SDN research directions

Promising problems to invest time on

Laurent Vanbever

ETH Zürich

SDNschool 2015

July, 3 2015

3 110

3 110

# of citations of the original
OpenFlow paper in ~6 years

SDN is still growing

SDN is reaching into always more CS communities

Networking	Systems	Distributed Algorithms	Security	PL
SIGCOMM	OSDI	PODC	CCS	PLDI
NSDI	SOSP	DISC	NDSS	POPL
HotNets	SOCC		Usenix Security	OOPSLA
CoNEXT				

Why?!

SDN finally enables us to innovate,
at a much faster pace

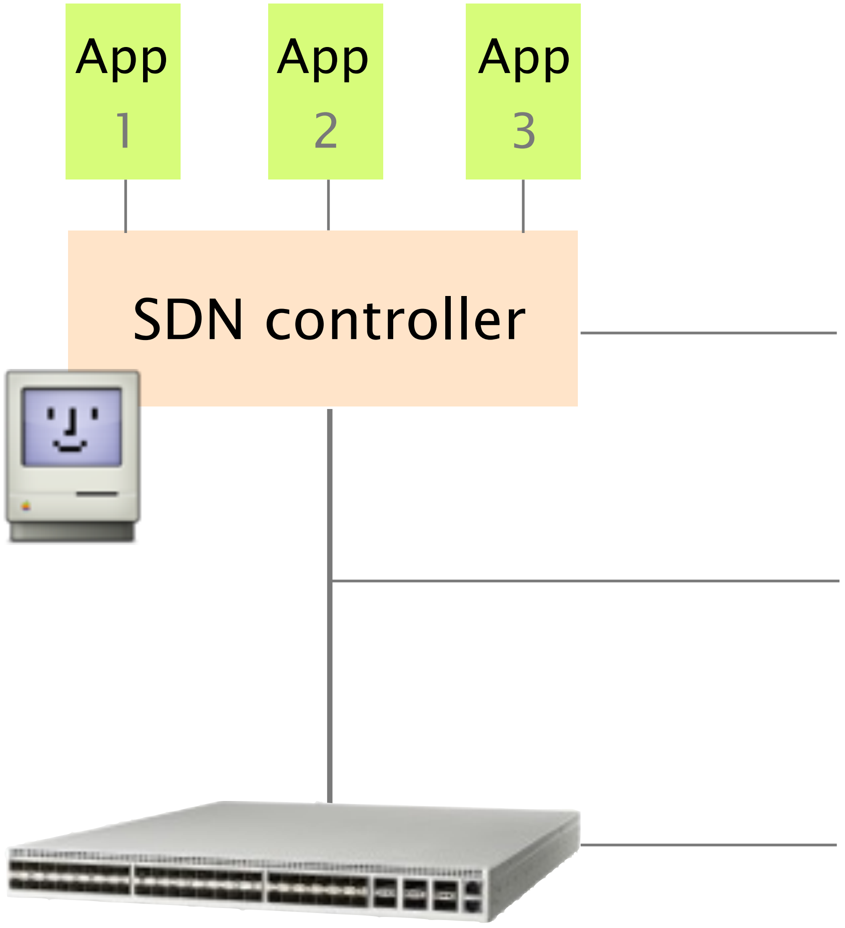
Before SDN

closed software

closed hardware

Cisco™ device

After SDN



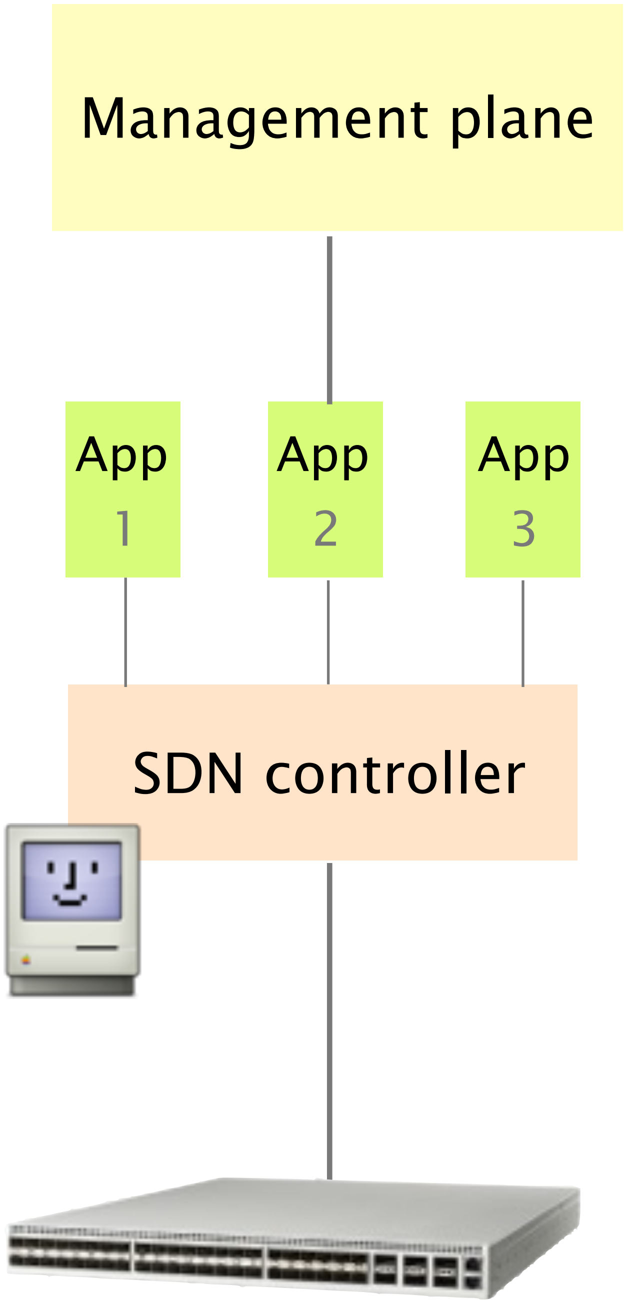
control software running on x86

standardized interface (OpenFlow)

standardized hardware

SDN device

Innovation is taking place
at each layer of the SDN stack



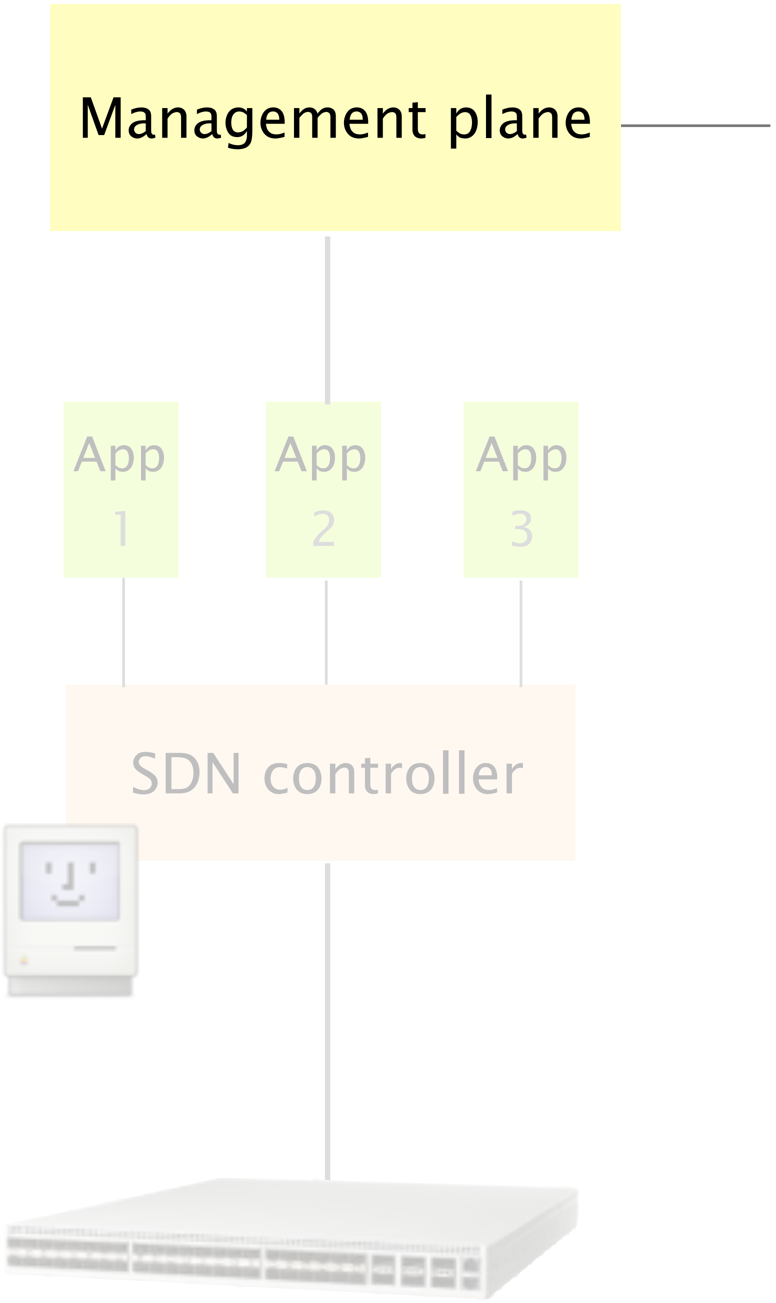
Management plane

App 1

App 2

App 3

SDN controller



Management plane

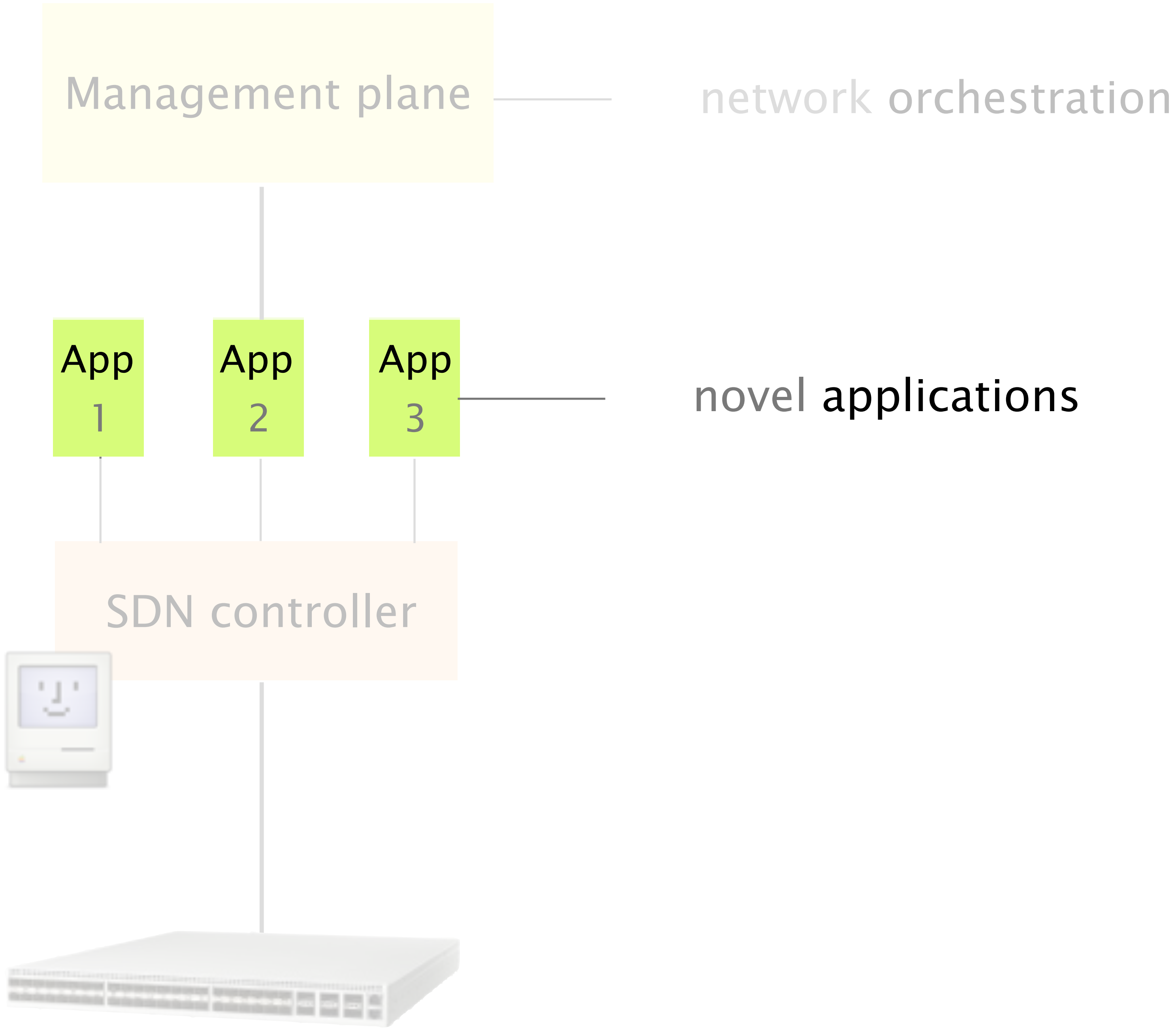
App 1

App 2

App 3

SDN controller

network orchestration



Management plane

network orchestration

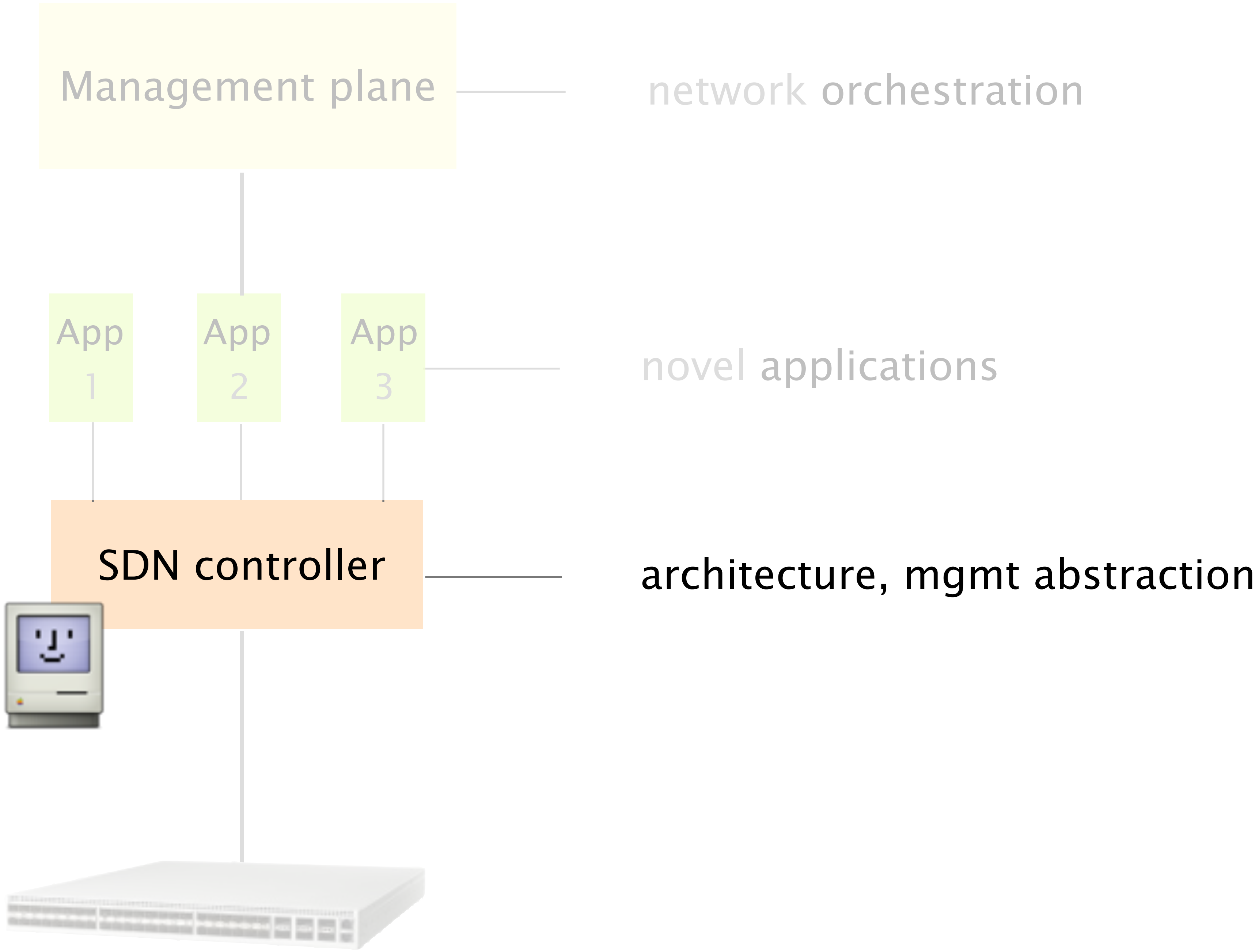
App 1

App 2

App 3

novel applications

SDN controller



Management plane

network orchestration

App 1

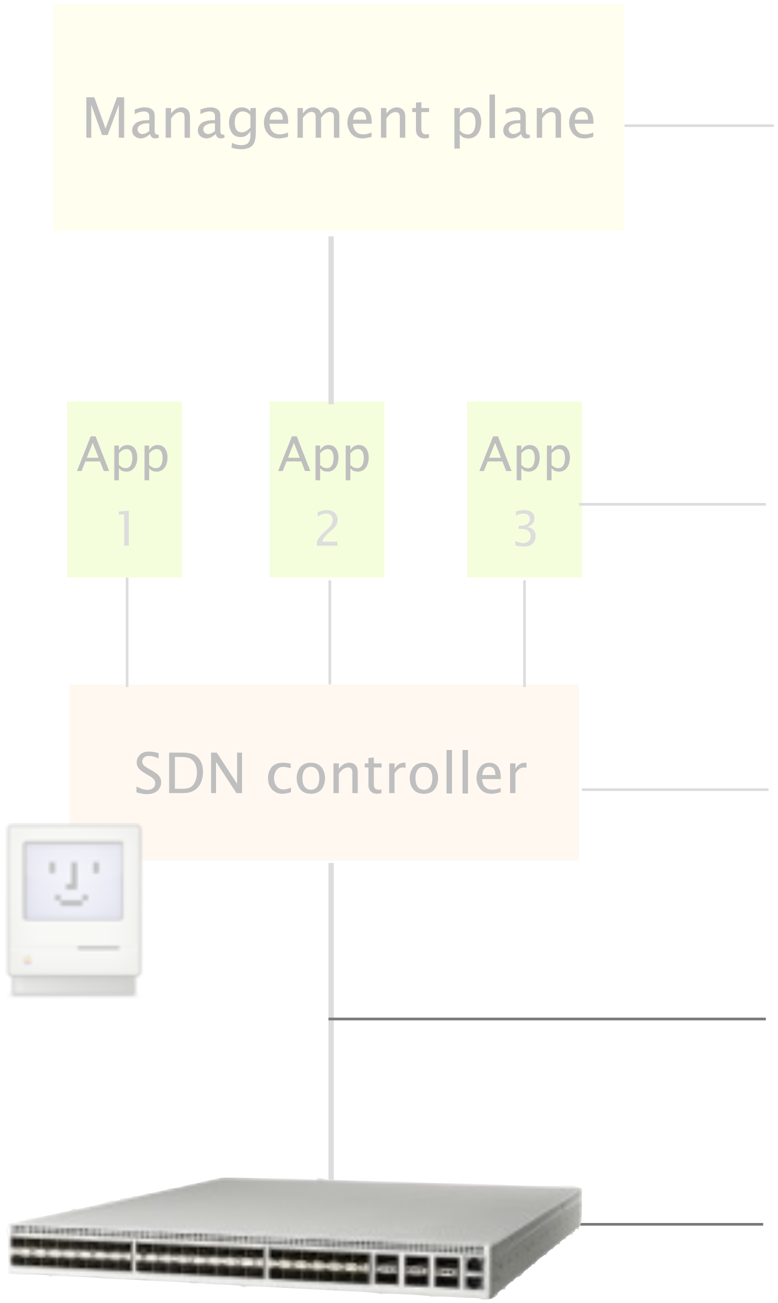
App 2

App 3

novel applications

SDN controller

architecture, mgmt abstraction



Management plane

App 1

App 2

App 3

SDN controller

network orchestration

novel applications

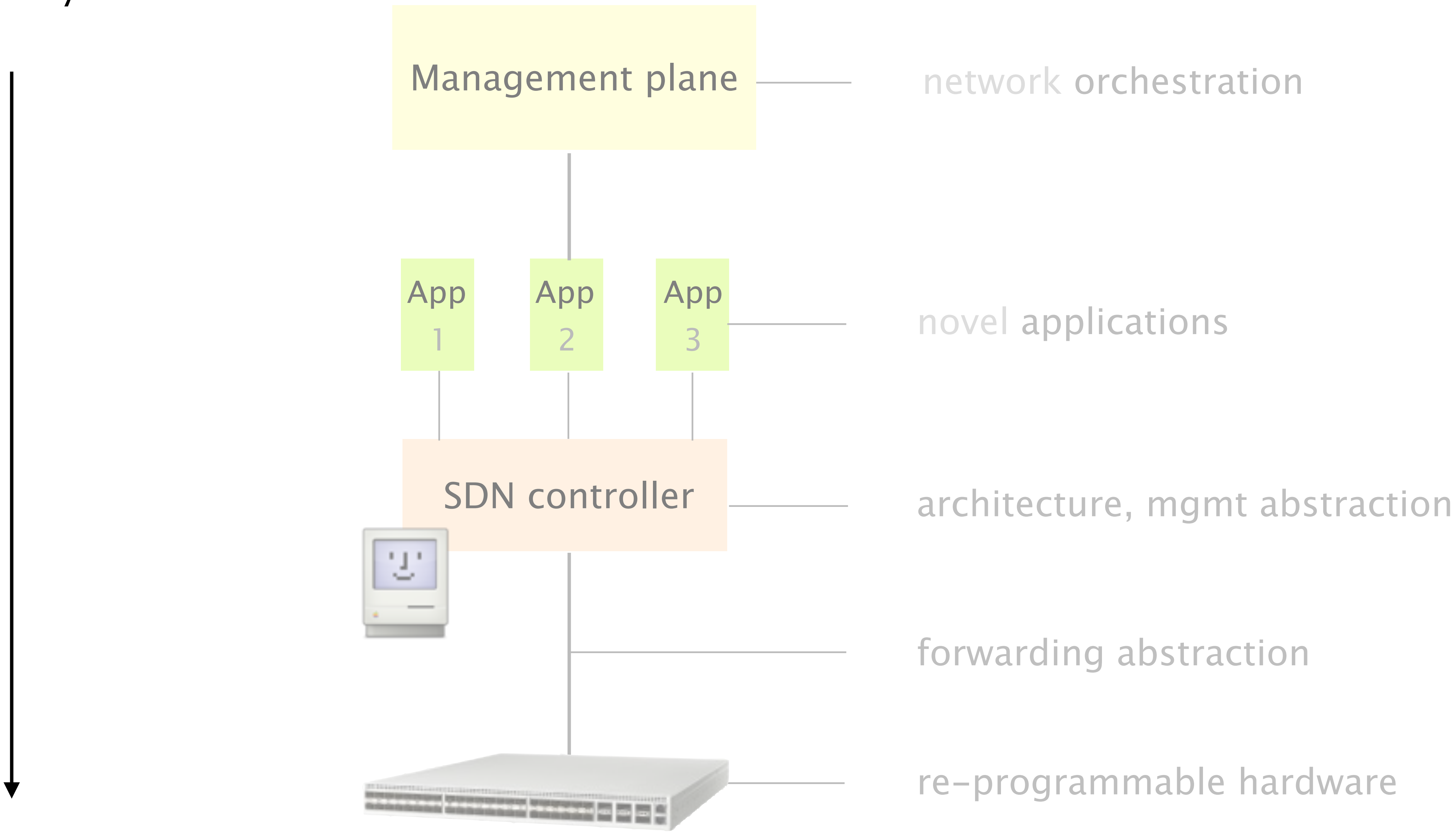
architecture, mgmt abstraction

forwarding abstraction

re-programmable hardware

Innovation is taking place
across layers of the SDN stack

security



Management plane

network orchestration

App 1

App 2

App 3

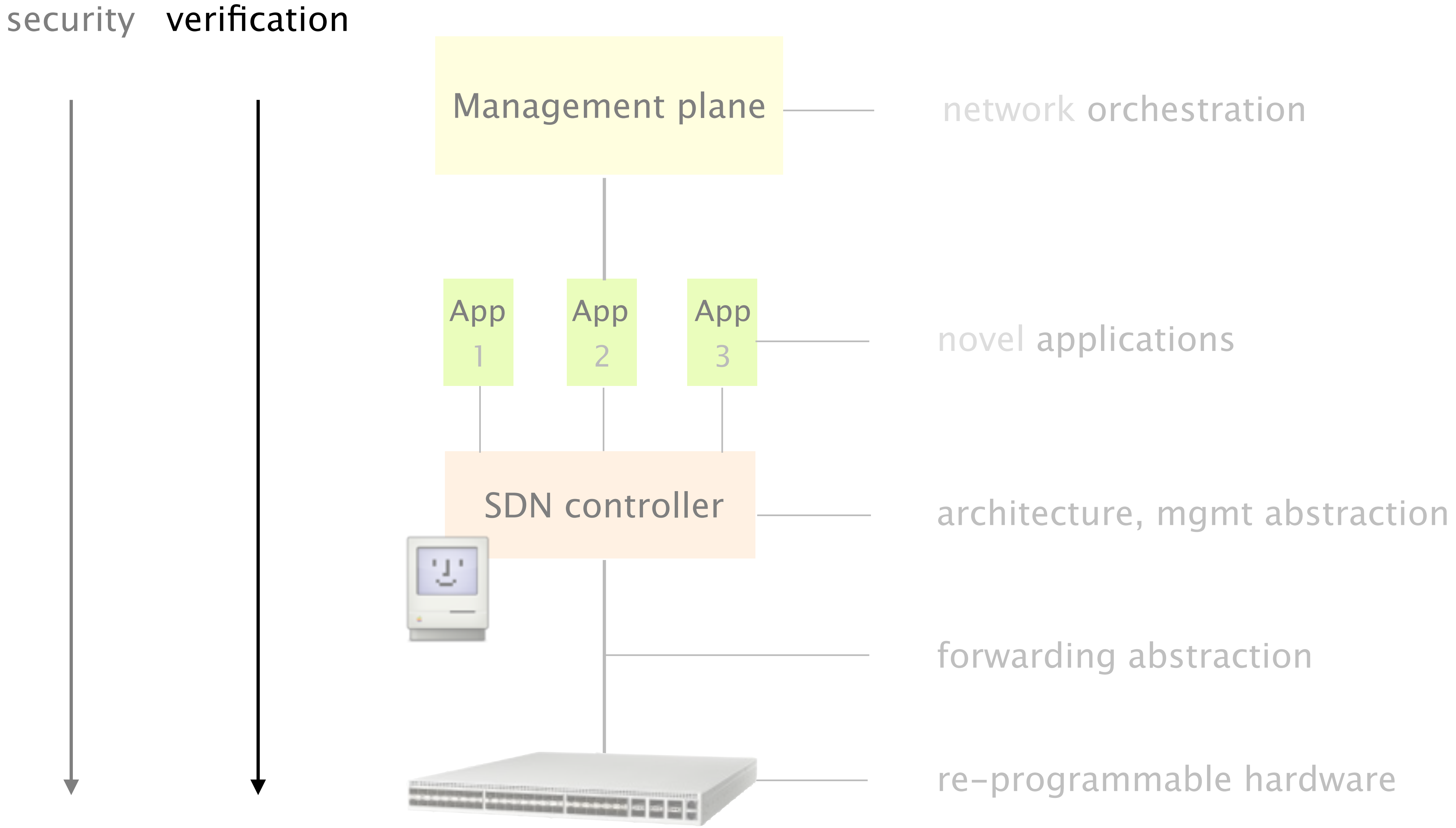
novel applications

SDN controller

architecture, mgmt abstraction

forwarding abstraction

re-programmable hardware



security

verification

Management plane

App 1

App 2

App 3

SDN controller

network orchestration

novel applications

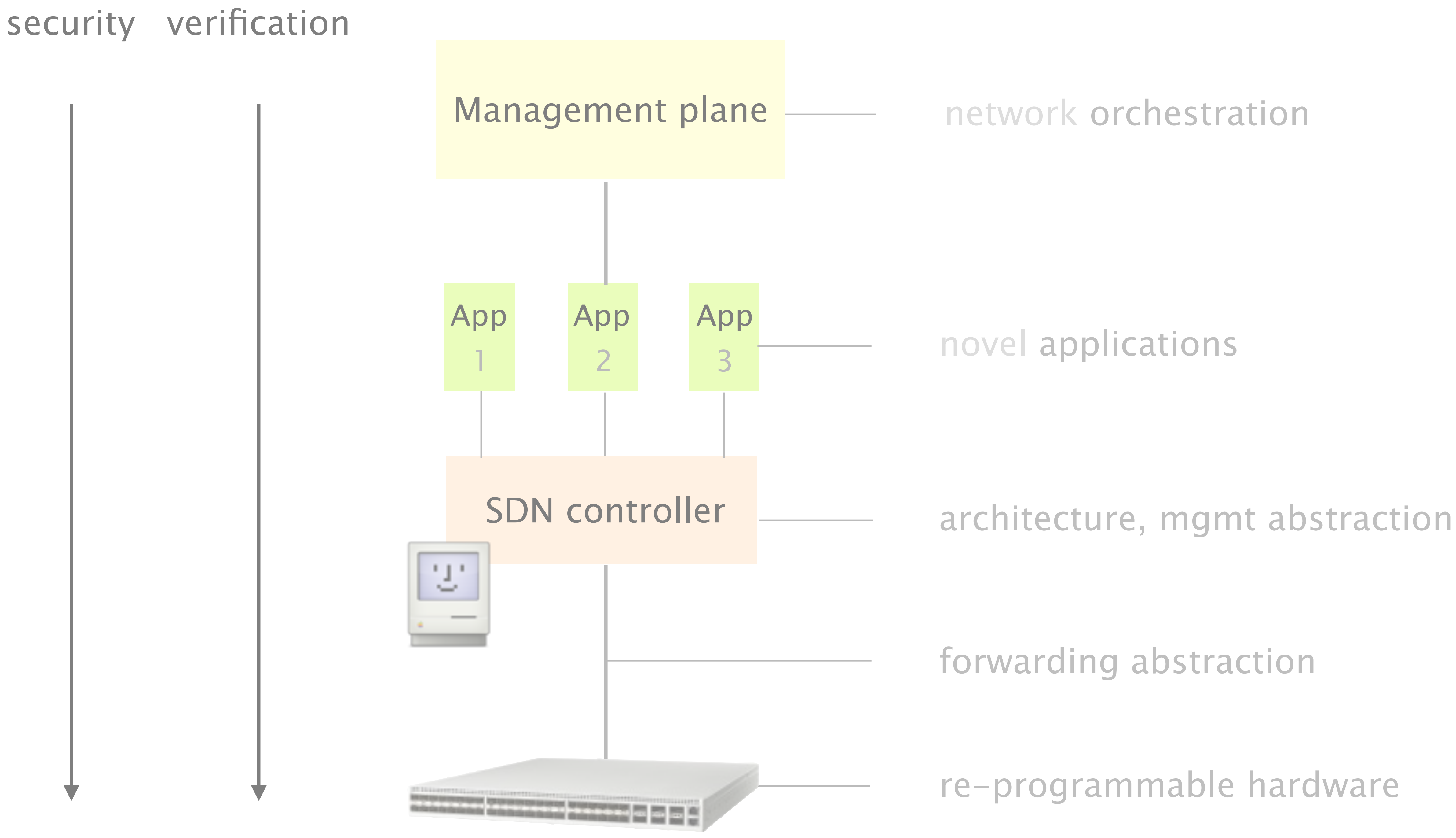
architecture, mgmt abstraction

forwarding abstraction

re-programmable hardware

Innovation is taking place to deploy SDN

deployment



security verification

Management plane

network orchestration

App 1

App 2

App 3

novel applications

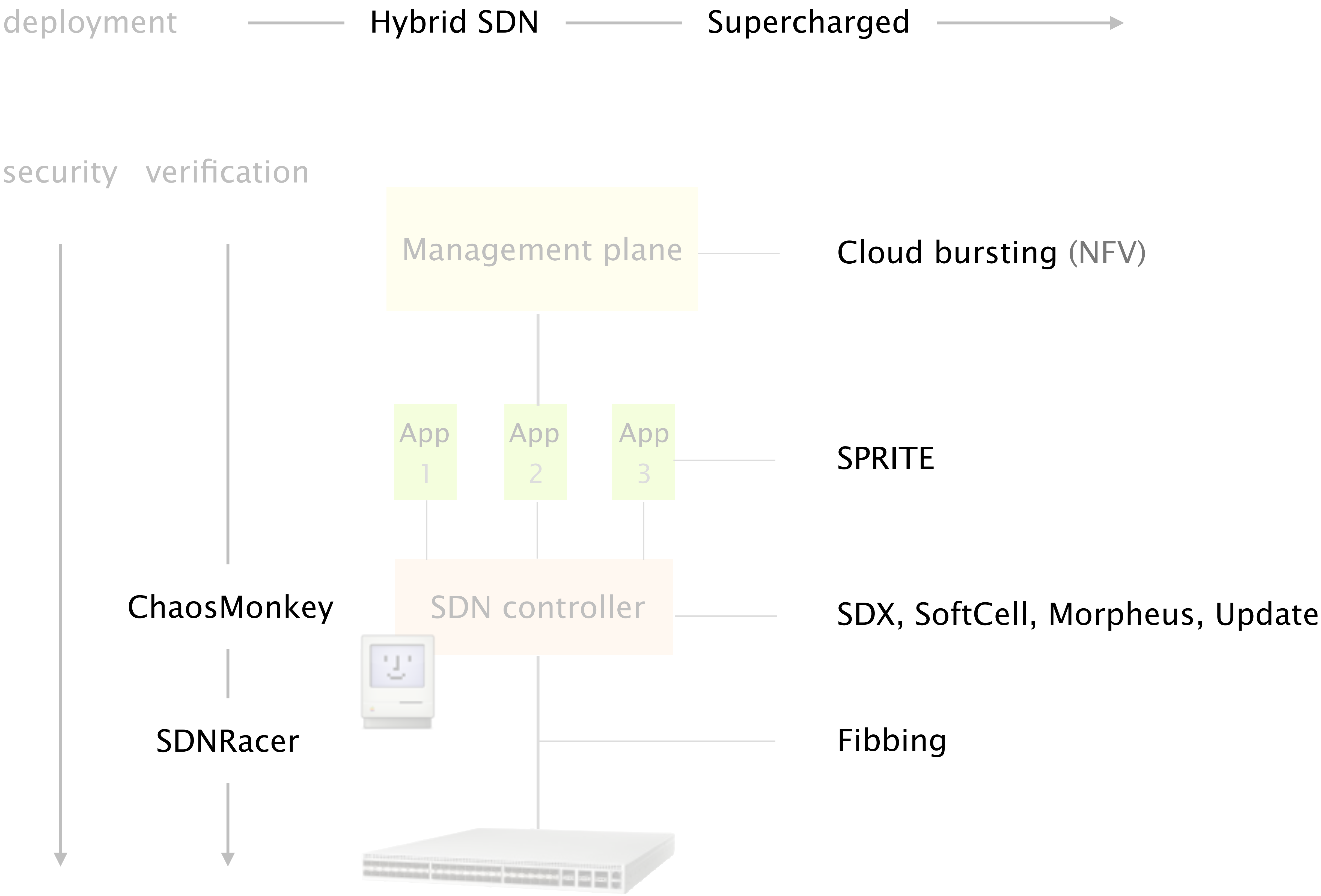
SDN controller

architecture, mgmt abstraction

forwarding abstraction

re-programmable hardware

My SDN research initiatives so far



deployment — Hybrid SDN — Supercharged →

security

verification

Management plane — Cloud bursting (NFV)

App 1

App 2

App 3 — SPRITE

ChaosMonkey

SDN controller — SDX, SoftCell, Morpheus, Update

SDNRacer

Fibbing

SDN research directions

Promising problems to invest time on

1. Go beyond OpenFlow
2. Secure SDN platforms
3. Incentivize deployment
4. Extend SDN reach

SDN research directions

Promising problems to invest time on

1 Go beyond OpenFlow

Secure SDN platforms

Incentivize deployment

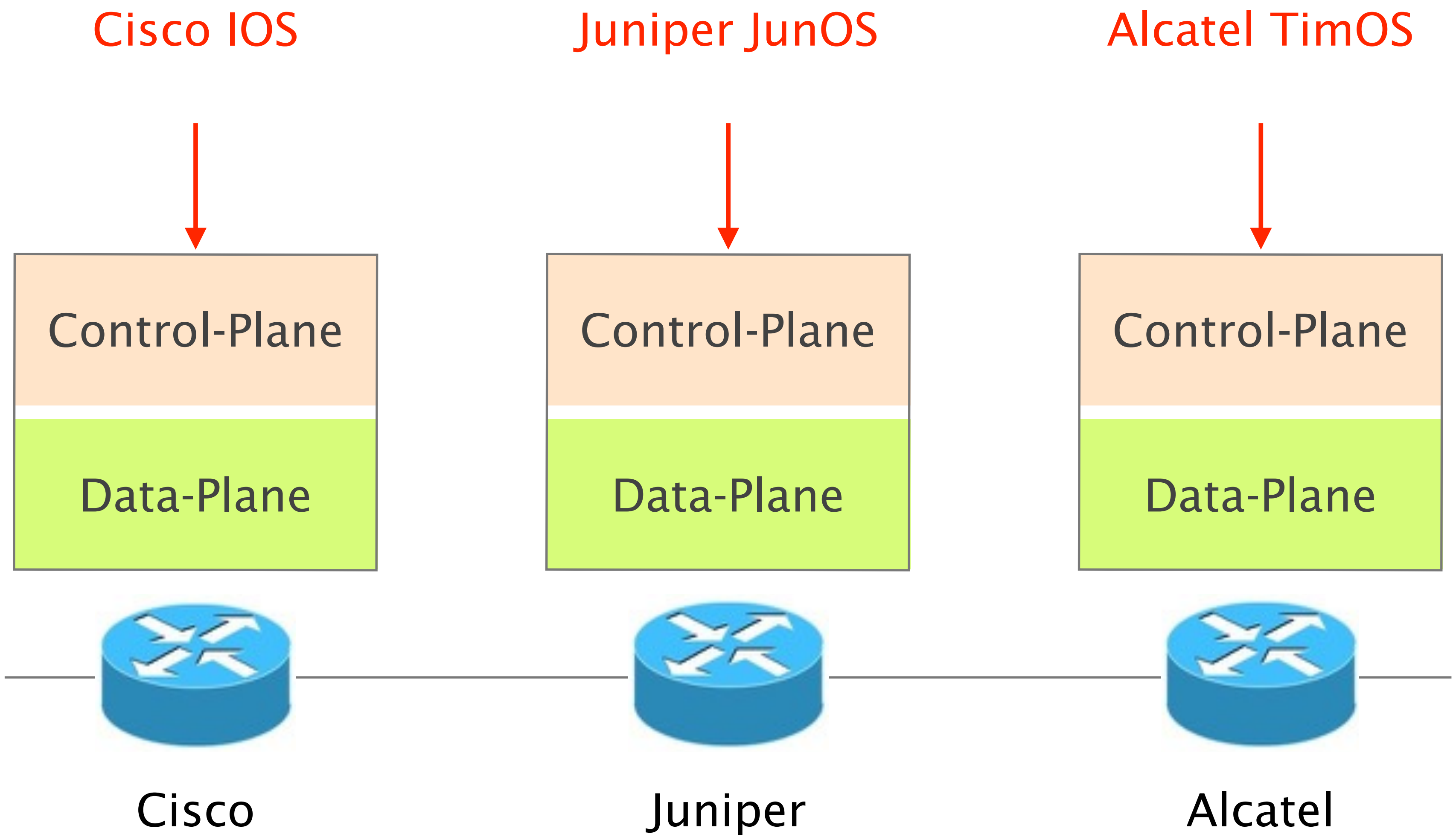
Extend SDN reach

Wouldn’t it be great to manage
an existing network “à la SDN”?

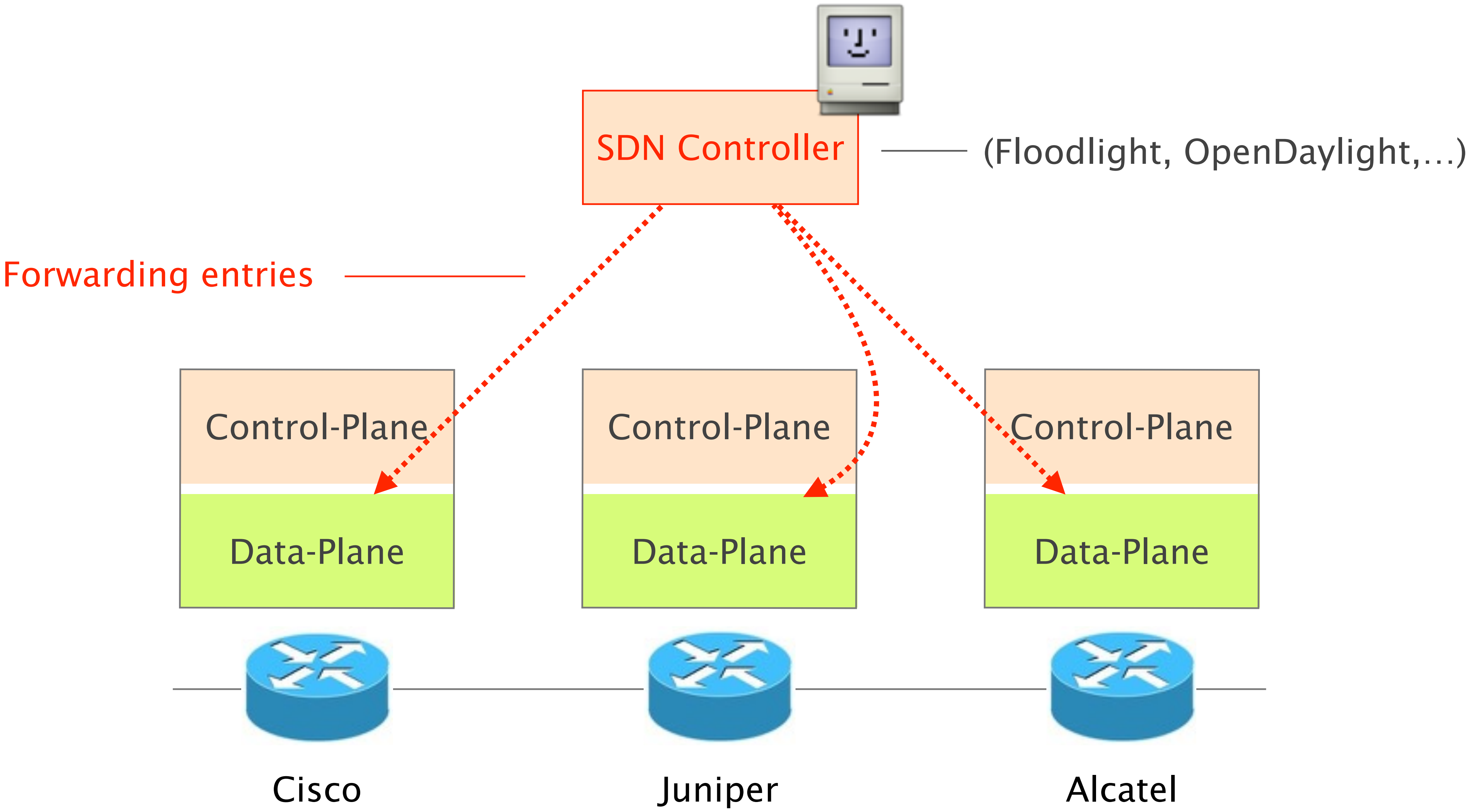
Wouldn’t it be great to manage an existing network “à la SDN”?

what does it mean?

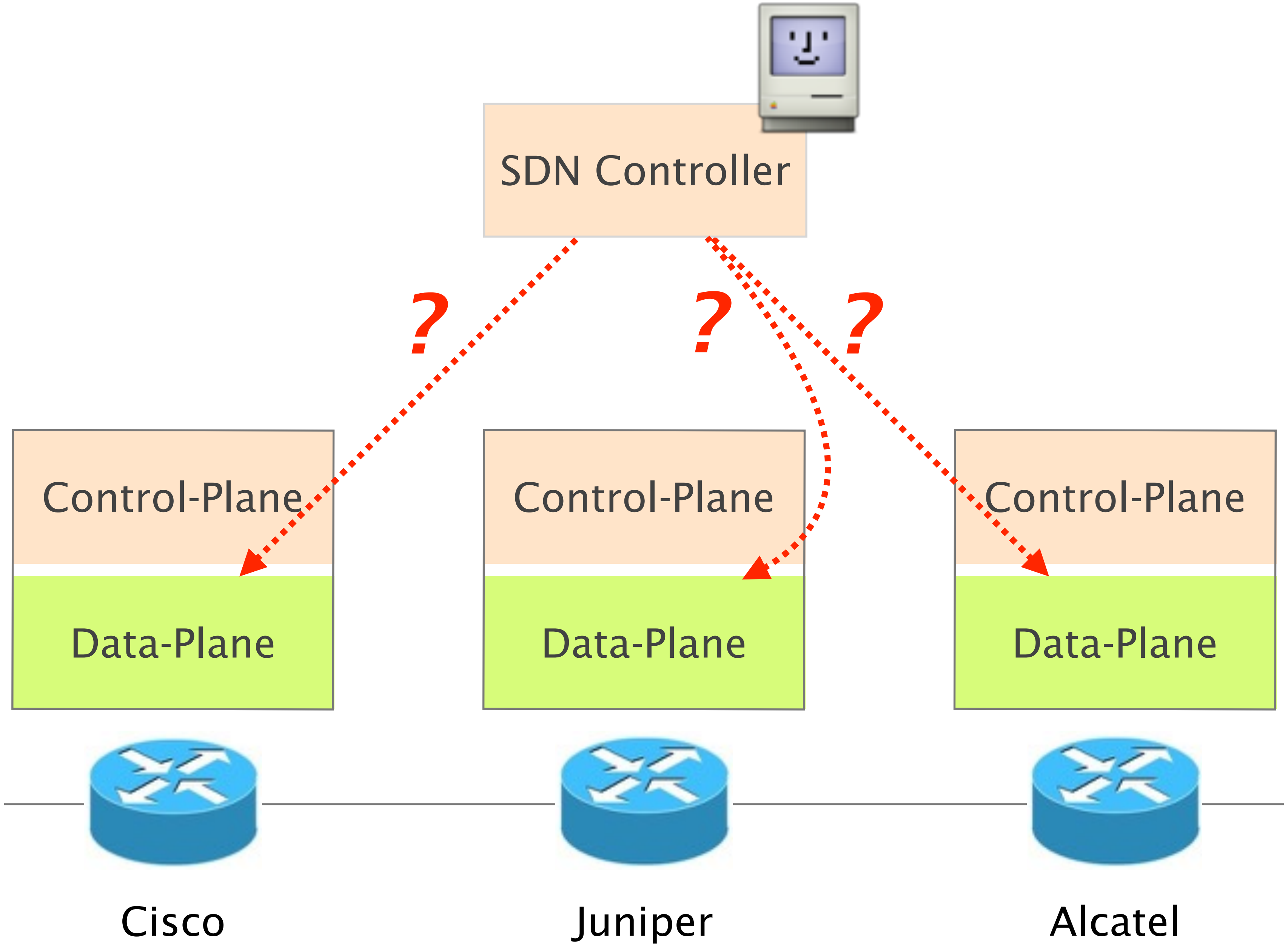
Instead of configuring a network using configuration “languages” …



… program it from a central SDN controller



For that, we need an API that *any* router can understand



Routing protocols are perfect candidates to act as such API

- messages are standardized

  routers must speak the same language

- behaviors are well-defined

  e.g., shortest-path routing

- implementations are widely available

  nearly all routers support OSPF

Fibbing

Fibbing

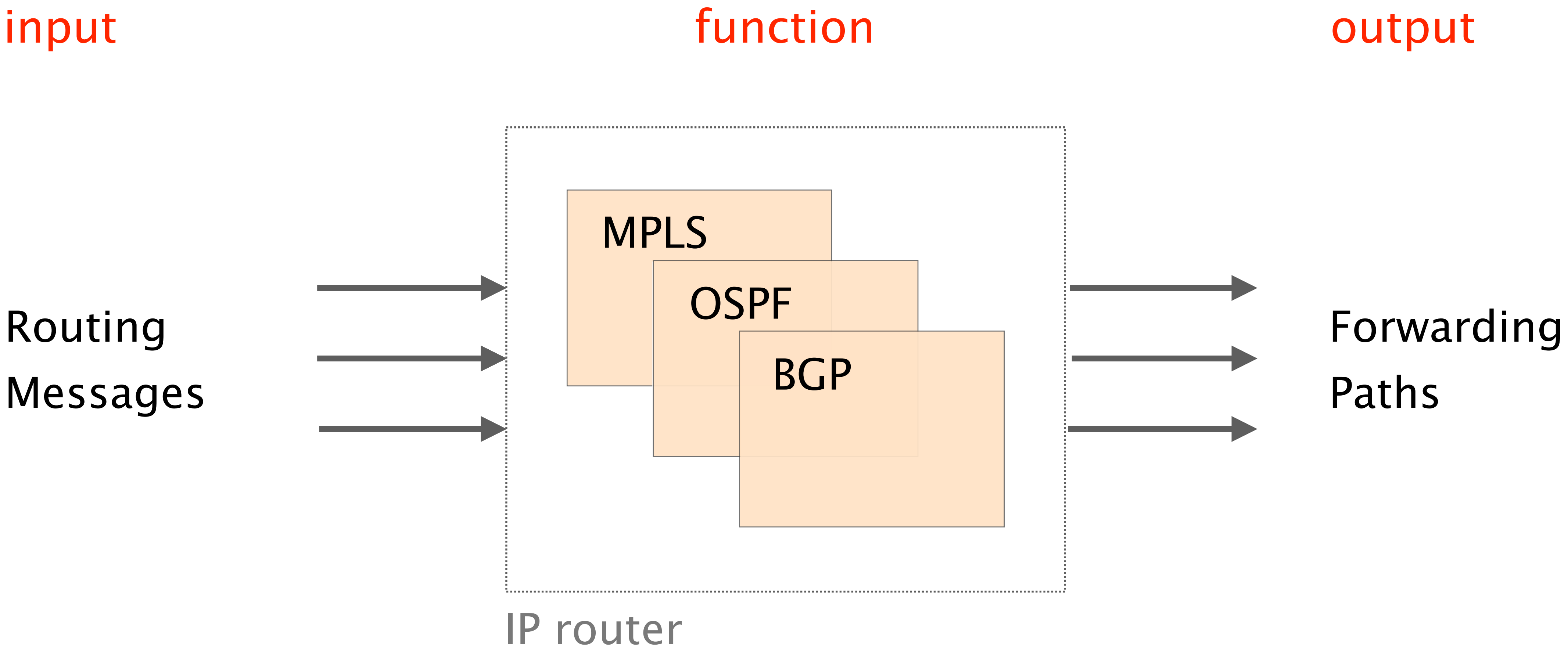
= lying

@SIGCOMM'15

Fibbing

to **control** router's forwarding table

A router implements a function from **routing messages** to **forwarding paths**



input

function

output

Routing Messages

MPLS

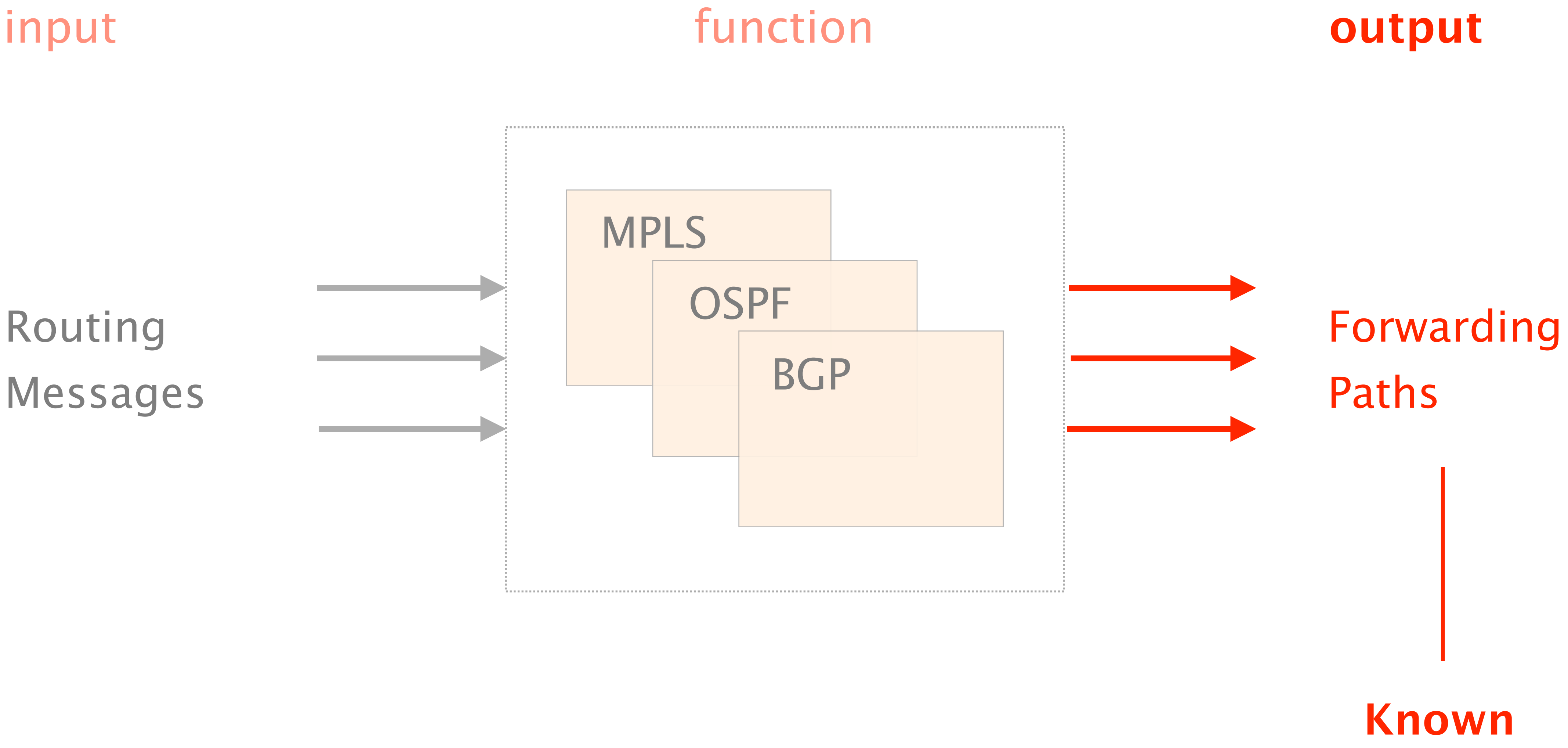
OSPF

BGP

IP router

Forwarding Paths

The forwarding paths are known, provided by the operators or by the controller



input

function

output

Routing Messages

MPLS

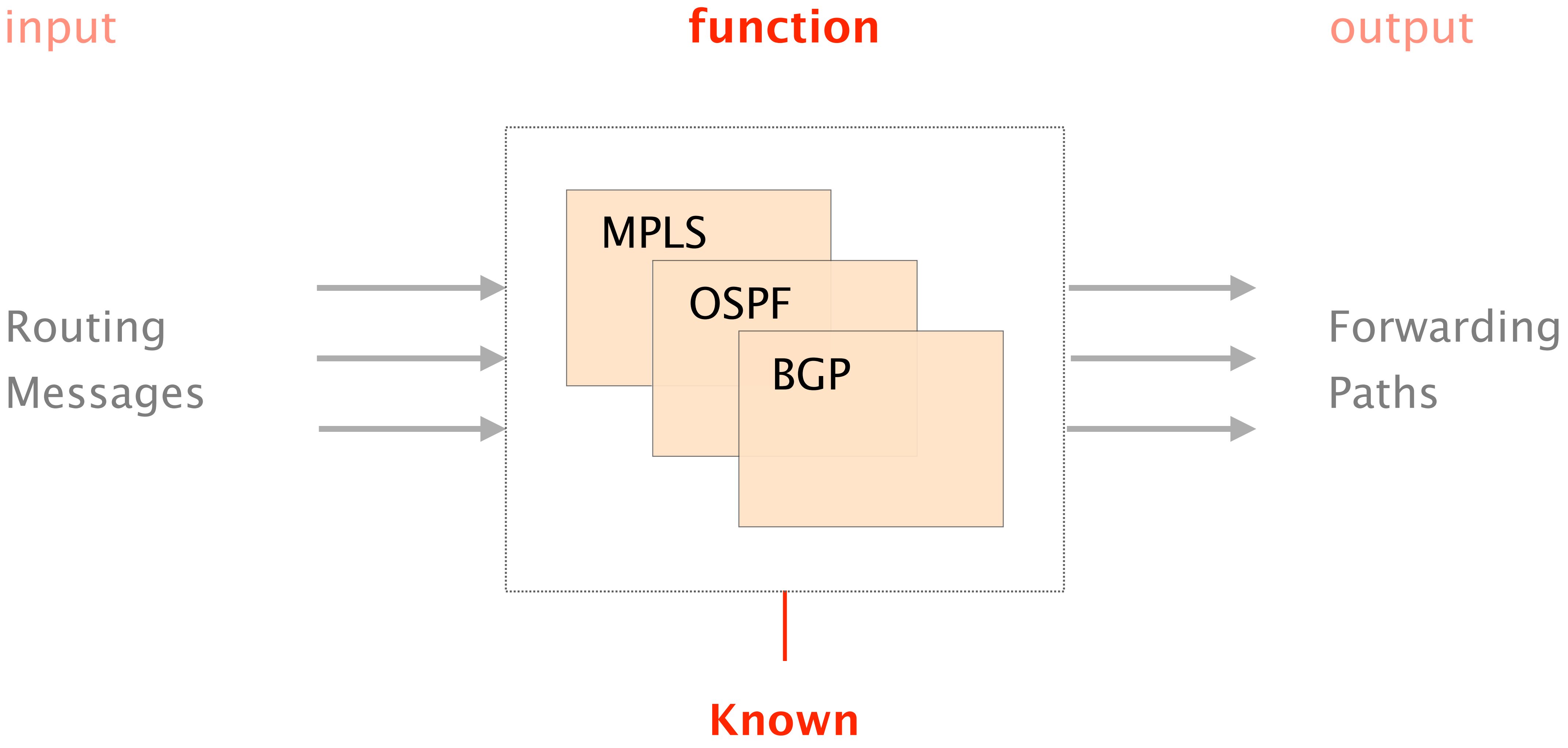
OSPF

BGP

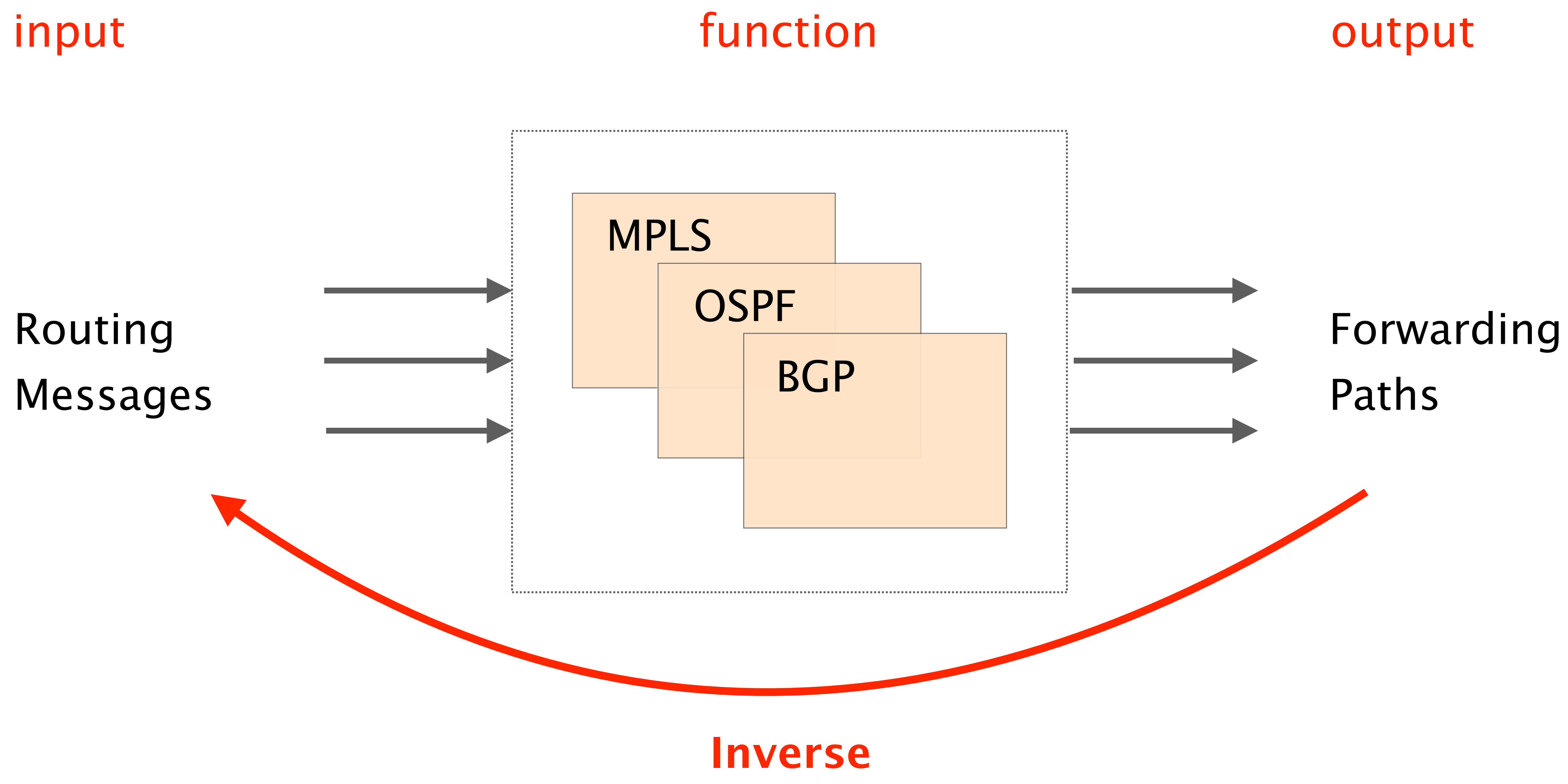
Forwarding Paths

Known

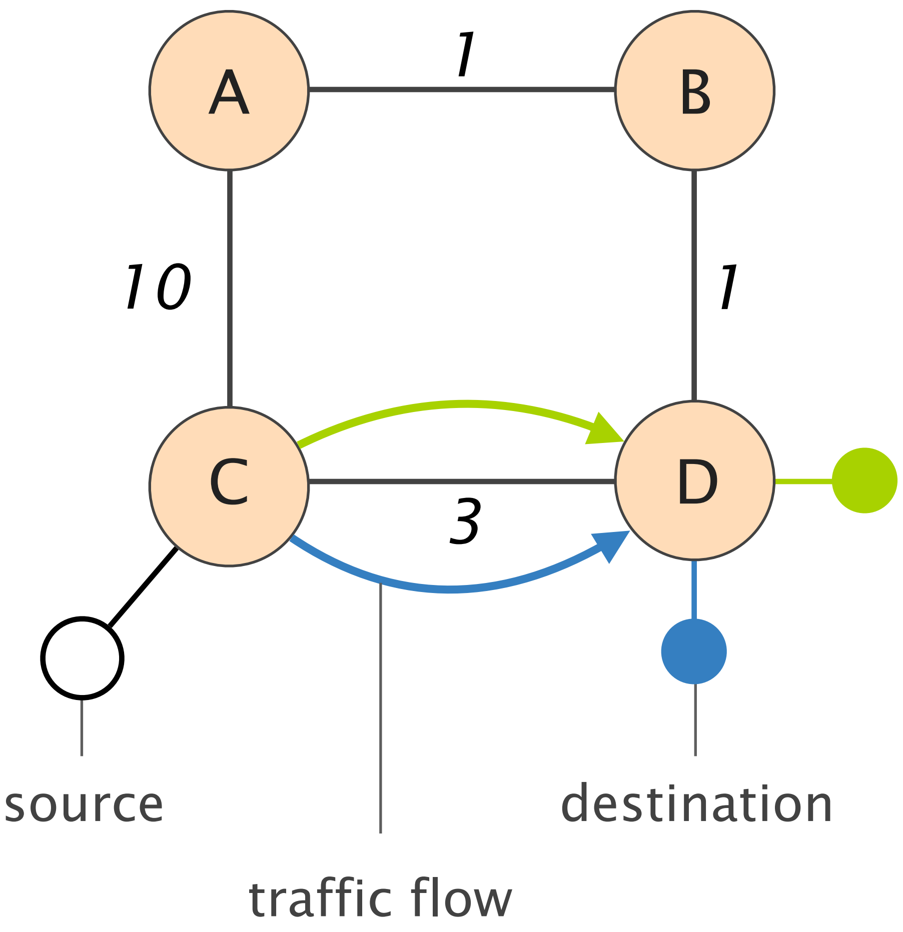
The function is known, from the protocols' specification & the configuration



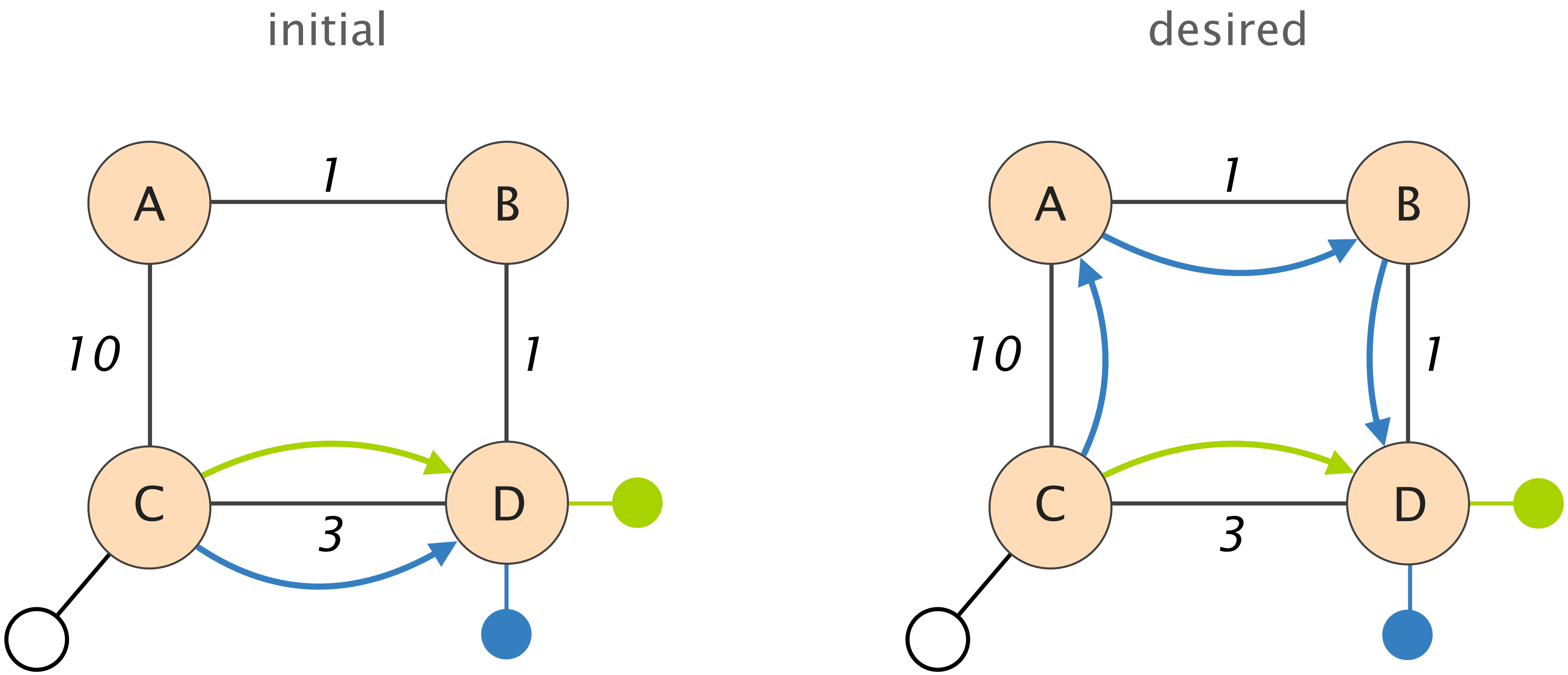
Given a path and a function, our framework computes corresponding routing messages by inverting the function



Consider this network where a source sends traffic to 2 destinations



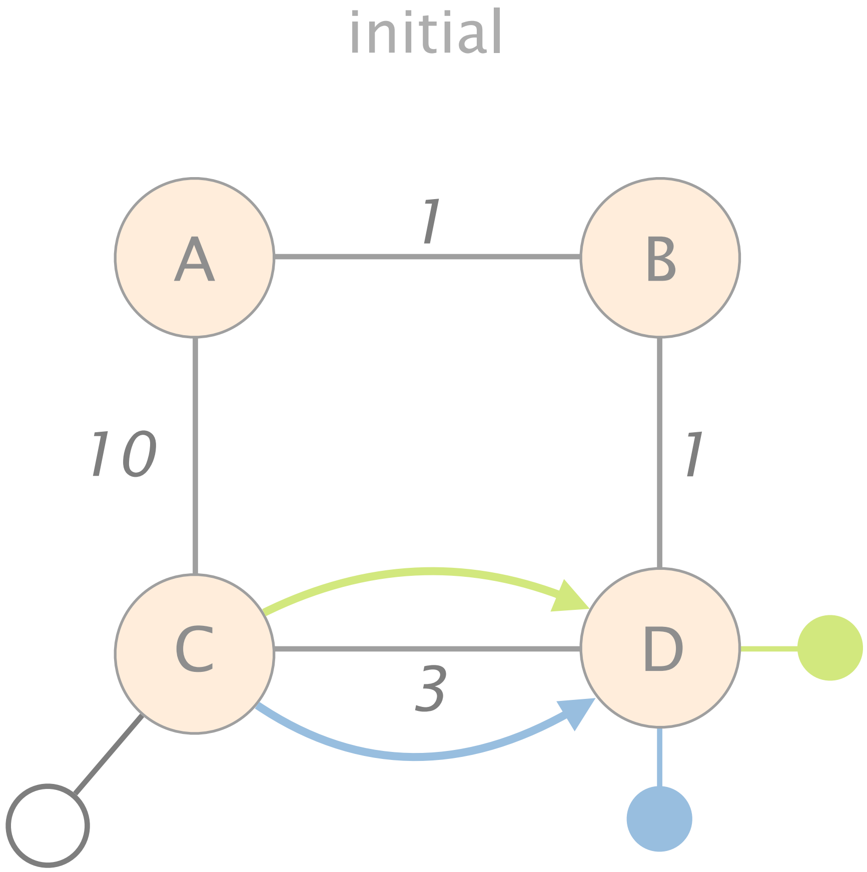
As congestion appears, the operator wants to shift away one flow from (C,D)

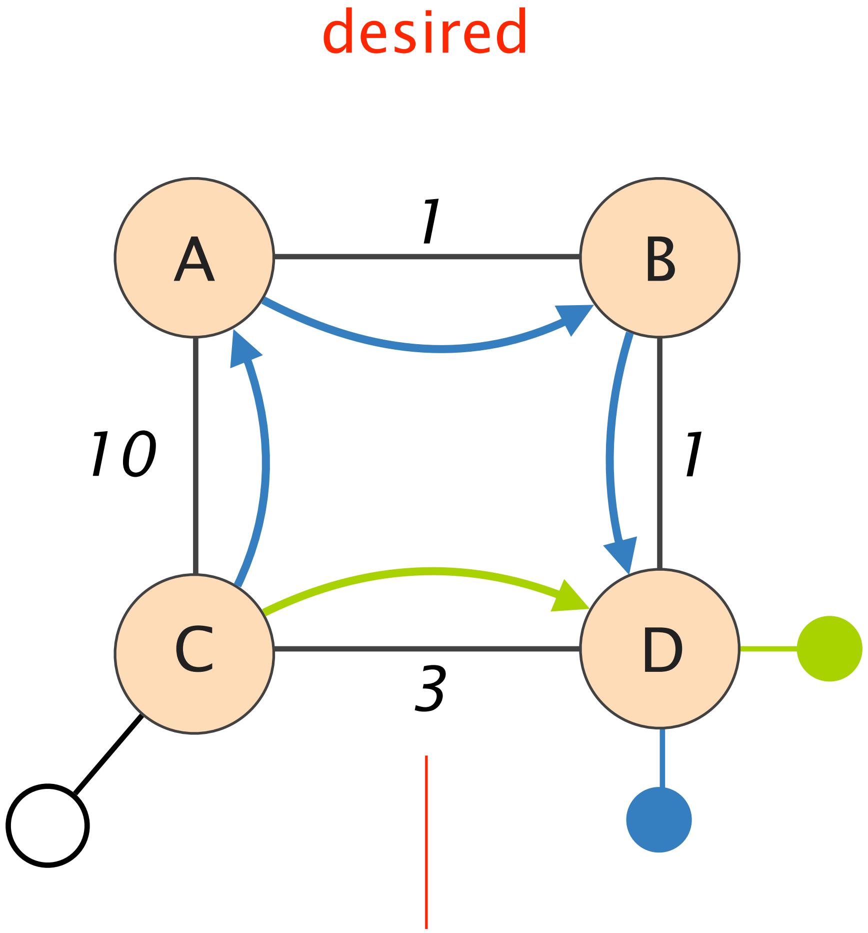


initial

desired

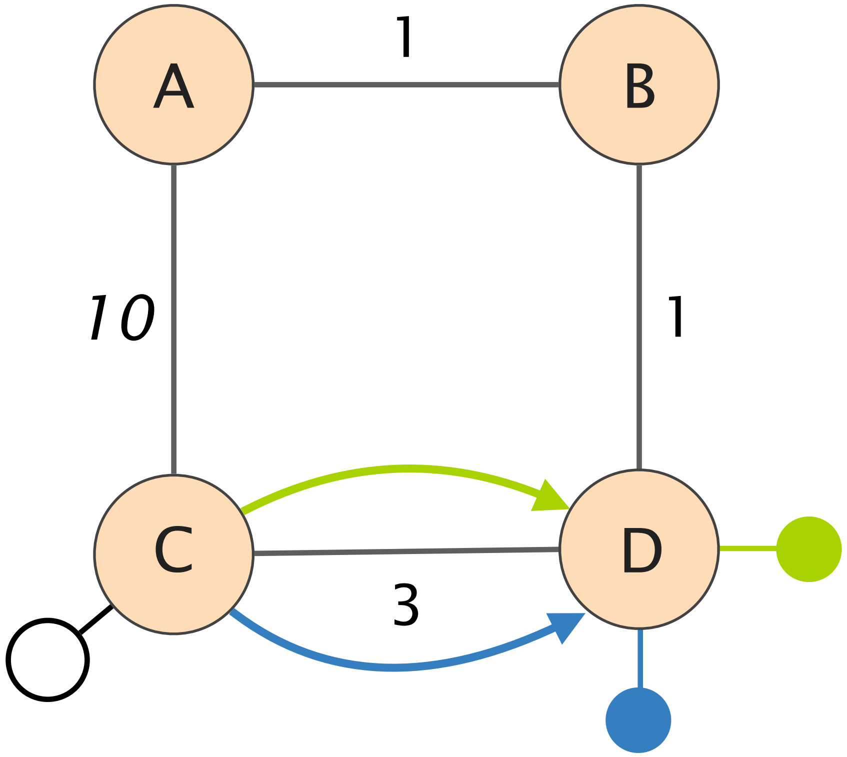
Moving only one flow is impossible though as both destinations are connected to D



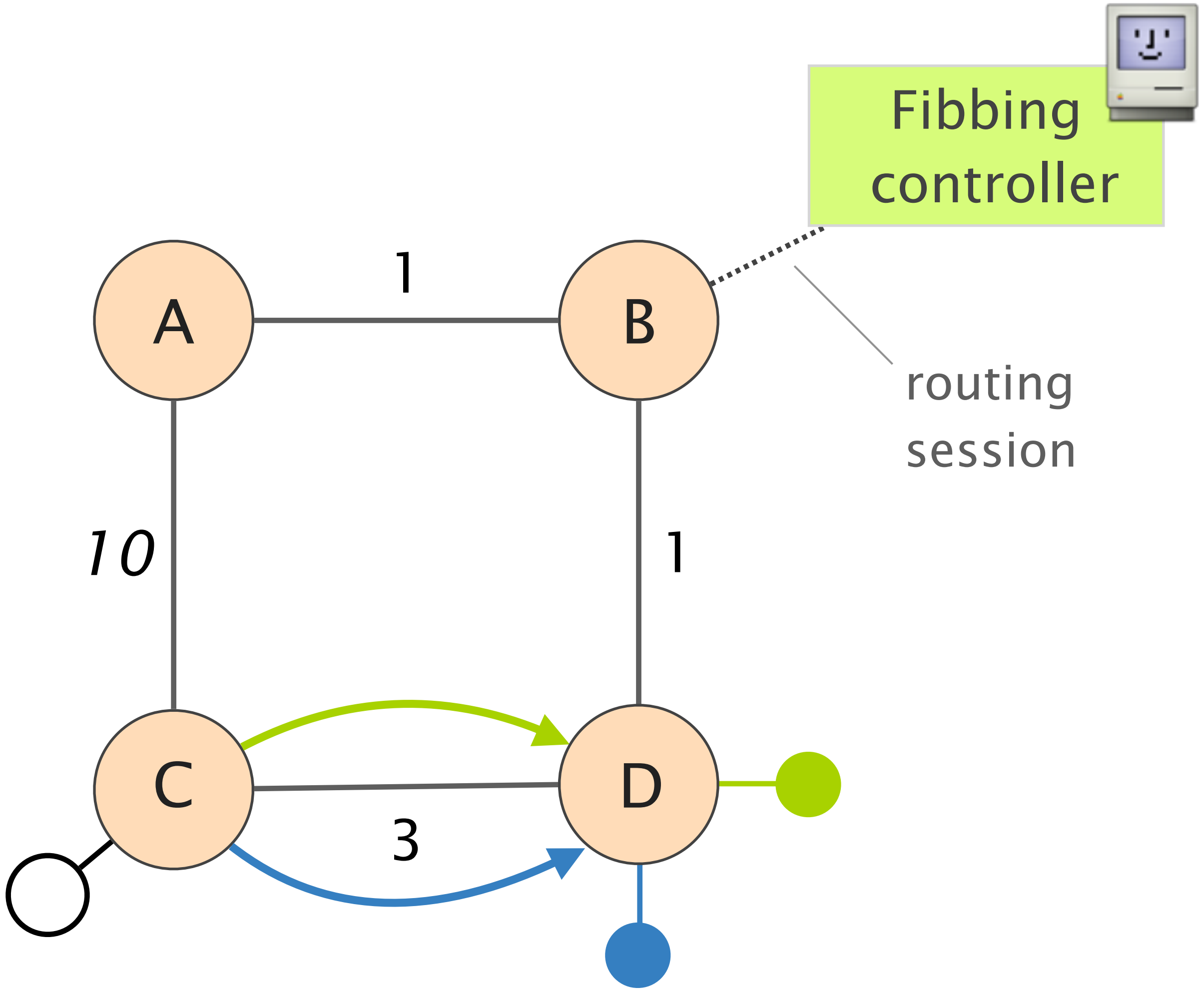


impossible to achieve by reweighing the links

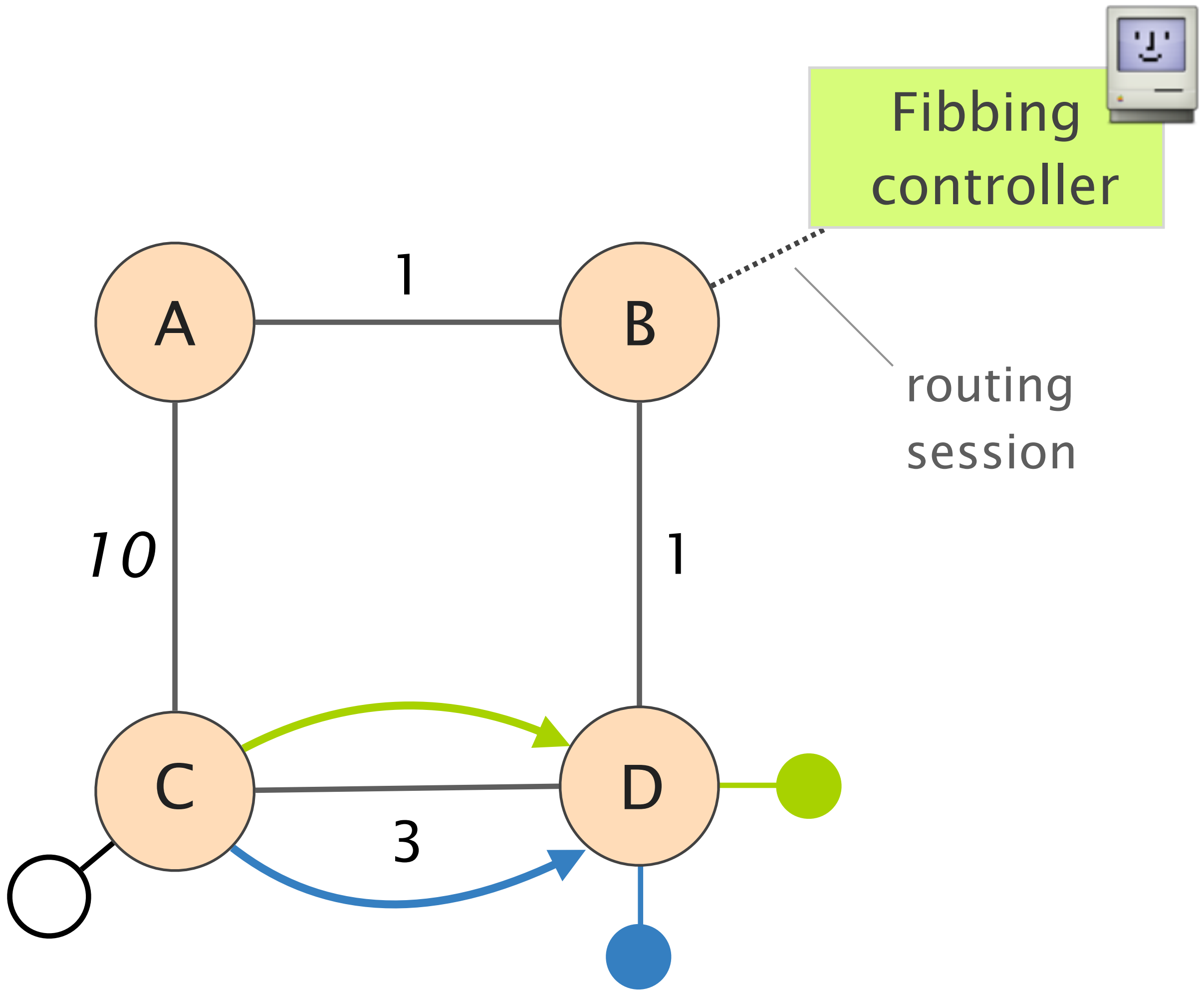
Let’s lie to the router



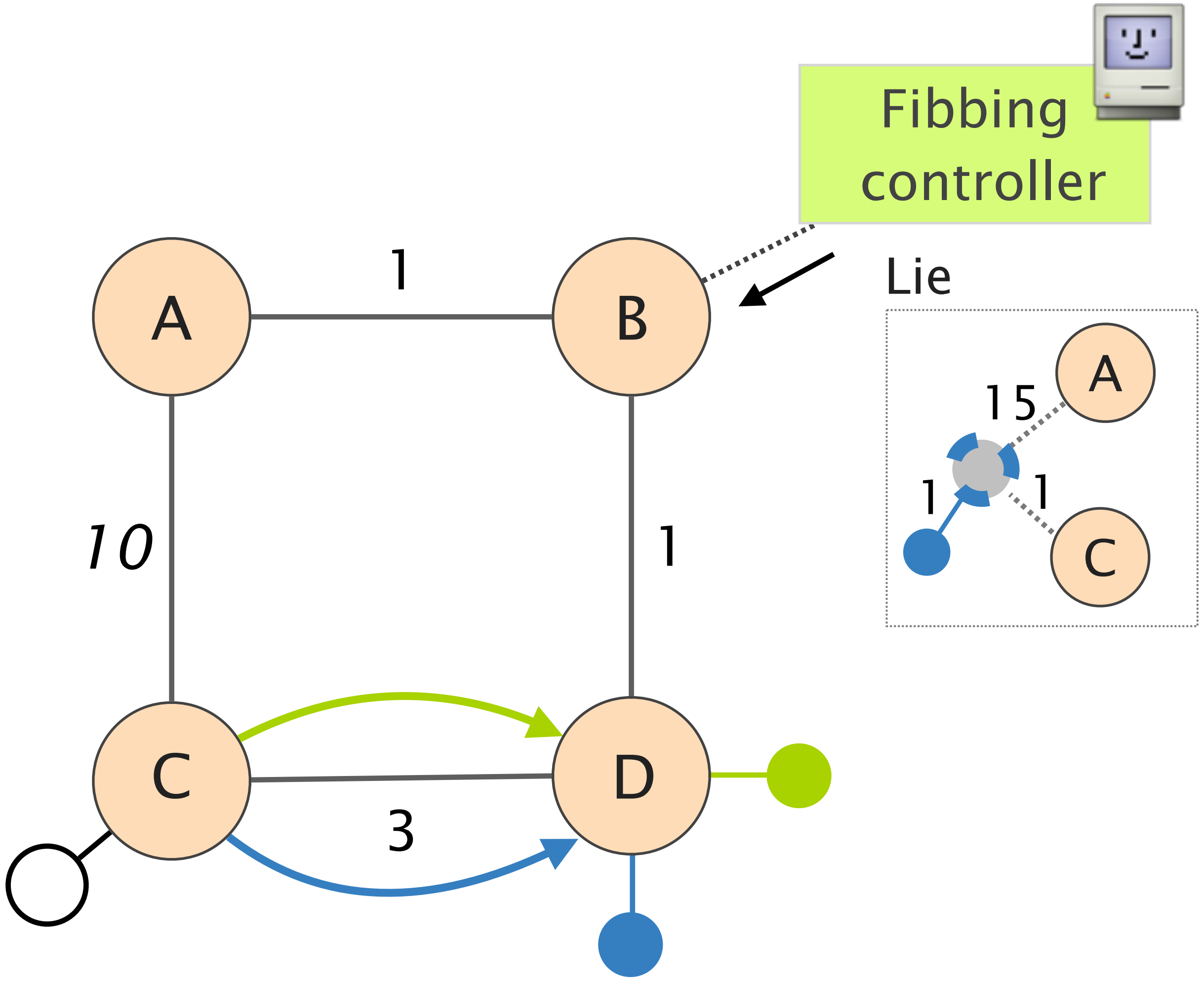
Let’s lie to the router



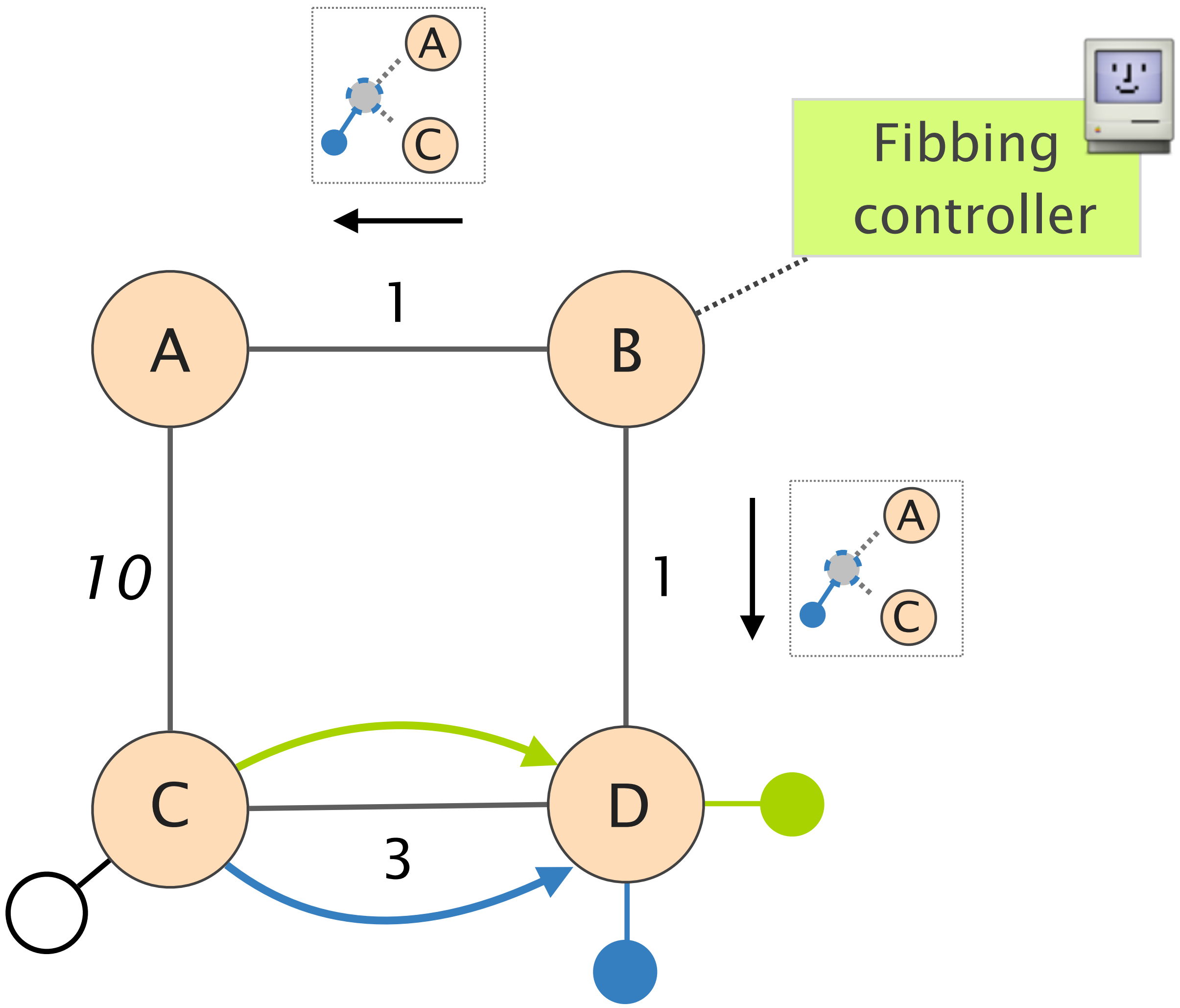
Let's lie to the router, by injecting fake nodes, links and destinations



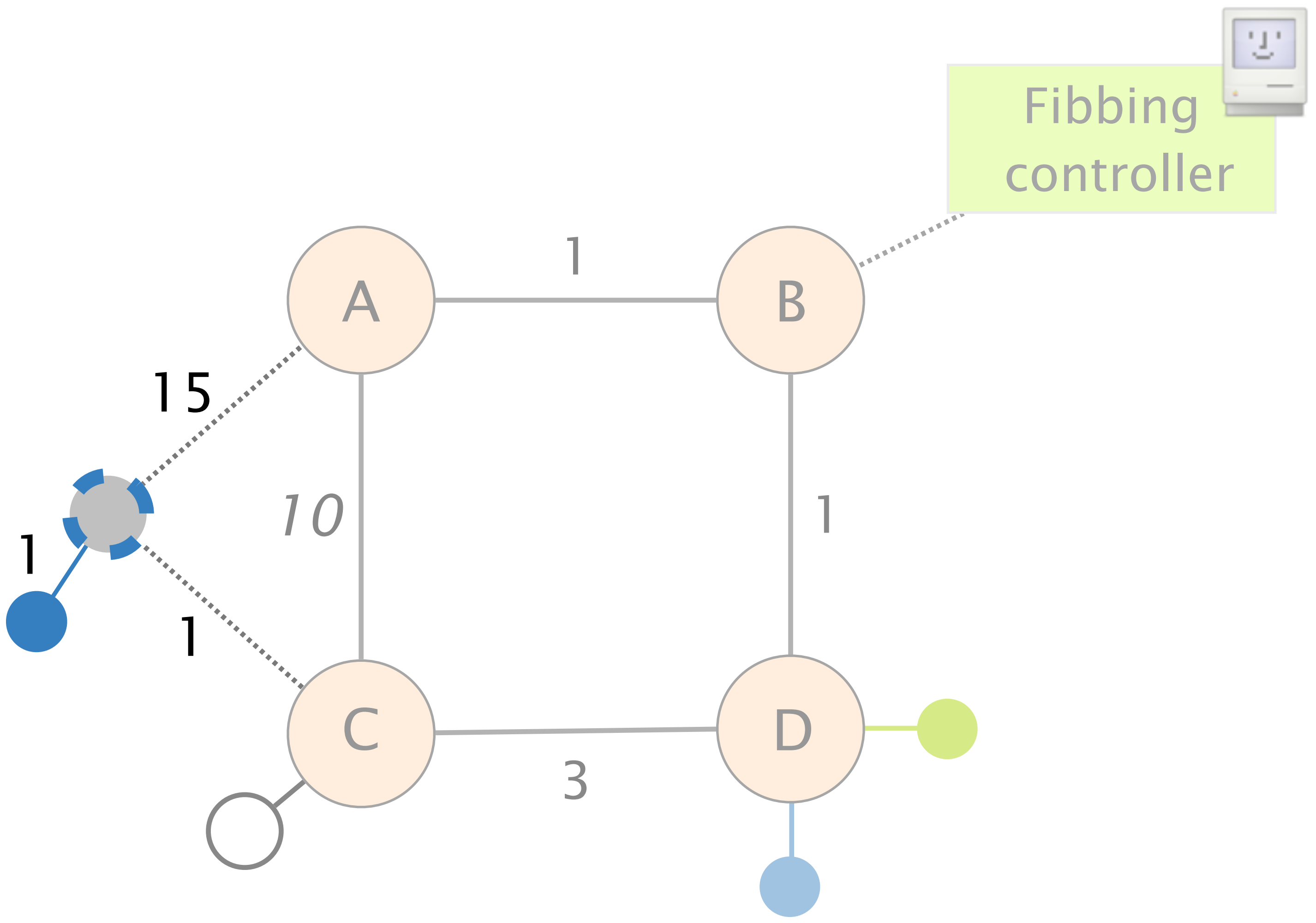
Let's **lie to the router**, by injecting fake nodes, links and destinations



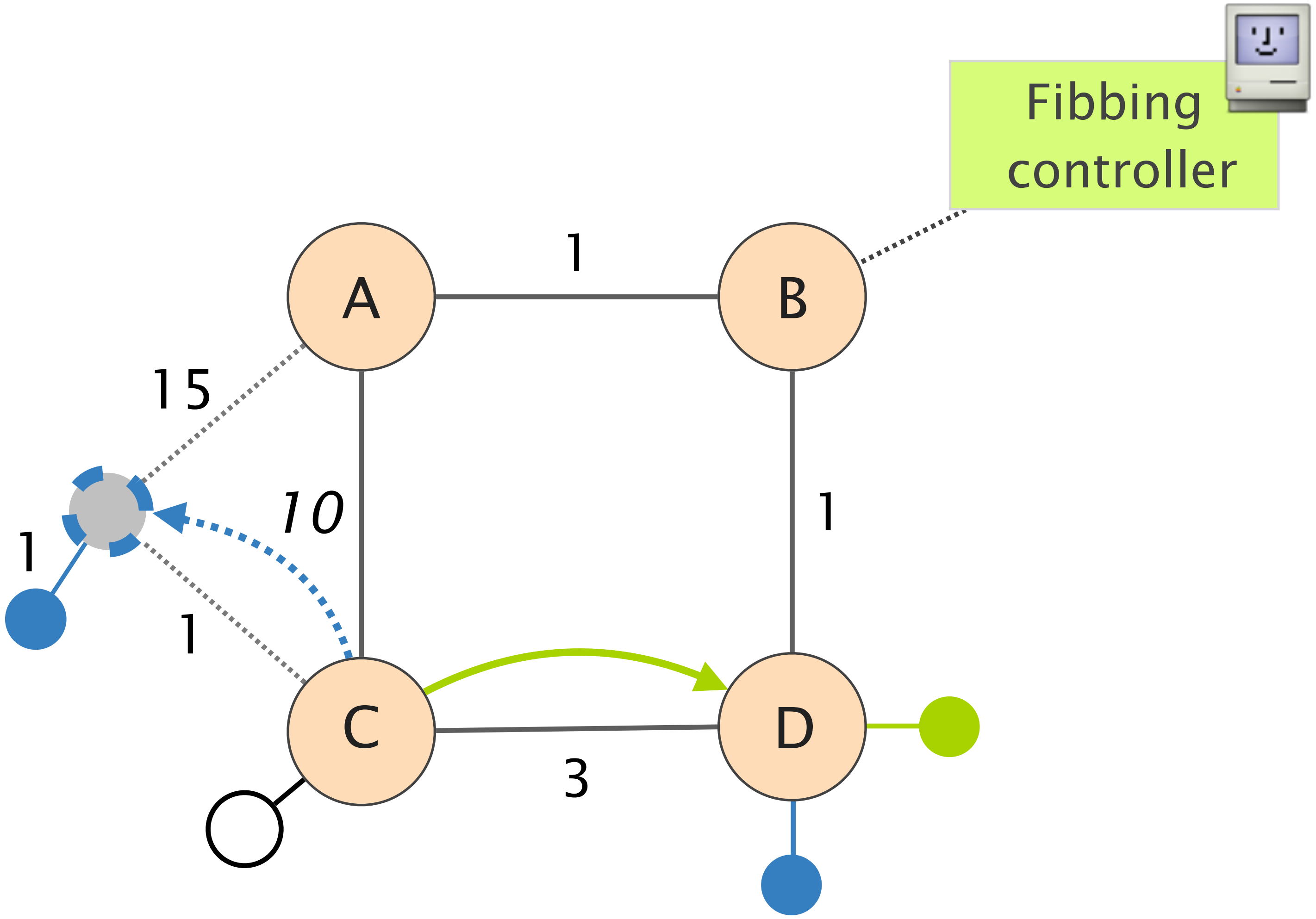
Lies are propagated network-wide by the protocol



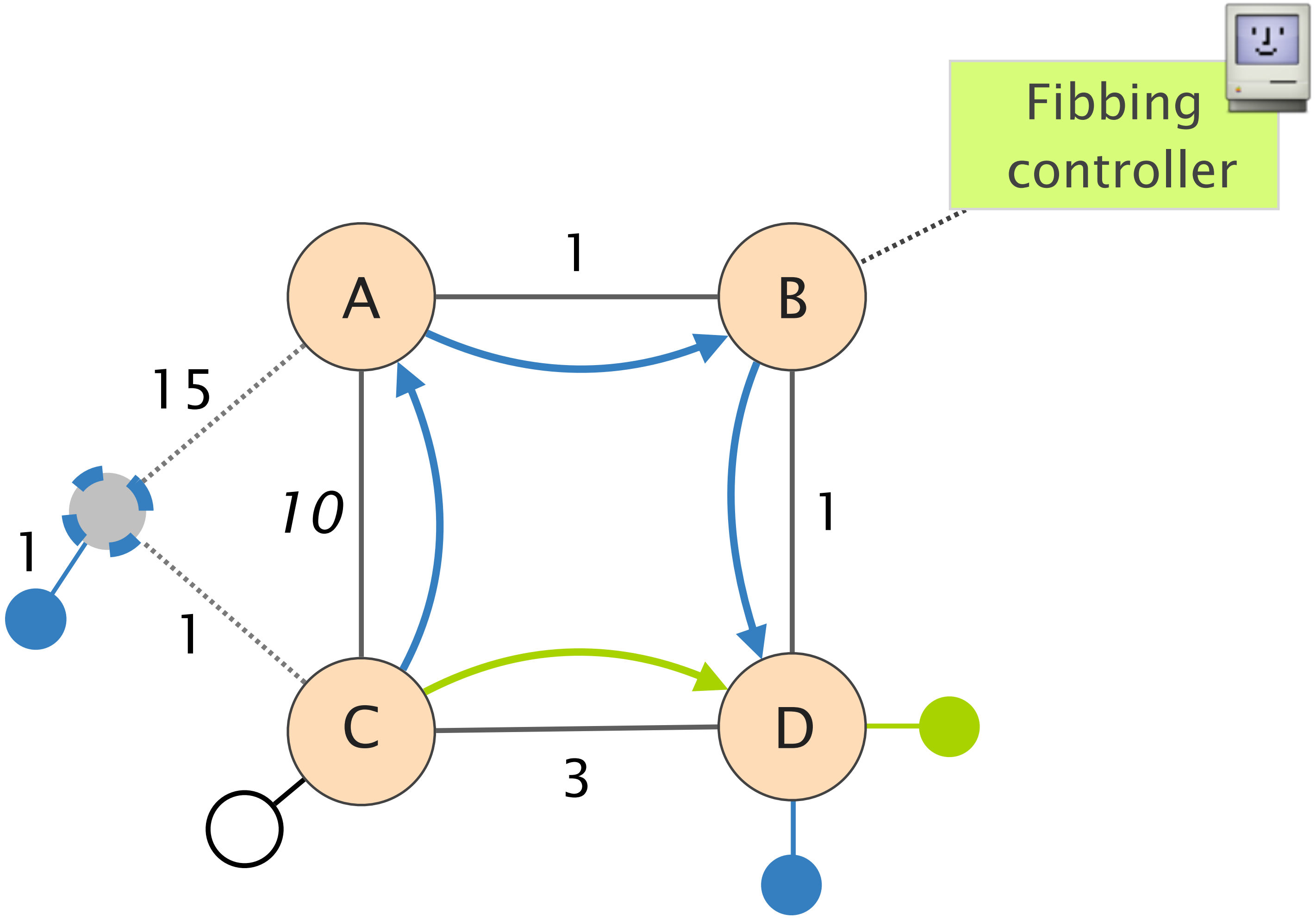
After the injection, this is the topology seen by all routers, on which they compute Dijkstra



Now, C prefers the virtual node (cost 2) to reach the blue destination…



As the virtual node does not really exist, actual traffic is *physically* sent to A



Fibbing is powerful

Fibbing is powerful

Theorem

Fibbing can program

any set of non-contradictory paths

Fibbing is powerful

Theorem

Fibbing can program
any set of non-contradictory paths

Fibbing is powerful

Theorem

Fibbing can program

any set of non-contradictory paths

- any path is **loop-free**

  (*e.g.*, [s1, a, b, a, d] is not possible)

- paths are **consistent**

  (*e.g.* [s1, a, b, d] and
  [s2, b, a, d] are inconsistent)

Fibbing scales

time

to compute lies

Augment topology within a sec.

space

# of lies

Augmented topologies are small. Much below what routers can support.

We implemented a fully-fledged Fibbing prototype and tested it against real routers

We implemented a fully-fledged Fibbing prototype and tested it against real routers

2 measurements

How many lies can a router sustain?

How long does it take to process a lie?

Existing routers can easily sustain Fibbing-induced load, even with huge topologies

# fake nodes	router memory (MB)	
1000	0.7	
5 000	6.8	
10 000	14.5	
50 000	76.0	
100 000	153	DRAM is cheap

Because it is entirely distributed, programming forwarding entries is fast

# fake nodes	installation time (s)	
1000	0.9	
5 000	4.5	
10 000	8.9	
50 000	44.7	
100 000	89.50	894.50 µs/entry

So… it’s done basically?

So… it's done basically?

No… far from it!

We want to create a momentum around Fibbing

Build applications on top of Fibbing

checkout www.fibbing.net (soon!)

Improve the Fibbing platform

e.g., fast (local) convergence, support for NFV

Build an OpenFlow to Fibbing interface

one network controller to rule them all

Fibbing is only a first step

One example where we successfully
abstracted the behavior of an existing technology

How can we abstract other technologies?

e.g., Telekinesis for L2 (SOSR'15)

How can we combine them—in a programmatic way

"classical" compilation problem

SDN research directions

Promising problems to invest time on

Go beyond OpenFlow

2 Secure SDN platforms

Incentivize deployment

Extend SDN reach

On the one hand,

SDN **reduces** the network attack surface

On the one hand,
SDN **reduces** the network attack surface

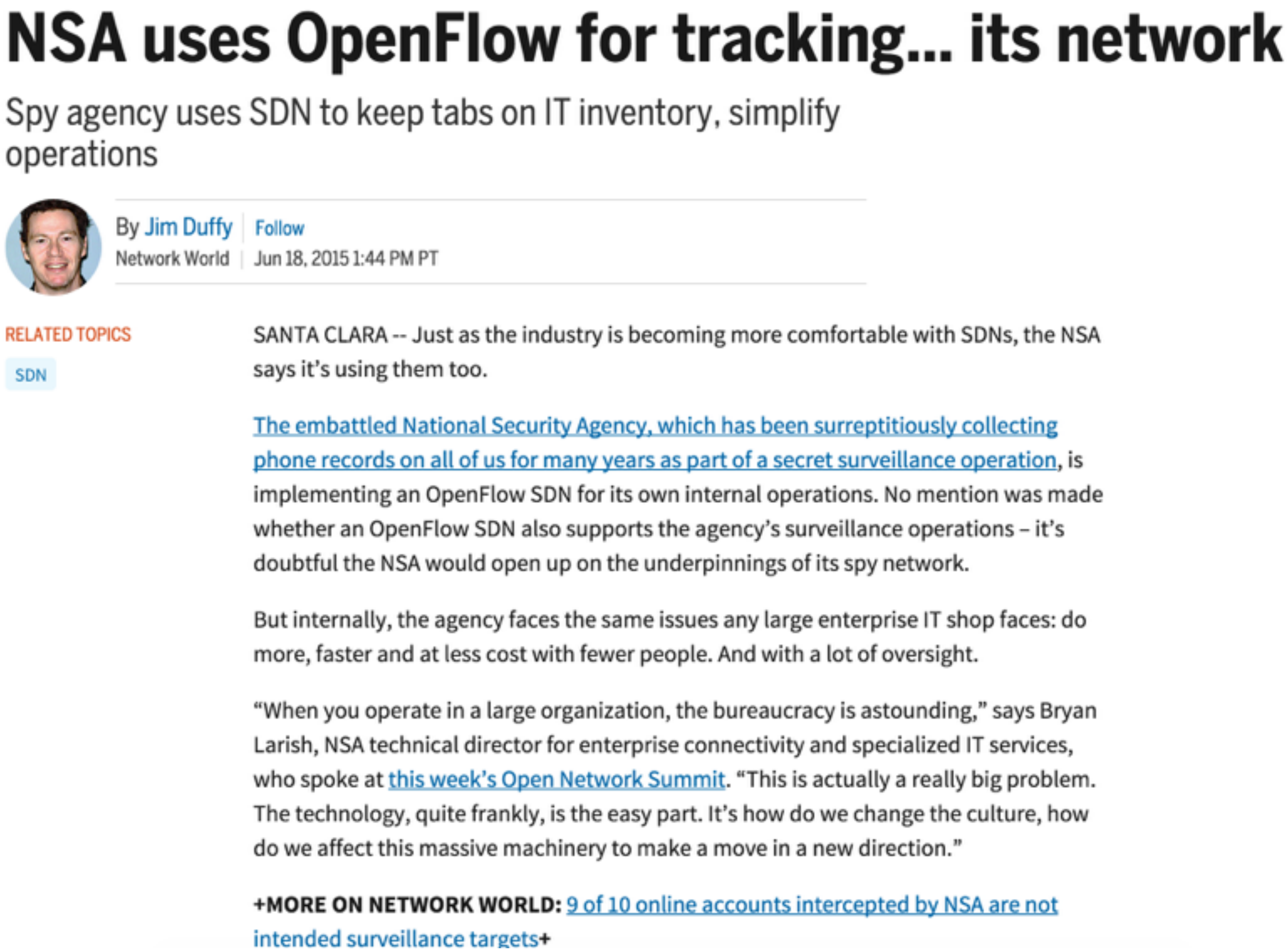
Traditional SDN

On the one hand,
SDN **reduces** the network attack surface

	Traditional	SDN
# code bases		
control		
visibility		
expressiveness		

On the one hand, SDN **reduces** the network attack surface

	Traditional	SDN
# code bases	dozens	1 (controller)
control	indirect	declarative
visibility	poor	network-wide
expressiveness	coarse-grained	fine-grained



NSA uses OpenFlow for tracking... its network

Spy agency uses SDN to keep tabs on IT inventory, simplify operations

By Jim Duffy | Follow
Network World | Jun 18, 2015 1:44 PM PT

SANTA CLARA -- Just as the industry is becoming more comfortable with SDNs, the NSA says it's using them too.

The embattled National Security Agency, which has been surreptitiously collecting phone records on all of us for many years as part of a secret surveillance operation, is implementing an OpenFlow SDN for its own internal operations. No mention was made whether an OpenFlow SDN also supports the agency's surveillance operations – it's doubtful the NSA would open up on the underpinnings of its spy network.

But internally, the agency faces the same issues any large enterprise IT shop faces: do more, faster and at less cost with fewer people. And with a lot of oversight.

"When you operate in a large organization, the bureaucracy is astounding," says Bryan Larish, NSA technical director for enterprise connectivity and specialized IT services, who spoke at this week's Open Network Summit. "This is actually a really big problem. The technology, quite frankly, is the easy part. It's how do we change the culture, how do we affect this massive machinery to make a move in a new direction."

+MORE ON NETWORK WORLD: 9 of 10 online accounts intercepted by NSA are not intended surveillance targets**+**

http://www.networkworld.com/article/2937787/sdn/nsa-uses-openflow-for-tracking-its-network.html

On the other hand,
SDN introduces new vectors of attacks

DDoS the controller

why kill a host if you can kill the network?

Hijack the controller

take control of the brain & the body

Hijack SDN applications

you say “yes”, I say “no”

Many novel research questions!

limit reactive app distributed controller	DDoS the controller why kill a host if you can kill the network?
protection & detection mechanisms	Hijack the controller take control of the brain & the body
authorization framework	Hijack SDN applications you say “yes”, I say “no”

SDN research directions

Promising problems to invest time on

Go beyond OpenFlow

Secure SDN platforms

3 Incentivize deployment

Extend SDN reach

To succeed, SDN-based technologies should possess at least 3 characteristics

Small investment

Low risk

High return

To succeed, SDN-based technologies should possess at least 3 characteristics

Small investment

provide benefits
under partial deployment
(ideally, with a single switch)

Low risk

High return

To succeed, SDN-based technologies should possess at least 3 characteristics

Small investment

Low risk

- require minimum changes to operational practices
- be compatible with existing technologies

High return

To succeed, SDN-based technologies should possess at least 3 characteristics

Small investment

Low risk

High return

solve a timely problem

Supercharged

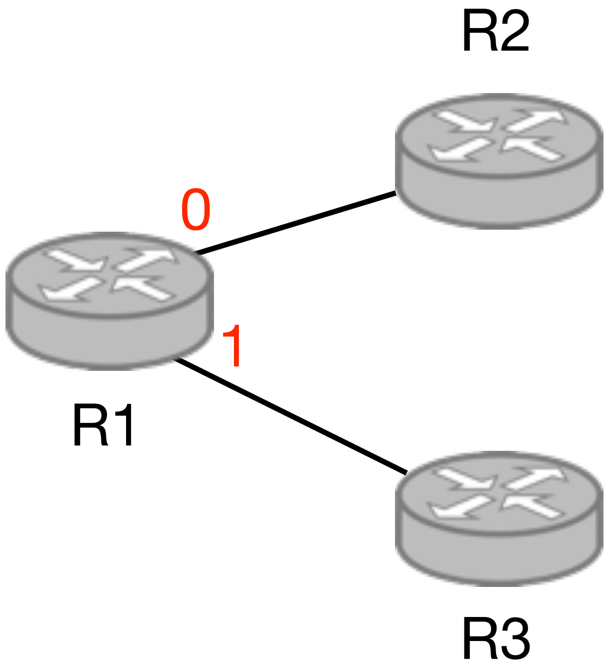
Supercharged

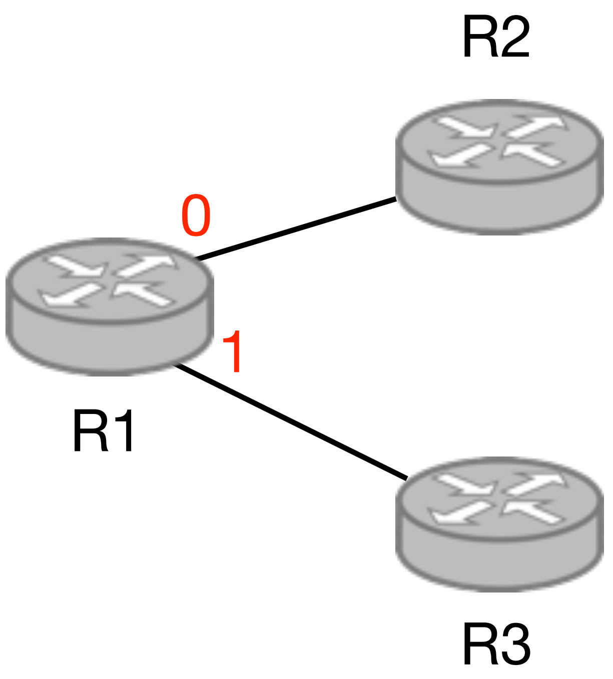
boost routers performance

by **combining** them with **SDN** devices

IP routers are pretty slow to converge upon link and node failures

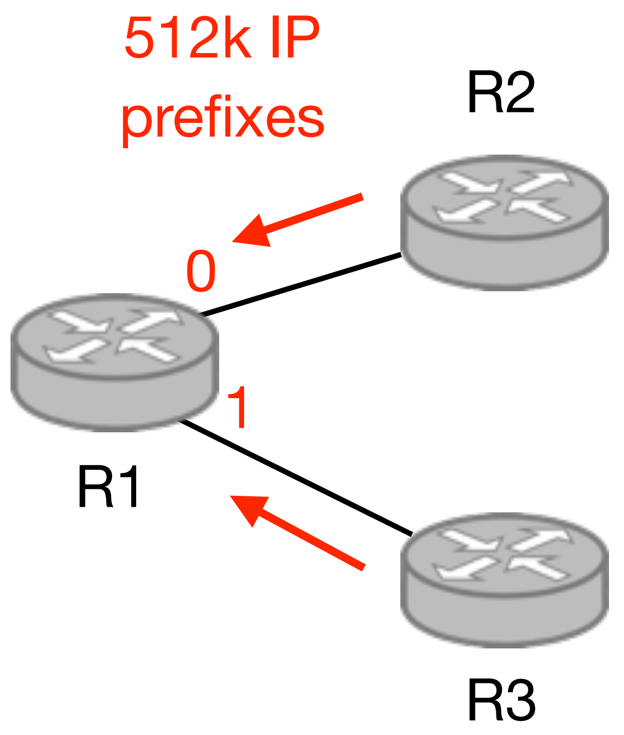
R1


R2
0
1
R1
R3



Provider #1 ($)
IP: 203.0.113.1
MAC: 01:aa

Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb



Provider #1 ($)

IP: 203.0.113.1

MAC: 01:aa

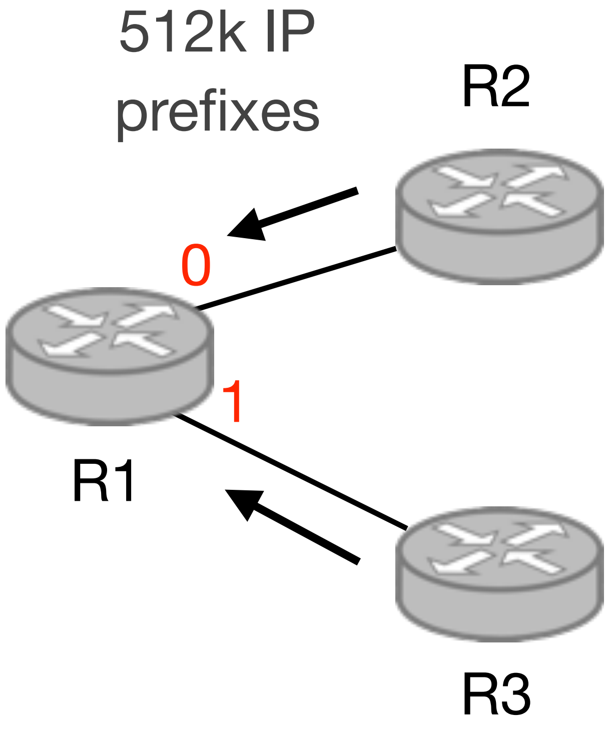
Provider #2 ($$)

IP: 198.51.100.2

MAC: 02:bb

R1’s Forwarding Table

prefix	Next-Hop



Provider #1 ($)
IP: 203.0.113.1
MAC: 01:aa

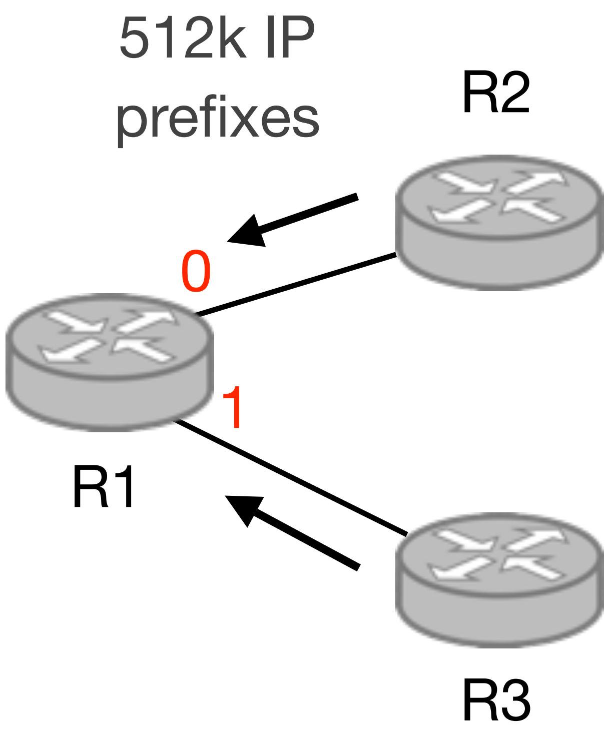
Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb

All 512k entries point to R2

because it is cheaper

R1's Forwarding Table

	prefix	Next-Hop
1	1.0.0.0/24	(01:aa, 0)
2	1.0.1.0/16	(01:aa, 0)
…	…	…
256k	100.0.0.0/8	(01:aa, 0)
…	…	…
512k	200.99.0.0/24	(01:aa, 0)



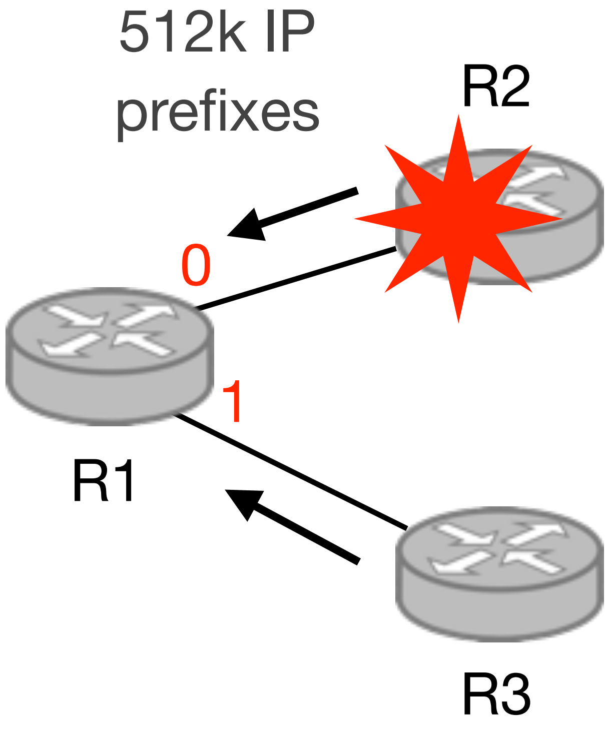
Provider #1 ($)
IP: 203.0.113.1
MAC: 01:aa

Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb

Upon failure of R2,
all 512k entries have to be updated

R1's Forwarding Table

	prefix	Next-Hop
1	1.0.0.0/24	(01:aa, 0)
2	1.0.1.0/16	(01:aa, 0)
…	…	…
256k	100.0.0.0/8	(01:aa, 0)
…	…	…
512k	200.99.0.0/24	(01:aa, 0)



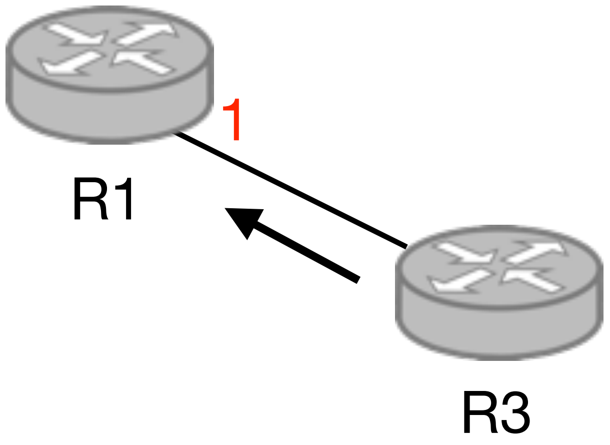
Provider #1 ($)
IP: 203.0.113.1
MAC: 01:aa

Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb

Upon failure of R2,
all 512k entries have to be updated

R1’s Forwarding Table

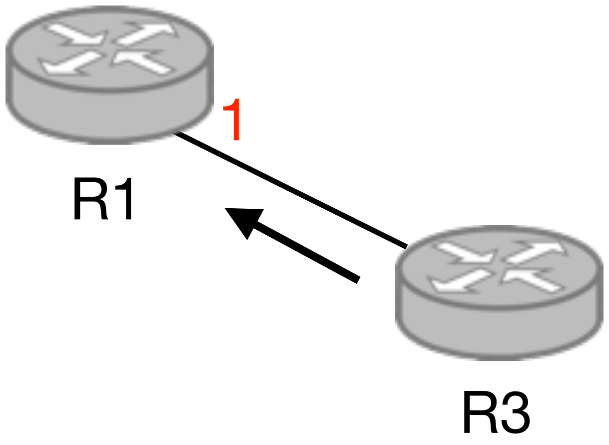
	prefix	Next-Hop
1	1.0.0.0/24	(01:aa, 0)
2	1.0.1.0/16	(01:aa, 0)
…	…	…
256k	100.0.0.0/8	(01:aa, 0)
…	…	…
512k	200.99.0.0/24	(01:aa, 0)



Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb

R1’s Forwarding Table

	prefix	Next-Hop
1	1.0.0.0/24	(02:bb, 1)
2	1.0.1.0/16	(01:aa, 0)
…	…	…
256k	100.0.0.0/8	(01:aa, 0)
…	…	…
512k	200.99.0.0/24	(01:aa, 0)



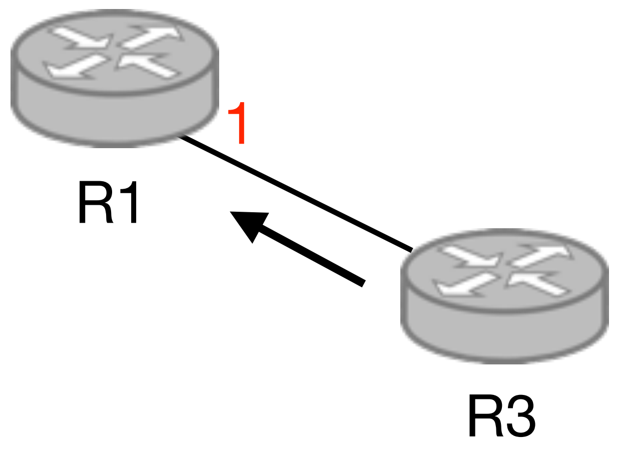
Provider #2 ($$)

IP: 198.51.100.2

MAC: 02:bb

R1’s Forwarding Table

	prefix	Next-Hop
1	1.0.0.0/24	(02:bb, 1)
2	1.0.1.0/16	(02:bb, 1)
…	…	…
256k	100.0.0.0/8	(01:aa, 0)
…	…	…
512k	200.99.0.0/24	(01:aa, 0)



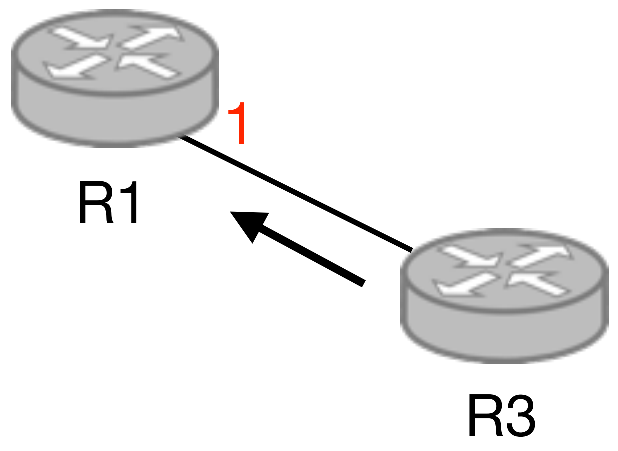
Provider #2 ($$)

IP: 198.51.100.2

MAC: 02:bb

R1's Forwarding Table

	prefix	Next-Hop
1	1.0.0.0/24	(02:bb, 1)
2	1.0.1.0/16	(02:bb, 1)
…	…	…
256k	100.0.0.0/8	(02:bb, 1)
…	…	…
512k	200.99.0.0/24	(01:aa, 0)



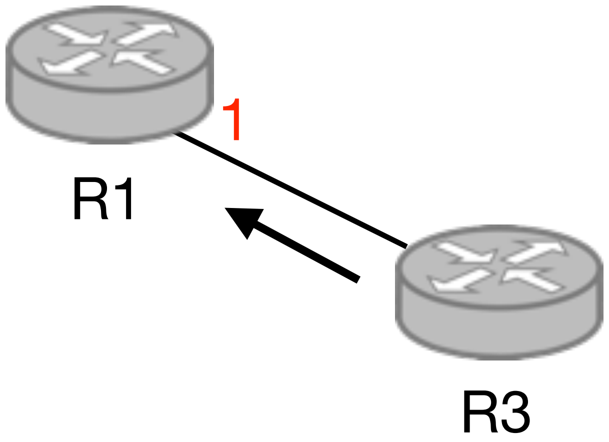
Provider #2 ($$)

IP: 198.51.100.2

MAC: 02:bb

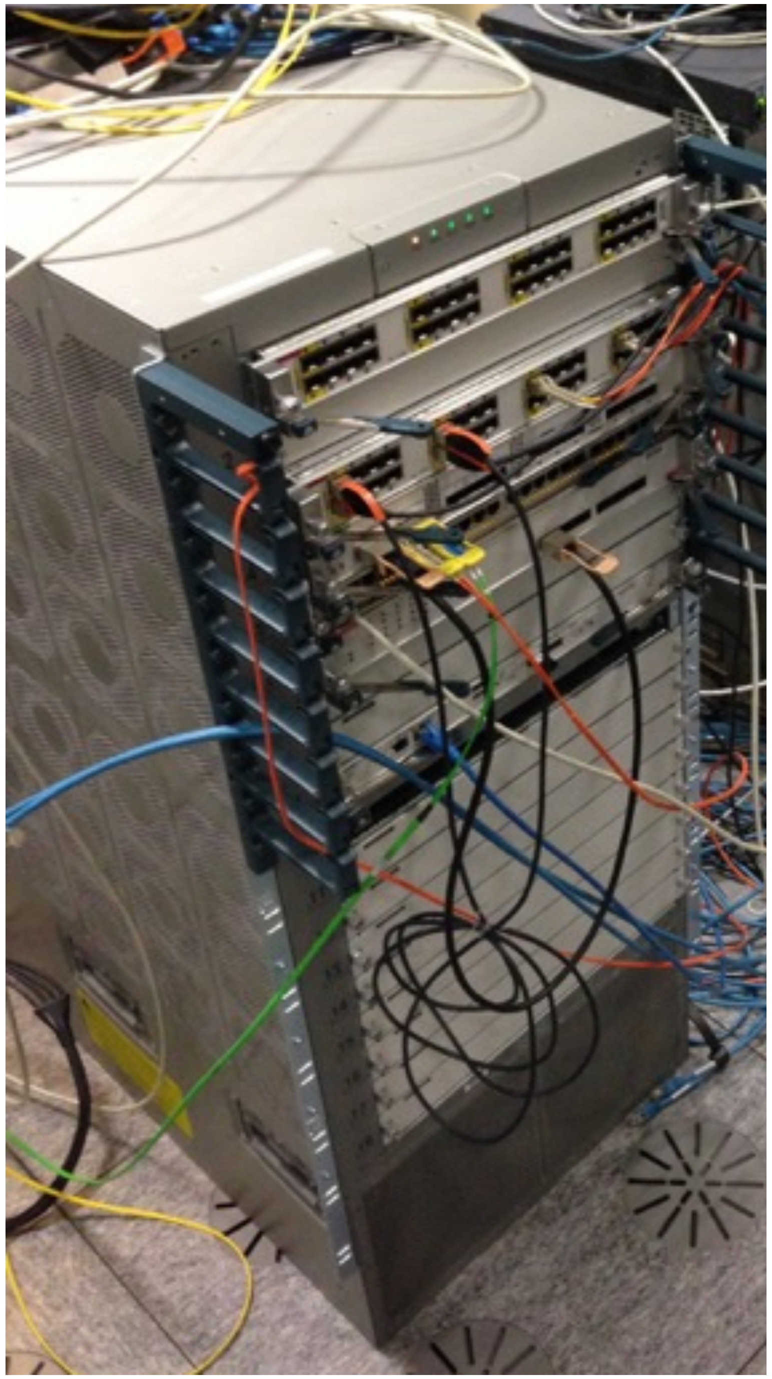
R1’s Forwarding Table

	prefix	Next-Hop
1	1.0.0.0/24	(02:bb, 1)
2	1.0.1.0/16	(02:bb, 1)
…	…	…
256k	100.0.0.0/8	(02:bb, 1)
…	…	…
512k	200.99.0.0/24	(02:bb, 1)



Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb

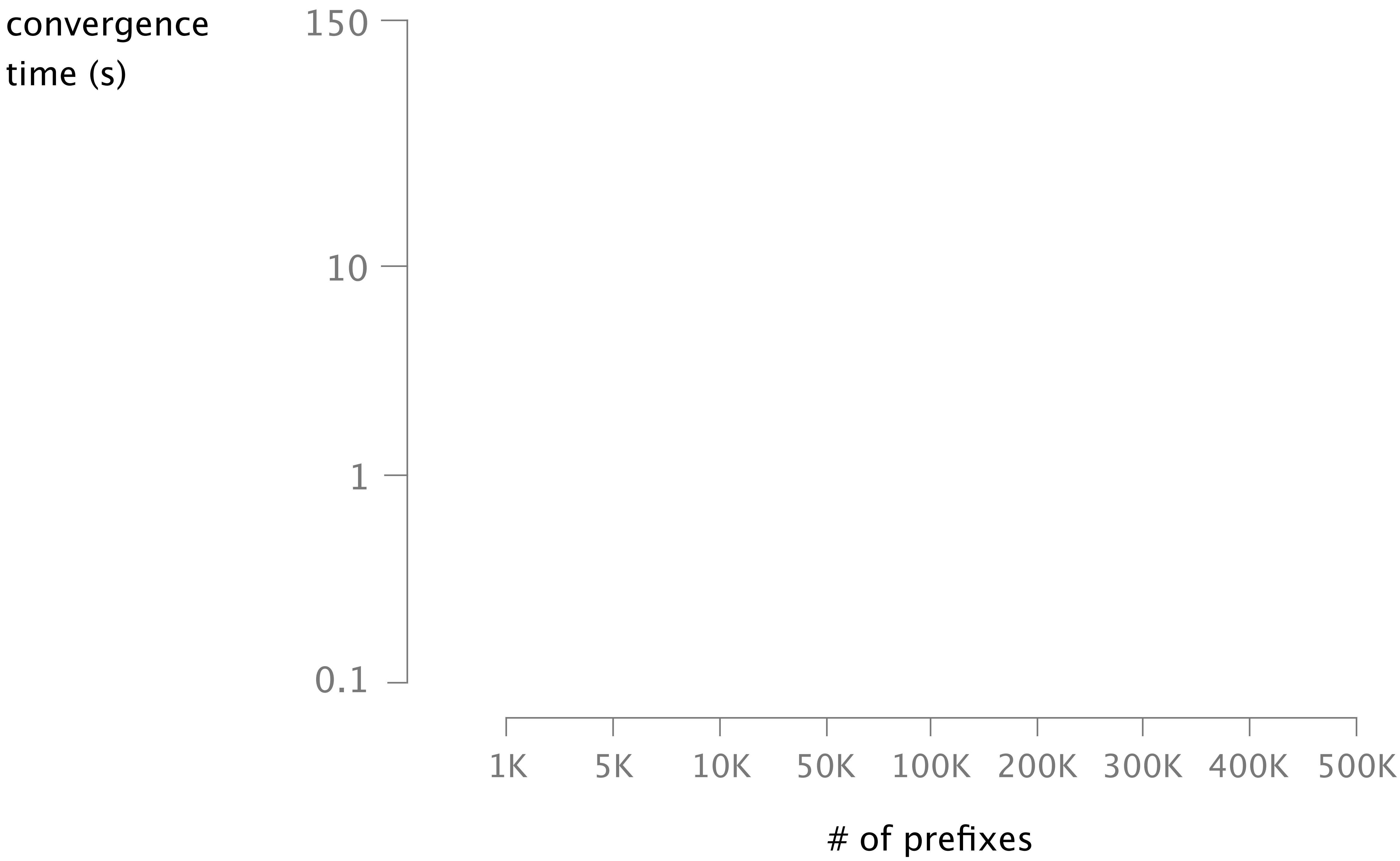
We measured how long it takes in our home network

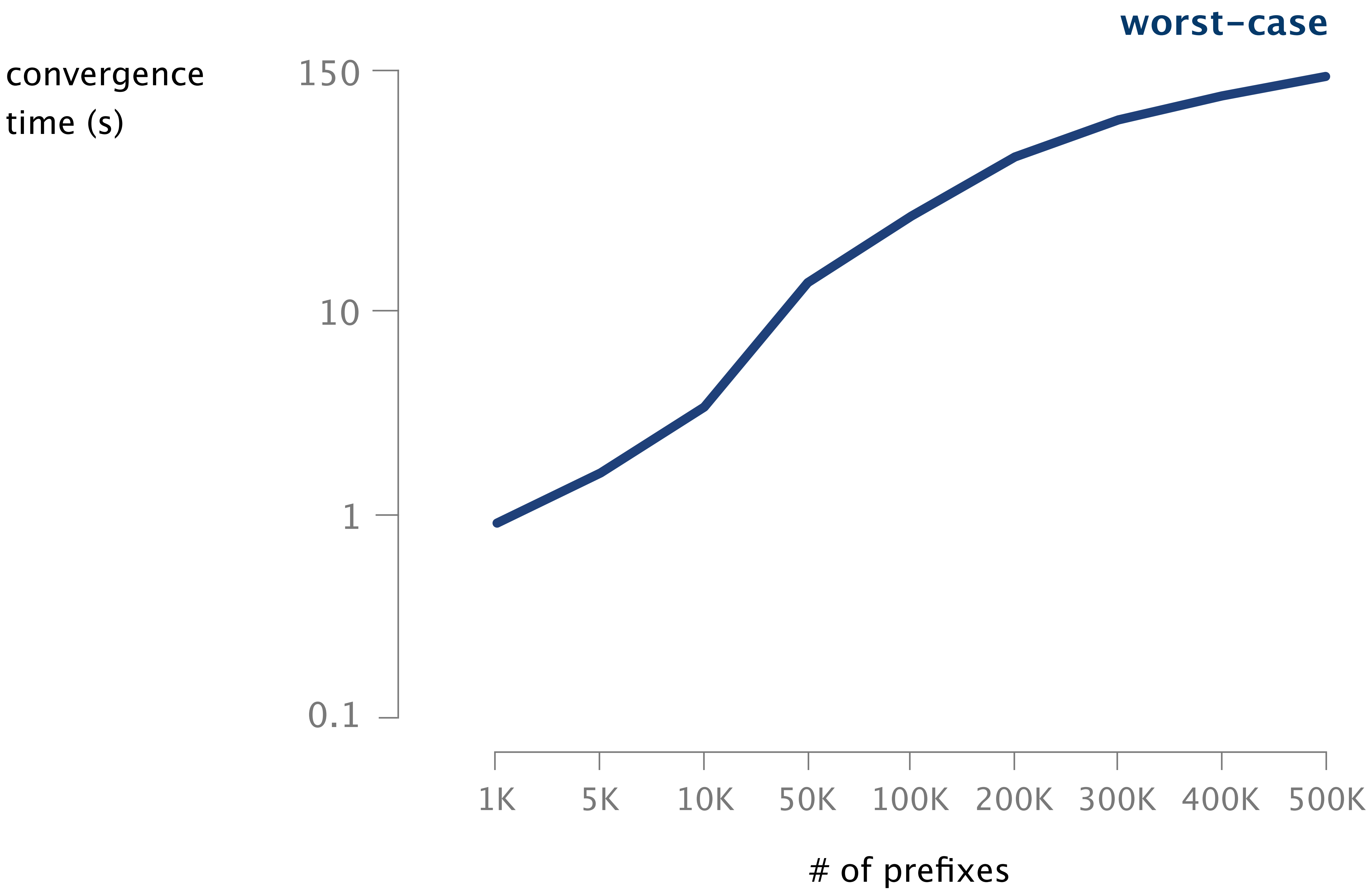


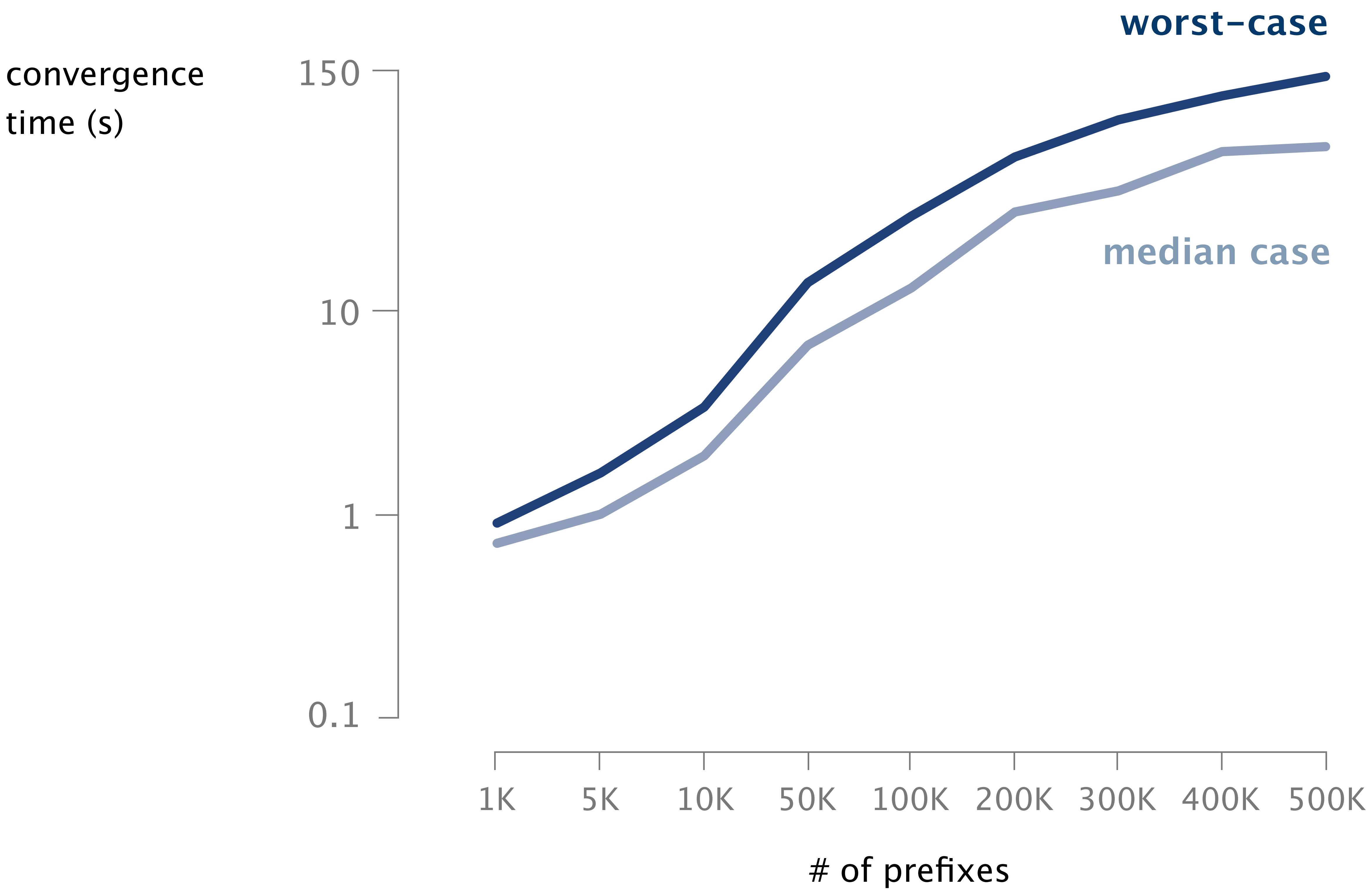
Cisco Nexus 9k

ETH recent routers

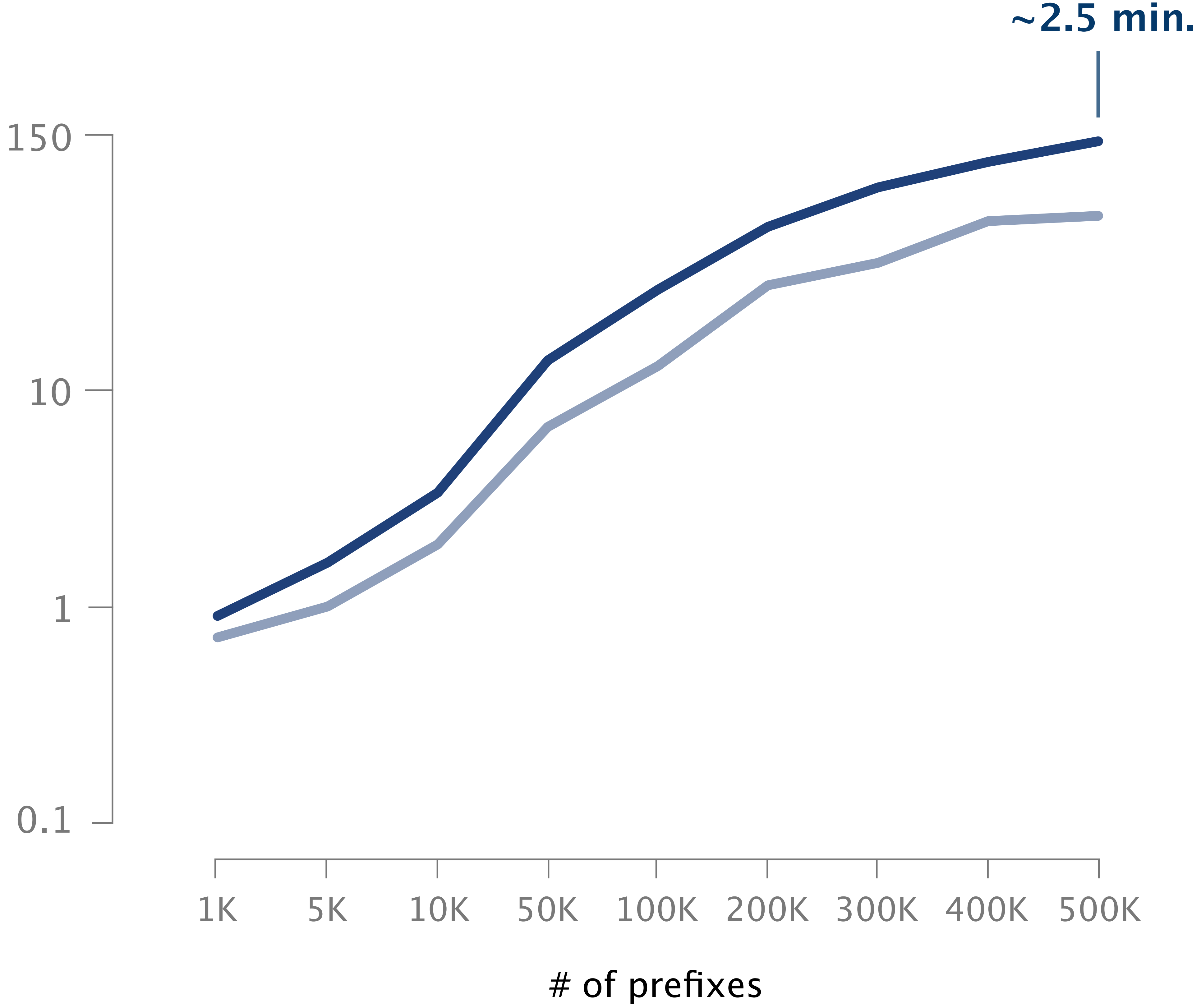
25 deployed


convergence
time (s)
150
10
1
0.1
1K
5K
10K
50K
100K
200K
300K
400K
500K
# of prefixes


worst-case
convergence
time (s)
150
10
1
0.1
1K
5K
10K
50K
100K
200K
300K
400K
500K
# of prefixes


worst-case
convergence
time (s)
150
10
1
0.1
median case
1K
5K
10K
50K
100K
200K
300K
400K
500K
# of prefixes

Traffic can be lost for several minutes



The problem is that forwarding tables are flat

Entries do not share any information

even if they are identical

Upon failure, all of them have to be updated

inefficient, but also unnecessary

The problem is that forwarding tables are flat

Entries do not share any information

even if they are identical

Upon failure, all of them have to be updated

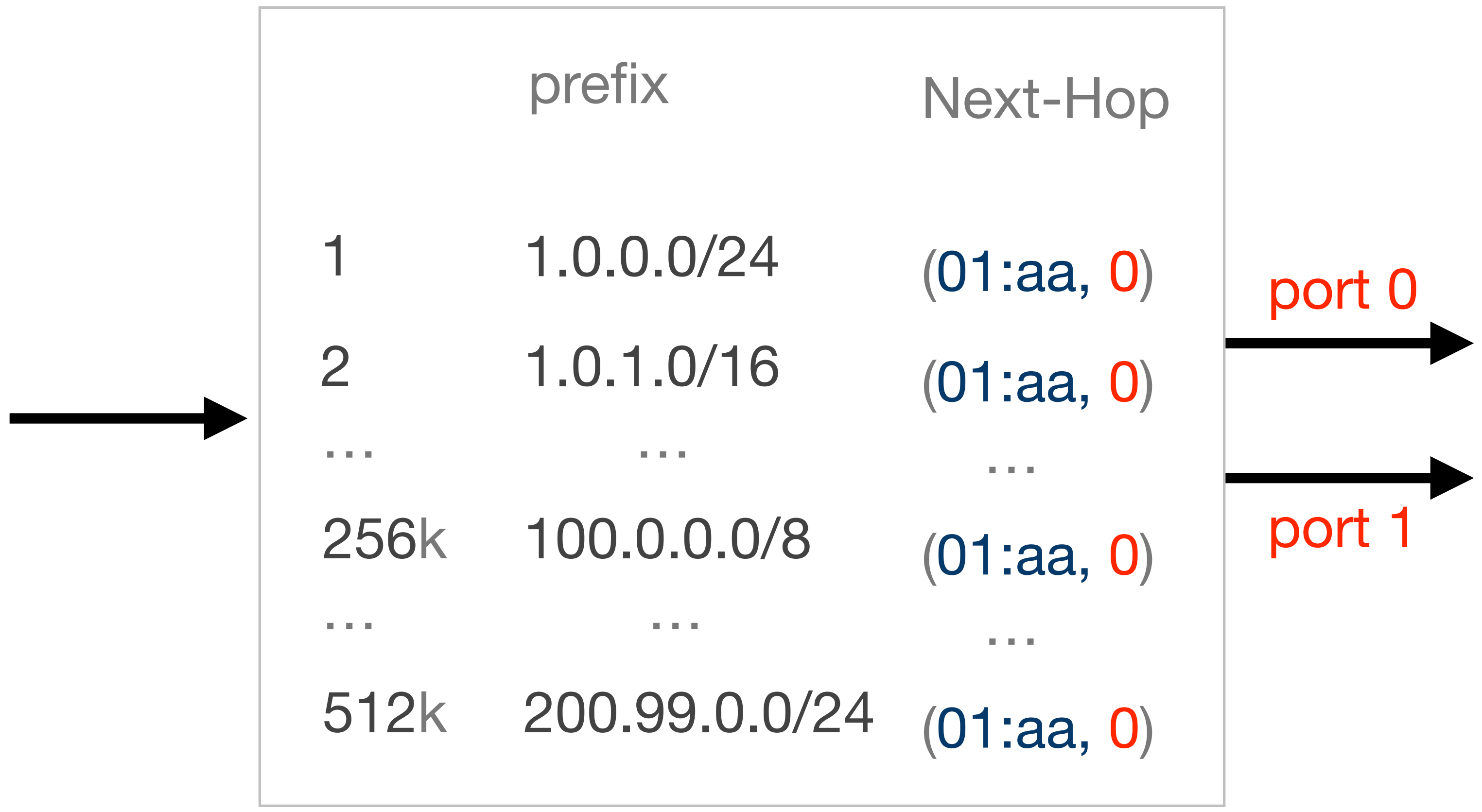
inefficient, but also unnecessary

Solution: introduce a hierarchy

as with any problem in CS…

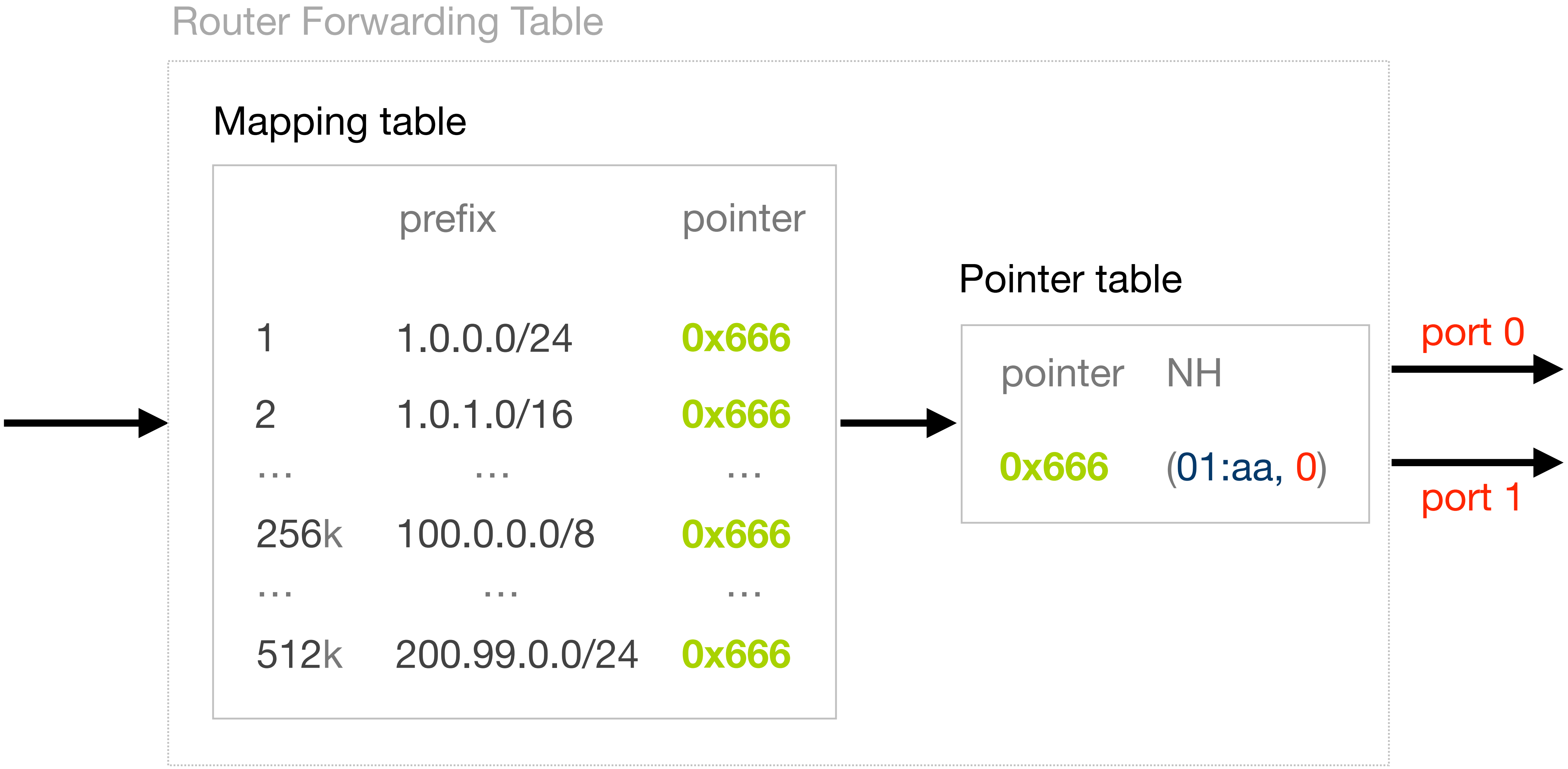
replace this…

Router Forwarding Table



	prefix	Next-Hop
1	1.0.0.0/24	(01:aa, 0)
2	1.0.1.0/16	(01:aa, 0)
…	…	…
256k	100.0.0.0/8	(01:aa, 0)
…	…	…
512k	200.99.0.0/24	(01:aa, 0)

… with that



Router Forwarding Table

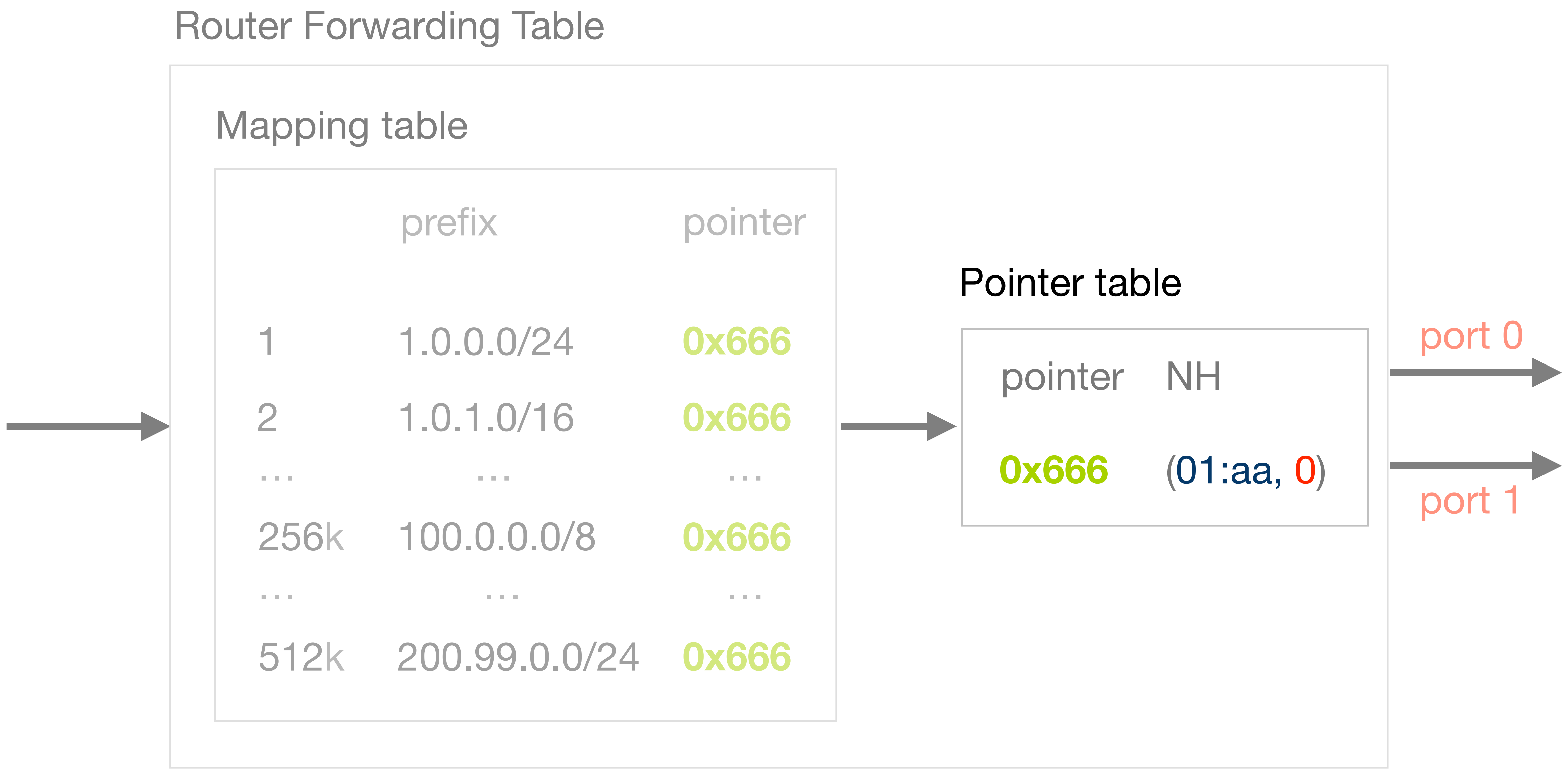
Mapping table

	prefix	pointer
1	1.0.0.0/24	0x666
2	1.0.1.0/16	0x666
…	…	…
256k	100.0.0.0/8	0x666
…	…	…
512k	200.99.0.0/24	0x666

Pointer table

pointer	NH
0x666	(01:aa, 0)

Upon failures, we update the pointer table



Router Forwarding Table

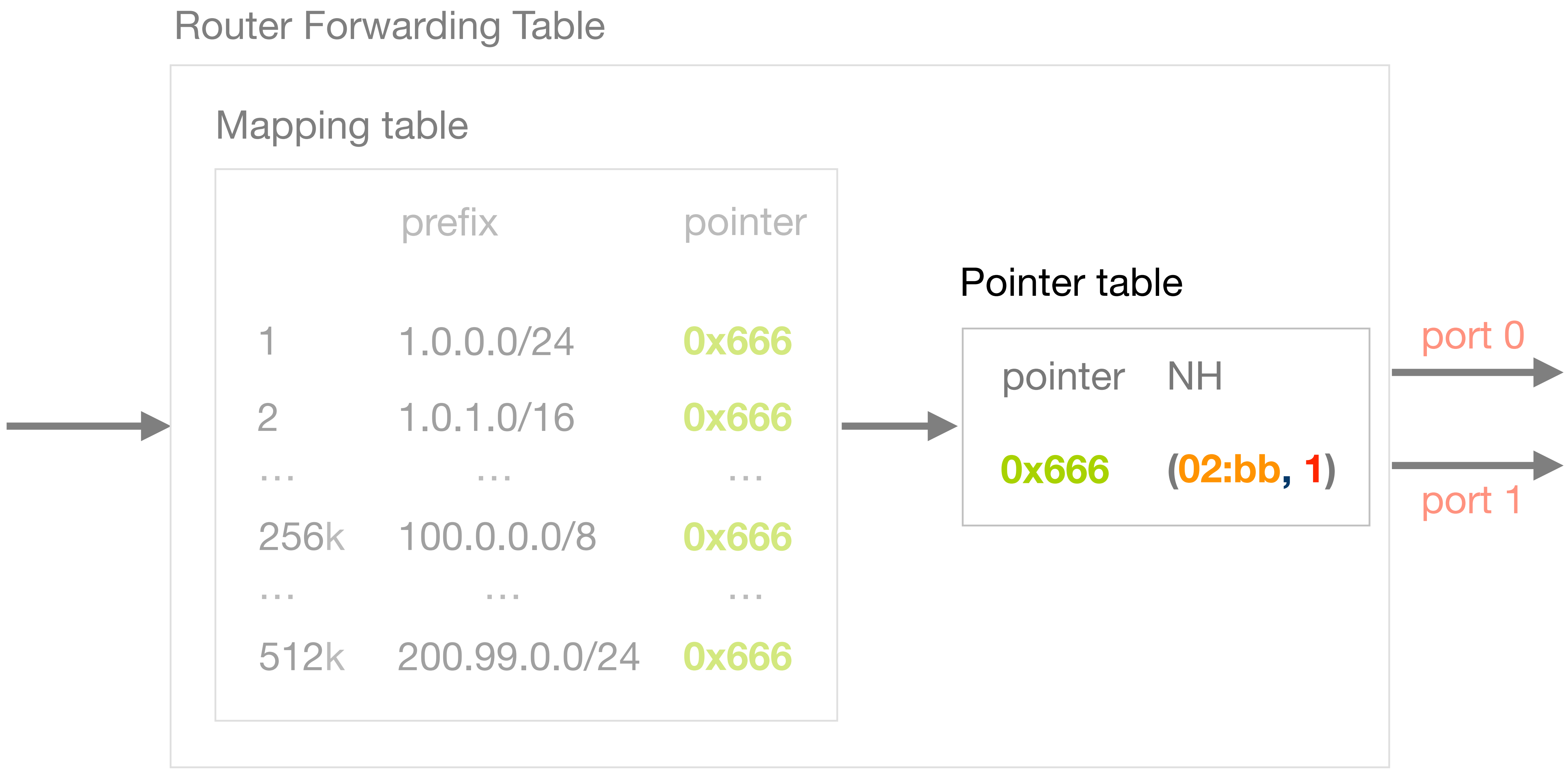
Mapping table

	prefix	pointer
1	1.0.0.0/24	0x666
2	1.0.1.0/16	0x666
…	…	…
256k	100.0.0.0/8	0x666
…	…	…
512k	200.99.0.0/24	0x666

Pointer table

pointer	NH
0x666	(01:aa, 0)

Here, we only need to do one update



Router Forwarding Table

Mapping table

	prefix	pointer
1	1.0.0.0/24	0x666
2	1.0.1.0/16	0x666
…	…	…
256k	100.0.0.0/8	0x666
…	…	…
512k	200.99.0.0/24	0x666

Pointer table

pointer	NH
0x666	(02:bb, 1)

Nowadays, only high-end routers have hierarchical forwarding table

Expensive

by orders of magnitude

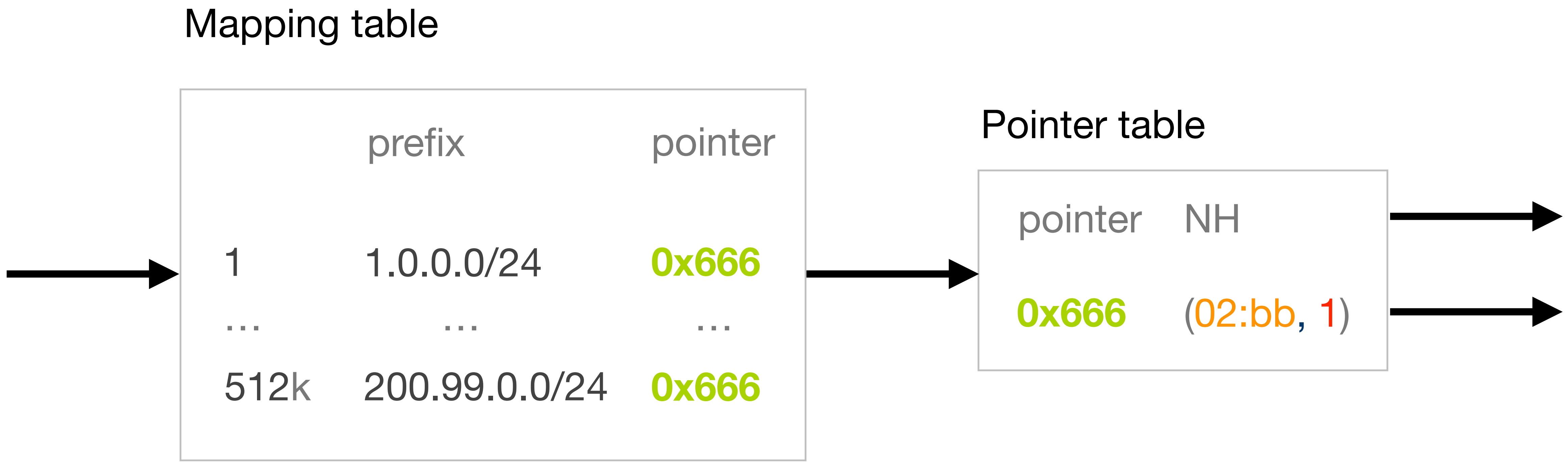
Limited availability

only a few vendors, on few models

Limited benefits

of fast convergence, if not used network-wide

We can build a hierarchical table



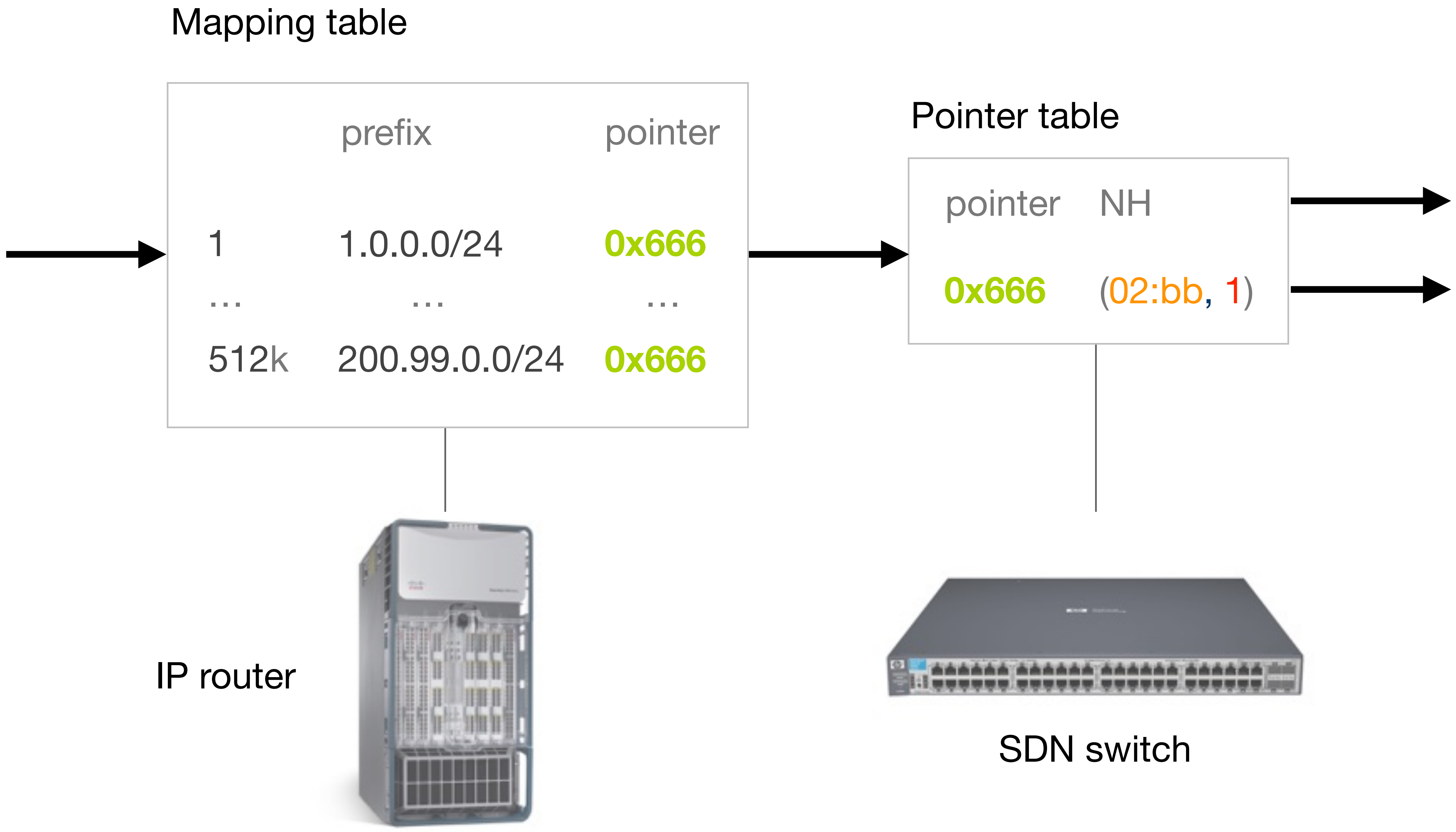
Mapping table

	prefix	pointer
1	1.0.0.0/24	0x666
…	…	…
512k	200.99.0.0/24	0x666

Pointer table

pointer	NH
0x666	(02:bb, 1)

We can build a hierarchical table
using two adjacent devices



Mapping table

	prefix	pointer
1	1.0.0.0/24	0x666
…	…	…
512k	200.99.0.0/24	0x666

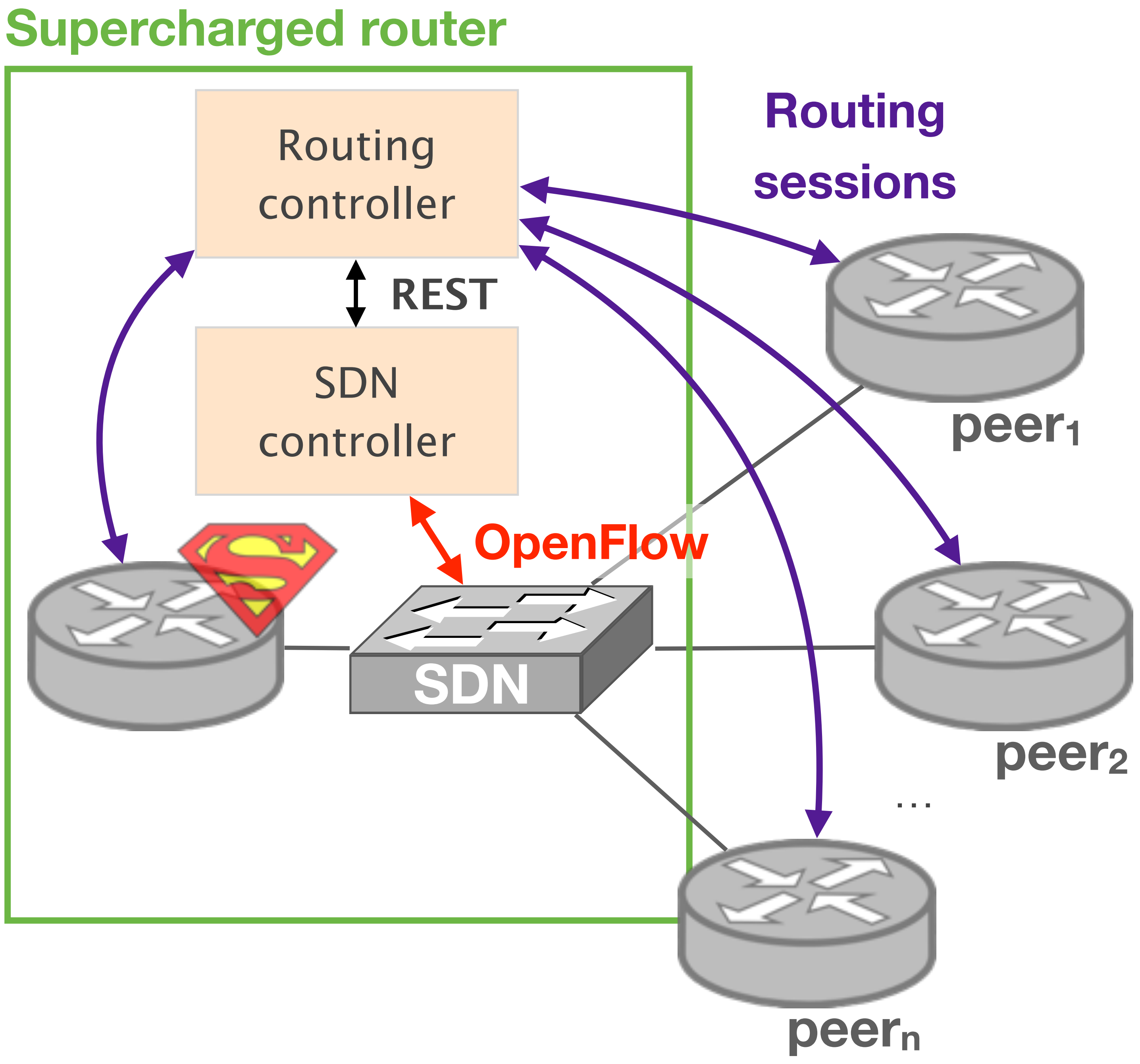
Pointer table

pointer	NH
0x666	(02:bb, 1)

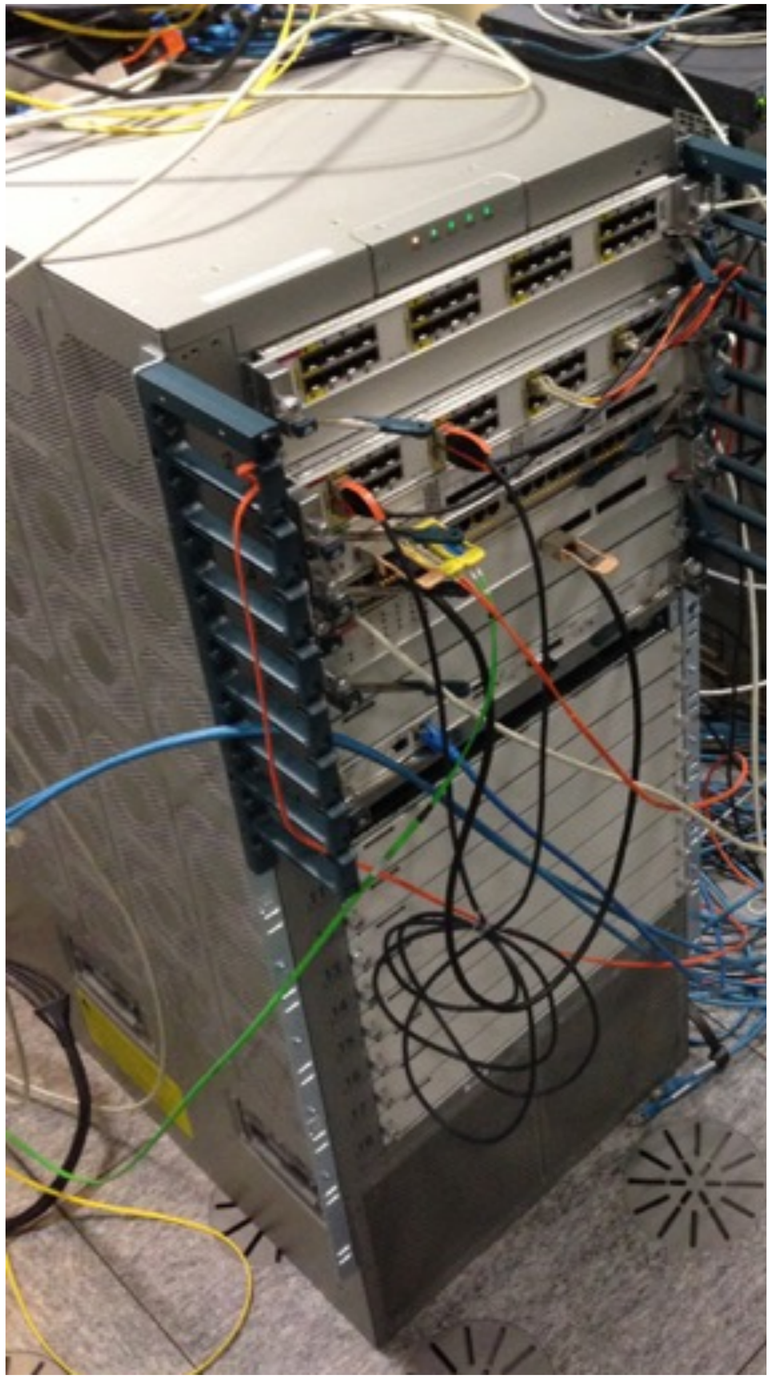
IP router

SDN switch

We have implemented a fully-functional “router supercharger”



We used it to supercharge the same router as before



Cisco Nexus 9k

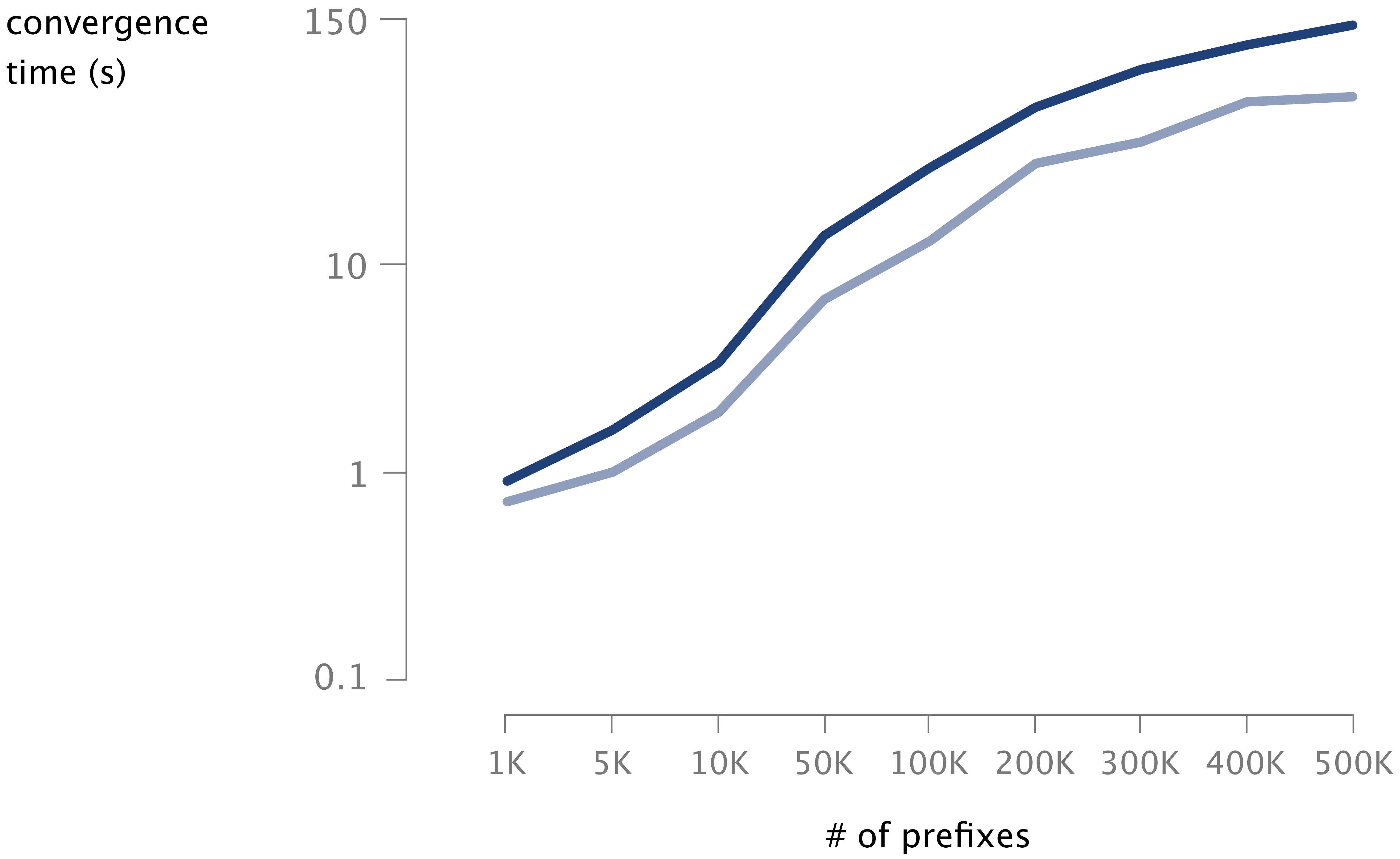
ETH recent routers

25 deployed

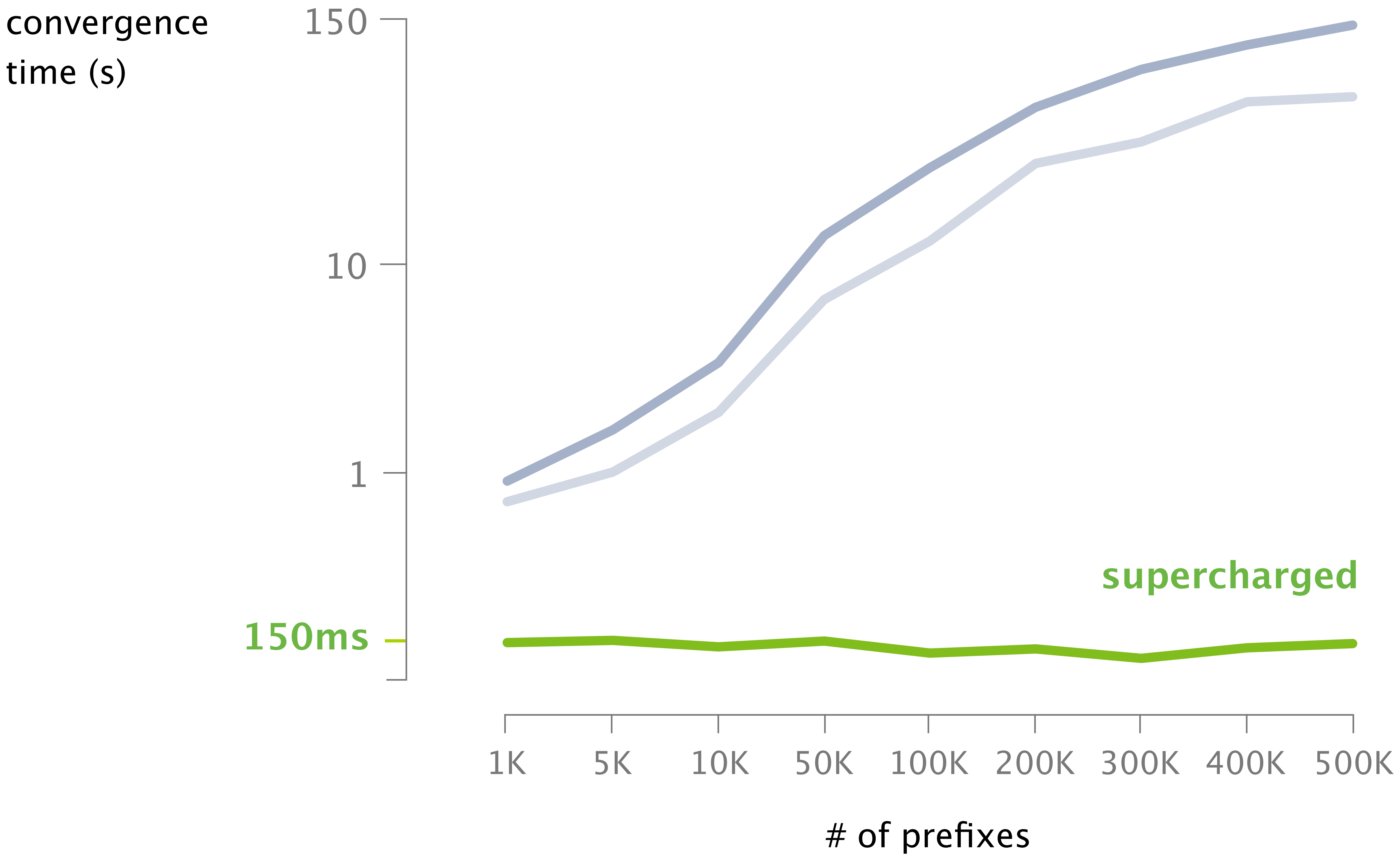
+ (old) SDN HP switch

~2k$ cost

While the router took more than 2 min to converge in the worst-case



The supercharged router systematically converged within 150ms



Other aspects of a router can be supercharged

- memory size

  offload to SDN if no local forwarding entry

- load-balancing

  monitor & overwrite poor routers decisions

- monitoring

  precise, micro-flow based measurements

SDN research directions

Promising problems to invest time on

Go beyond OpenFlow

Secure SDN platforms

Incentivize deployment

4 Extend SDN reach

So far, SDN reach has been limited to few network types

So far, SDN reach has been limited to few network types

Data-Center network

Cellular network

Wide-Area network

Enterprise network

There are many more terrain to conquer!

Data-Center network

Cellular network

Wide-Area network

Enterprise network

On-chip network

Campus network

Access network

Transit network

Today SDN targeted the operation of switches *within* a single domain

Tomorrow Let’s bring SDN to the Internet

Internet ❤ SDN

How do you deploy SDN in a network composed of 50,000 subnetworks?

How do you deploy SDN in a network composed of 50,000 subnetworks?

Well, you don’t …

Instead, you aim at finding locations where deploying SDN can have the most impact

Instead, you aim at finding locations where deploying SDN can have the most impact

Deploy SDN in locations that

- connect a large number of networks
- carry a large amount of traffic
- are opened to innovation

Internet eXchange Points (IXP) meet all the criteria

Deploy SDN in locations that

- connect a large number of networks
- carry a large amount of traffic
- are opened to innovation

AMS-IX

721 networks

3.7 Tb/s (peak)

BGP Route Server

Mobile peering

Open peering…

https://www.ams-ix.net

A single deployment can have a large impact

Deploy SDN in locations that

- connect a large number of networks
- carry a large amount of traffic
- are opened to innovation

AMS-IX

721 networks

3.7 Tb/s (peak)

BGP Route Server

Mobile peering

Open peering…

https://www.ams-ix.net

@SIGCOMM'14

SDX = SDN + IXP

http://sdx.cs.princeton.edu/

SDX = SDN + IXP

Augment the IXP data-plane with SDN capabilities

keeping default forwarding and routing behavior

Enable fine-grained inter domain policies

bringing new features while simplifying operations

SDX = SDN + IXP

Augment the IXP data-plane with SDN capabilities

keeping default forwarding and routing behavior

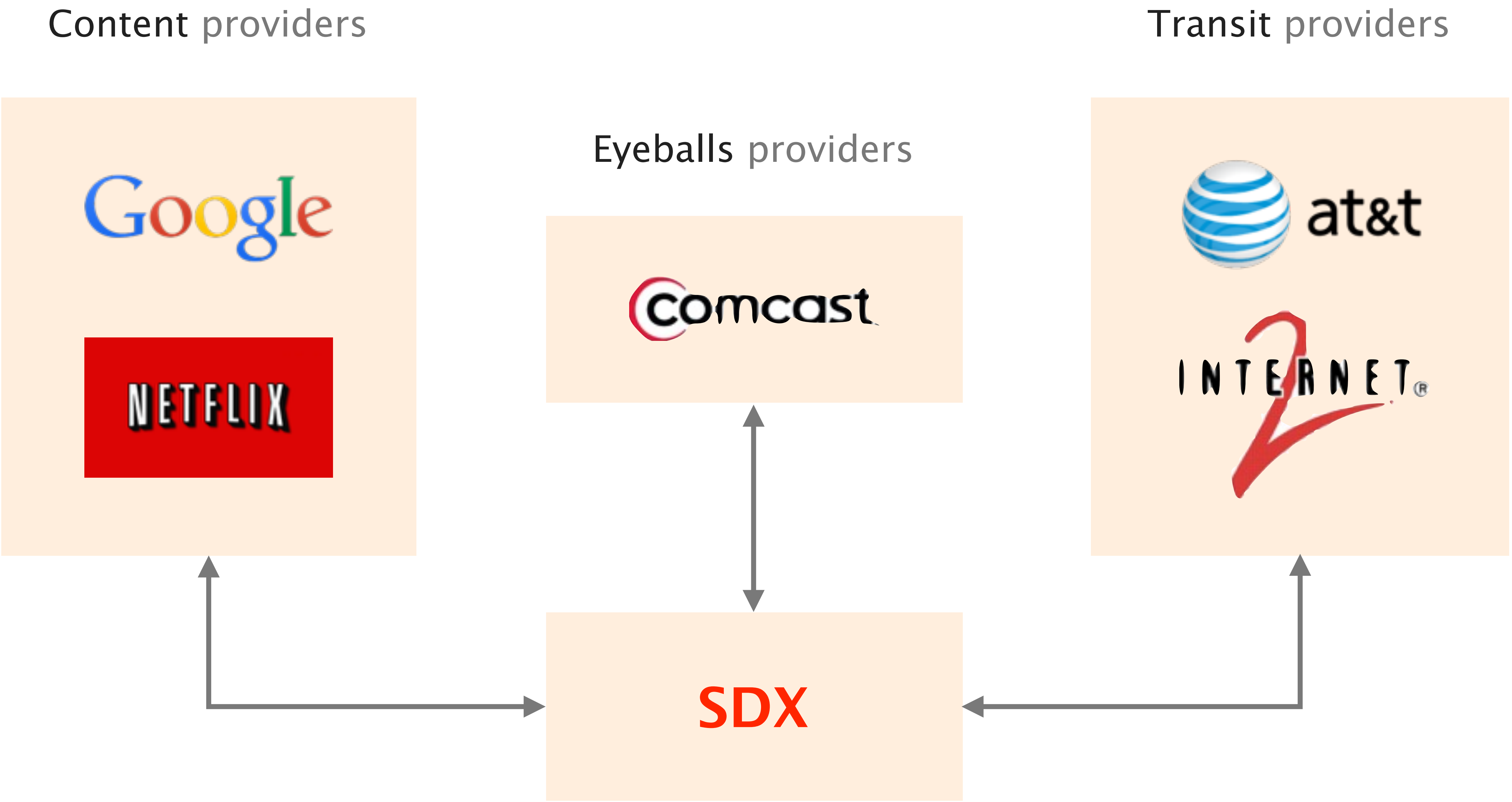
Enable fine-grained inter domain policies

bringing new features while simplifying operations

… with scalability and correctness in mind

supporting the load of a large IXP and resolving conflicts

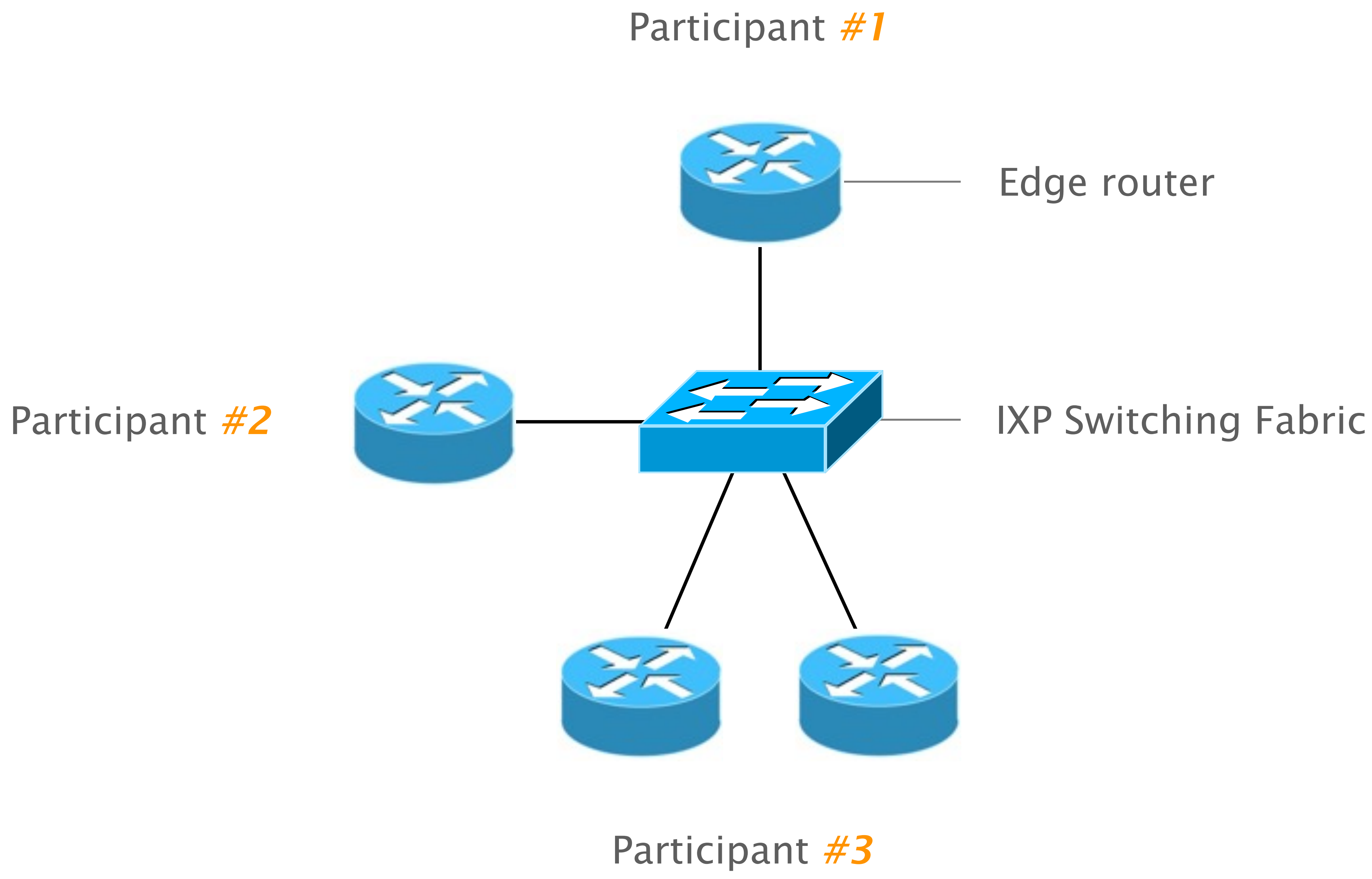
SDX is a platform that enables multiple stakeholders to define policies/apps over a shared infrastructure



SDX enables a wide range of novel applications

security	Prevent/block policy violation
	Prevent participants communication
	Upstream blocking of DoS attacks
forwarding optimization	Middlebox traffic steering
	Traffic offloading
	Inbound Traffic Engineering
	Fast convergence
peering	Application-specific peering
remote-control	Influence BGP path selection
	Wide-area load balancing

An IXP is a large layer-2 domain



Participant #1

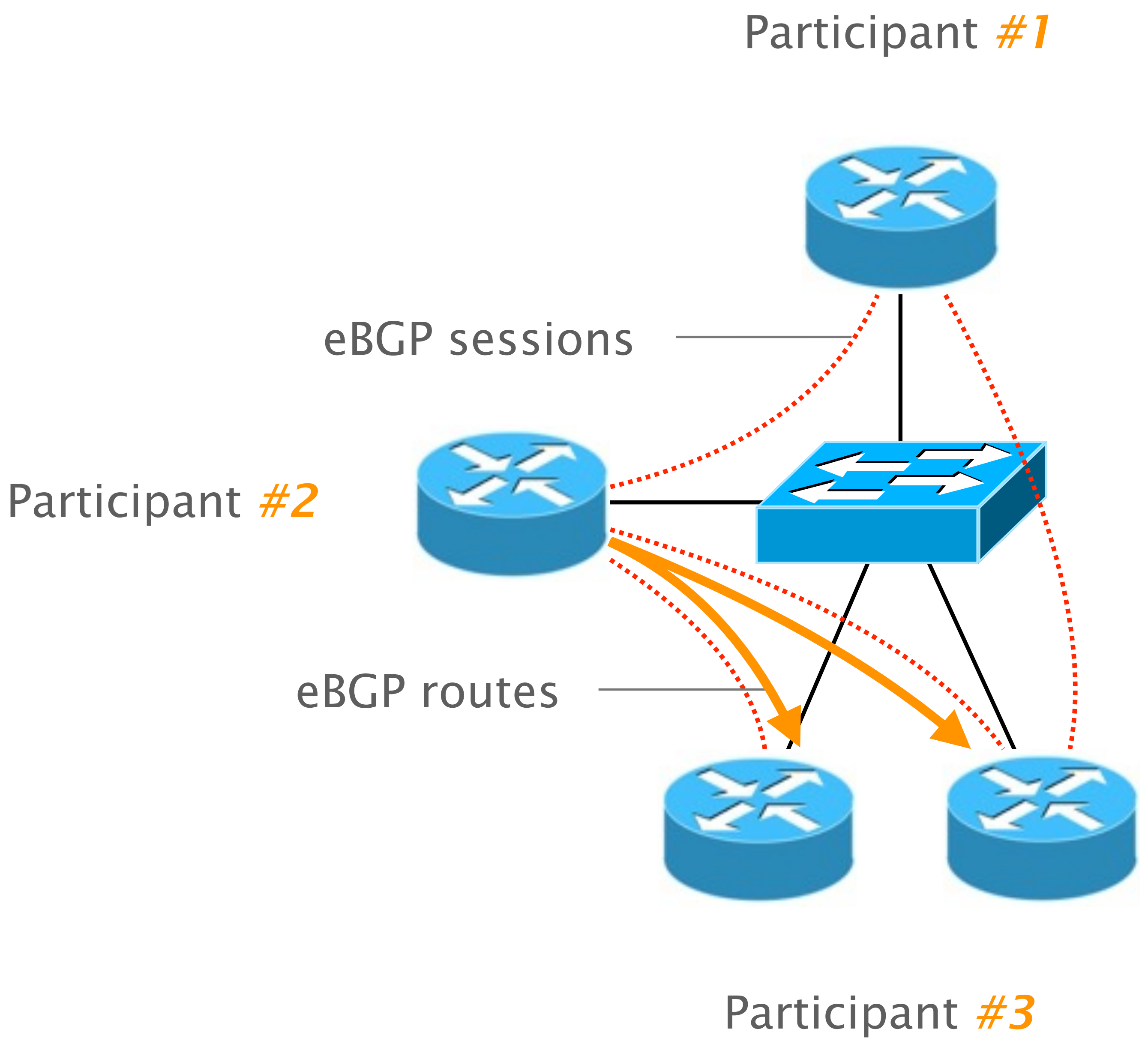
Edge router

Participant #2

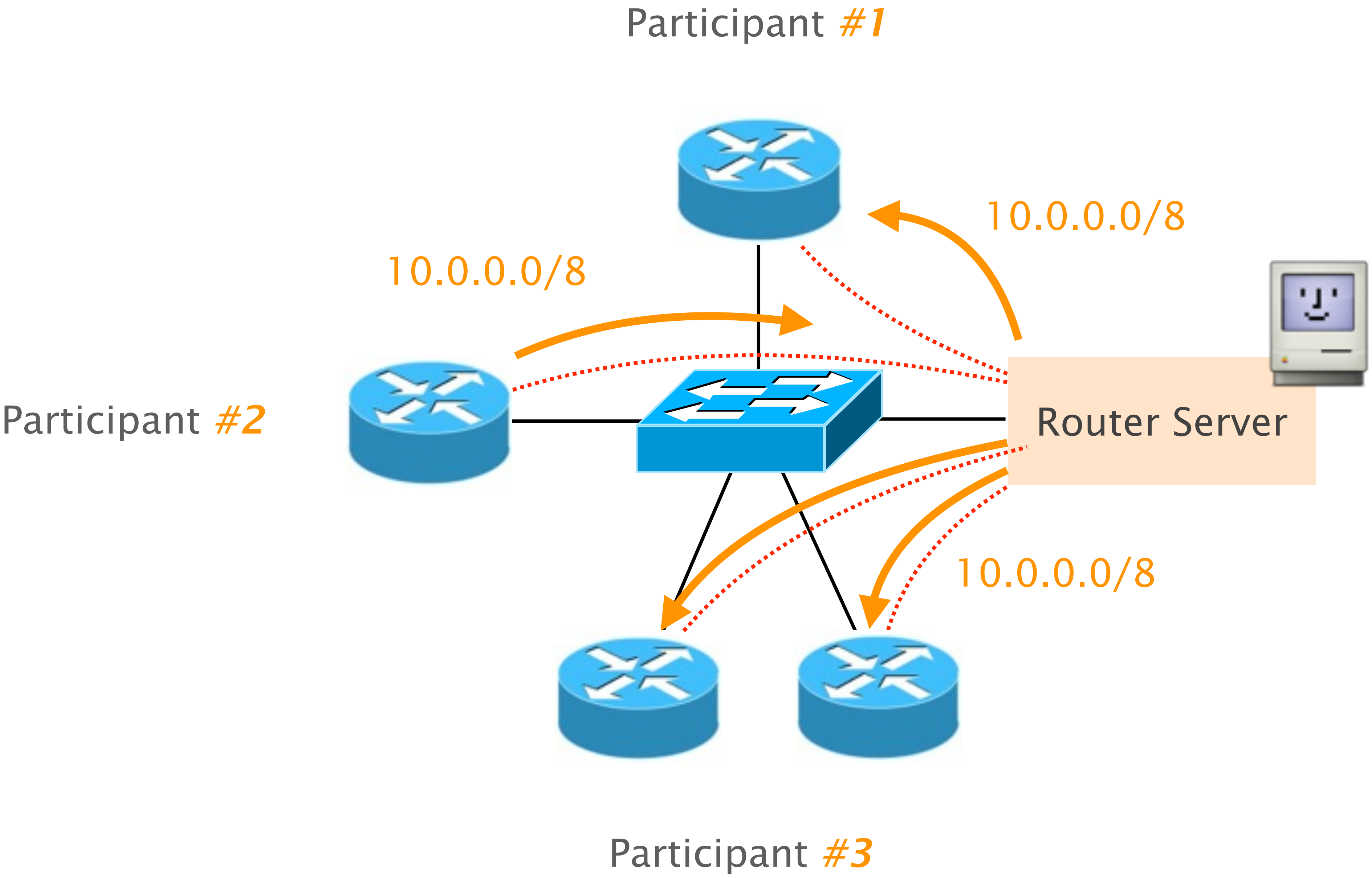
IXP Switching Fabric

Participant #3

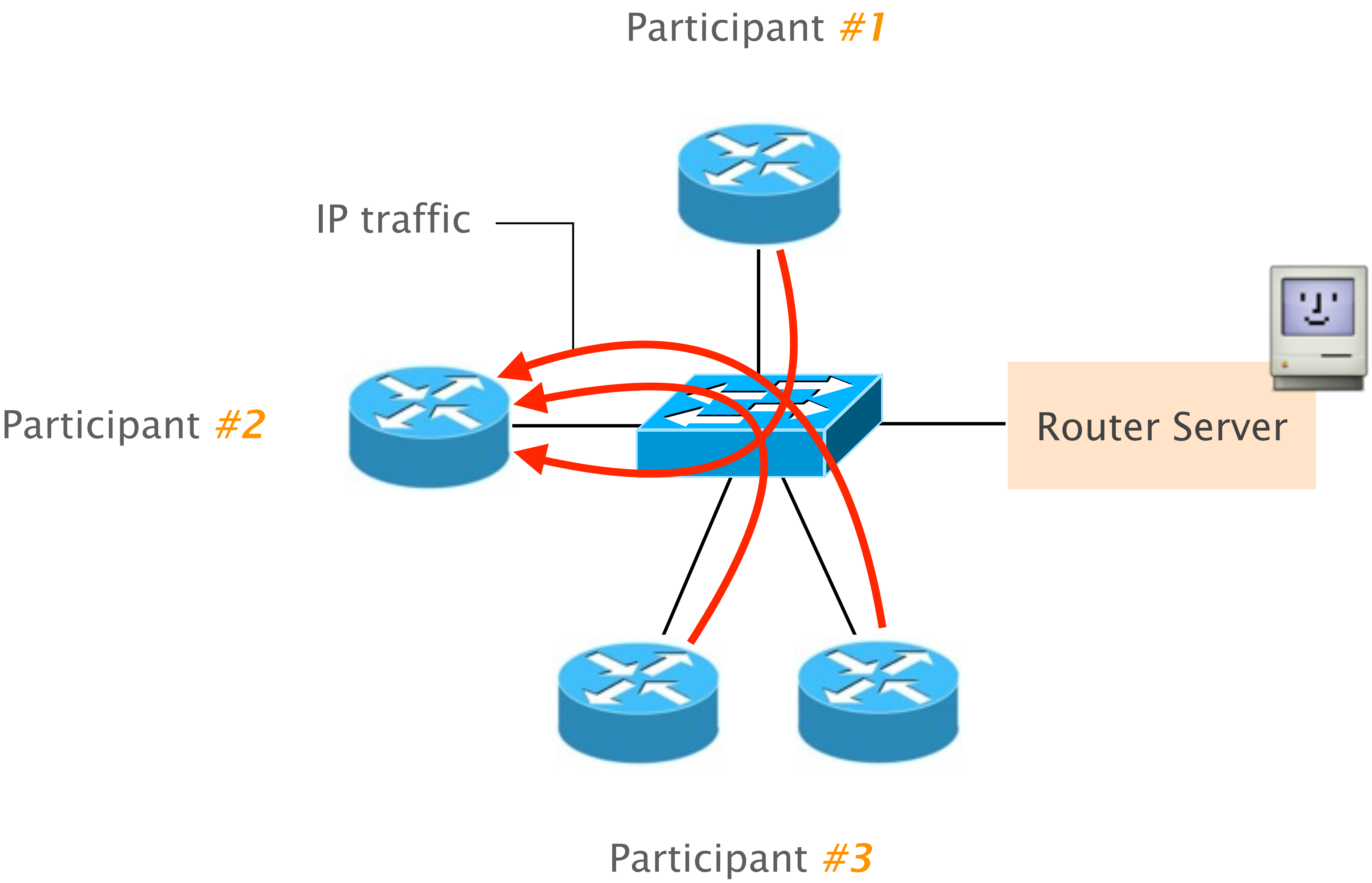
An IXP is a large layer-2 domain where participant routers exchange routes using BGP



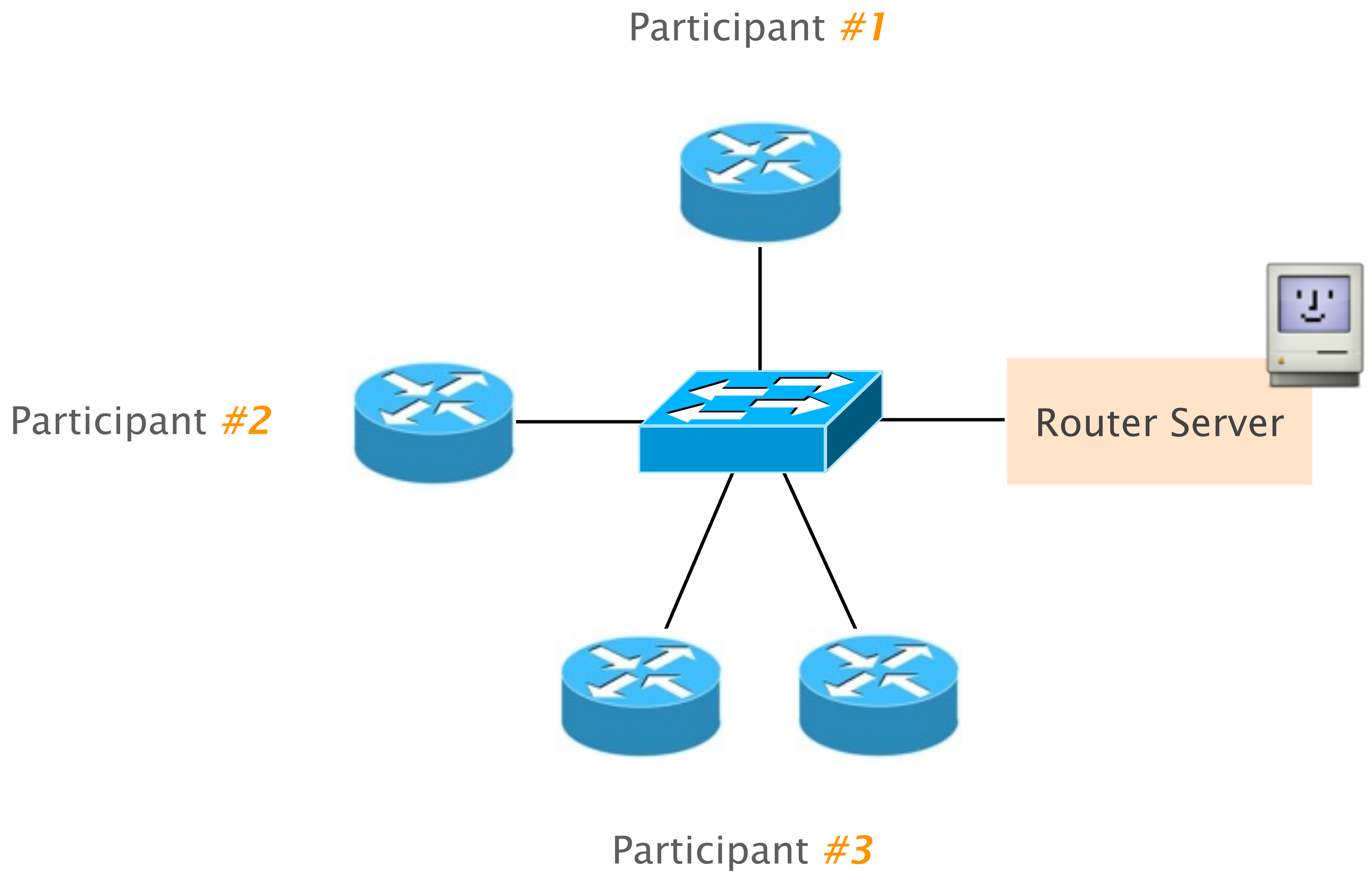
To alleviate the need of establishing eBGP sessions, IXP often provides a Route Server (route multiplexer)



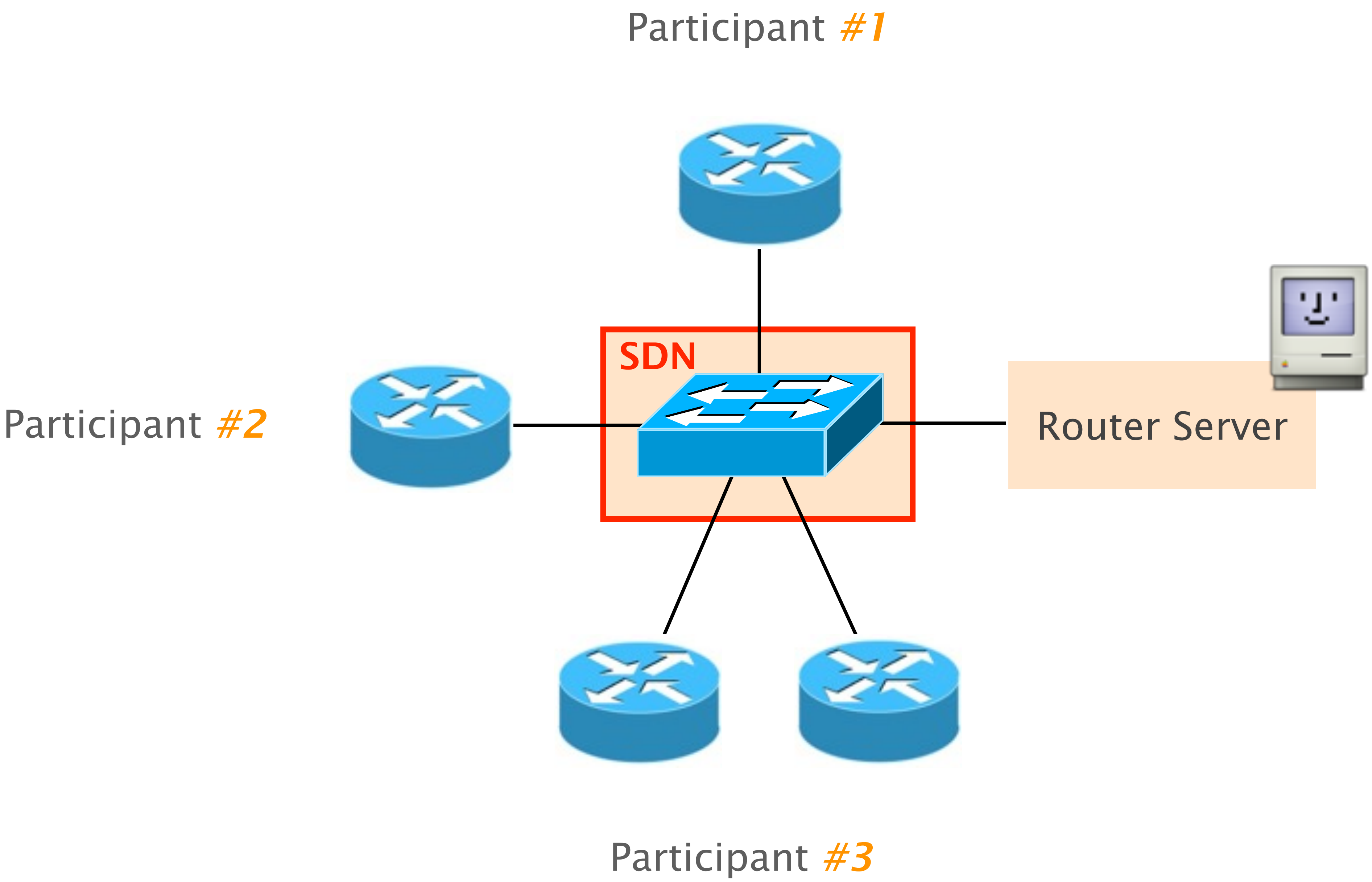
IP traffic is exchanged directly between participants—IXP is forwarding transparent



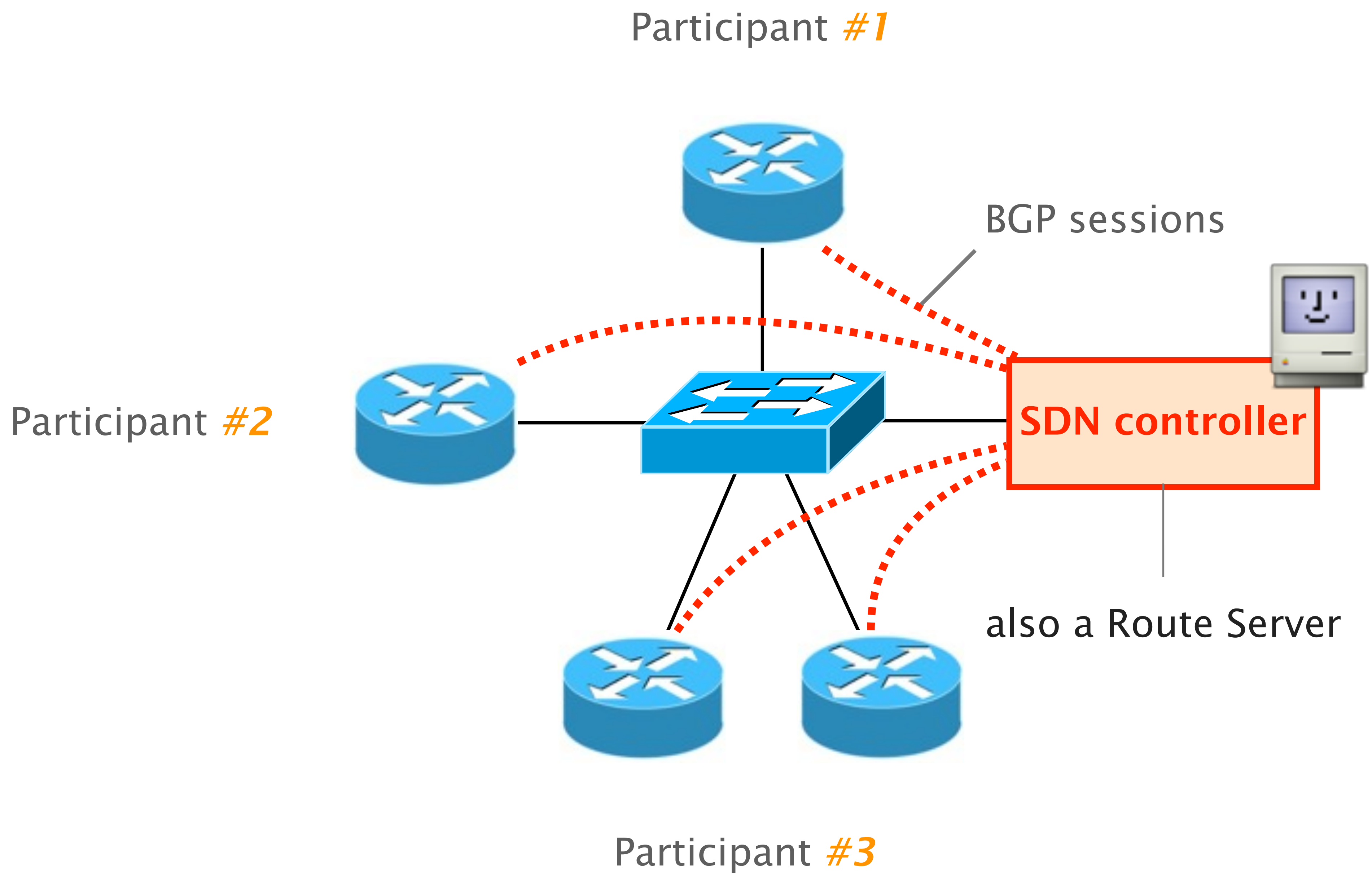
With respect to a traditional IXP, SDX…



With respect to a traditional IXP, SDX's data-plane relies on SDN-capable devices



With respect to a traditional IXP, SDX's control-plane relies on a SDN controller



SDX participants express their forwarding policies in a high-level language (*)

(*) http://frenetic-lang.org/pyretic/

SDX policies are composed of
a *pattern* and some *actions*

```
match (     Pattern     ), then (  Actions  )
```

Pattern selects packets based on any header fields

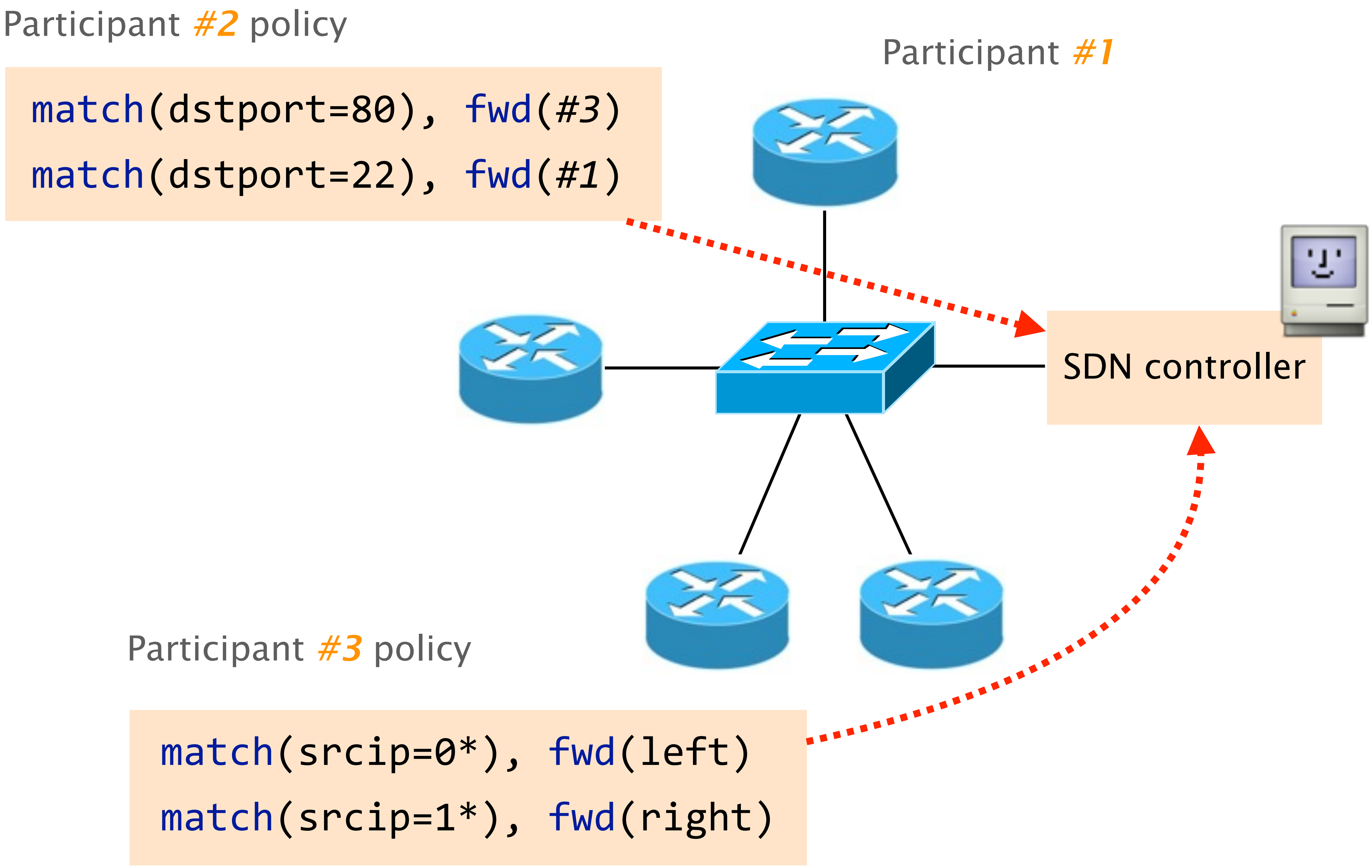
```
            Pattern

            eth_type
            vlan_id
            srcmac
match (     dstmac    , &&, ||  ), then (  Actions  )
            protocol
            dstip
            tos
            srcip
            srcport
            dstport
```

Pattern selects packets based on any header fields,
while actions forward or modify the selected packets

Actions

```
                                      drop
match (      Pattern      ), then (   forward    )
                                      rewrite
```

Each participant writes policies independently and transmits them to the controller



Participant #2 policy

```
match(dstport=80), fwd(#3)
match(dstport=22), fwd(#1)
```

Participant #1

SDN controller

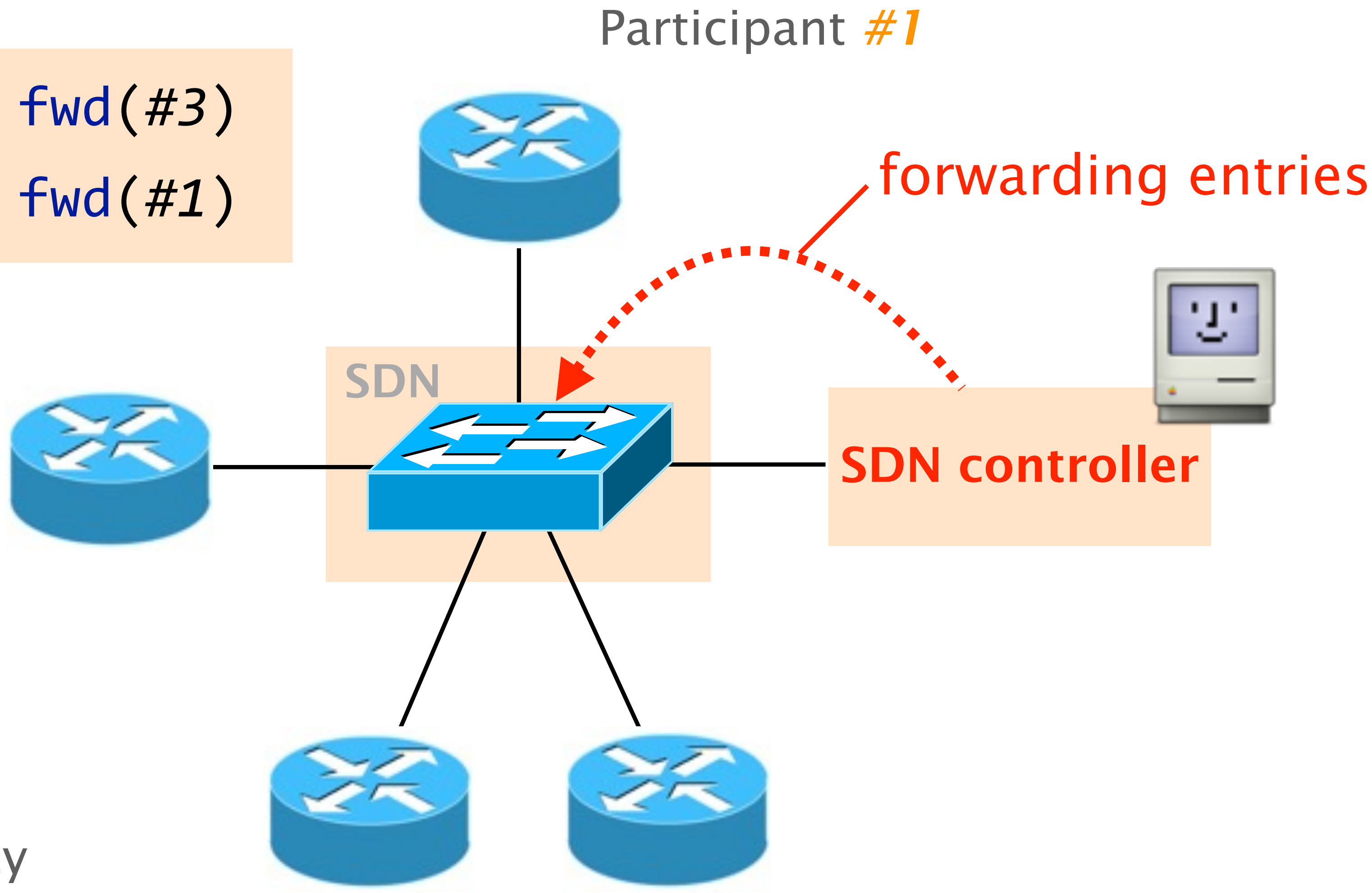
Participant #3 policy

```
match(srcip=0*), fwd(left)
match(srcip=1*), fwd(right)
```

Given the participant policies, the controller compiles them to SDN forwarding rules

Participant *#2* policy

```
match(dstport=80), fwd(#3)
match(dstport=22), fwd(#1)
```



Participant *#3* policy

```
match(srcip=0*), fwd(left)
match(srcip=1*), fwd(right)
```

Given the participant policies, the controller compiles them to SDN forwarding rules

Ensuring isolation

Resolving policies conflict

Ensuring compatibility with BGP

Given the participant policies,
the controller compiles them to SDN forwarding rules

Ensuring isolation

Each participant controls
one virtual switch

connected to participants
it can communicate with

Resolving policies conflict

Ensuring compatibility with BGP

Given the participant policies,
the controller compiles them to SDN forwarding rules

Ensuring isolation

Resolving policies conflict

Participant policies are sequentially composed

in an order that respects business relationships

Ensuring compatibility with BGP

Given the participant policies, the controller compiles them to SDN forwarding rules

Ensuring isolation

Resolving policies conflict

Ensuring compatibility with BGP

policies are augmented with BGP information

guaranteed correctness and reachability

SDX is a promising first step towards fixing Internet routing

It runs

check out https://github.com/sdn-ixp/sdx-ryu (new!)

It scales

to 100+ of participants

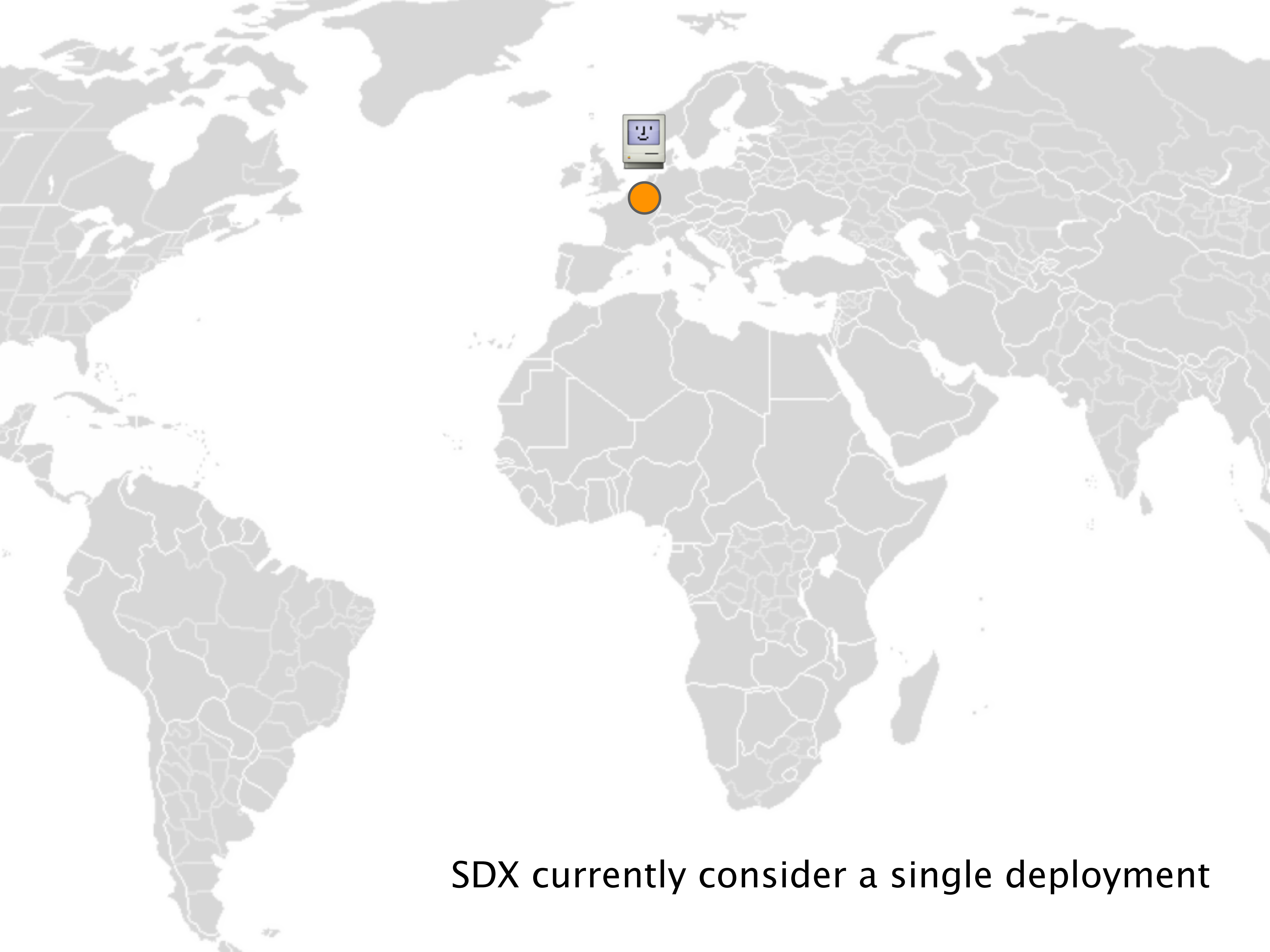
It is getting deployed

NSA plans to use it to connect federal agencies

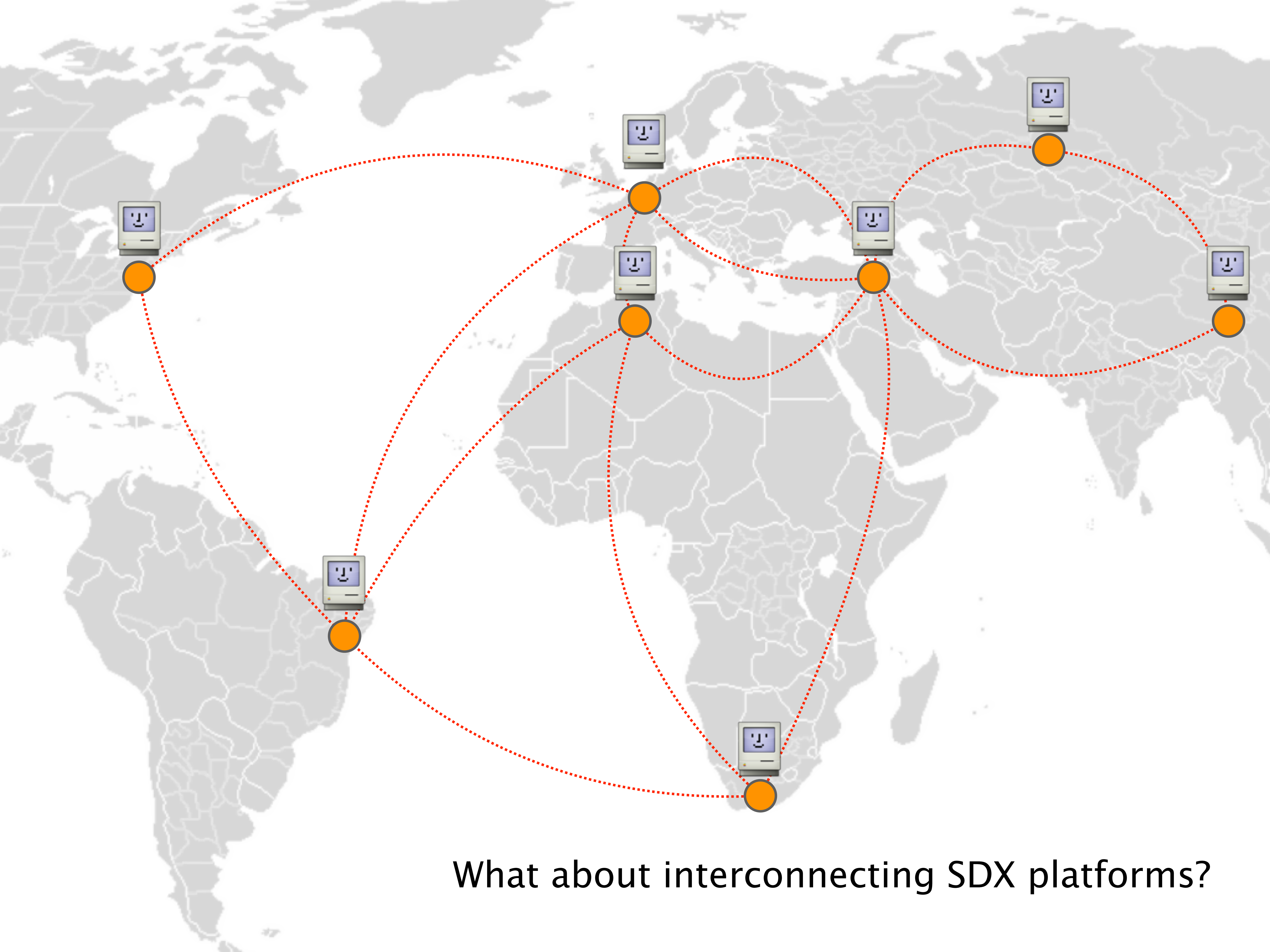
So… it's done basically?

So… it’s done basically?

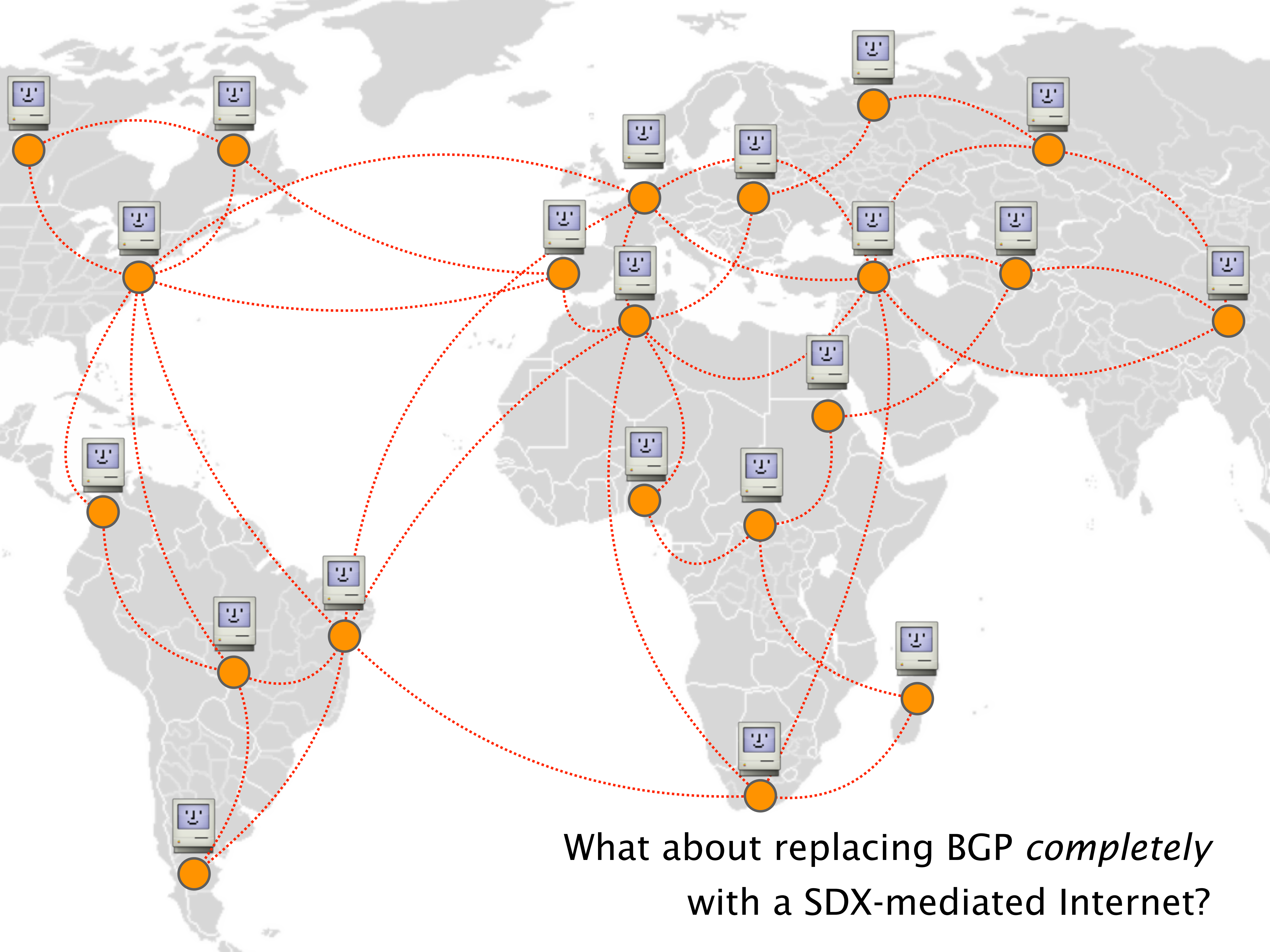
No… far from it!



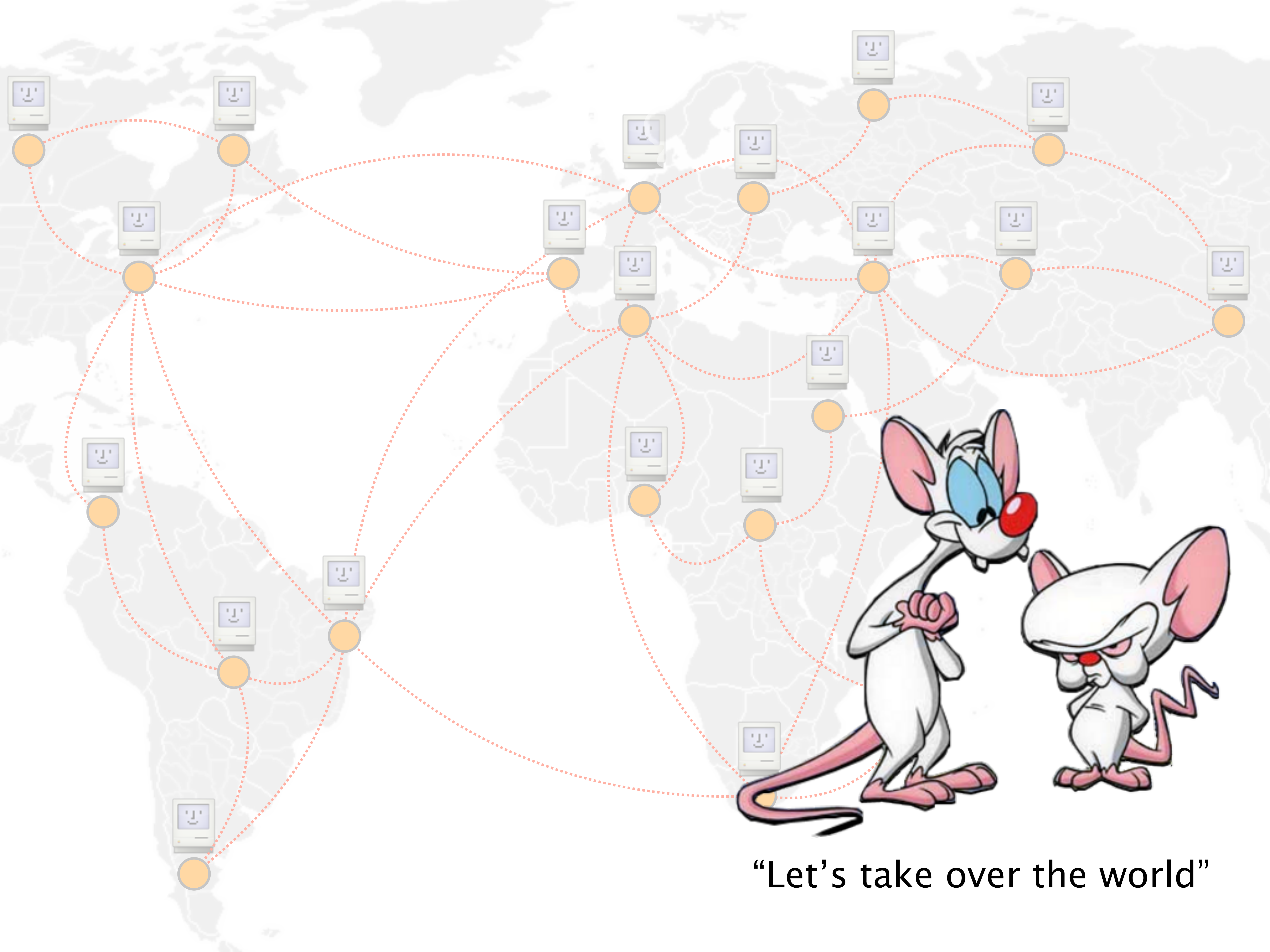
SDX currently consider a single deployment



What about interconnecting SDX platforms?



What about replacing BGP *completely*
with a SDX-mediated Internet?



“Let’s take over the world”

Towards a SDX-mediated Internet

New endpoint peering paradigm

more flexible, tailored to the traffic exchanged

Simple, scalable & policy neutral Internet core

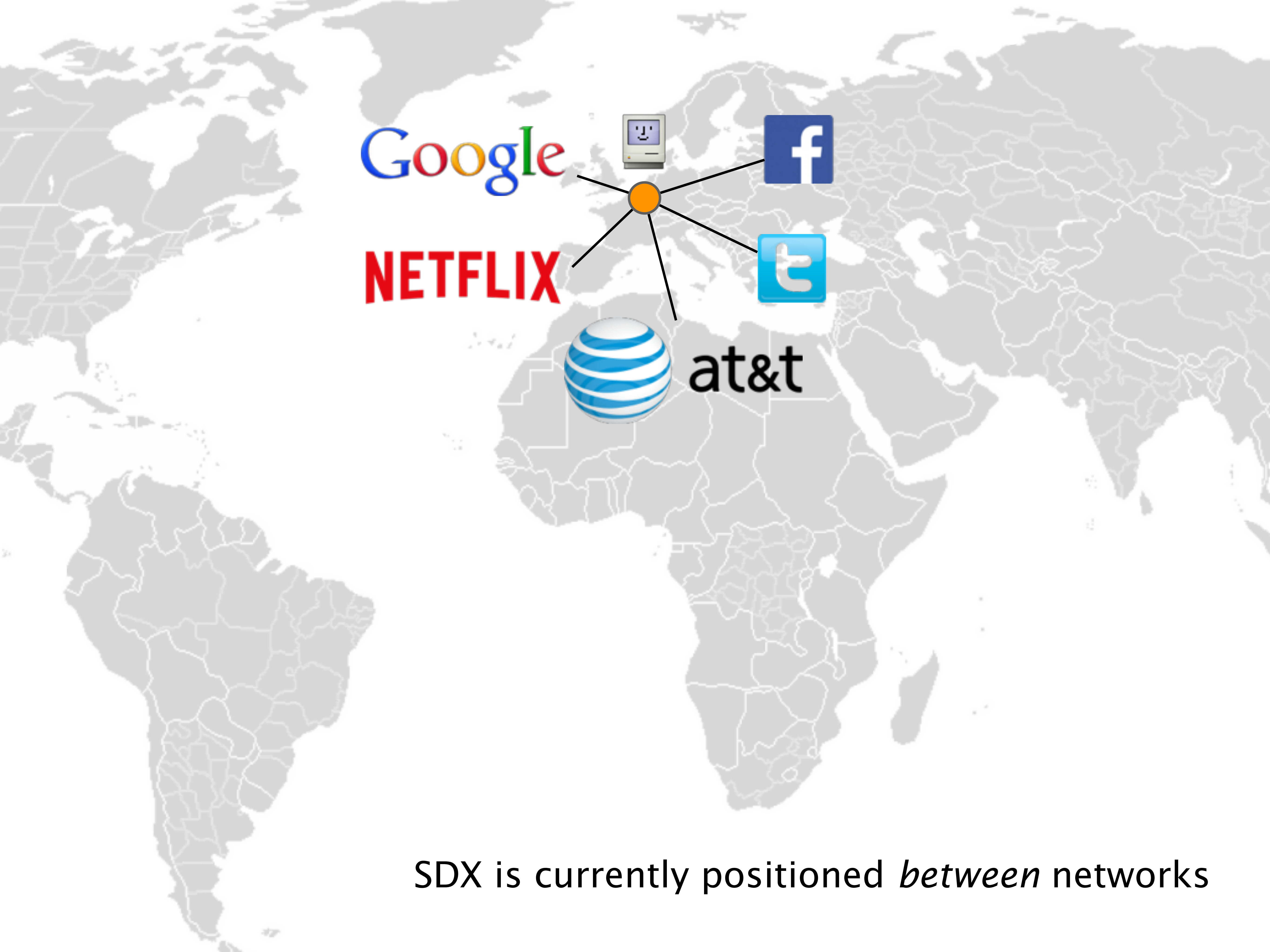
SDX-to-SDX only, just carry bits

In-synch with the current Internet ecosystem

content consumer *vs* content provider *vs* transit network

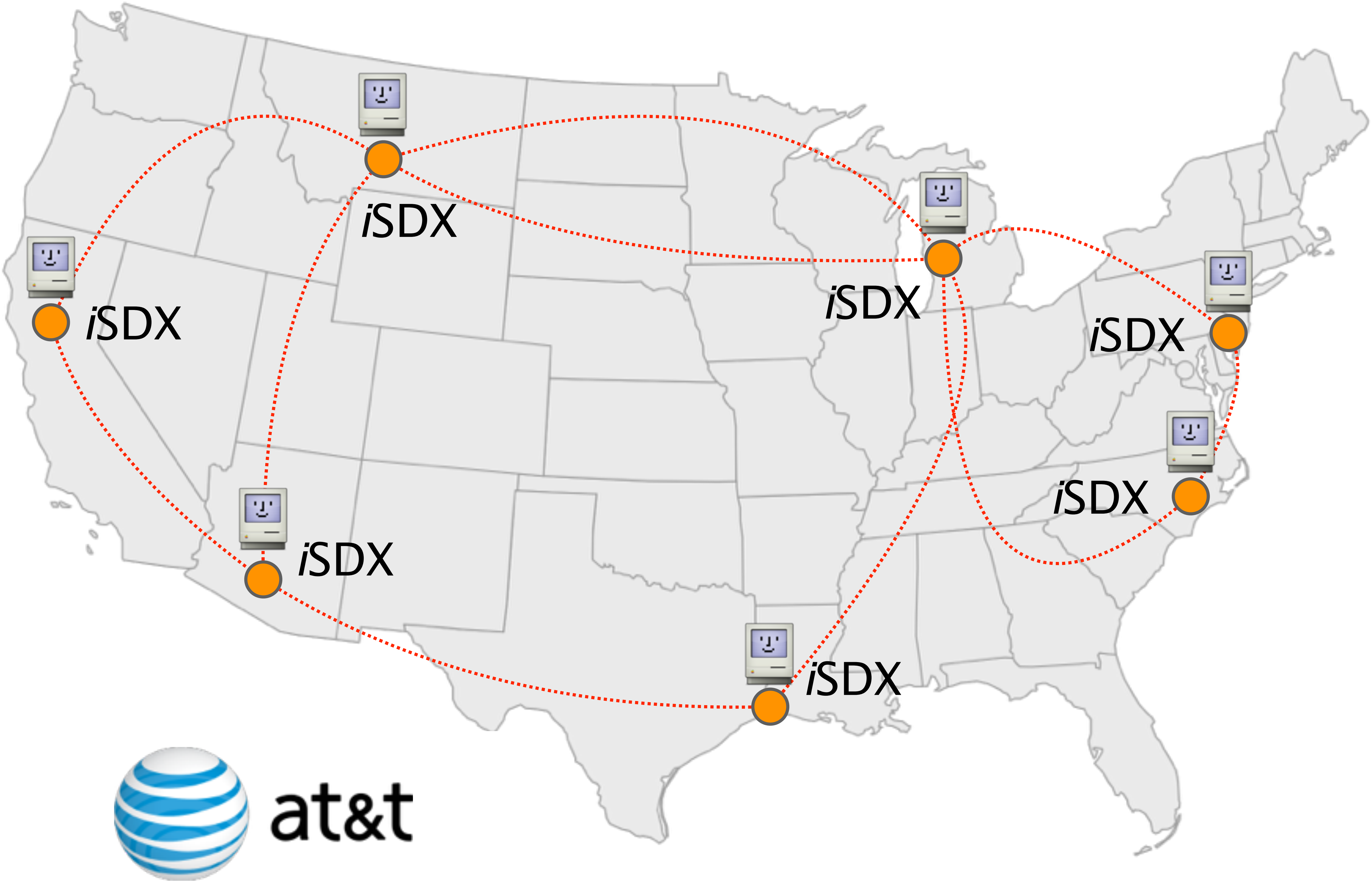
Many novel research questions!

policy analysis?	New endpoint peering paradigm more flexible, tailored to the traffic exchanged
routing mechanism?	Simple, scalable & policy neutral Internet core SDX-to-SDX only, just carry bits
new provider type?	In-synch with the current Internet ecosystem content consumer *vs* content provider *vs* transit network

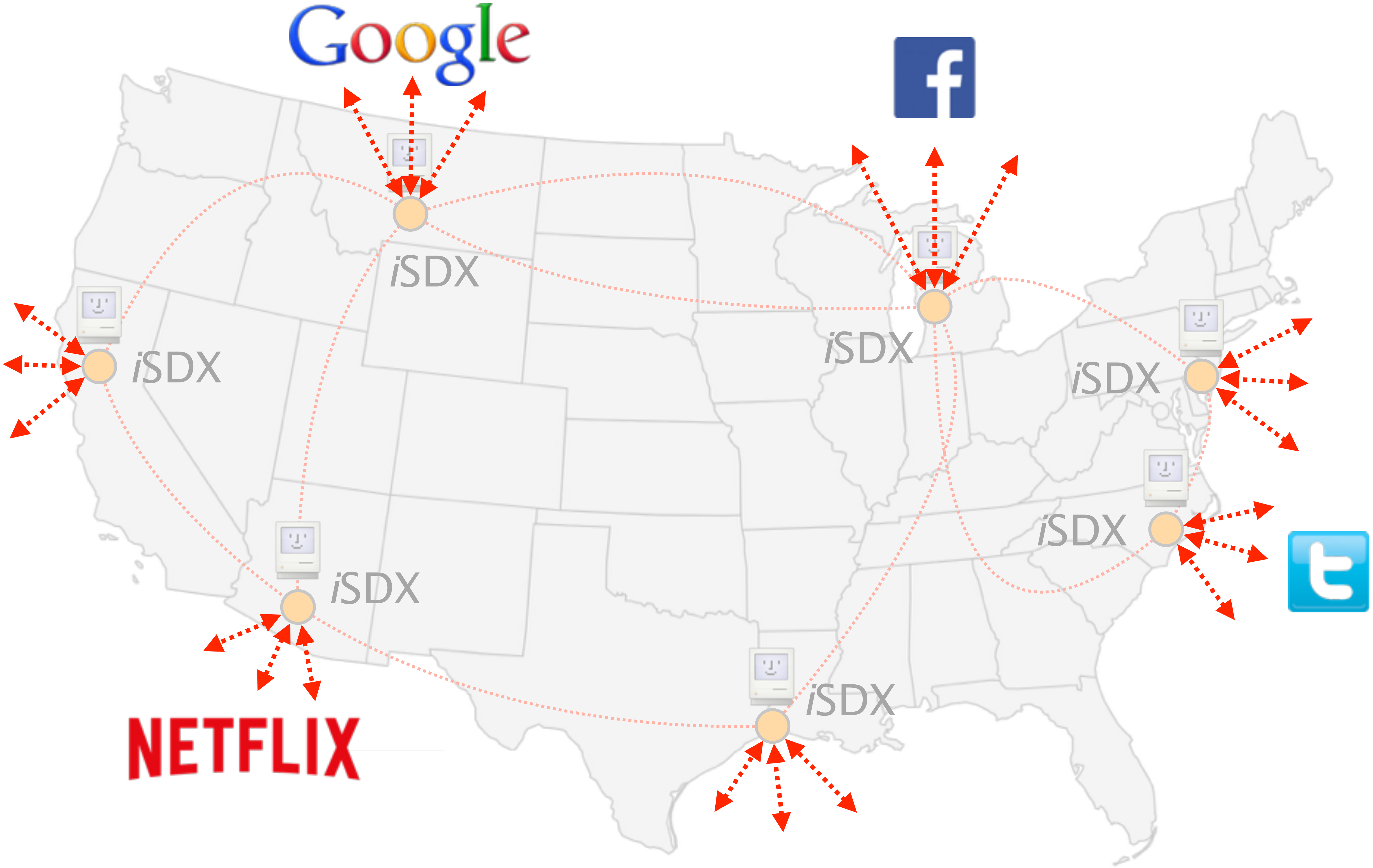


SDX is currently positioned *between* networks

What about using the SDX platform internally…



What about using the SDX platform internally…



…to better manage peerings with neighbouring ASes

Current transit networks are still managed archaically

per-neighbor configuration

one session at the time

static configuration

while Internet traffic is inherently dynamic

lack of visibility

coarse-grained measurements (mostly for billing)

SDX-mediated peering would bring much-needed flexibility

high-level, declarative objective

“equally load-balance Netflix on 3 given links”

automated & dynamic optimization

to ensure compliance and ease network provisioning

fine-grained, network-wide visibility

improved decisions, troubleshooting & billing (!)

Many novel research questions!

policy	high-level, declarative objective
language?	“equally load-balance Netflix on 3 given links”
correctness	automated & dynamic optimization
guarantees?	to ensure compliance and ease network provisioning
scalability?	fine-grained, network-wide visibility
	improved decisions, troubleshooting & billing (!)

SDN research directions

Promising problems to invest time on

Go beyond OpenFlow

Secure SDN platforms

Incentivize deployment

Extend SDN reach

SDN holds great research opportunities

SDN is still in its infancy

lots of moving parts—and opportunities

SDN is exciting

tons of interest—from academia & industry

SDN is happening

some success already

SDN research directions

Promising problems to invest time on

Laurent Vanbever

www.vanbever.eu

Wishing you every success

in your future SDN research