

Balancing on the edge

Transport affinity without network state

NSDI 2018 | João Taveira Araújo, Lorenzo Saino, Raul Landa and Lennert Buytenhek



this is the last slide (sort of)

FaILD decomposes load balancing as a division of labour

- leverage hardware wherever possible - **no latency cost in expected case**
- push functions requiring state towards hosts - **low latency overhead in worst case**
- efficient, stateless, while ensuring graceful completion of flows
- in production at Fastly since 2013

Read paper if you have an interest in transport protocols, Internet architecture

the problem

[Messages](#)[Academia](#)[Contact](#)

2013

I'm joining a startup



Text Message



< Messages

Academia

Contact

2013

I'm joining a startup

cool, is it cloud



Text Message



< Messages

Academia

Contact

2013

I'm joining a startup

cool, is it cloud

no, it's a CDN



Text Message



< Messages

Academia

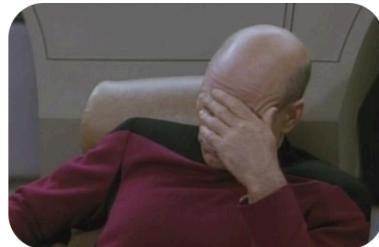
Contact

2013

I'm joining a startup

cool, is it cloud

no, it's a CDN



Text Message



< Messages

Academia

Contact

2013

I'm joining a startup

cool, is it cloud

no, it's a CDN



CDN is a solved problem,
vendor X already exists



Text Message



2009 ~ 2012



market



- incumbents addressed shrinking use case: large static assets
- a lot of suppressed demand

cost of entry



- maturity of open source reverse proxies (nginx, varnish)
- network topology flattened and bandwidth costs dropped

technology

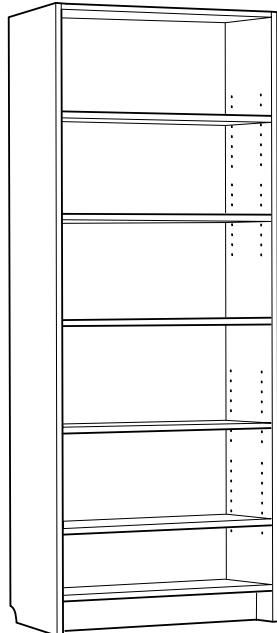


- SSD, multicore, 10Gbps NICs
- network programmability

start over

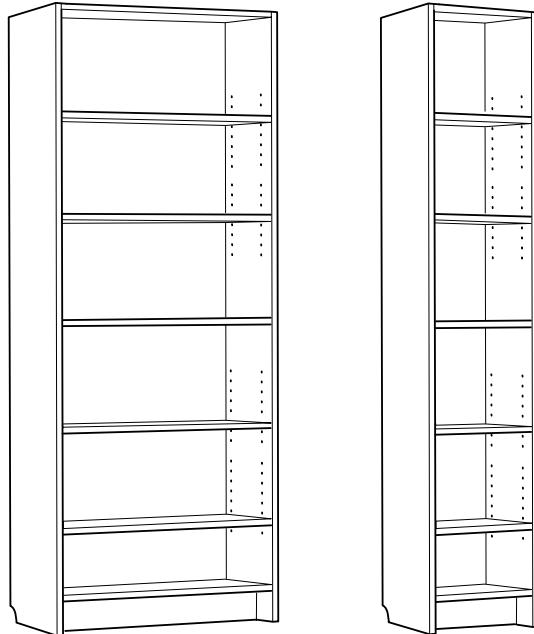
PÖP

(point of presence)



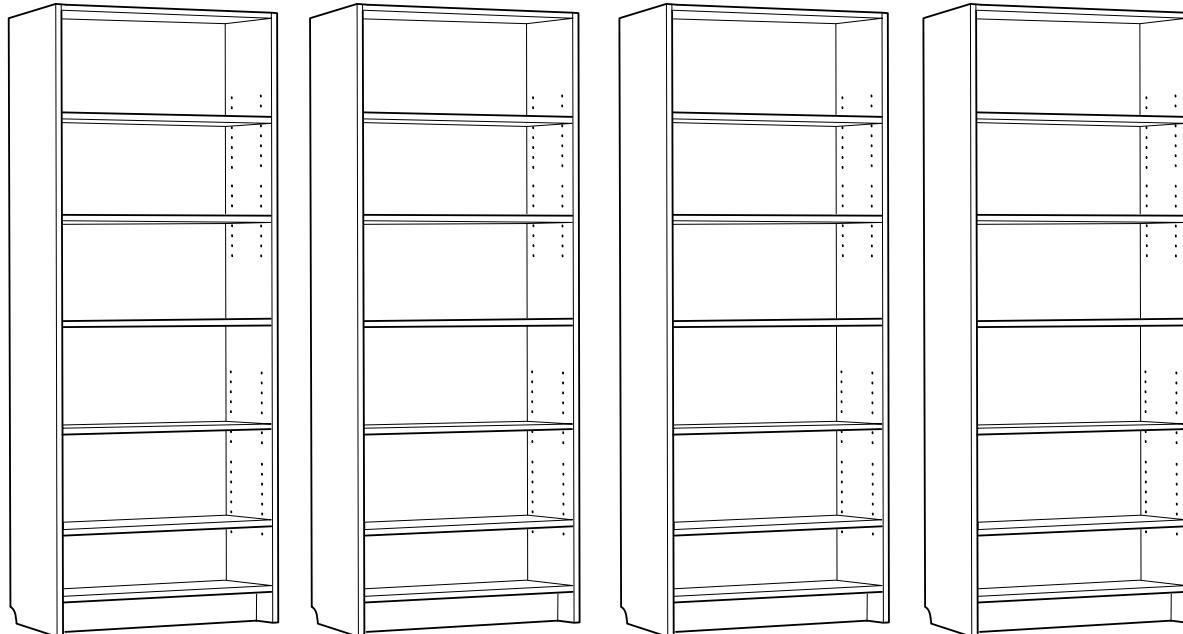
PÖP

(point of presence)



PÖP

(point of presence)



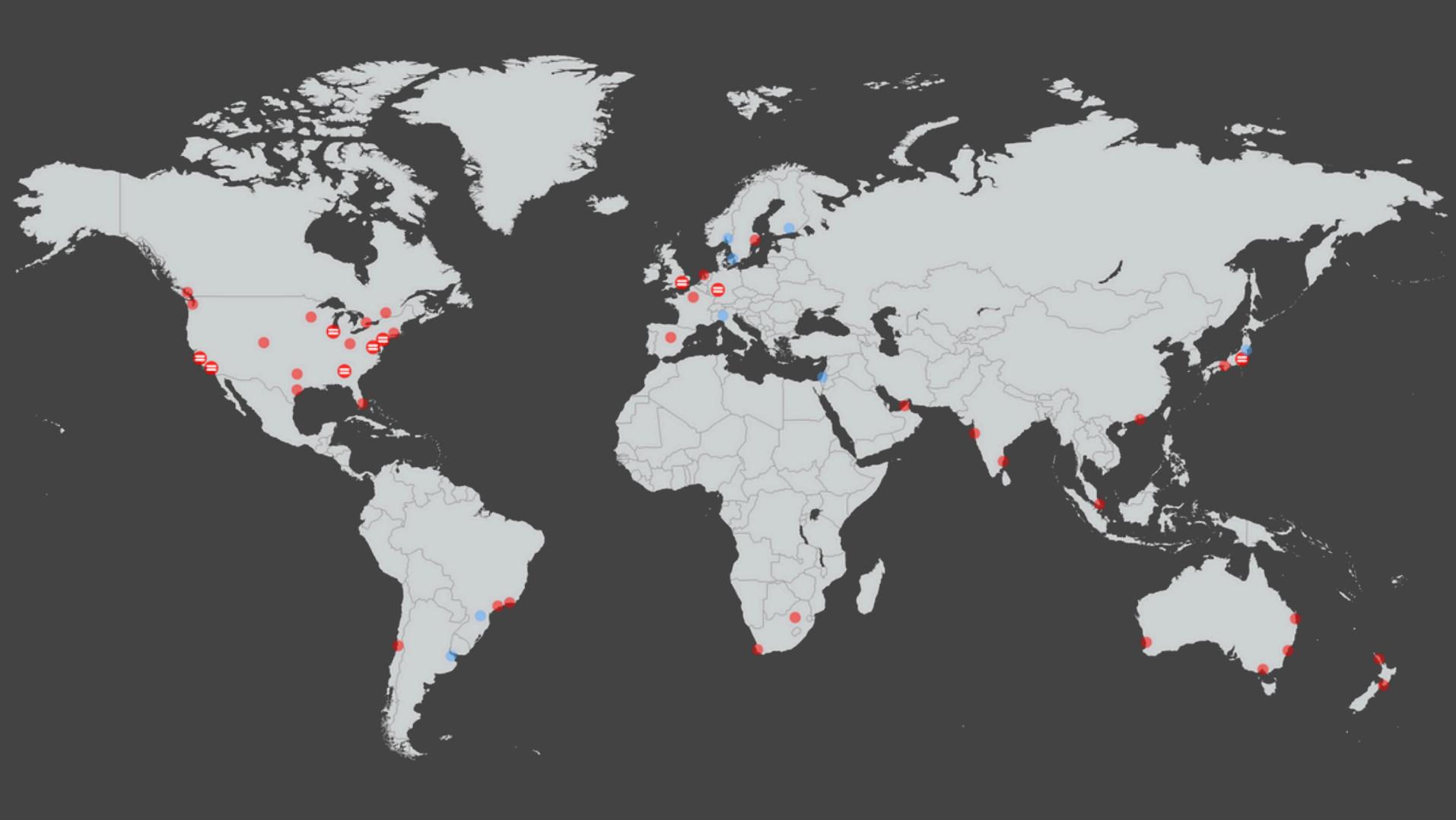
clöud

clöud

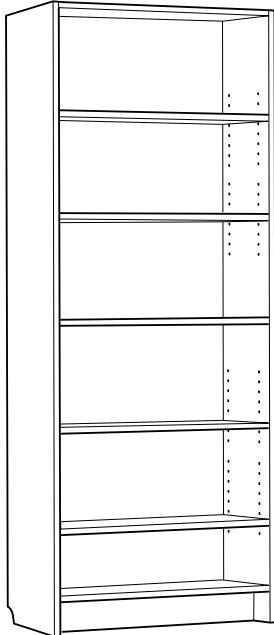
A horizontal red brushstroke underline is positioned below the word "clöud". The stroke is thick and has a slightly irregular, hand-painted appearance.



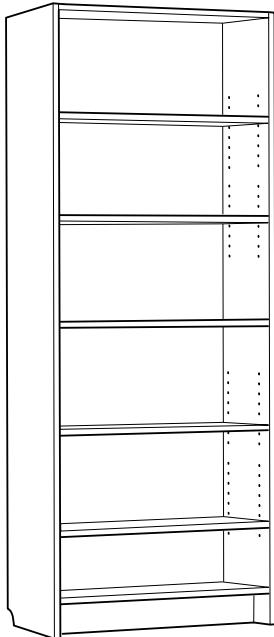




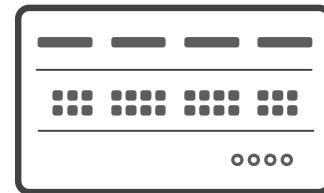
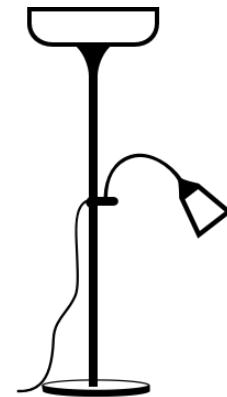




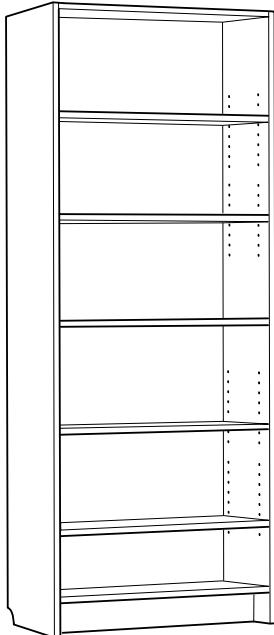
RÅCK



RÅCK

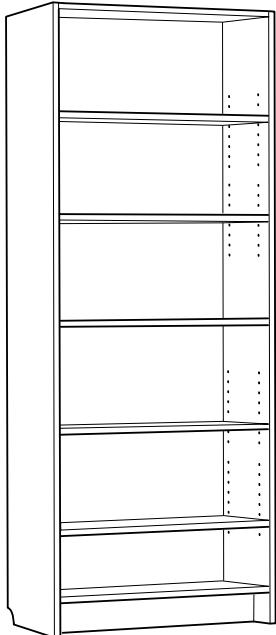


Requirements



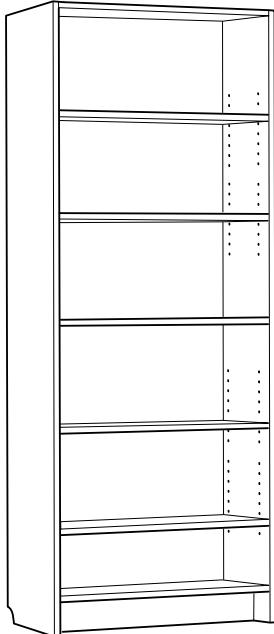
- ▶ maximize RPS
- ▶ low latency
- ▶ absorb DDOS attacks
- ▶ no single points of failure
- ▶ no service disruption on maintenance

Requirements



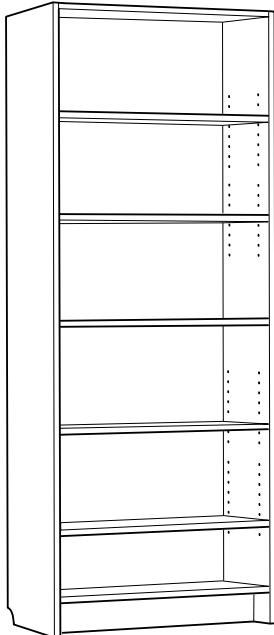
- ▶ maximize RPS and cache hit ratio
- ▶ low latency
- ▶ absorb DDOS attacks
- ▶ no single points of failure
- ▶ no service disruption on maintenance

Requirements



- ▶ maximize RPS and cache hit ratio
- ▶ low latency
- ▶ absorb DDOS attacks
- ▶ no single points of failure
- ▶ no service disruption on maintenance

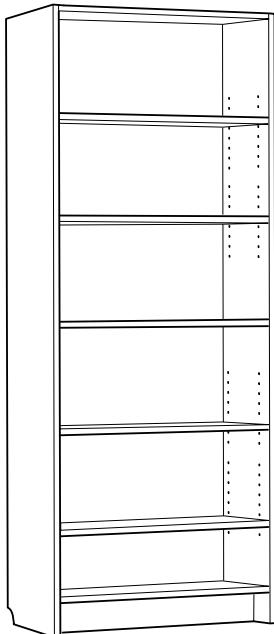
Problem statement



Given fixed physical footprint, how do you design a load balancing architecture which is **efficient, resilient** and **graceful**?

the topology

Guidelines



maximize number of hosts



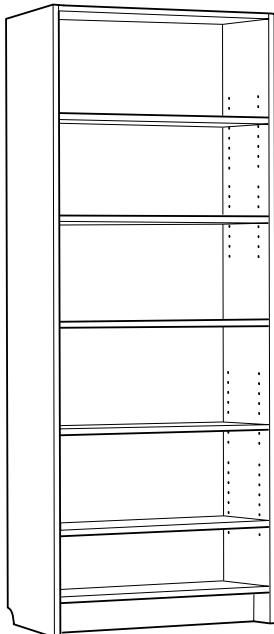
need switches for connecting to upstreams



avoid dedicated network hardware:

- ▶ unneeded features (routers)
- ▶ load balancer appliances
- ▶ multiple switch tiers

Guidelines



maximize number of hosts



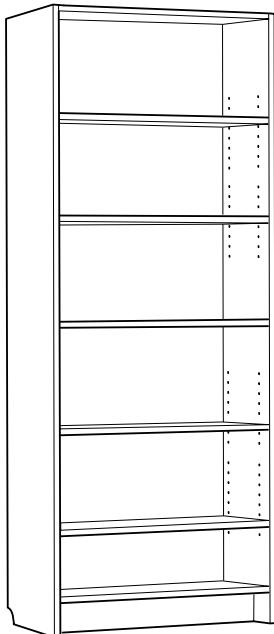
need switches for connecting to upstreams



avoid dedicated network hardware:

- ▶ unneeded features (routers)
- ▶ load balancer appliances
- ▶ multiple switch tiers

Guidelines



maximize number of hosts



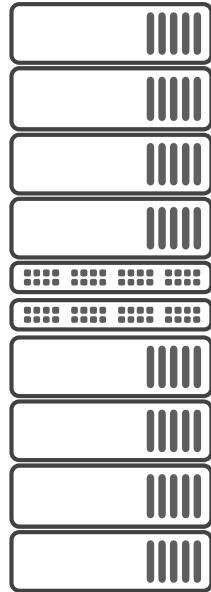
need switches for connecting to upstreams



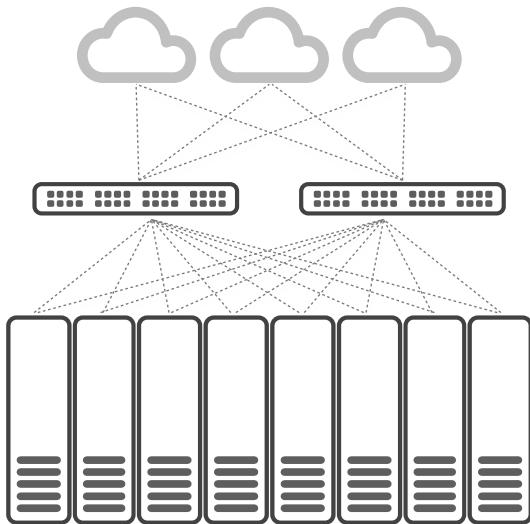
avoid dedicated network hardware:

- ▶ unneeded features (routers)
- ▶ load balancer appliances
- ▶ multiple switch tiers

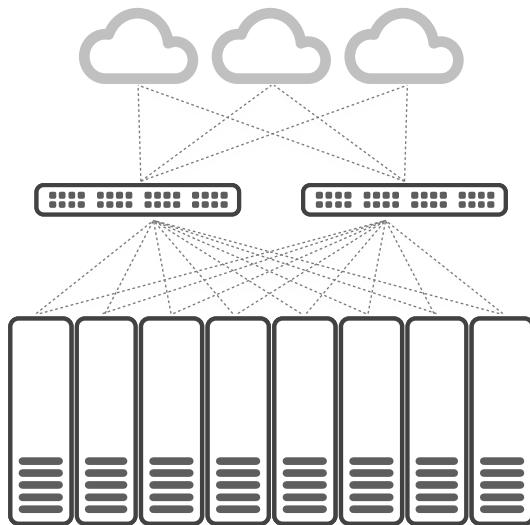
POP topology



POP topology



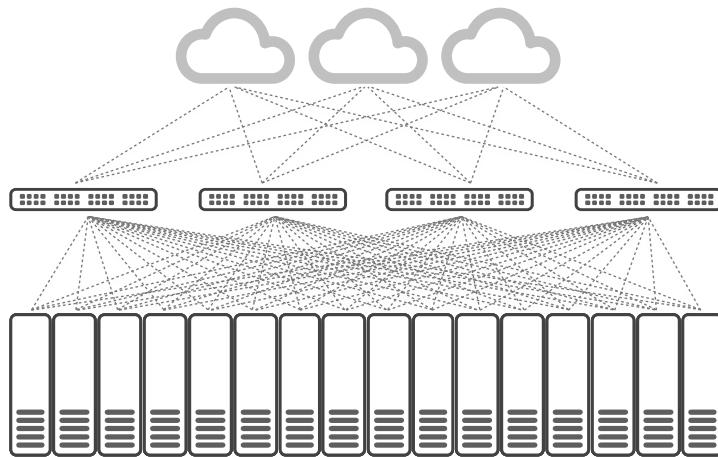
POP topology



Hosts	Racks	Bandwidth*	RPS	Storage
		(Gbps)	(millions)	(TB)
8	0.5	200	0.32	768

* notional host bandwidth on fabric accounting for loss of one switch

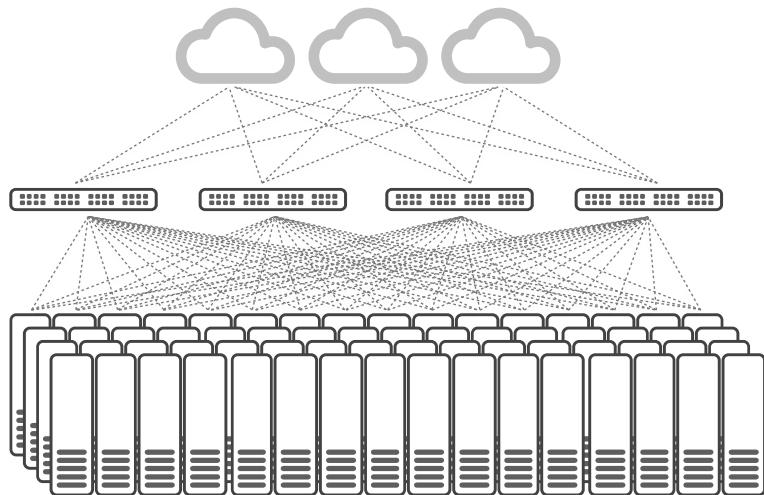
POP topology



Hosts	Racks	Bandwidth*	RPS	Storage
		(Gbps)	(millions)	(TB)
8	0.5	200	0.32	768
16	1	1200	0.64	1536

* notional host bandwidth on fabric accounting for loss of one switch

POP topology



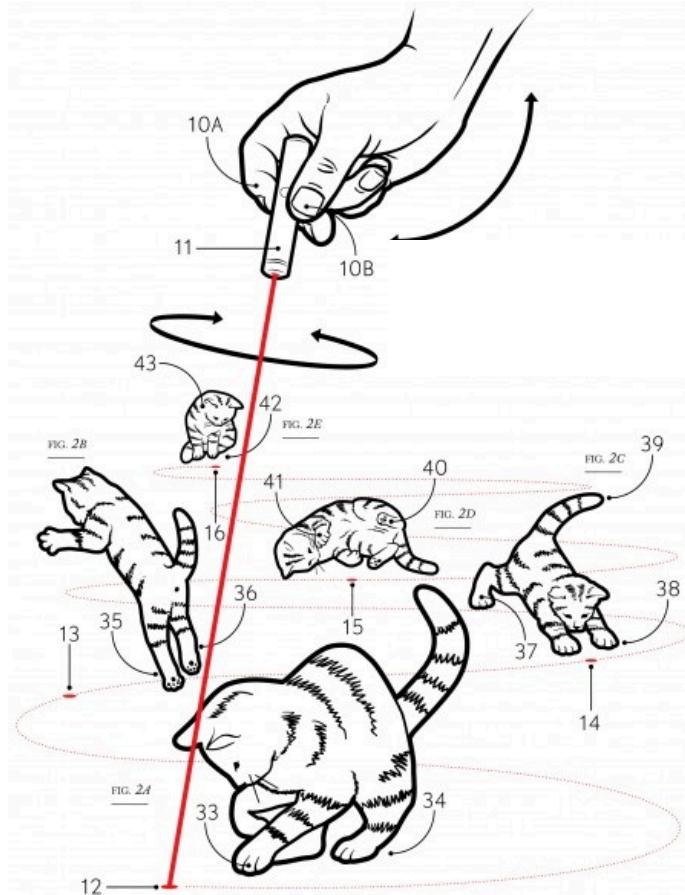
Hosts	Racks	Bandwidth*	RPS	Storage
		(Gbps)	(millions)	(TB)
8	0.5	400	0.32	768
16	1	1200	0.64	1536
32	2	2400	1.28	3072
64	4	4800	2.56	6144

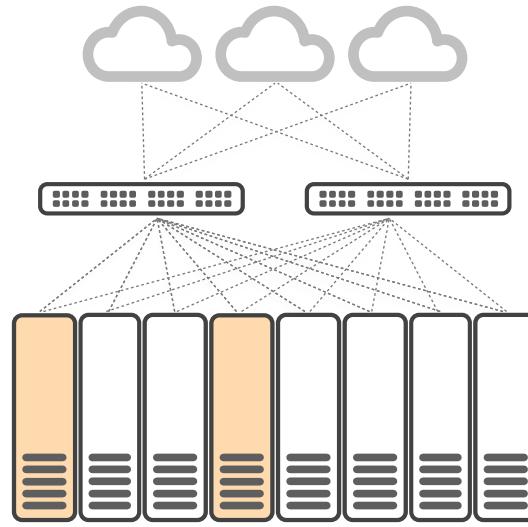
* notional host bandwidth on fabric accounting for loss of one switch

load balancing

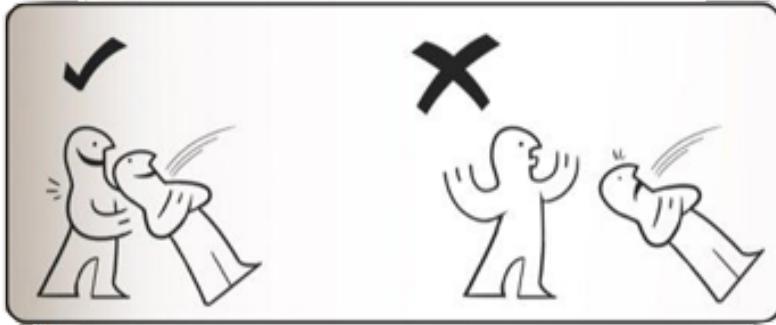
anything that maintains state is easy to DDOS

- load balancer appliances
- Ananta [SIGCOMM'13]
- Duet [SIGCOMM'14]
- MagLev [NSDI'16]
- SilkRoad [SIGCOMM'17]





**software-only load balancers can't
make use of full bisection bandwidth**

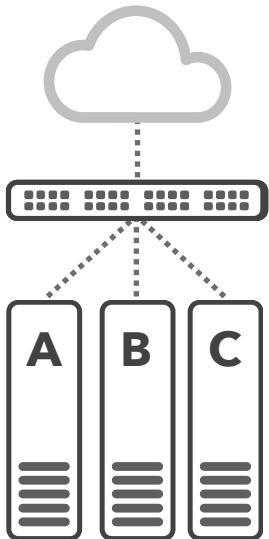


stateless solutions are not graceful

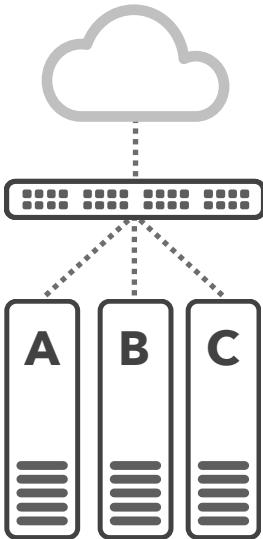
- ▶ many switch ECMP implementations result in rehashing
- ▶ even ones that don't will break ongoing connections

faild

Faild



Faild



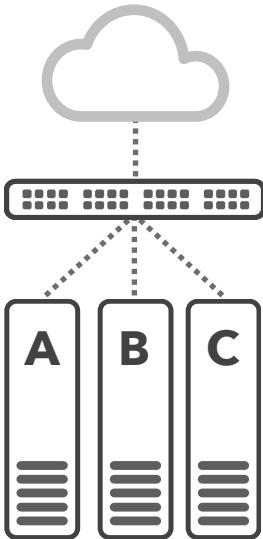
A diagram of a switch's Forwarding Information Base (FIB). It features a blue header bar with a dotted pattern at the top. Below it is a dashed-line box containing a table. The table has two columns: 'Destination prefix' and 'Next hop IP'. There are six rows in the table, each corresponding to a destination prefix of 192.168.0.0/24 and a different next hop IP address (10.0.1.A through 10.0.2.C). The table is labeled 'FIB' at the bottom center.

Destination prefix	Next hop IP
192.168.0.0/24	10.0.1.A
192.168.0.0/24	10.0.2.A
192.168.0.0/24	10.0.1.B
192.168.0.0/24	10.0.2.B
192.168.0.0/24	10.0.1.C
192.168.0.0/24	10.0.2.C

on switch, map VIP to static set of virtual next hops

- ▶ using static set avoids rehashing
- ▶ ECMP width determines granularity with which we load balance

Faild



A table titled "FIB" (Forwarding Information Base) showing a list of destination prefixes and their corresponding next hop IP addresses. The table has two columns: "Destination prefix" and "Next hop IP". There are six entries, all with a destination prefix of 192.168.0.0/24 and different next hop IP addresses. An orange arrow points from the handwritten note "made up IPs" to the table.

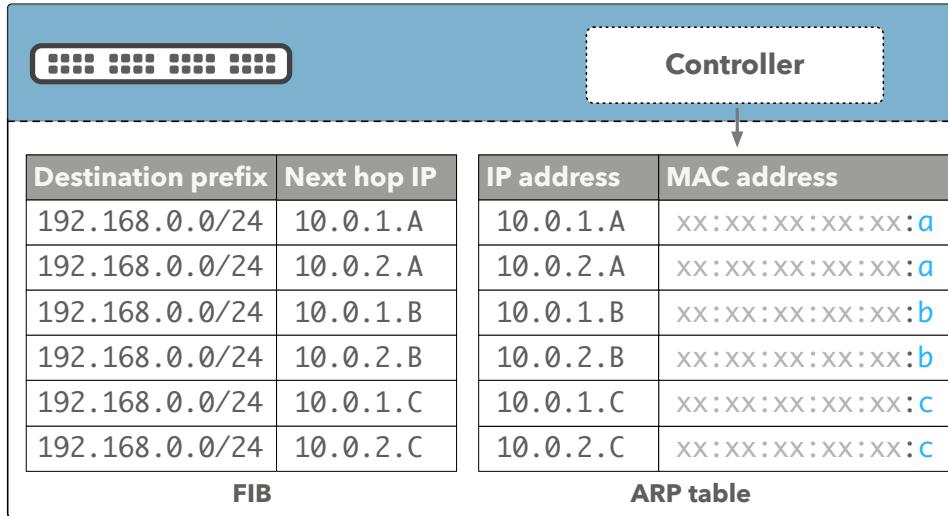
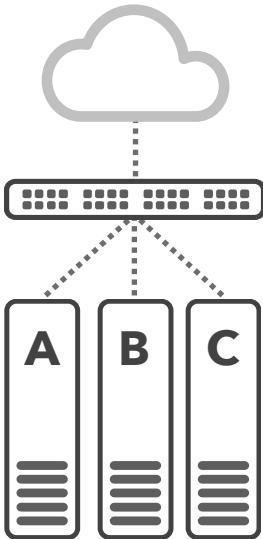
Destination prefix	Next hop IP
192.168.0.0/24	10.0.1.A
192.168.0.0/24	10.0.2.A
192.168.0.0/24	10.0.1.B
192.168.0.0/24	10.0.2.B
192.168.0.0/24	10.0.1.C
192.168.0.0/24	10.0.2.C

FIB

on switch, map VIP to static set of virtual next hops

- ▶ using static set avoids rehashing
- ▶ ECMP width determines granularity with which we load balance

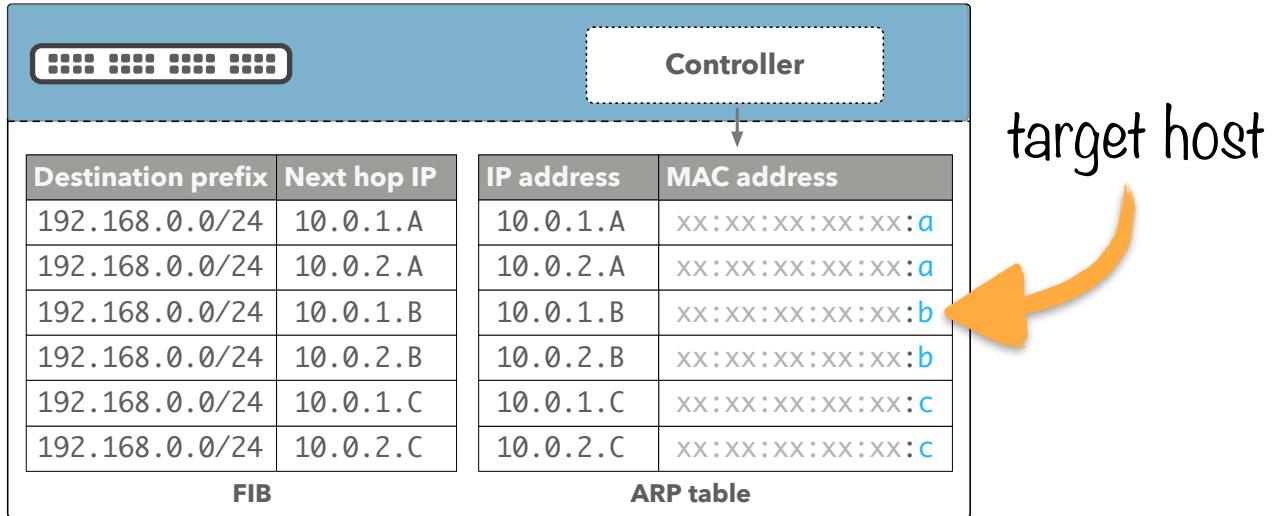
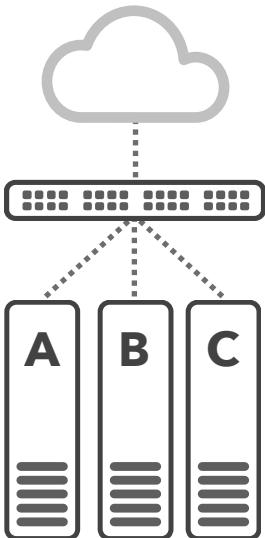
Faild



on switch, control forwarding by manipulating ARP entry

- controller maps virtual next hop to virtual MAC address
- encode information about current host in virtual MAC address
- bridging table configured to map virtual MAC to egress port

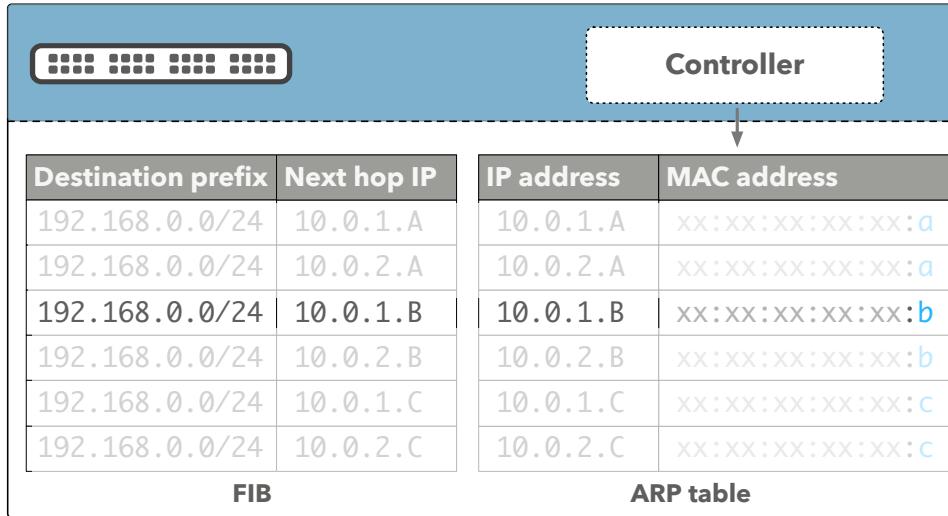
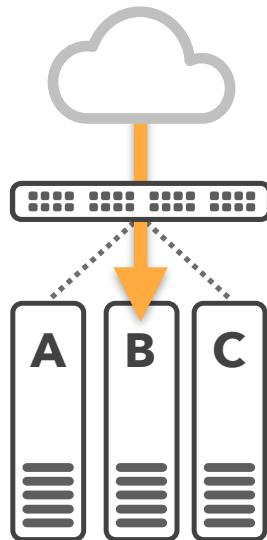
Faild



on switch, control forwarding by manipulating ARP entry

- controller maps virtual next hop to virtual MAC address
- encode information about current host in virtual MAC address
- bridging table configured to map virtual MAC to egress port

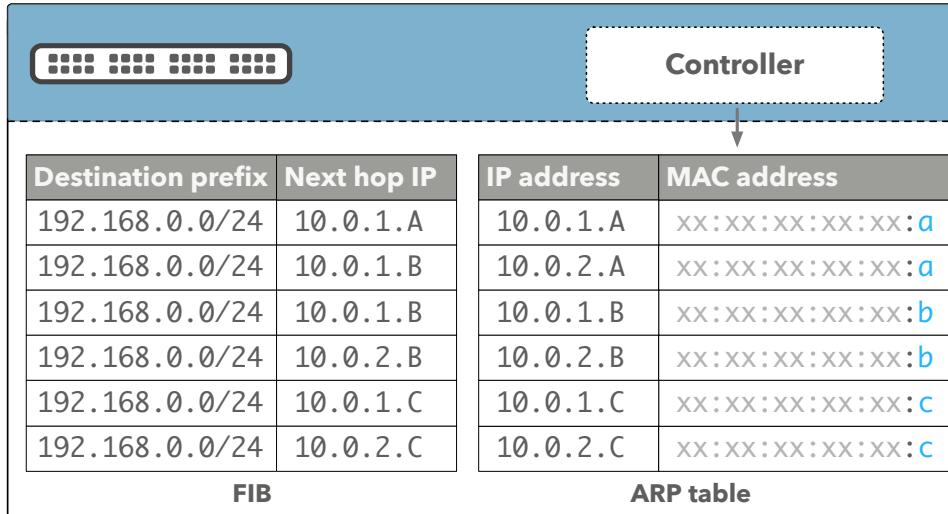
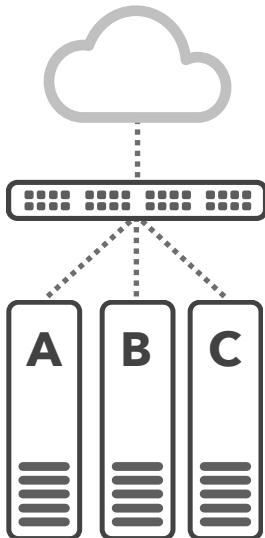
Faild



on switch, control forwarding by manipulating ARP entry

- controller maps virtual next hop to virtual MAC address
- encode information about current host in virtual MAC address
- bridging table configured to map virtual MAC to egress port

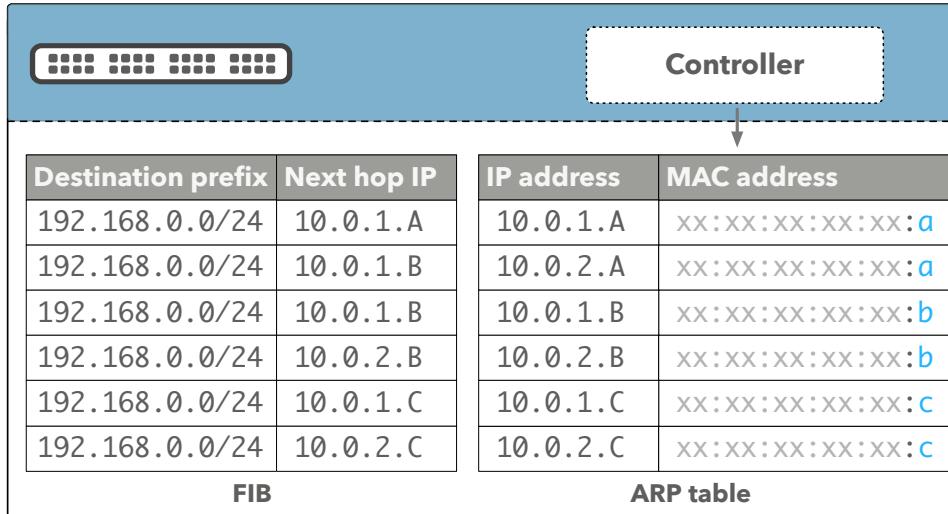
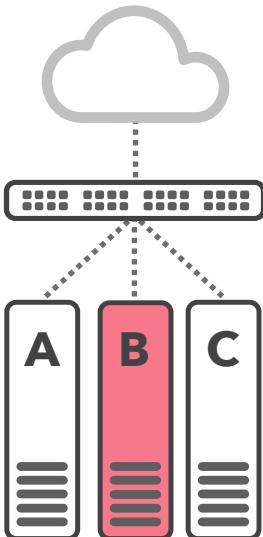
Faild



hosts send health status to controller

- ▶ on drain, update ARP entry
- ▶ balance virtual next hops across available servers

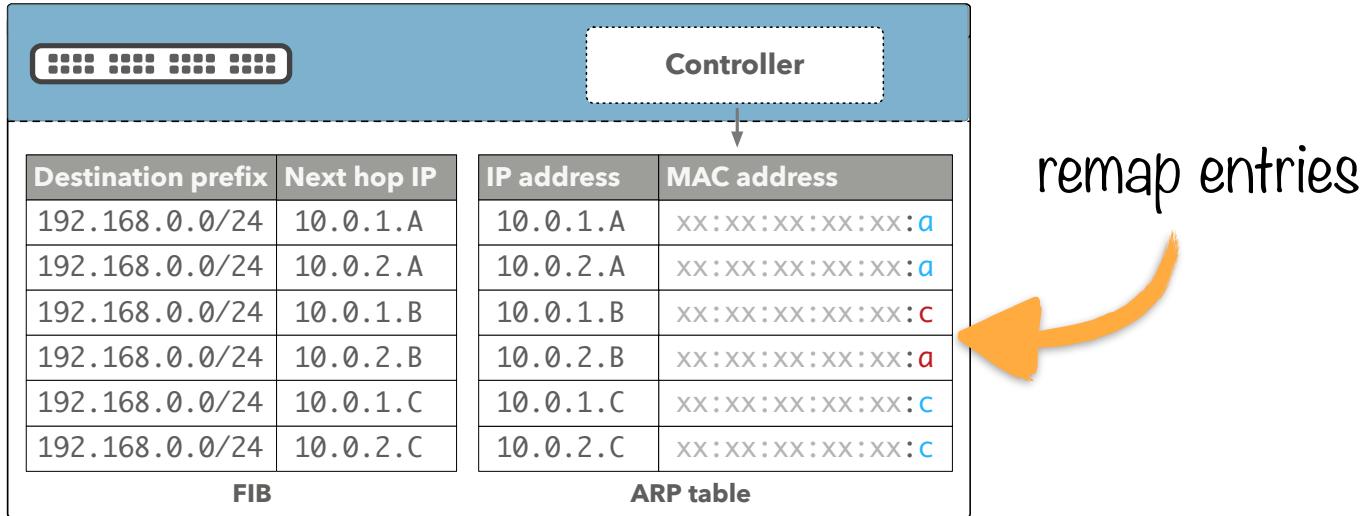
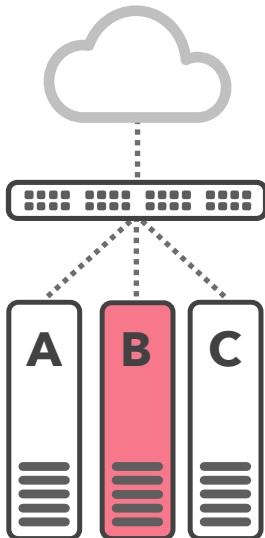
Faild



hosts send health status to controller

- ▶ on drain, update ARP entry
- ▶ balance virtual next hops across available servers

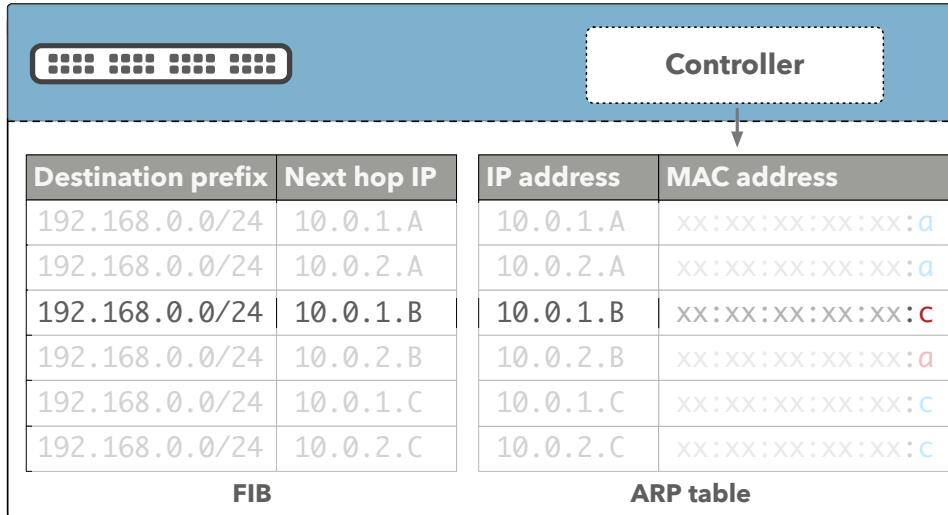
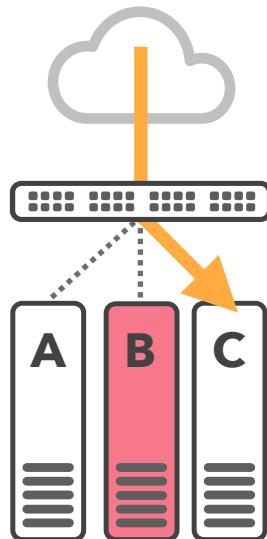
Faild



hosts send health status to controller

- ▶ on drain, update ARP entry
- ▶ balance virtual next hops across available servers

Faild



hosts send health status to controller

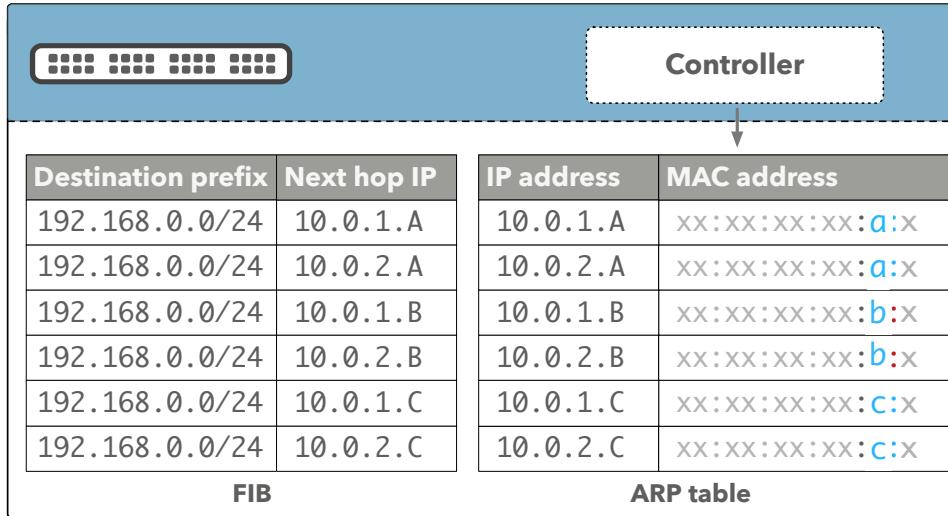
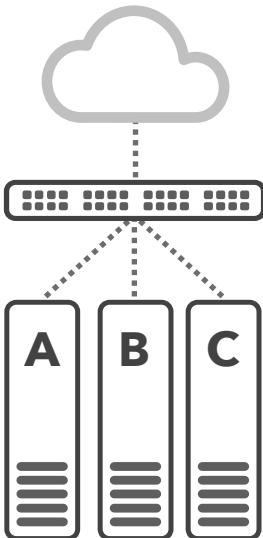
- ▶ on drain, update ARP entry
- ▶ balance virtual next hops across available servers

isn't this just consistent hashing?

isn't this just consistent hashing?

yes, but we can extend mechanism and avoid resets entirely

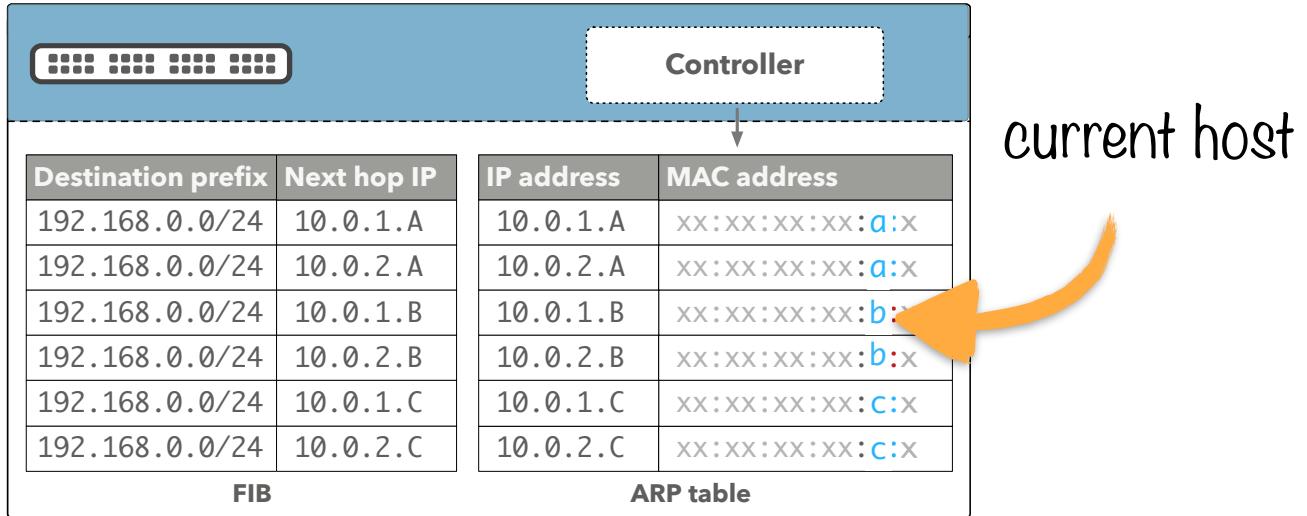
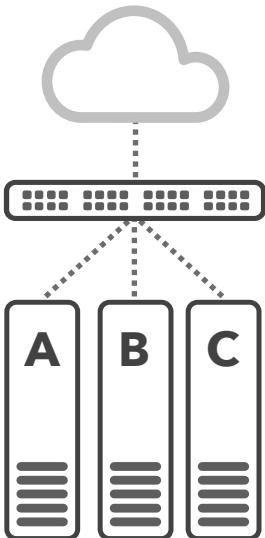
Faild



embed mapping history in MAC address

- append previous target as part of MAC address
- still results in resets, but...
- ...conveys necessary information down to the host

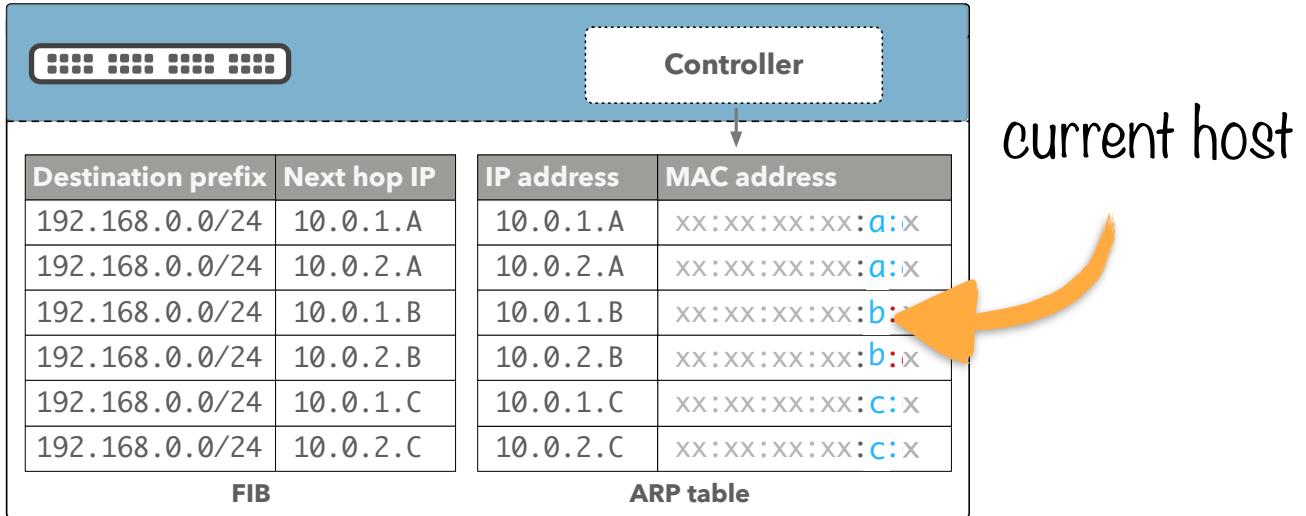
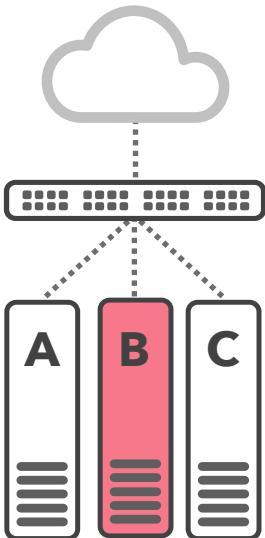
Faild



embed mapping history in MAC address

- append previous target as part of MAC address
- still results in resets, but...
- ...conveys necessary information down to the host

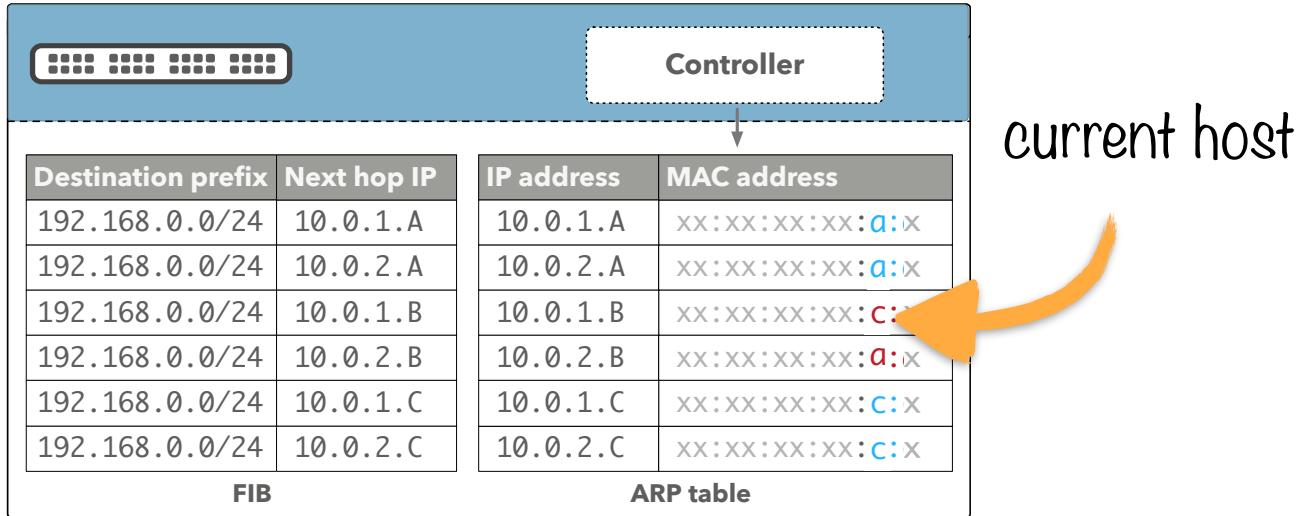
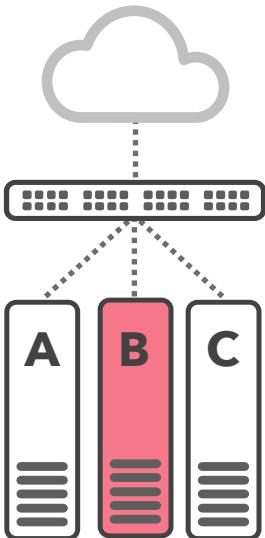
Faild



embed mapping history in MAC address

- append previous target as part of MAC address
- still results in resets, but...
- ...conveys necessary information down to the host

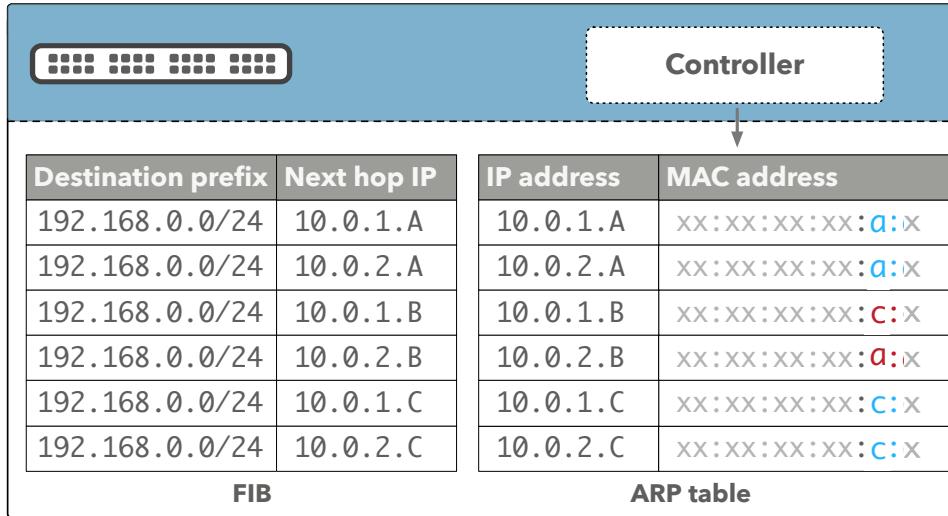
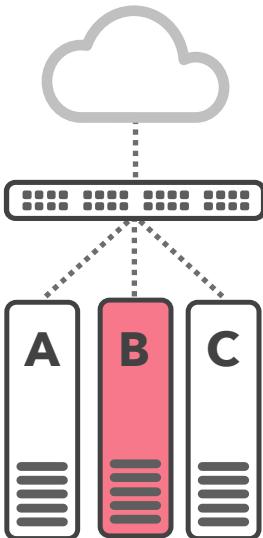
Faild



embed mapping history in MAC address

- append previous target as part of MAC address
- still results in resets, but...
- ...conveys necessary information down to the host

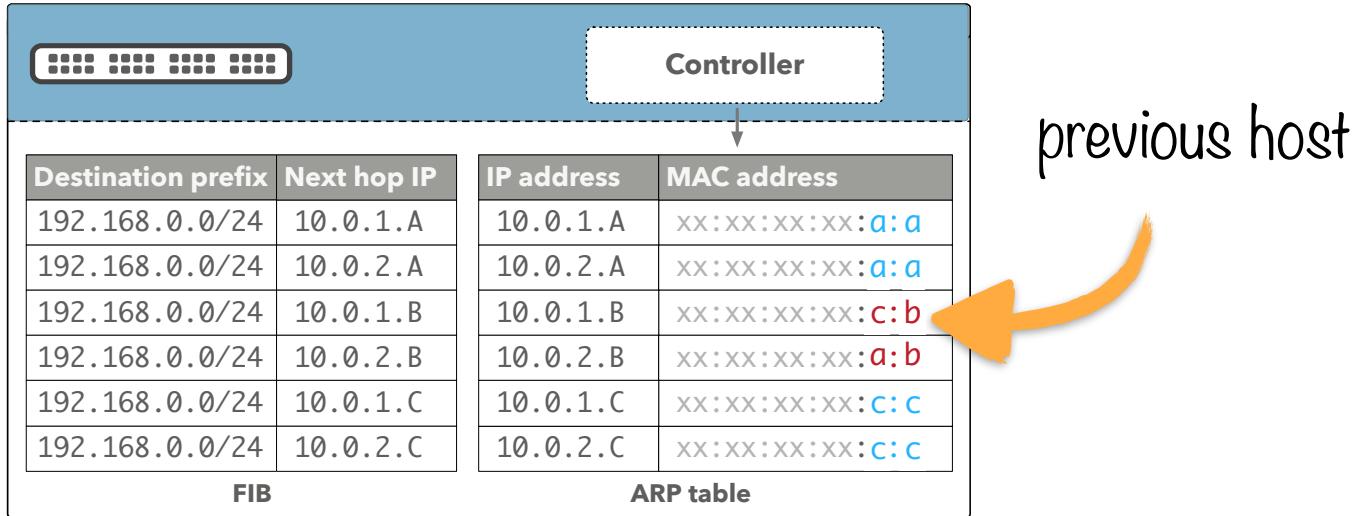
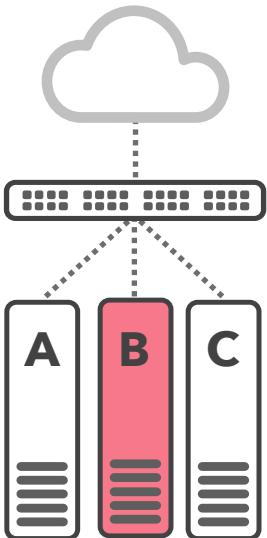
Faild



embed mapping history in MAC address

- append previous target as part of MAC address
- still results in resets, but...
- ...conveys necessary information down to the host

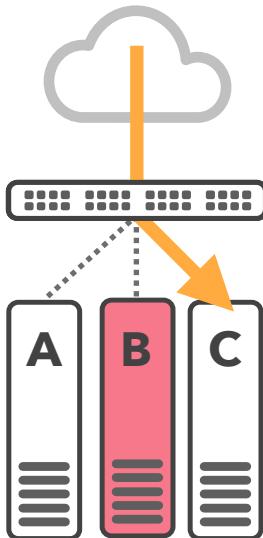
Faild



embed mapping history in MAC address

- append previous target as part of MAC address
- still results in resets, but...
- ...conveys necessary information down to the host

Faild

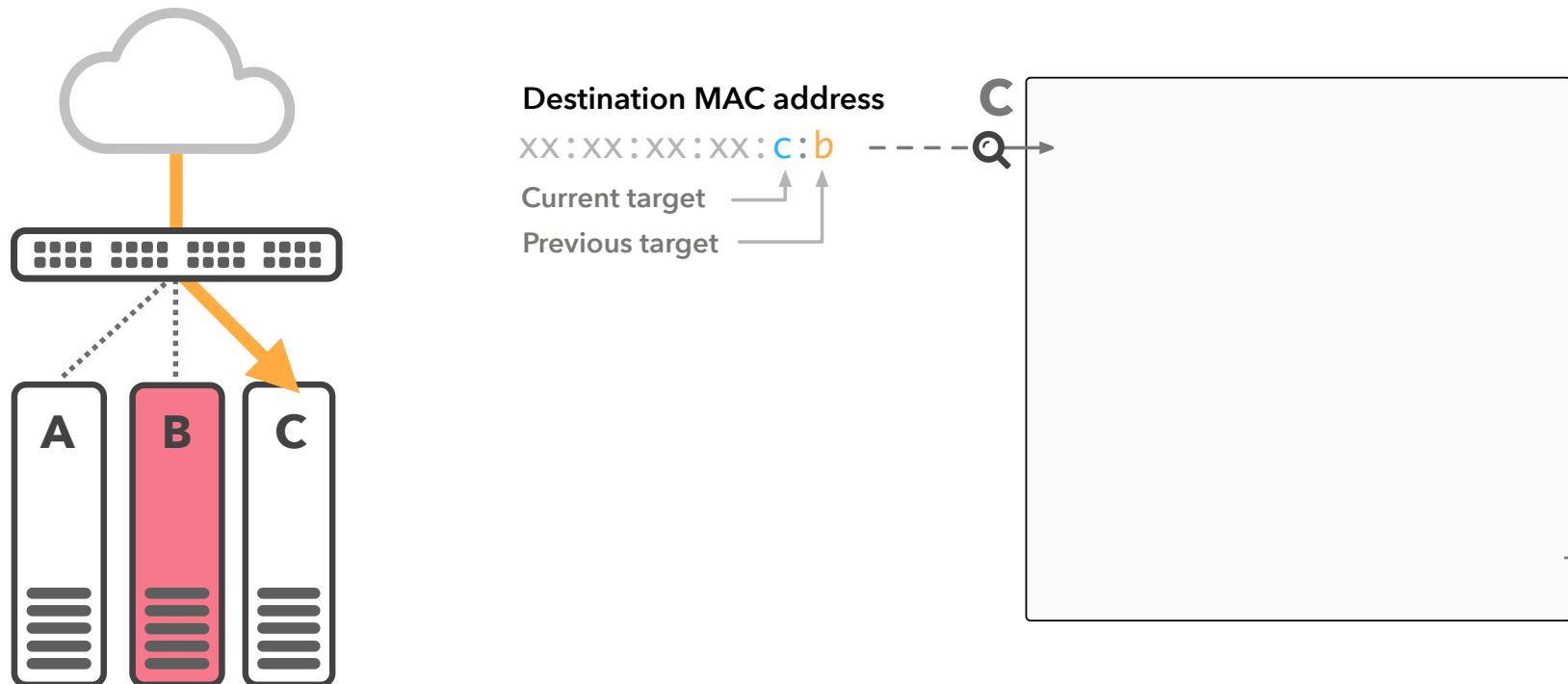


FIB		ARP table	
Destination prefix	Next hop IP	IP address	MAC address
192.168.0.0/24	10.0.1.A	10.0.1.A	xx:xx:xx:xx:a:a
192.168.0.0/24	10.0.2.A	10.0.2.A	xx:xx:xx:xx:a:a
192.168.0.0/24	10.0.1.B	10.0.1.B	xx:xx:xx:xx:c:b
192.168.0.0/24	10.0.2.B	10.0.2.B	xx:xx:xx:xx:a:b
192.168.0.0/24	10.0.1.C	10.0.1.C	xx:xx:xx:xx:c:c
192.168.0.0/24	10.0.2.C	10.0.2.C	xx:xx:xx:xx:c:c

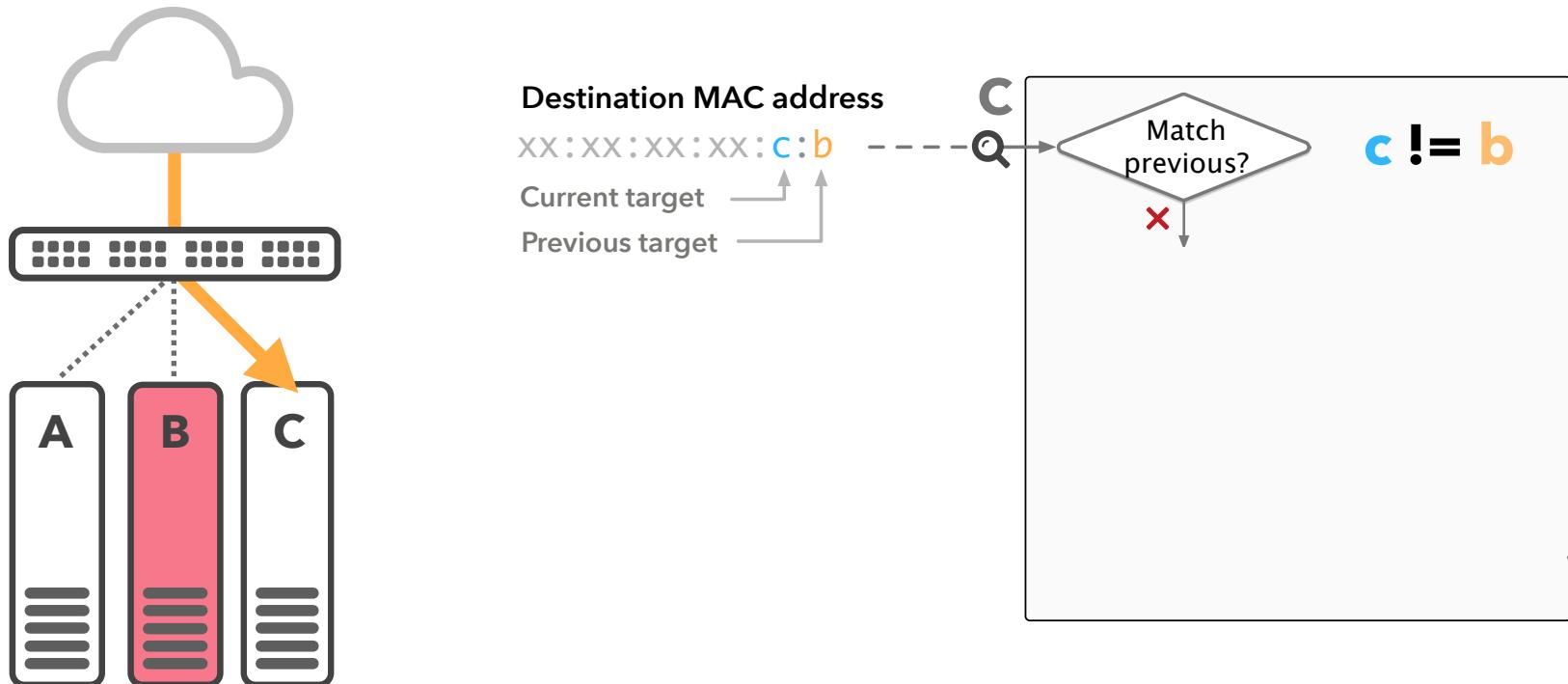
embed mapping history in MAC address

- append previous target as part of MAC address
- still results in resets, but...
- ...conveys necessary information down to the host

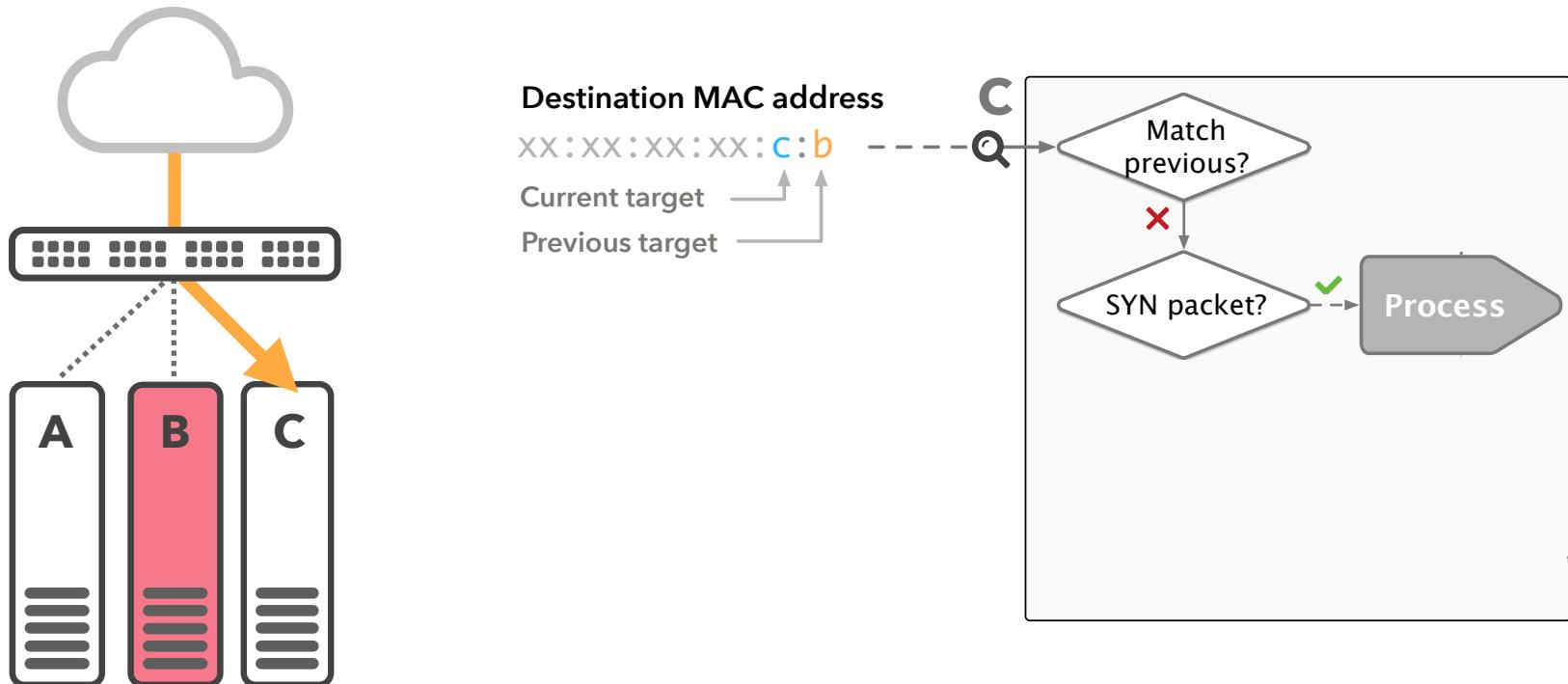
Host processing



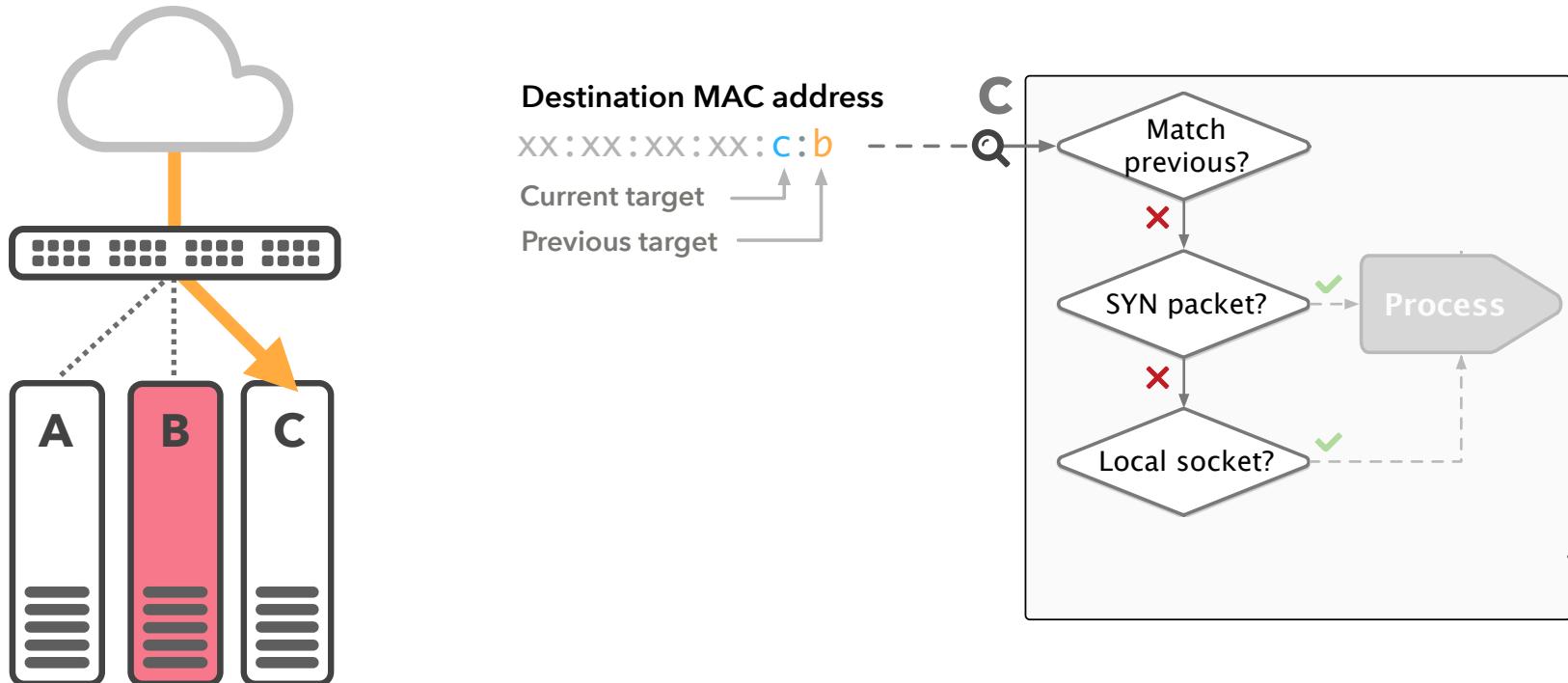
Host processing



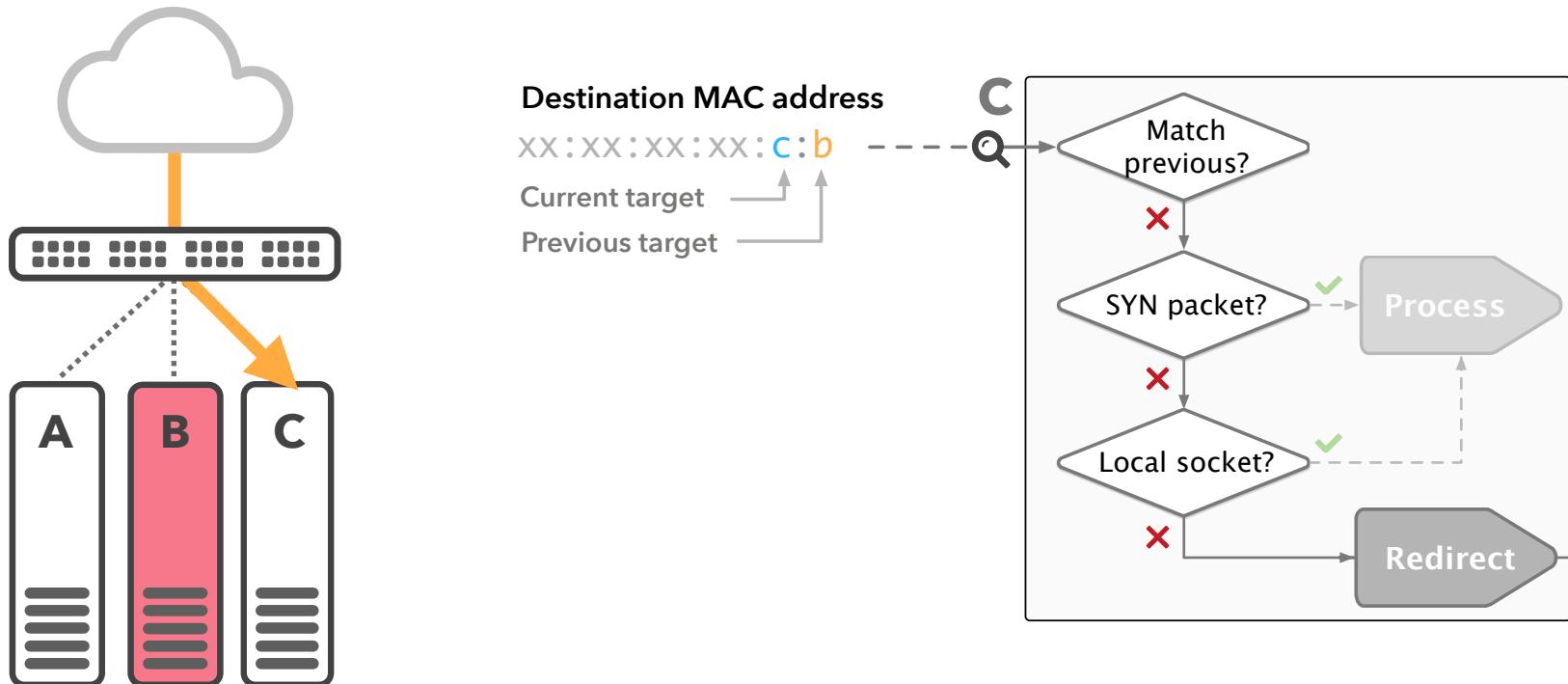
Host processing



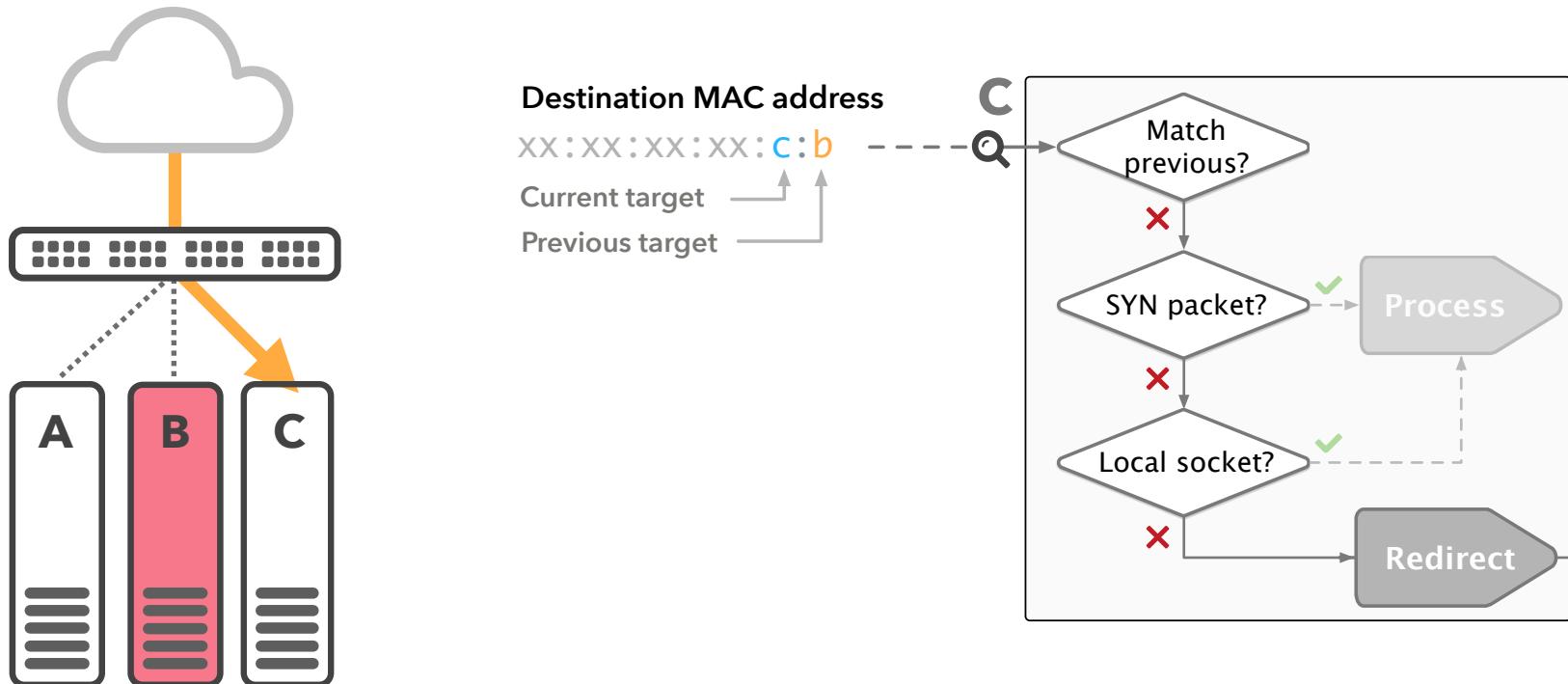
Host processing



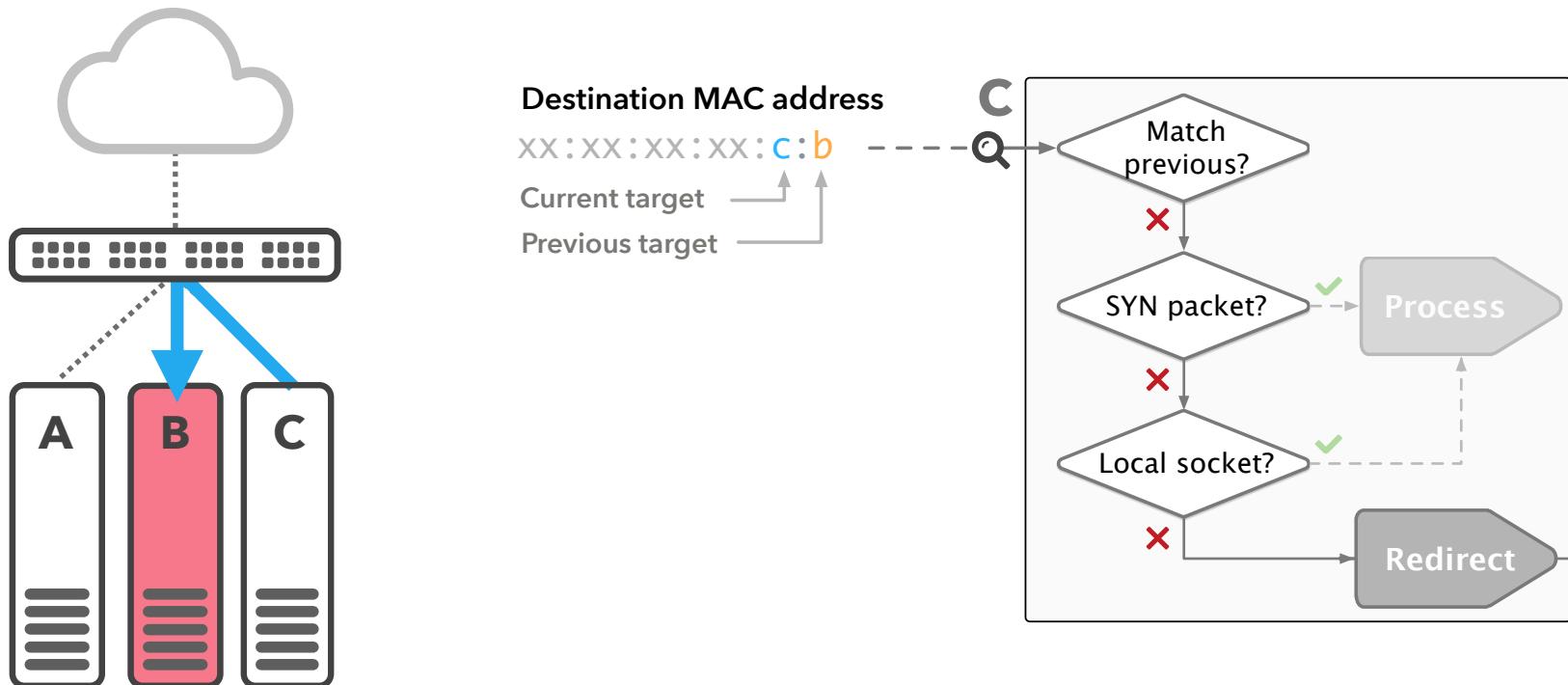
Host processing



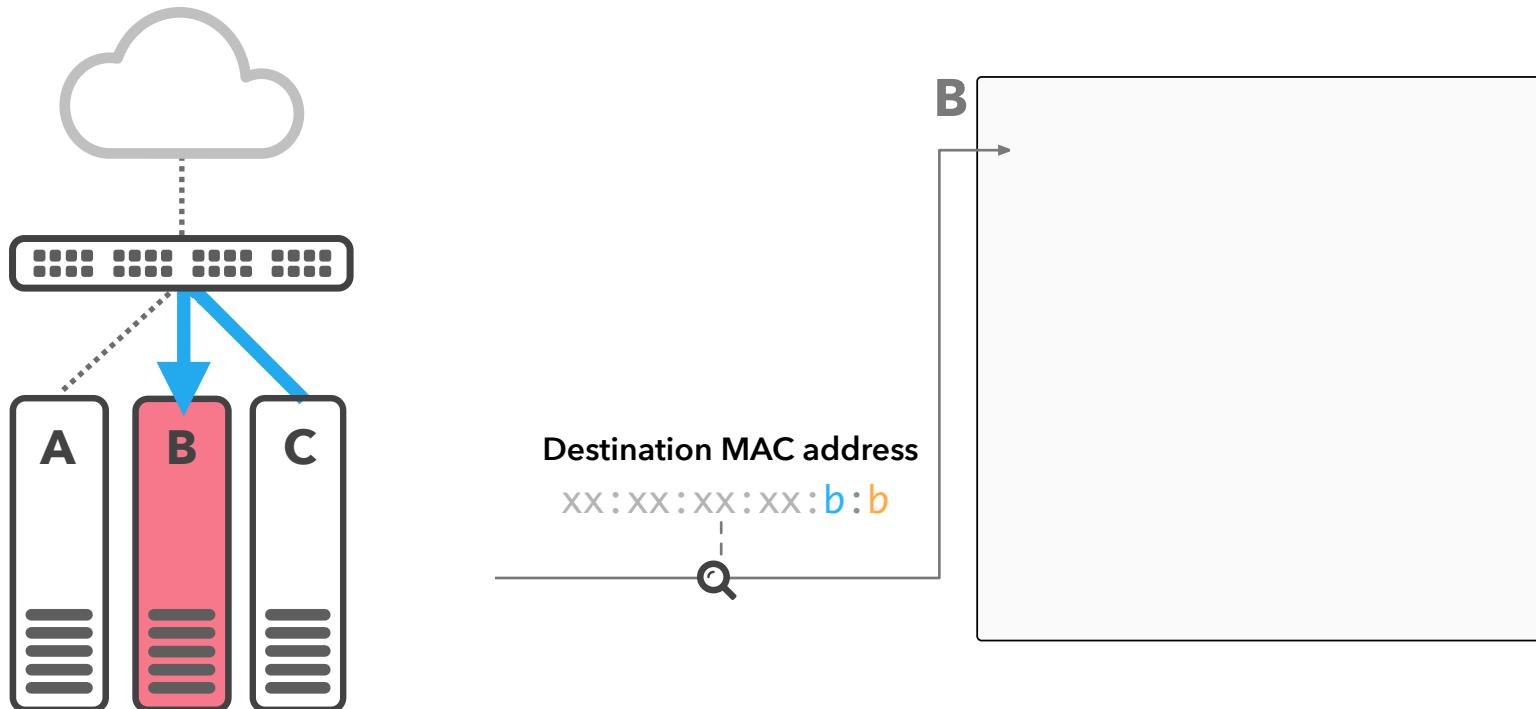
Host processing



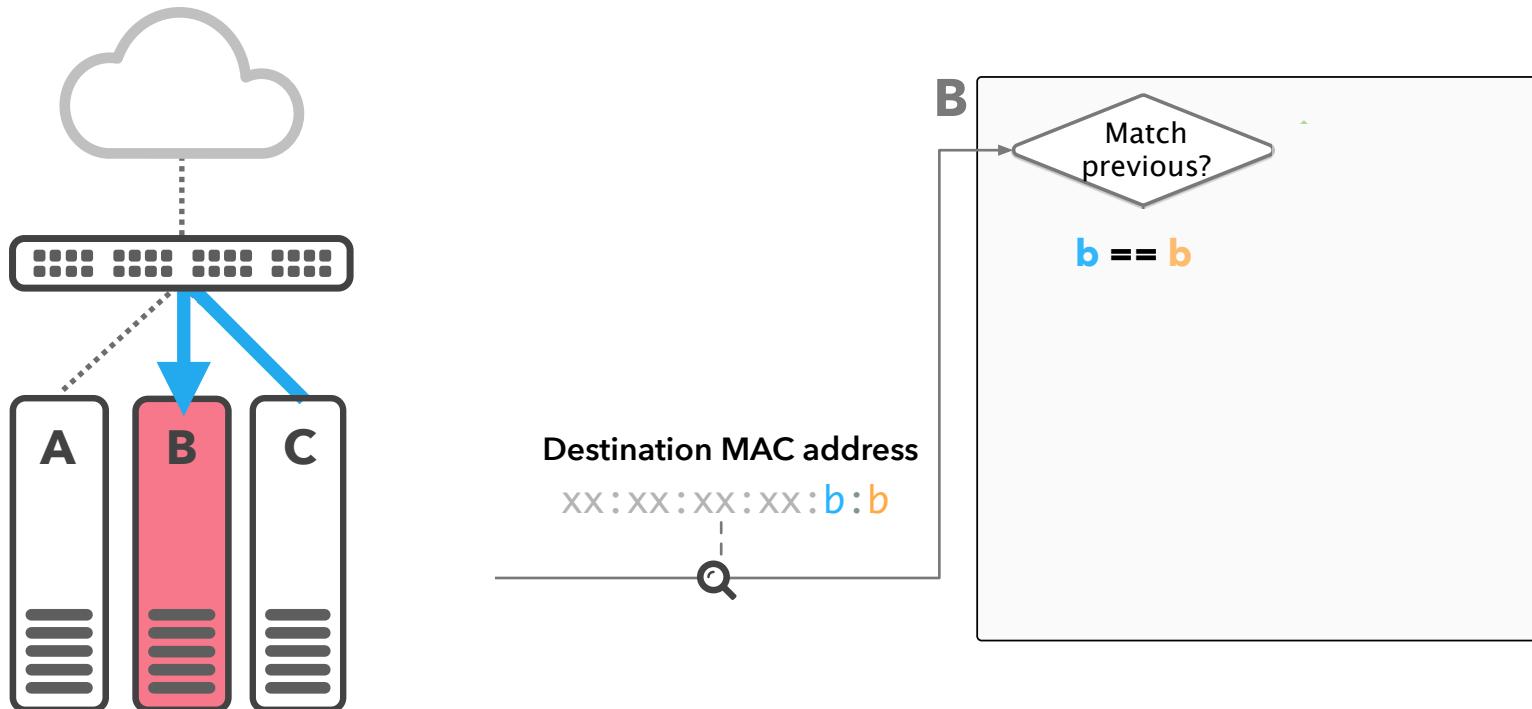
Host processing



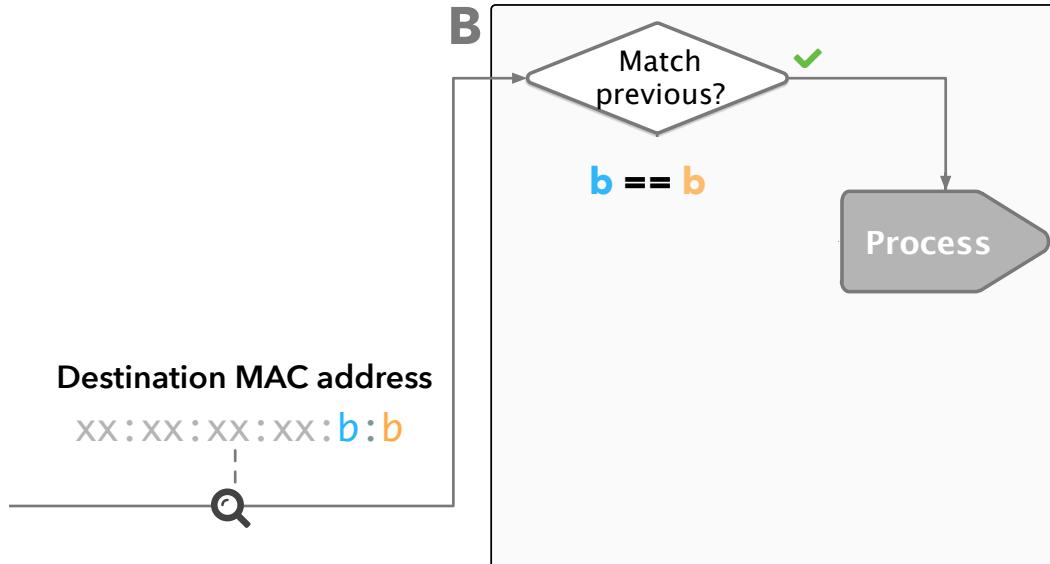
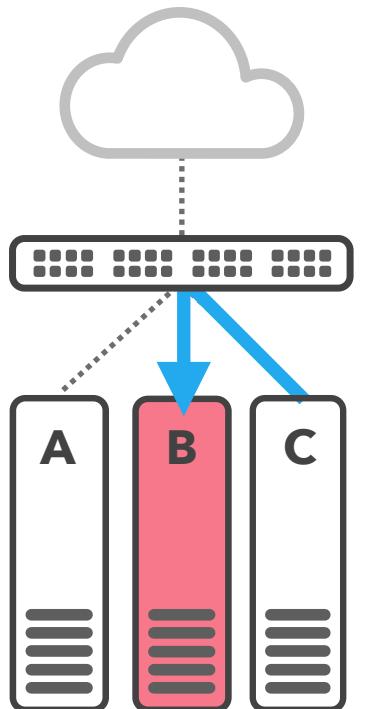
Host processing



Host processing



Host processing



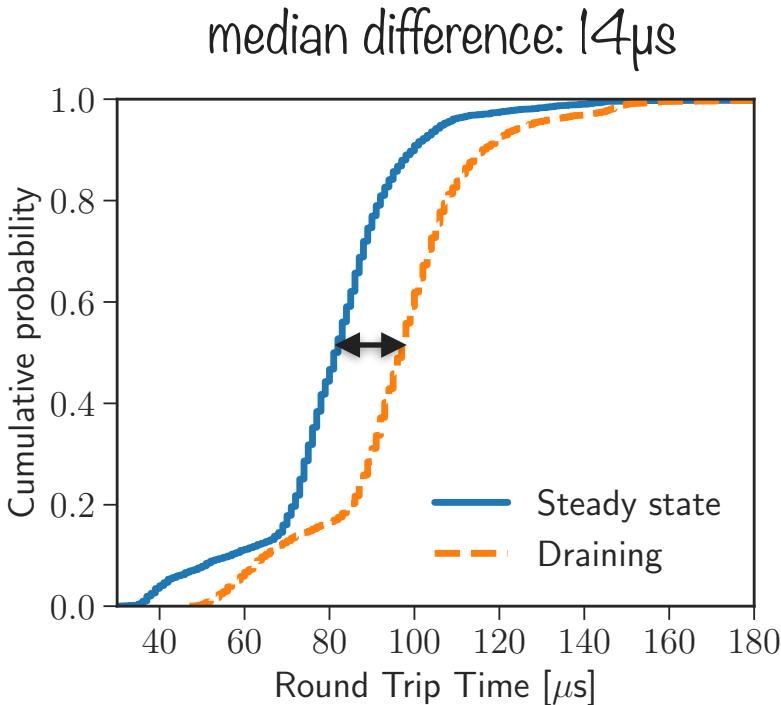
Host processing

Low latency

- expected case: switches do all heavy lifting
- worst case: detour routing costs $14\mu\text{s}$

Negligible impact on CPU utilization

- impact only when refilling
- peak CPU utilization below 0.3%



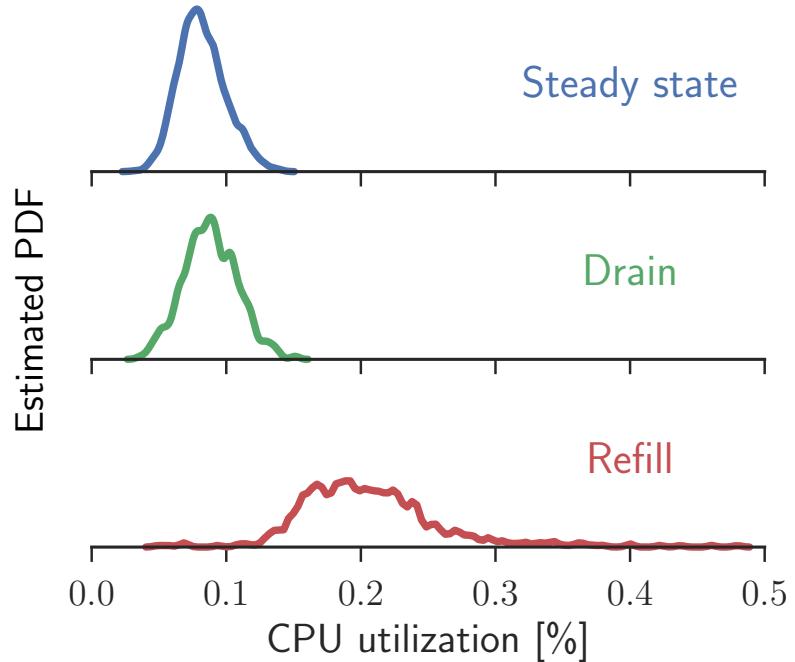
Host processing

Low latency

- expected case: switches do all heavy lifting
- worst case: detour routing costs 20 μ s

Negligible impact on CPU utilization

- impact only when refilling (transient)
- peak CPU utilization below 0.3%



Timeline

2012

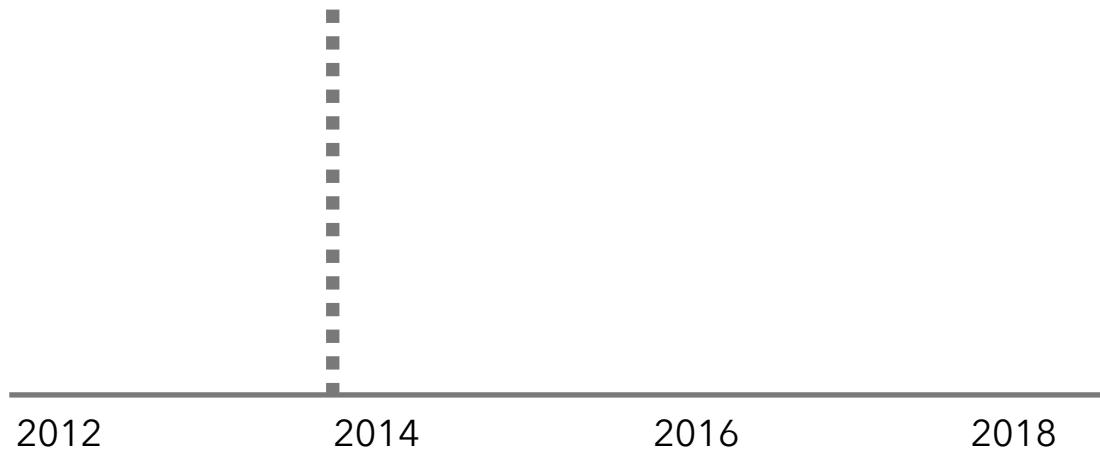
2014

2016

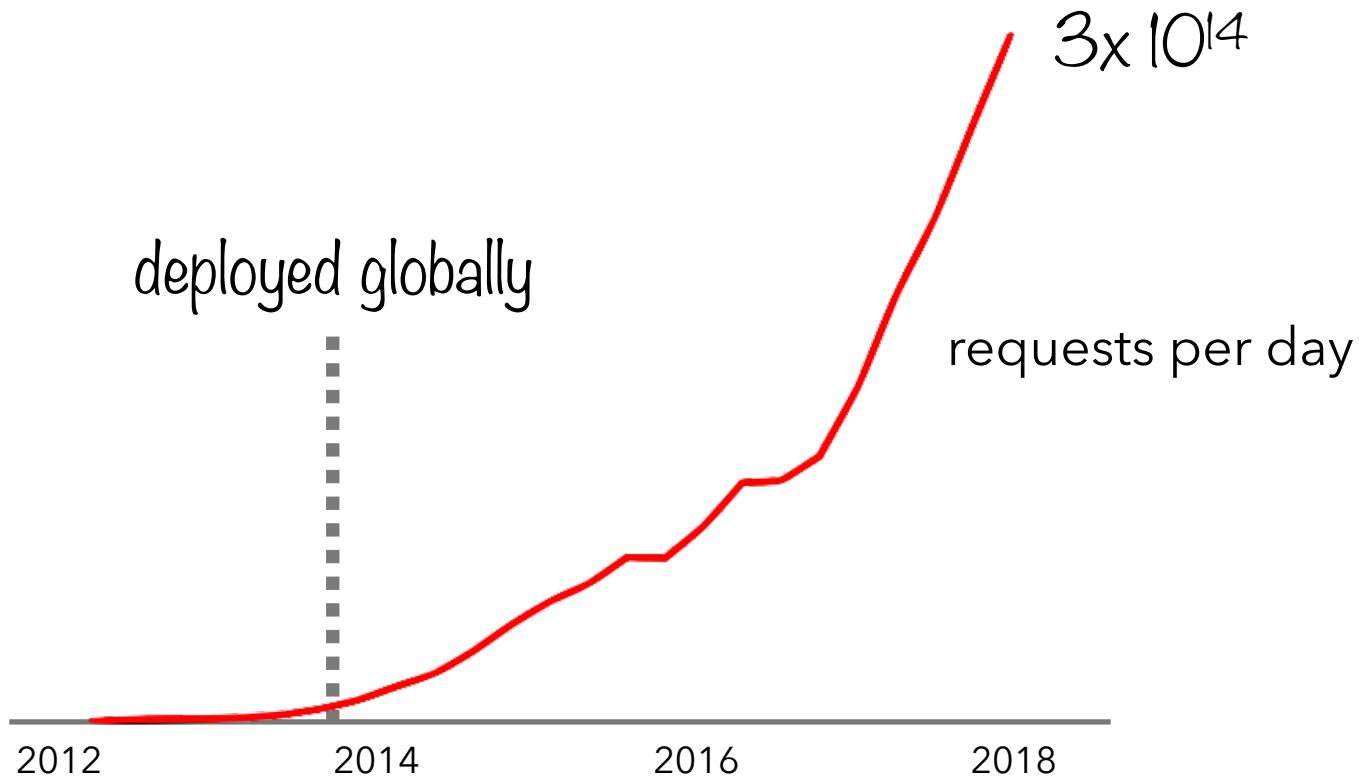
2018

Timeline

deployed globally



Timeline

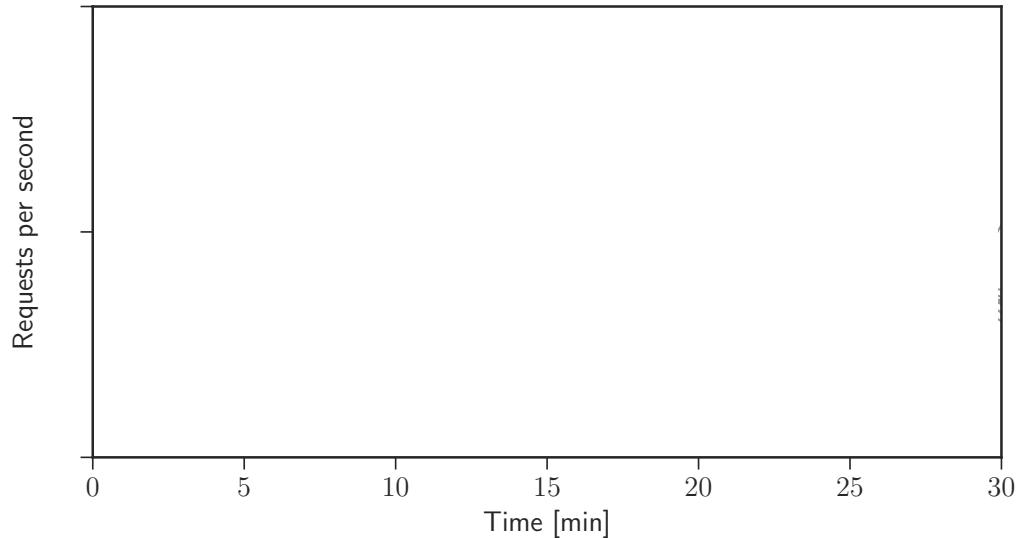
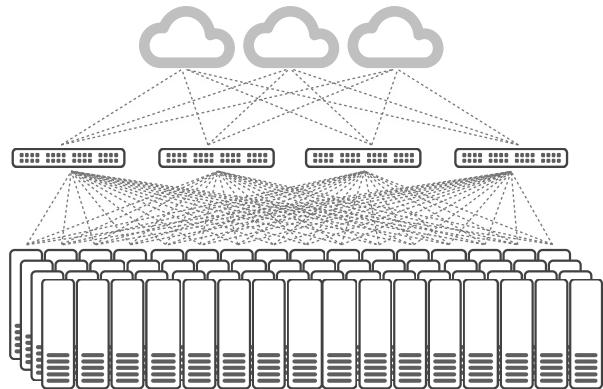


we suspect it works

Assumption #1

hash buckets are equally loaded

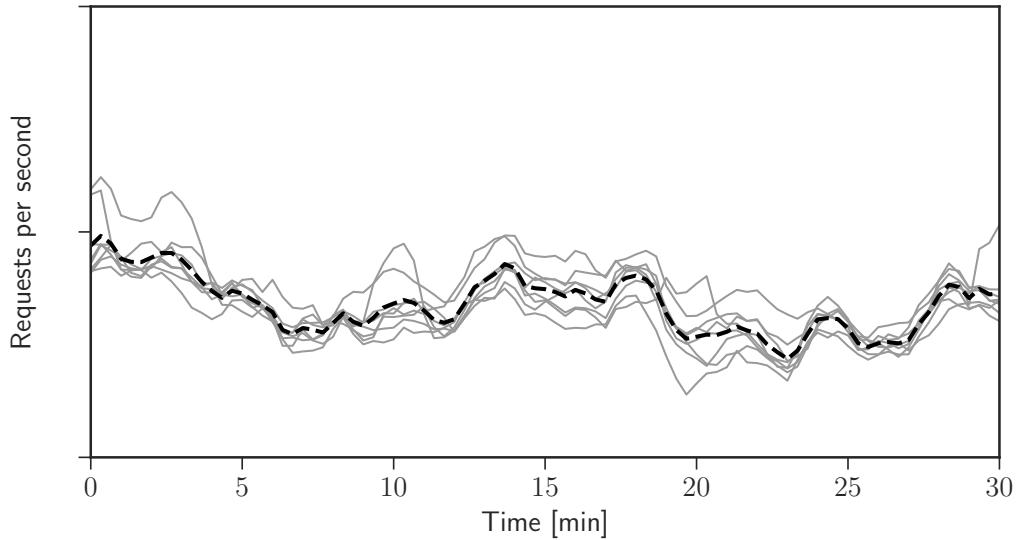
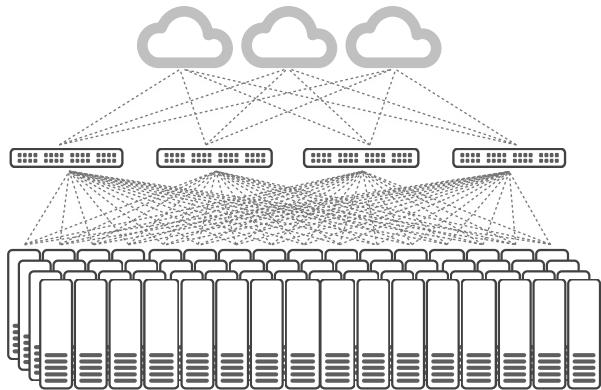
Hashing



Implications for capacity planning

- ▶ you are bound by most loaded host in a cluster

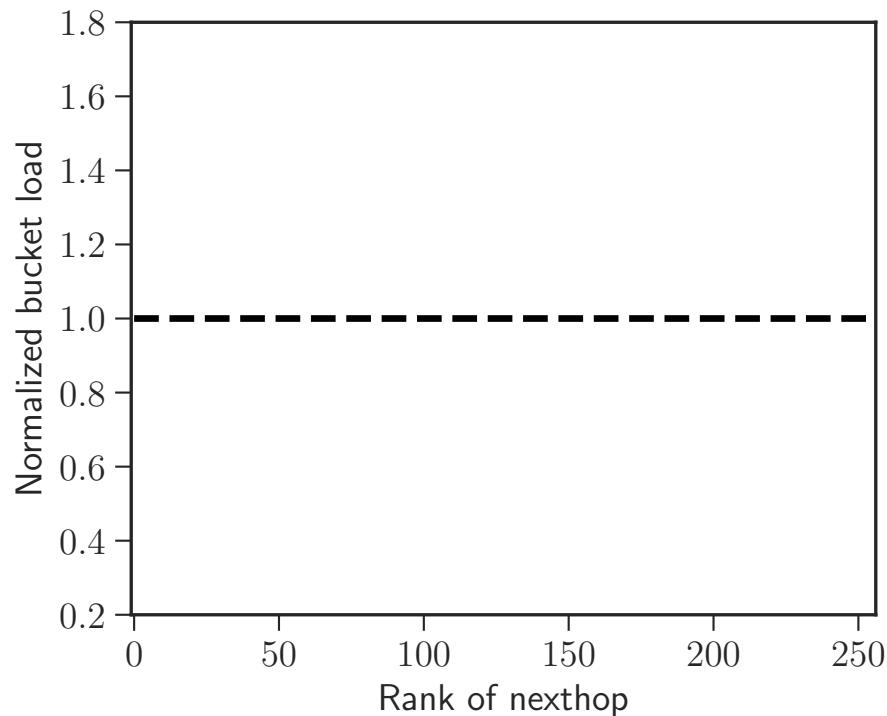
Hashing



Implications for capacity planning

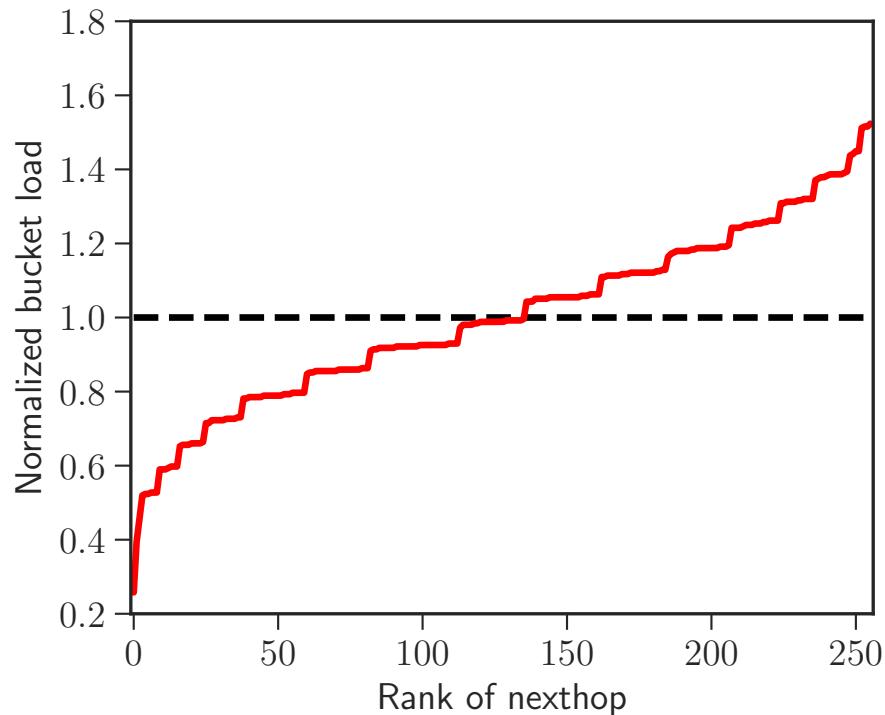
- ▶ you are bound by most loaded host in a cluster

Uneven hashing



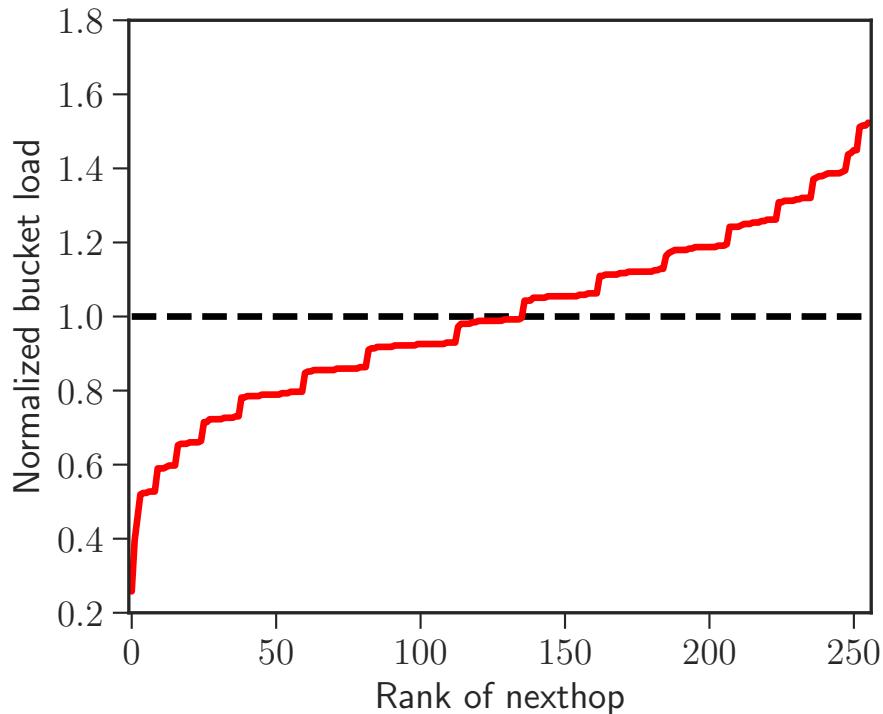
Inject synthetic, equally distributed traffic

Uneven hashing



Inject synthetic, equally distributed traffic

Uneven hashing

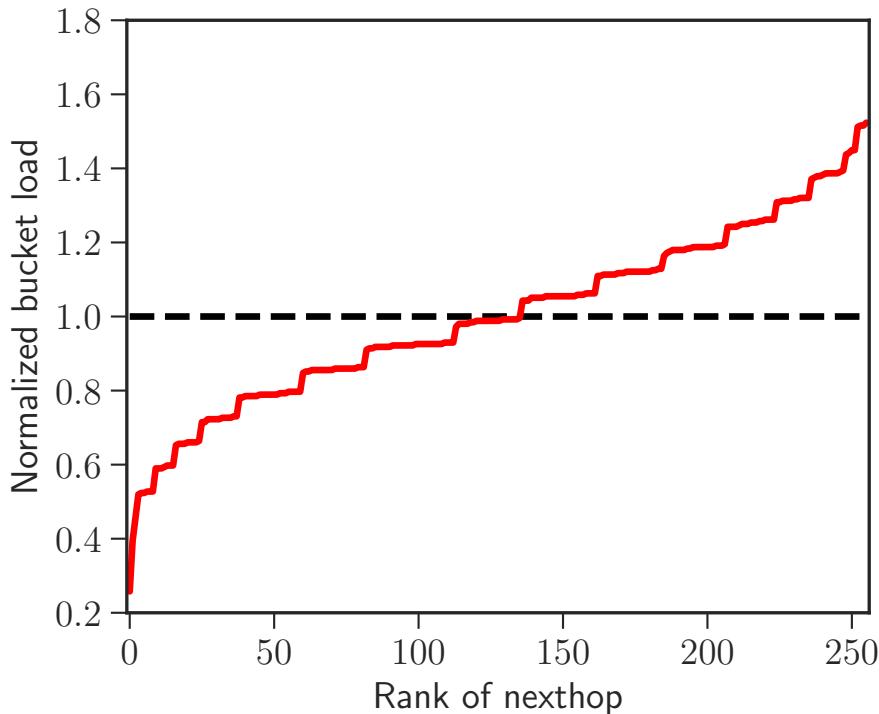


Inject synthetic, equally distributed traffic

Significant skew

- ▶ most loaded bucket 6 times more loaded than the least loaded

Uneven hashing



Inject synthetic, equally distributed traffic

Significant skew

- most loaded bucket 6 times more loaded than the least loaded

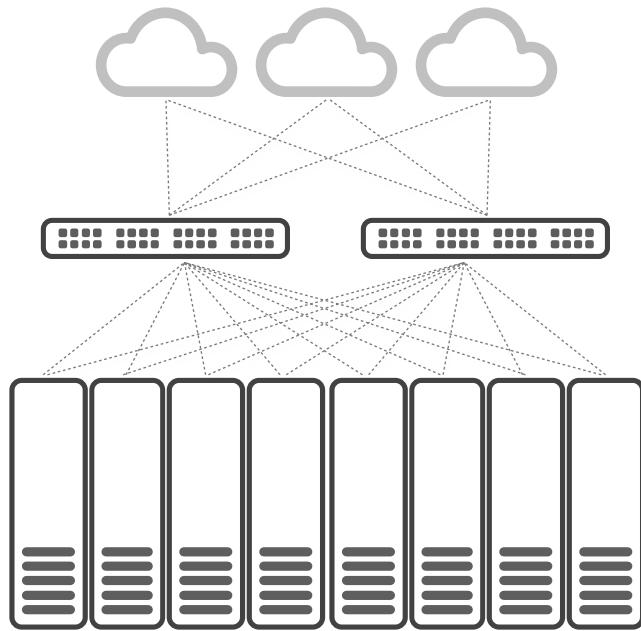
Behaviour can depend on number of nexthops

- some buckets received **no** traffic for specific number of configured nexthops

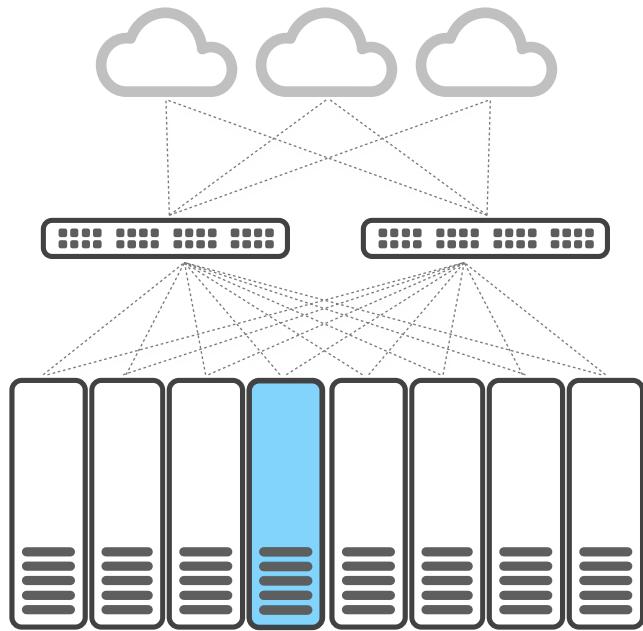
Assumption #2

switches hash identically

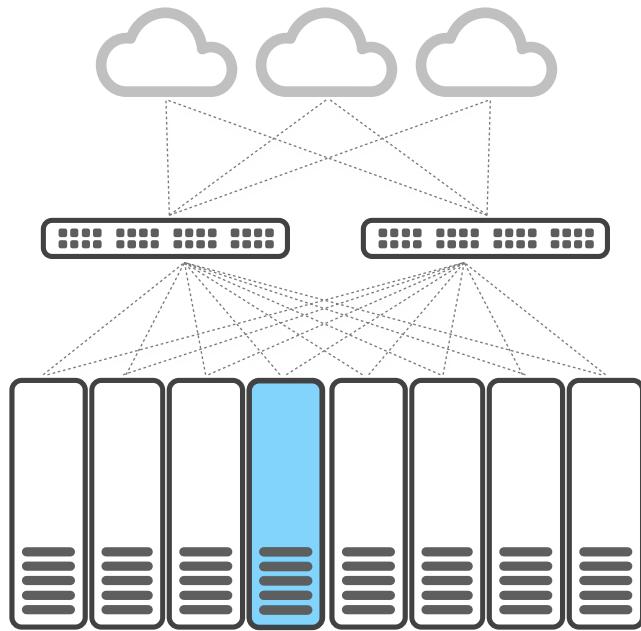
Hash polarization



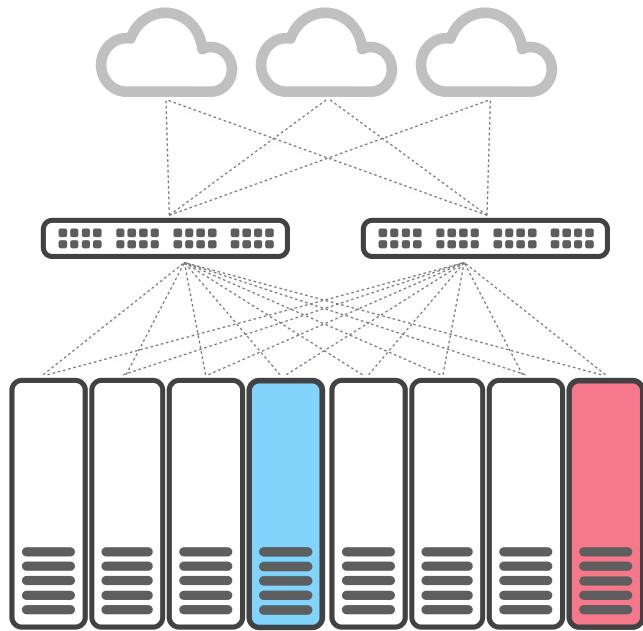
Hash polarization



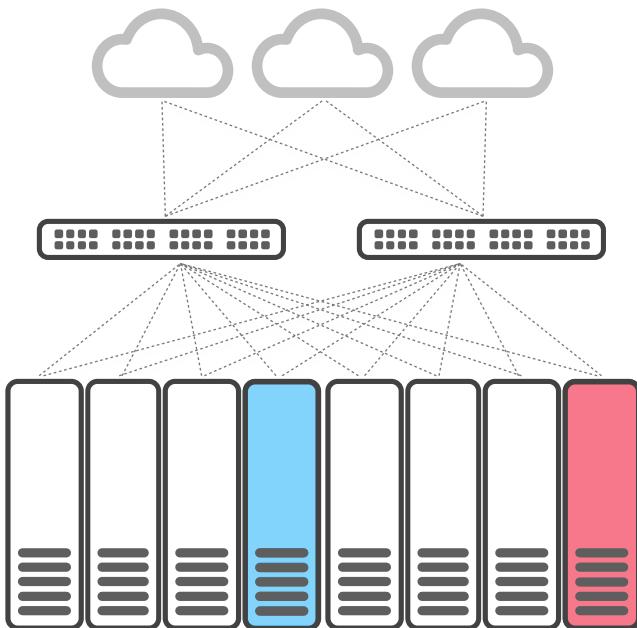
Hash polarization



Hash polarization



Hash polarization



Vendors were told hash polarization was bad

- in many cases you can't configure seed
- in one case, you can configure the seed, but vendor additionally uses boot order of linecards to add entropy

Assumption #3

packets in a flow use same network path

Nope, things break

Fragmentation

- returning ICMP packets hash on outer header
- took draft to IETF in 2014

ECN

- some middleboxes hash on TOS field
- ended up turning ECN negotiation off, breaks anycast too
- still looking for vendor(s) behind this, affected multiple ISPs

SYN proxies

- recent trend in enterprise appliances
- route lookup after connection handoff results in new path
- one vendor fixed implementation

paper has lots more stuff

- SYN cookie handling
- ARP reconfiguration measurements
- evaluation of switch and host draining
- switch controller details
- host-side implementation quirks
- ECMP skew results
- switch memory
- real flow measurements
- vendors that don't test their products
- ...

●●●○ Sprint LTE

5:18 PM

75% 

 Messages

Reviewers

Details



Text Message



 Messages **Reviewers**

Details

Layer 2 doesn't scale



Text Message



[Messages](#)[Reviewers](#)[Details](#)

Layer 2 doesn't scale

You didn't use P4, so it's
not portable



Text Message



[Messages](#)

Reviewers

[Details](#)

Layer 2 doesn't scale

You didn't use P4, so it's
not portable

Why didn't you use eBPF
or DPDK?



Text Message



NSDI

the value is not in the implementation

 Messages

Reviewers

Details

Layer 2 doesn't scale

You can use any encap
method



Text Message



[Messages](#)

Reviewers

[Details](#)

Layer 2 doesn't scale

You can use any encap
method

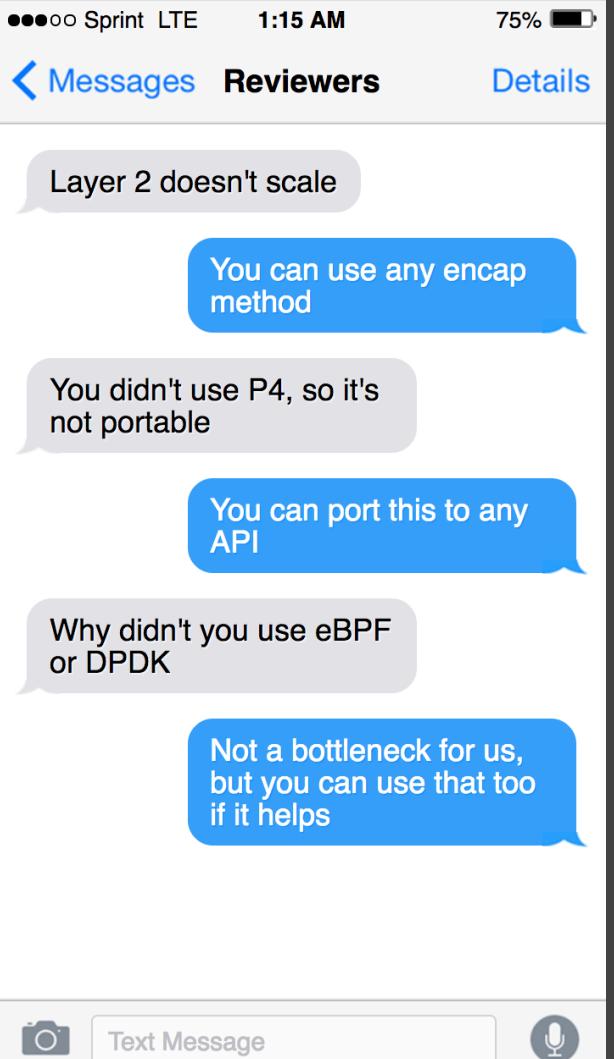
You didn't use P4, so it's
not portable

You can port this to any
API



Text Message





NSDI

the value is in the design

NSDI

the value is in the design

Faild decomposes load balancing as a division of labour

- ▶ leverage hardware wherever possible - **no latency cost in expected case**
- ▶ push functions requiring state towards hosts - **low latency overhead in worst case**
- ▶ result is **efficient, resilient** and **graceful**
- ▶ many of the design patterns applicable to other networking environments

NSDI

...the design is now part of the architecture

Faile part of shift in economics of edge delivery, has since percolated through industry

Five years of dealing with the consequences of changing a fraction of the Internet:

- ▶ PMTUD in ECMP networks
- ▶ talking to vendors about broken hashing implementations and middleboxes
- ▶ raising awareness within transport community and academia

If you propose protocol changes, please take this paper into account

Additional materials

- 2015
 - Networking @Scale [presentation](#)
 - SREcon [presentation](#)
- 2016
 - [RFC7690](#) workarounds for PMTUD in ECMP
 - [blogpost](#) covering Faild design
- 2017
 - [NANOG 70](#) presentation on broken hashing