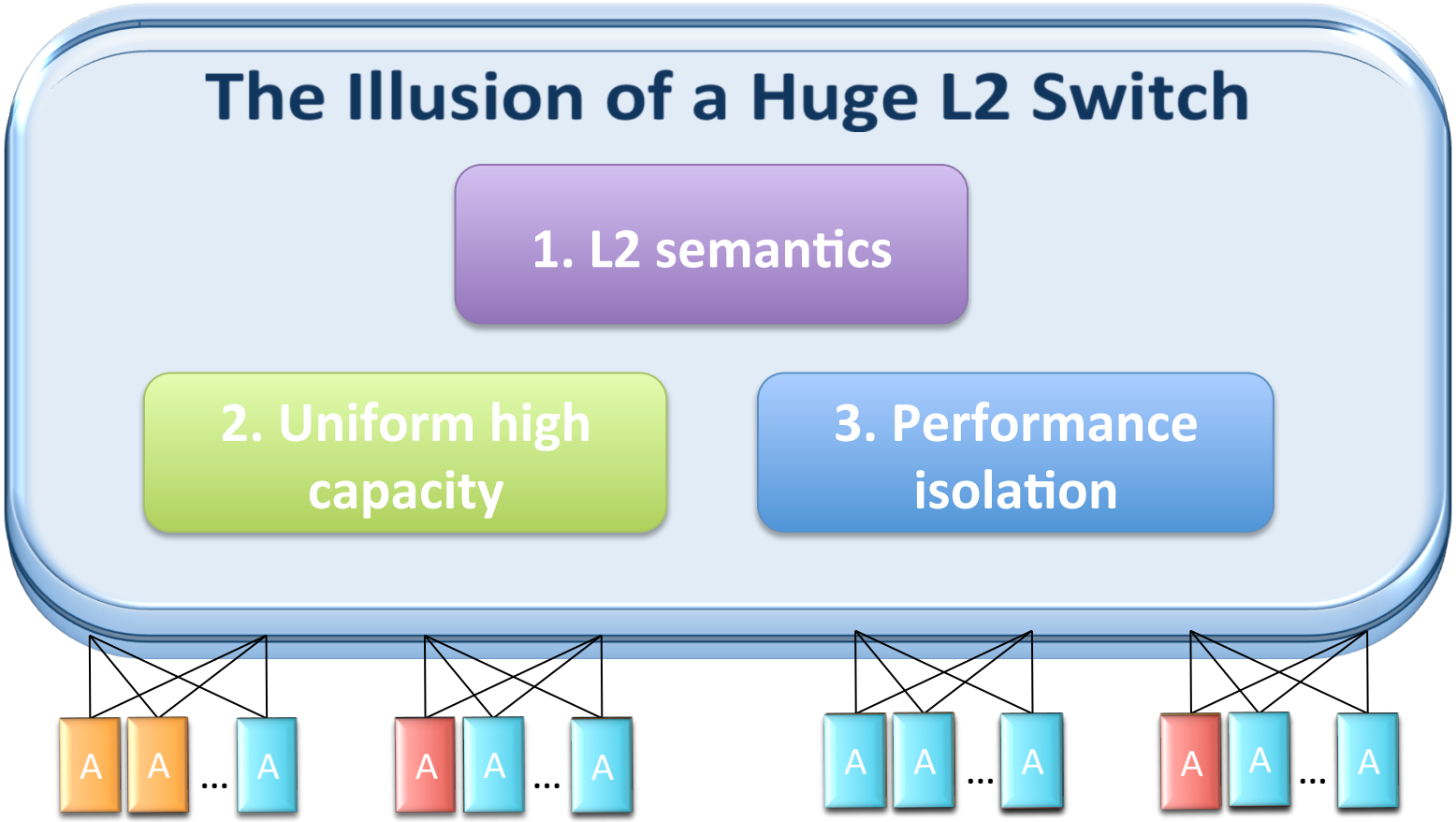


What You Said

“In 3.2, the paper states that randomizing large flows won't cause much perpetual congestion if misplaced since large flows are only 100 MB and thus take 1 second to transmit on a 1 Gbps link. Isn't 1 second sufficiently high to harm the isolation that VL2 tries to provide?”

Virtual Layer 2 Switch

VL2 Goals


The Illusion of a Huge L2 Switch
1. L2 semantics
2. Uniform high capacity
3. Performance isolation
A
A
...
A
A
A
...
A
A
A
...
A
A
A
...
A

VL2 Design Principles

Randomizing to Cope with Volatility

- Tremendous variability in traffic matrices

Separating Names from Locations

- Any server, any service

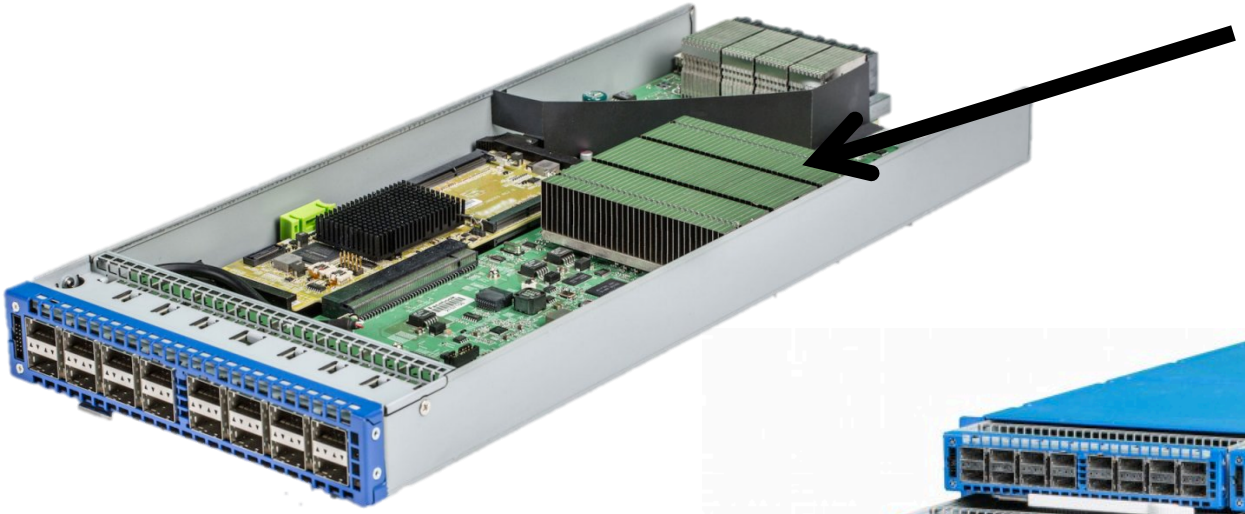
Embracing End Systems

- Leverage the programmability & resources of servers
- Avoid changes to switches

Building on Proven Networking Technology

- Build with parts shipping today
- Leverage low cost, powerful merchant silicon ASICs, though do not rely on any one vendor

Single-Chip “Merchant Silicon” Switches



Switch ASIC

Wedge

6 pack

✧ Image courtesy of Facebook

Specific Objectives and Solutions

Objective	Approach	Solution
1. Layer-2 semantics	Employ flat addressing	Name-location separation & resolution service
2. Uniform high capacity between servers	Guarantee bandwidth for hose-model traffic	Flow-based random traffic indirection (Valiant LB)
3. Performance Isolation	Enforce hose model using existing mechanisms only	TCP

Discussion

What You Said

“It is interesting that this paper is from 2009. It seems that a large number of the suggestions in this paper are used in practice today.”

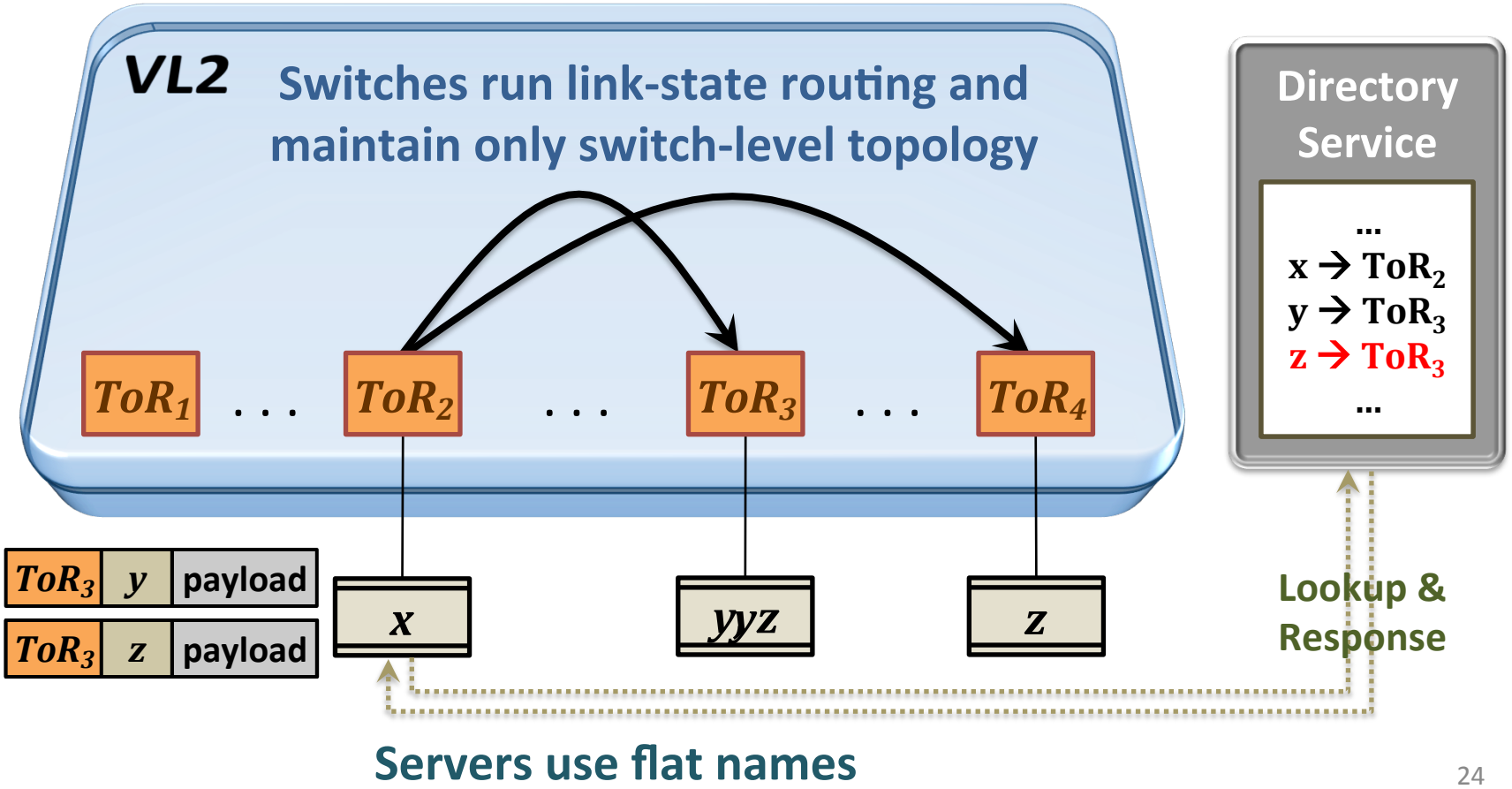
What You Said

"For address resolution, why not have applications use hostnames and use DNS to resolve hostnames to IP addresses (the mapping from hostname to IP could be updated when a service moved)? Is the directory system basically just DNS but with IPs instead of hostnames?"

"it was unclear why the hash of the 5 tuple is required."

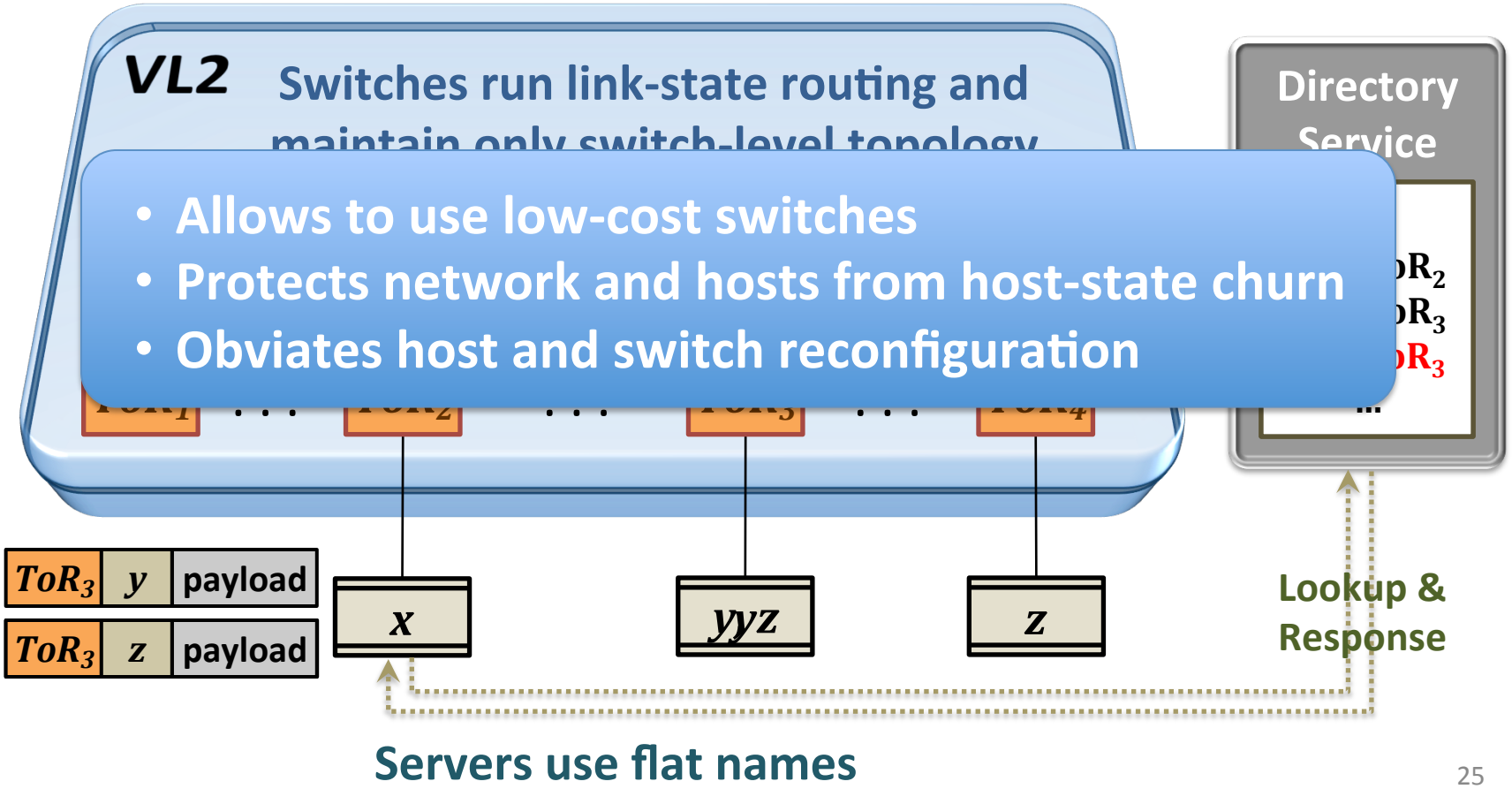
Addressing and Routing: Name-Location Separation

Cope with host churns with very little overhead



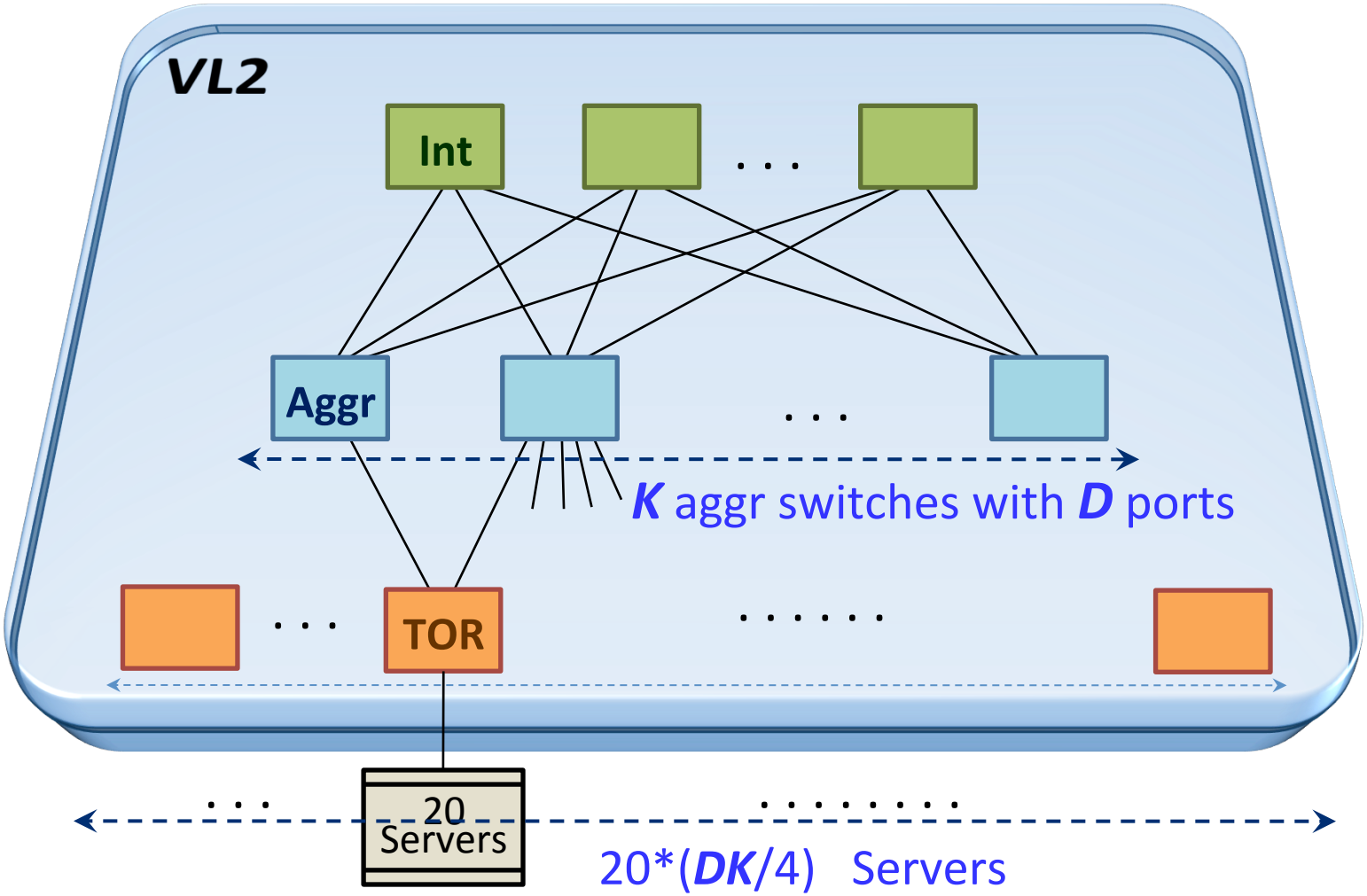
Addressing and Routing: Name-Location Separation

Cope with host churns with very little overhead



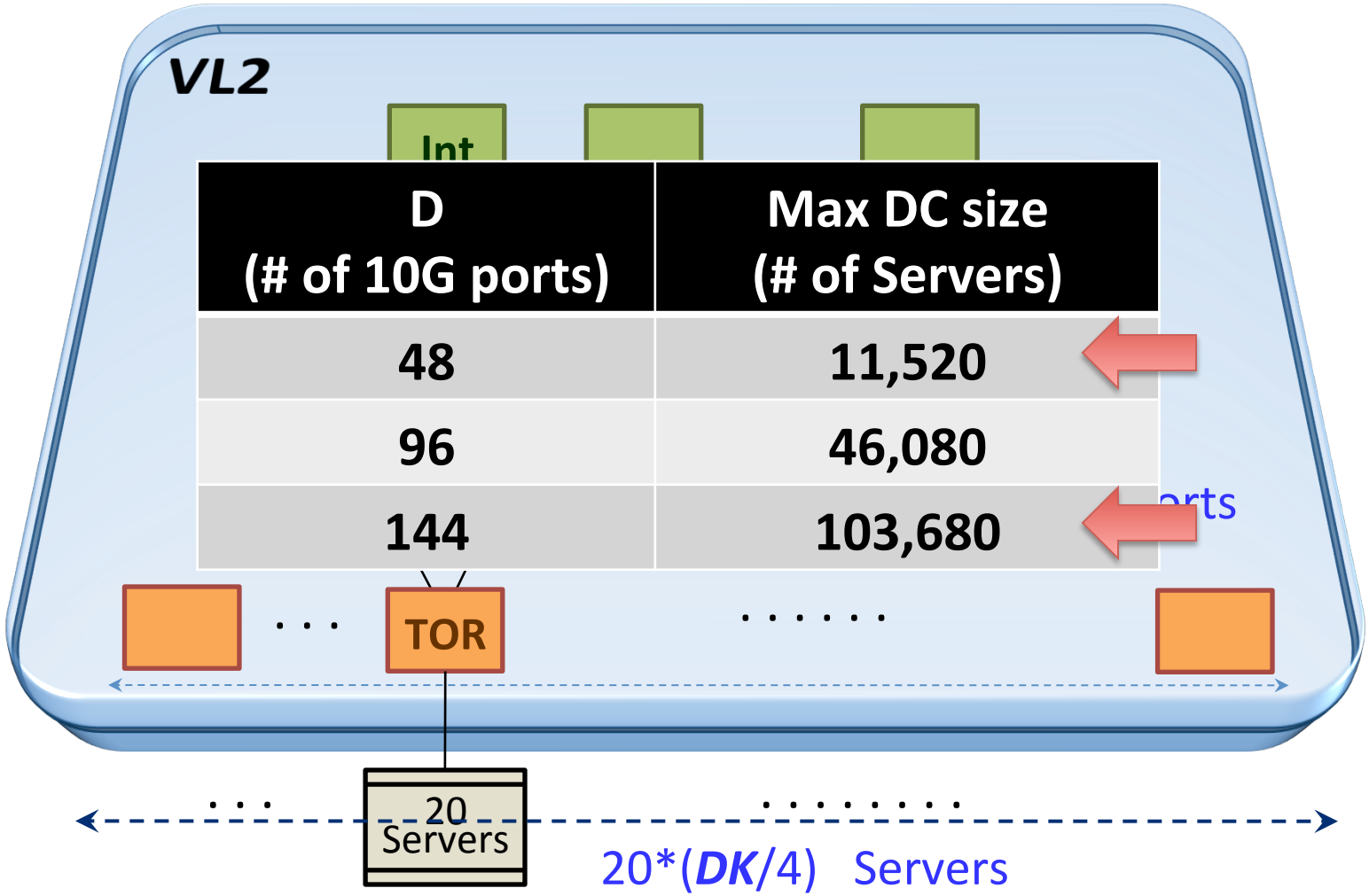
Example Topology: Clos Network

Offer huge aggr capacity and multi paths at modest cost



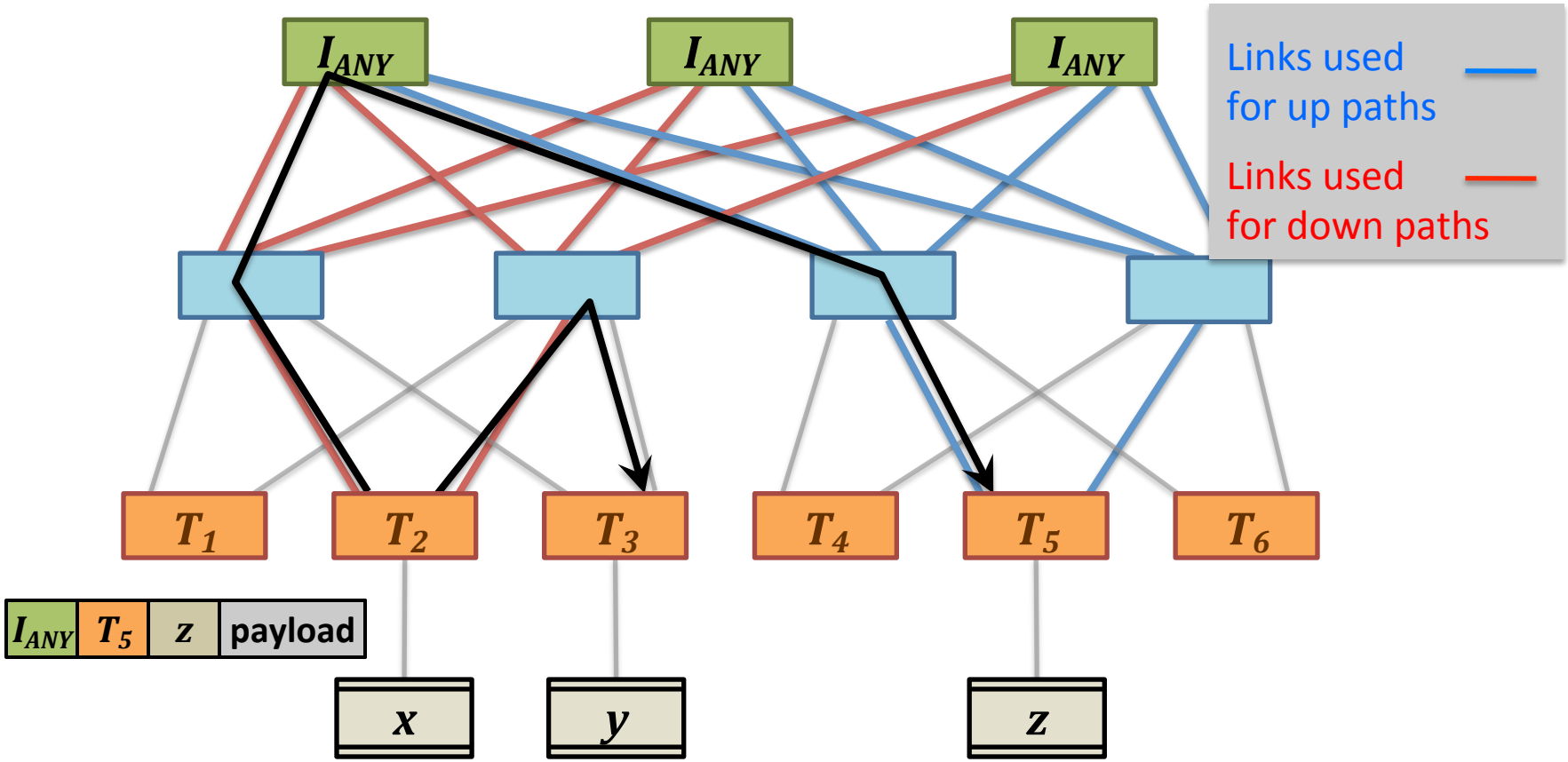
Example Topology: Clos Network

Offer huge aggr capacity and multi paths at modest cost



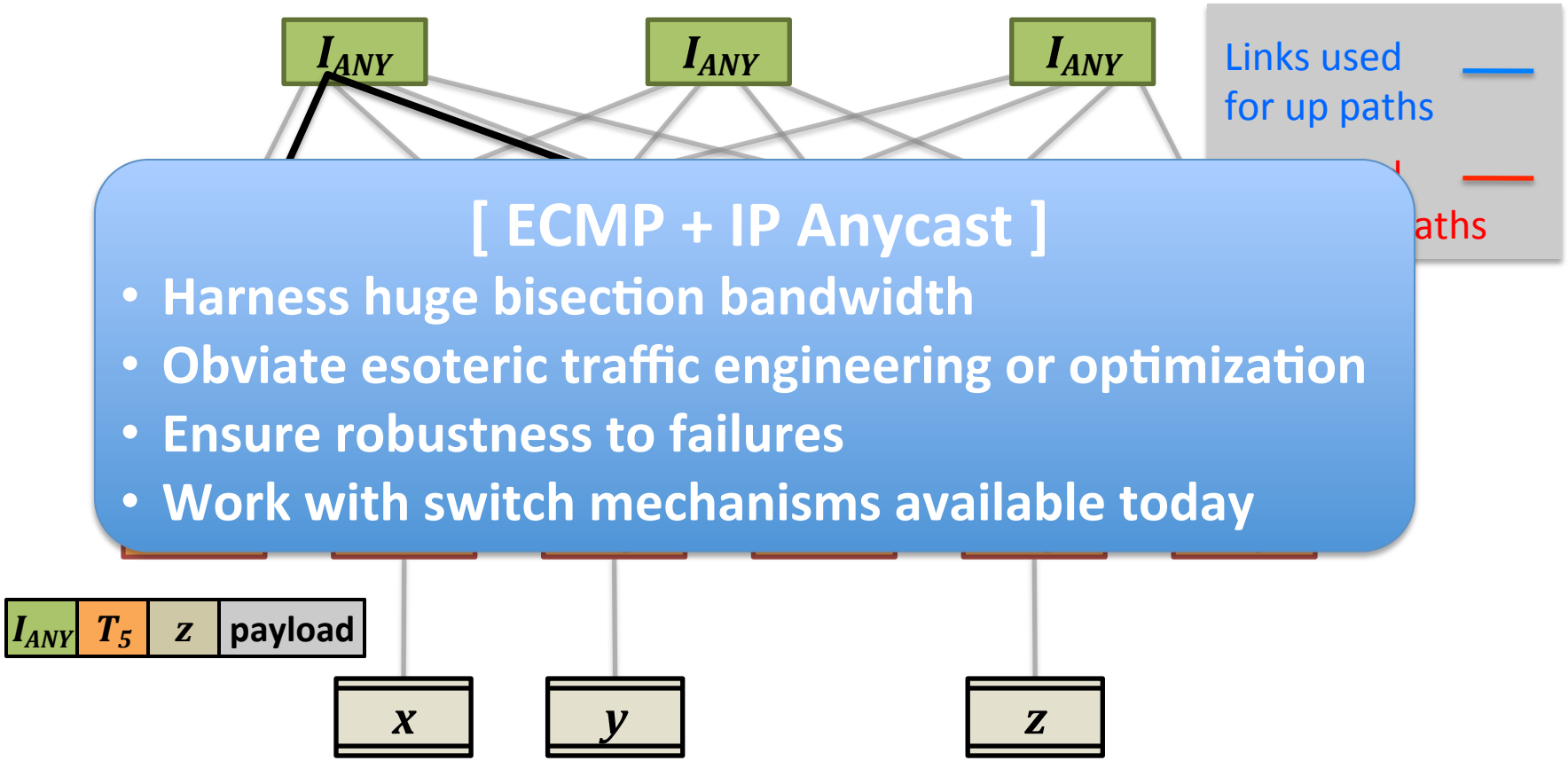
Traffic Forwarding: Random Indirection

Cope with arbitrary TMs with very little overhead



Traffic Forwarding: Random Indirection

Cope with arbitrary TMs with very little overhead

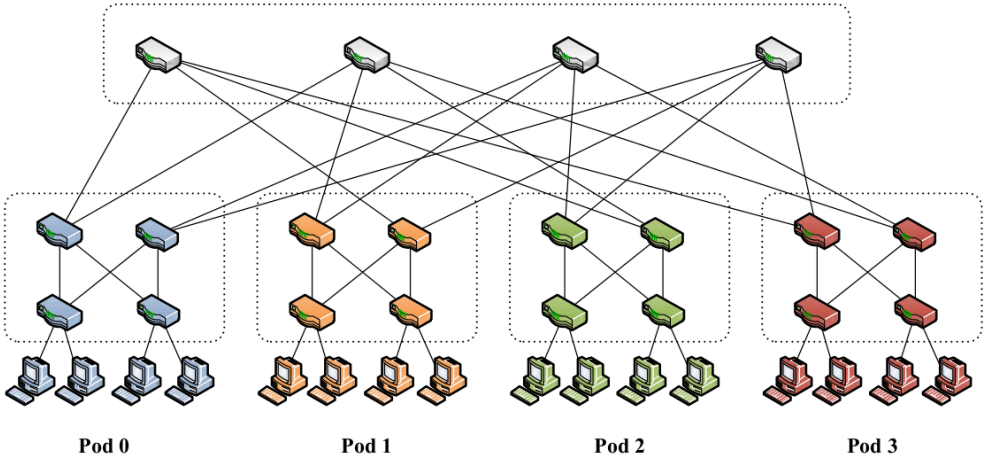


What you said

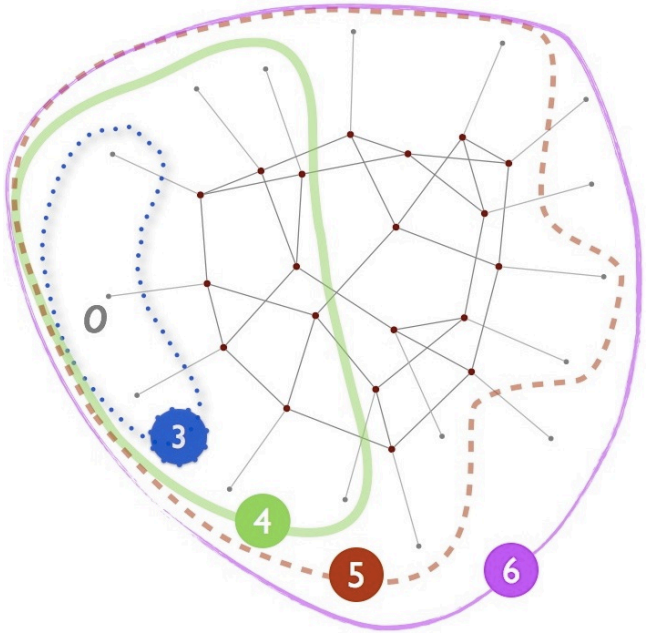
"… the heterogeneity of racks and the incremental deployment of new racks may introduce asymmetry to the topology. In this case, more delicate topology design and routing algorithms are needed. "

Some other DC network designs…

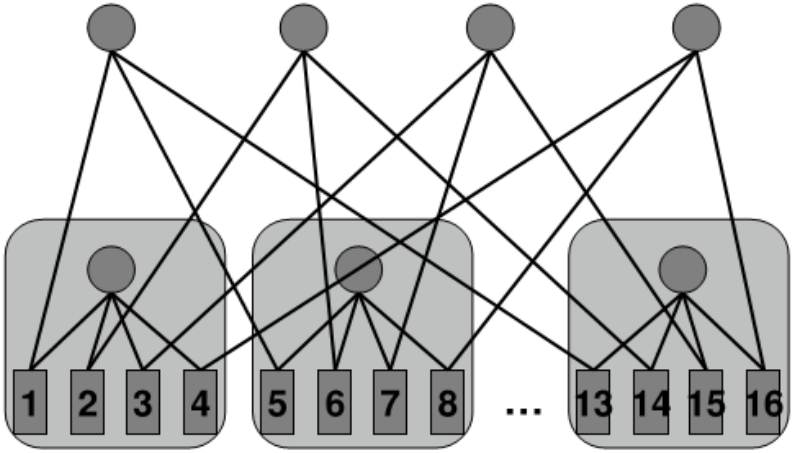
Fat-tree [SIGCOMM'08]



Jellyfish (random) [NSDI'12]



BCube [SIGCOMM'10]



Next time: Congestion Control

Data Center TCP (DCTCP)

Mohammad Alizadeh[‡†], Albert Greenberg[†], David A. Maltz[†], Jitendra Padhye
Parveen Patel[†], Balaji Prabhakar[‡], Sudipta Sengupta[†], Murari Sridharan[†]

[†]Microsoft Research [‡]Stanford University
{albert, dmaltz, padhye, parveenp, sudipta, muraris}@microsoft.com
{alizade, balaji}@stanford.edu

ABSTRACT

Cloud data centers host diverse applications, mixing workloads that require small predictable latency with others requiring large sustained throughput. In this environment, today's state-of-the-art TCP protocol falls short. We present measurements of a 6000 server production cluster and reveal impairments that lead to high application latencies, rooted in TCP's demands on the limited buffer space available in data center switches. For example, bandwidth hungry "background" flows build up queues at the switches, and thus impact the performance of latency sensitive "foreground" traffic.

To address these problems, we propose DCTCP, a TCP-like protocol for data center networks. DCTCP leverages Explicit Congestion Notification (ECN) in the network to provide multi-bit feedback to the end hosts. We evaluate DCTCP at 1 and 10Gbps speeds using commodity, shallow buffered switches. We find DCTCP delivers the same or better throughput than TCP, while using 90% less buffer space. Unlike TCP, DCTCP also provides high burst tolerance and low latency for short flows. In handling workloads derived from operational measurements, we found DCTCP enables the applications to handle 10X the current background traffic, without impacting foreground traffic. Further, a 10X increase in foreground traffic does not cause any timeouts, thus largely eliminating incast problems.

eral recent research proposals envision creating economi
to-manage data centers using novel architectures built at
commodity switches [2, 12, 15].

Is this vision realistic? The answer depends in large part
well the commodity switches handle the traffic of real dat
ter applications. In this paper, we focus on soft real-time ap
tions, supporting web search, retail, advertising, and recomm
tion systems that have driven much data center construction. T
applications generate a diverse mix of short and long flows,
require three things from the data center network: low latency
short flows, high burst tolerance, and high utilization for long flow

The first two requirements stem from the *Partition/Aggrega*
(described in §2.1) workflow pattern that many of these applica
tions use. The near real-time deadlines for end results translate into
latency targets for the individual tasks in the workflow. These tar-
gets vary from ∼10ms to ∼100ms, and tasks not completed before
their deadline are cancelled, affecting the final result. Thus, *appli-
cation requirements for low latency directly impact the quality of
the result returned and thus revenue.* Reducing network latency al-
lows application developers to invest more cycles in the algorithms
that improve relevance and end user experience.

The third requirement, high utilization for large flows, stems
from the need to continuously update internal data structures of

TIMELY: RTT-based Congestion Control fo
Datacenter

Radhika Mittal*(UC Berkeley), Vinh The Lam, Nandita Dukkipati, Emily Blem, H
Monia Ghobadi*(Microsoft), Amin Vahdat, Yaogong Wang, David Wetherall,

Google, Inc.

ABSTRACT

Datacenter transports aim to deliver low latency messaging together with high throughput. We show that simple packet delay, measured as round-trip times at hosts, is an effective congestion signal without the need for switch feedback. First, we show that advances in NIC hardware have made RTT measurement possible with microsecond accuracy, and that these RTTs are sufficient to estimate switch queueing. Then we describe how TIMELY can adjust transmission rates using RTT gradients to keep packet latency low while delivering high bandwidth. We implement our design in host software running over NICs with OS-bypass capabilities. We show using experiments with up to hundreds of machines on a Clos network topology that it provides excellent performance: turning on TIMELY for OS-bypass messaging over a fabric with PFC lowers 99 percentile tail latency by 9X while maintaining near line-rate throughput. Our system also outperforms DCTCP running in an optimized kernel, reducing tail latency by 13X. To the best of our knowledge, TIMELY is the first delay-based congestion control protocol for use in the datacenter, and it achieves its results despite having an order of magnitude fewer RTT signals (due to NIC offload) than earlier delay-based schemes such as Vegas.

1. INTRODUCTION

Datacenter networks run tightly-c
that must be responsive to users,
end computers may exchange info
request, and all of the transfers
enough to let the complete respon
100 ms [24]. To meet these requi
ports must simultaneously deliver
and utilization at low latency (≪
aspects of performance are at o
tency matters because even a sma
can cause a ripple effect that d
mance [21]. As a result, datace
bound latency and packet loss.

Since traditional loss-based
strict requirements, new datace
37, 47], take advantage of netw
set of congestion (e.g., DCTC
ECN), introduce flow abstrac
latency, cede scheduling to a
However, in this work we t
simpler, immediately deploy

The crux of our search is
signal would have several