Bringing SDN to the Internet, one exchange point at the time

Joint work with: Arpit Gupta, Muhammad Shahbaz, Sean P. Donovan, Russ Clark, Brandon Schlinker, E. Katz-Bassett, Nick Feamster, Jennifer Rexford and Scott Shenker

Laurent Vanbever

Princeton University

COS 561

November, 11 2014

BGP is notoriously inflexible
and difficult to manage

BGP is notoriously inflexible and difficult to manage

	BGP	SDN
Fwd paradigm		
Fwd control		
Fwd influence		

BGP is notoriously inflexible and difficult to manage

	BGP	SDN
Fwd paradigm	destination-based	
Fwd control	indirect configuration	
Fwd influence	local BGP session	

SDN can enable fine-grained, flexible and direct expression of interdomain policies

	BGP	SDN
Fwd paradigm	destination-based	any source addr, ports,…
Fwd control	indirect configuration	direct open API (e.g., OpenFlow)
Fwd influence	local BGP session	global remote controller control

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

675 networks

3.2 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

675 networks

3.2 Tb/s (peak)

BGP Route Server

Mobile peering

Open peering…

https://www.ams-ix.net

SDX = SDN + IXP

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 & simplifying operations

SDX = SDN + IXP

Augment the IXP data-plane with SDN capabilities

keeping default forwarding and routing behavior

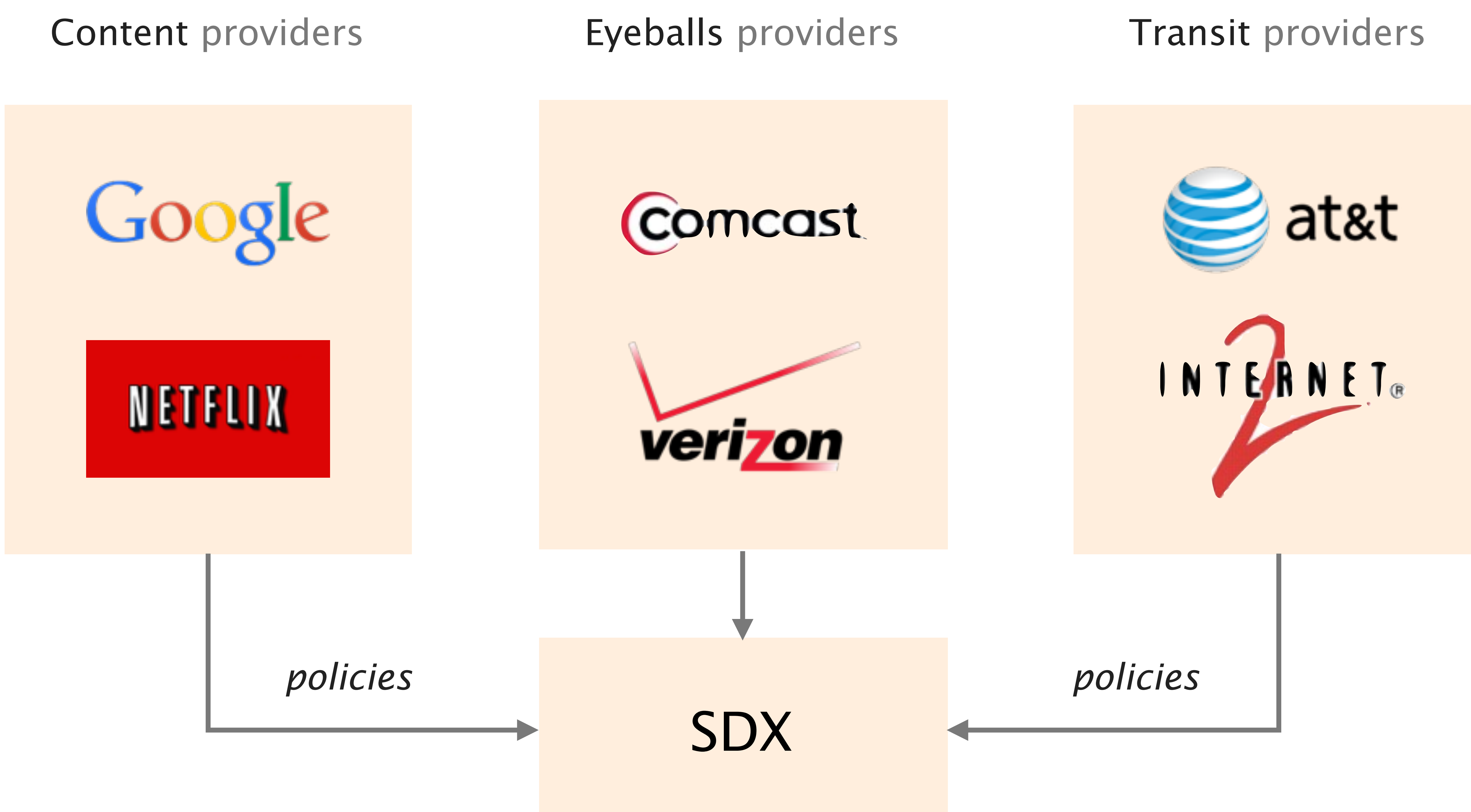
Enable fine-grained inter-domain policies

bringing new features & simplifying operations

… with scalability and correctness in mind

supporting large IXP load and resolving conflicts

SDX enables multiple stakeholders to implement policies and apps over a shared infrastructure



Bringing SDN to the Internet, one exchange point at the time

1 **Architecture**
programming model

2 **Scalability**
control- & data-plane

3 **Applications**
inter domain bonanza

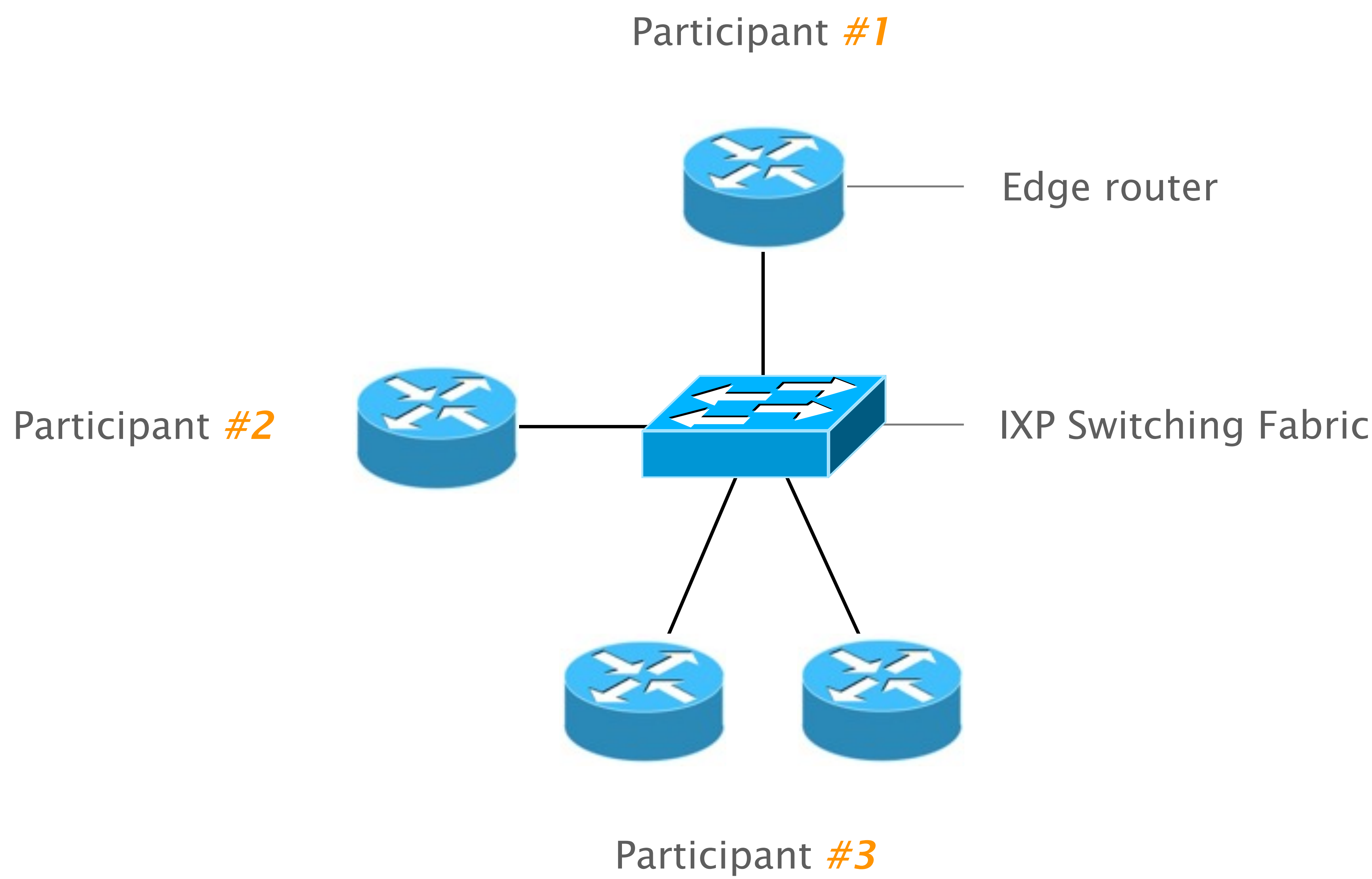
Bringing SDN to the Internet, one exchange point at the time

1 **Architecture**
programming model

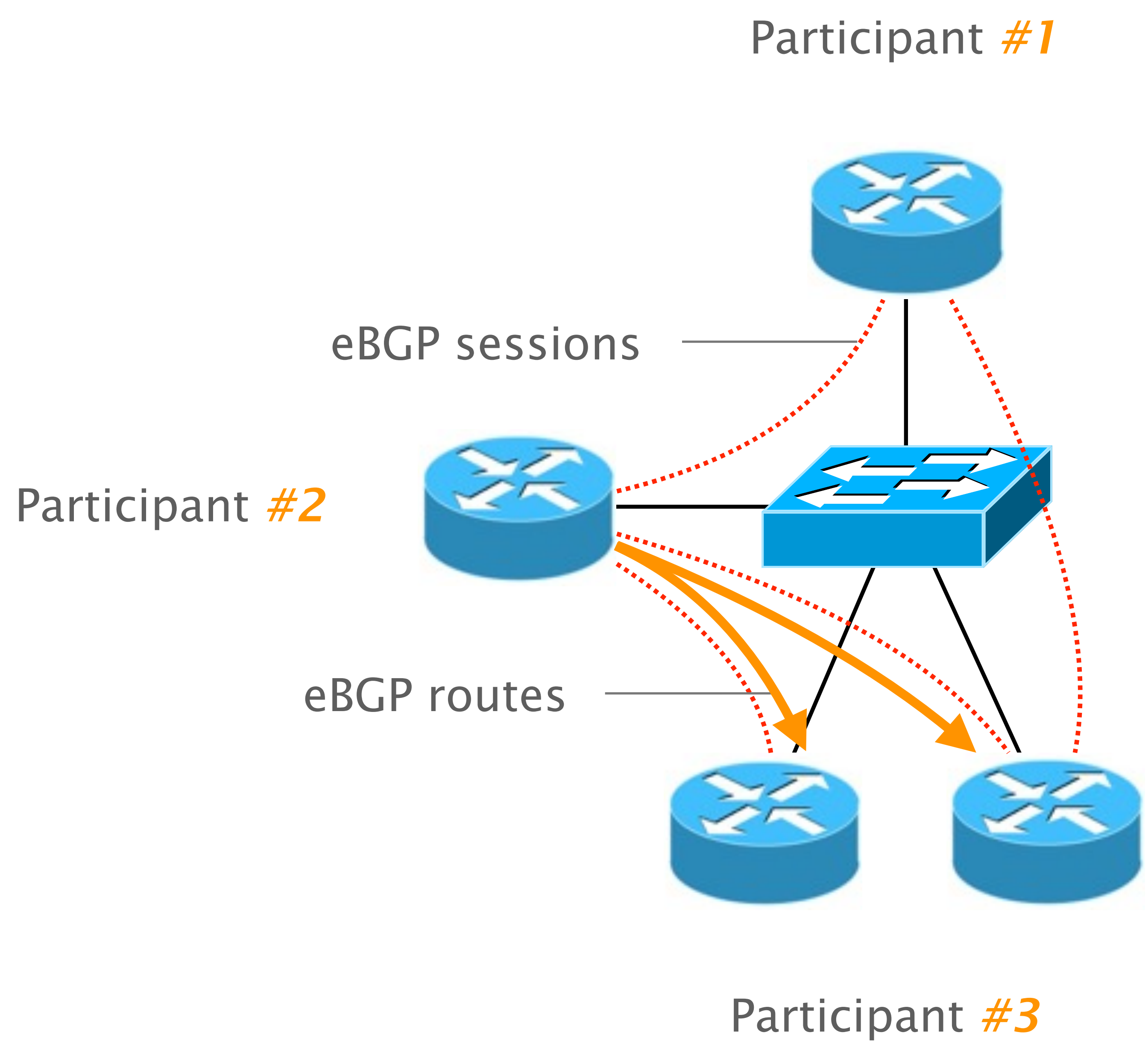
Scalability
control- & data-plane

Applications
inter domain bonanza

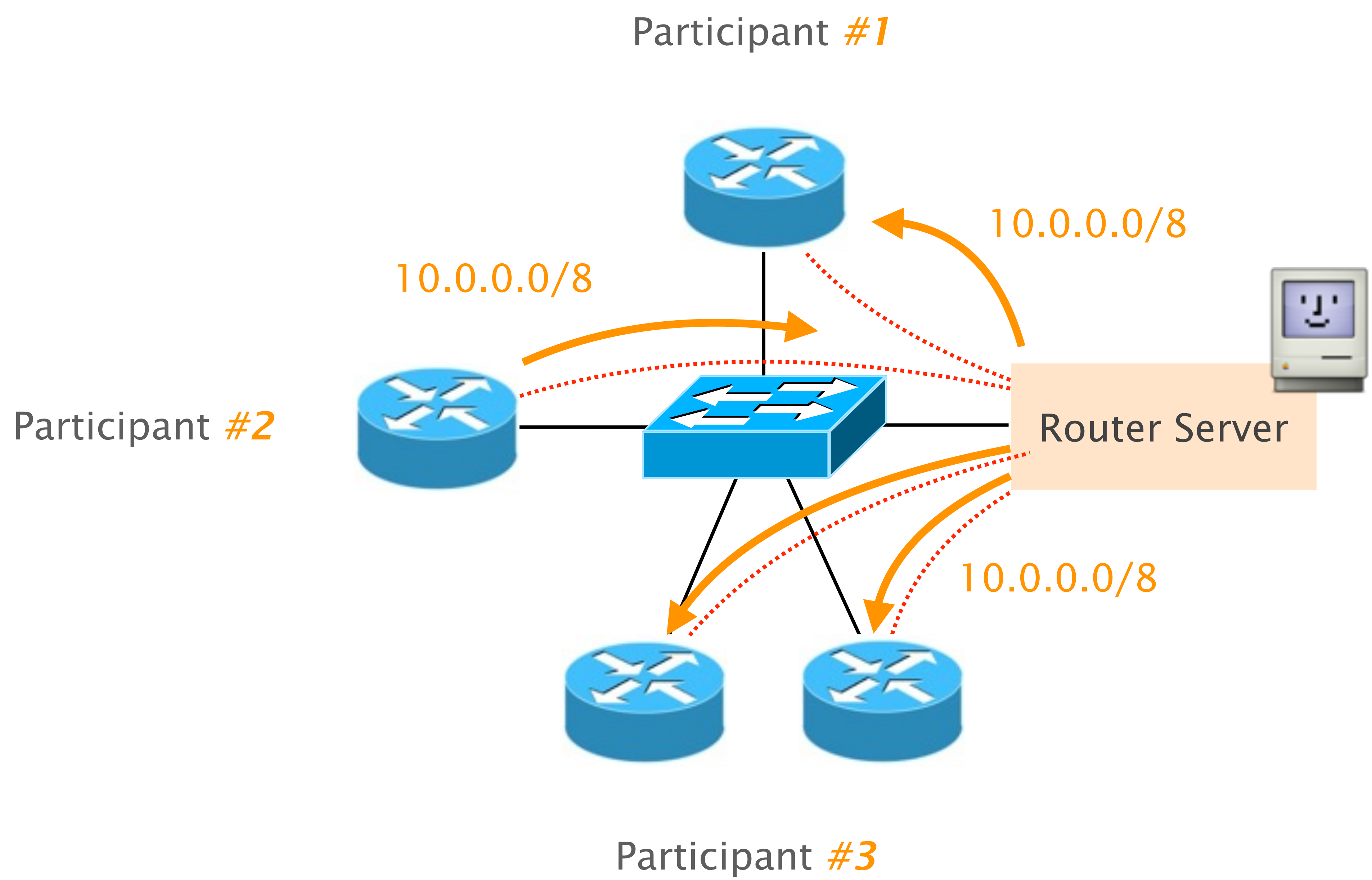
An IXP is a large layer-2 domain



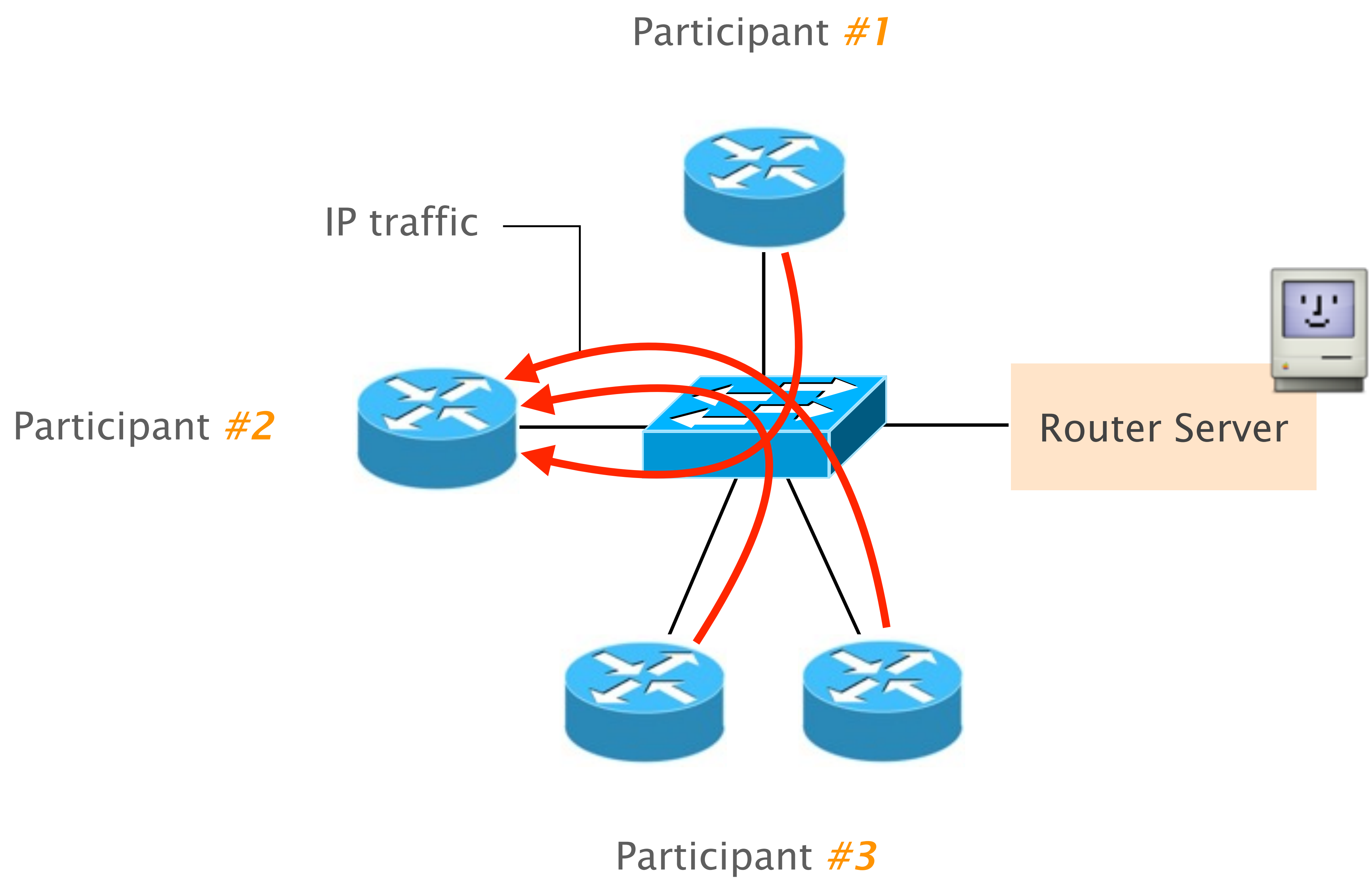
An IXP is a large layer-2 domain **where participant routers exchange routes using BGP**



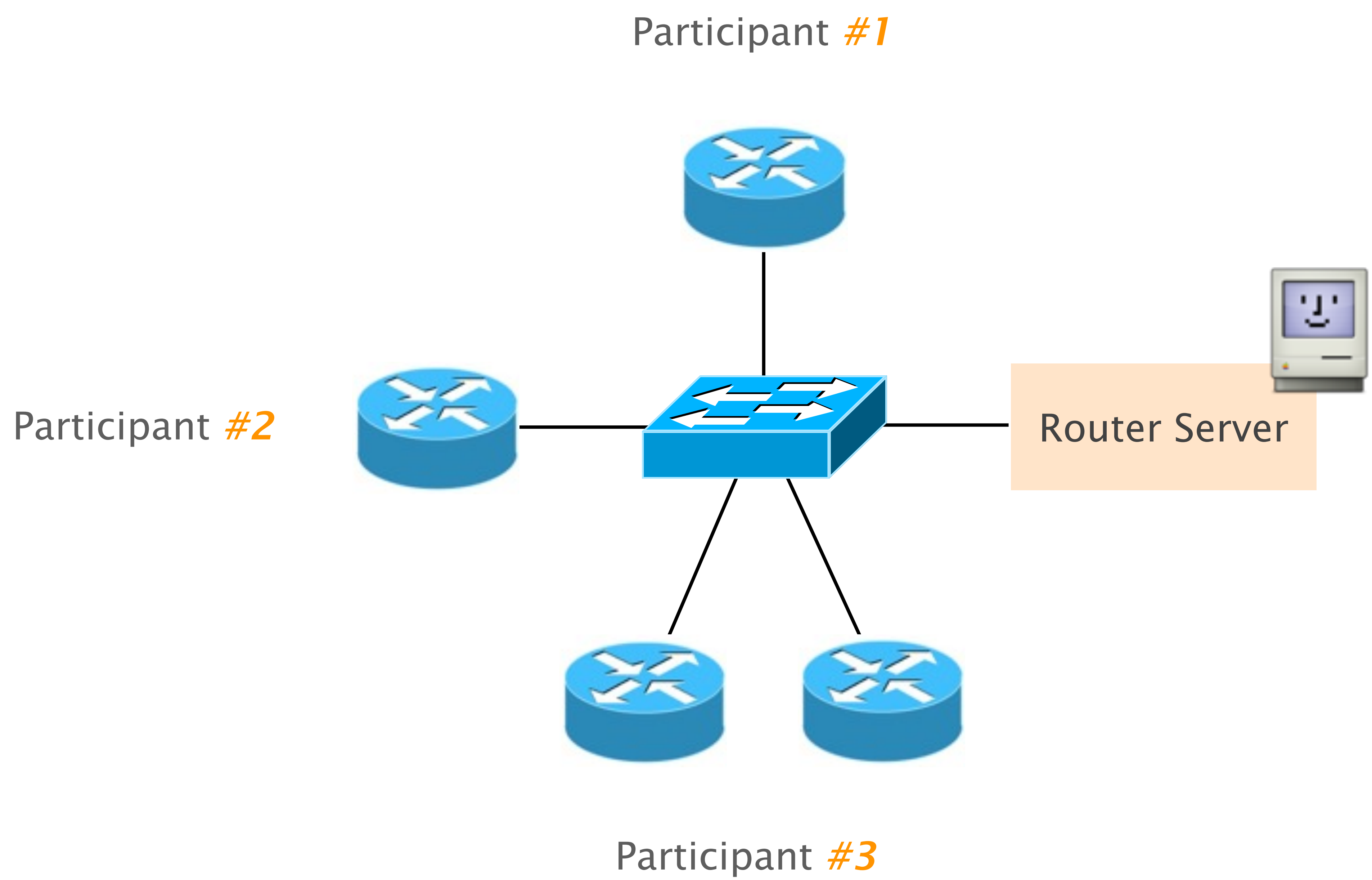
To alleviate the need of establishing eBGP sessions, IXP often provides a Route Server (route multiplexer)



IP traffic is exchanged directly between participants

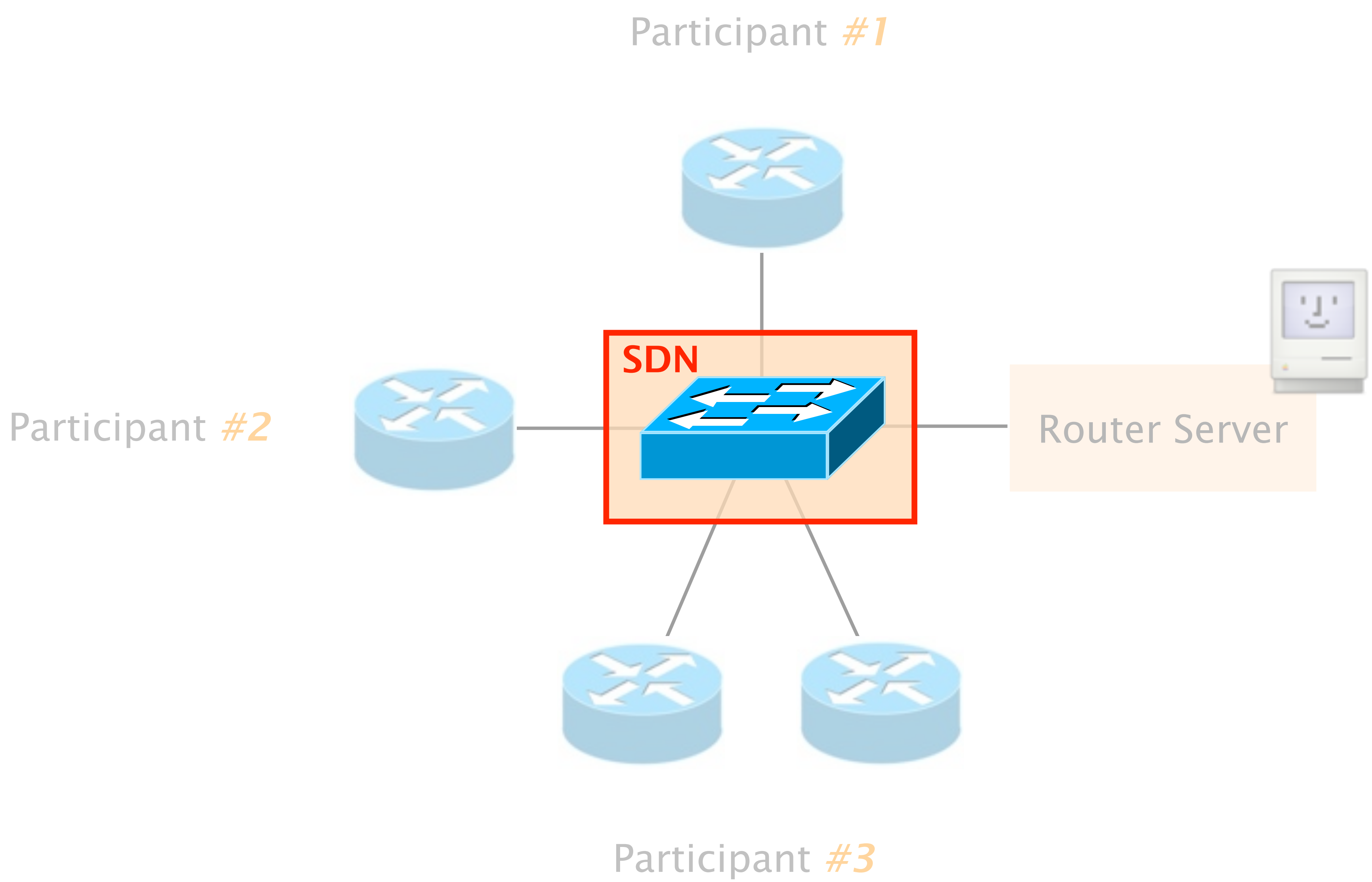


With respect to a traditional IXP,



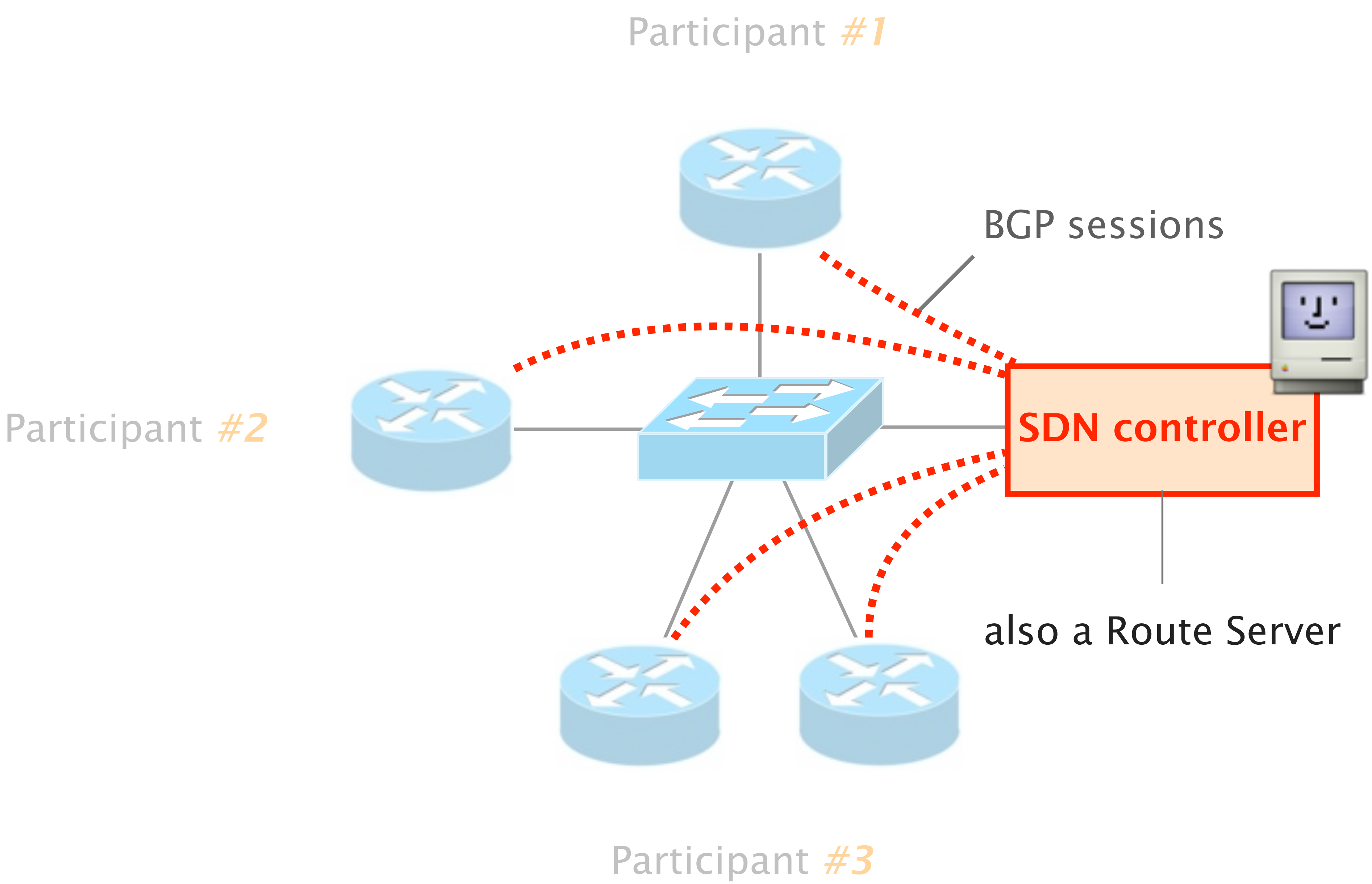
With respect to a traditional IXP,

SDX data-plane relies on SDN-capable devices



With respect to a traditional IXP,

SDX control-plane relies on a SDN controller



SDX participants express their forwarding policies in a high-level language, built on top of Pyretic (*)

(*) 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

```
match ( eth_type
        vlan_id
        srcmac
        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

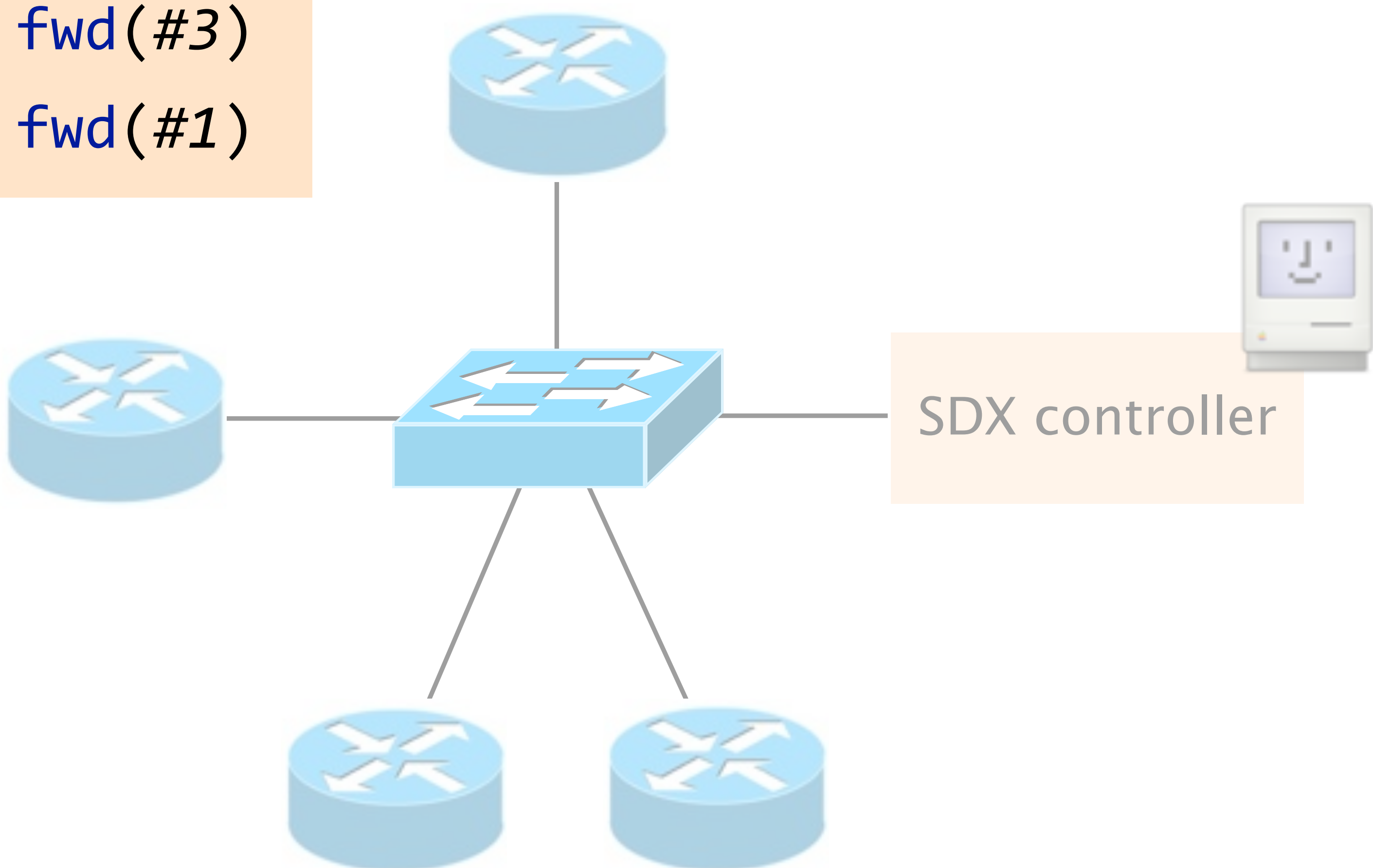
Actions

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

Each SDX participant writes her policies independently

Participant *#2* policy

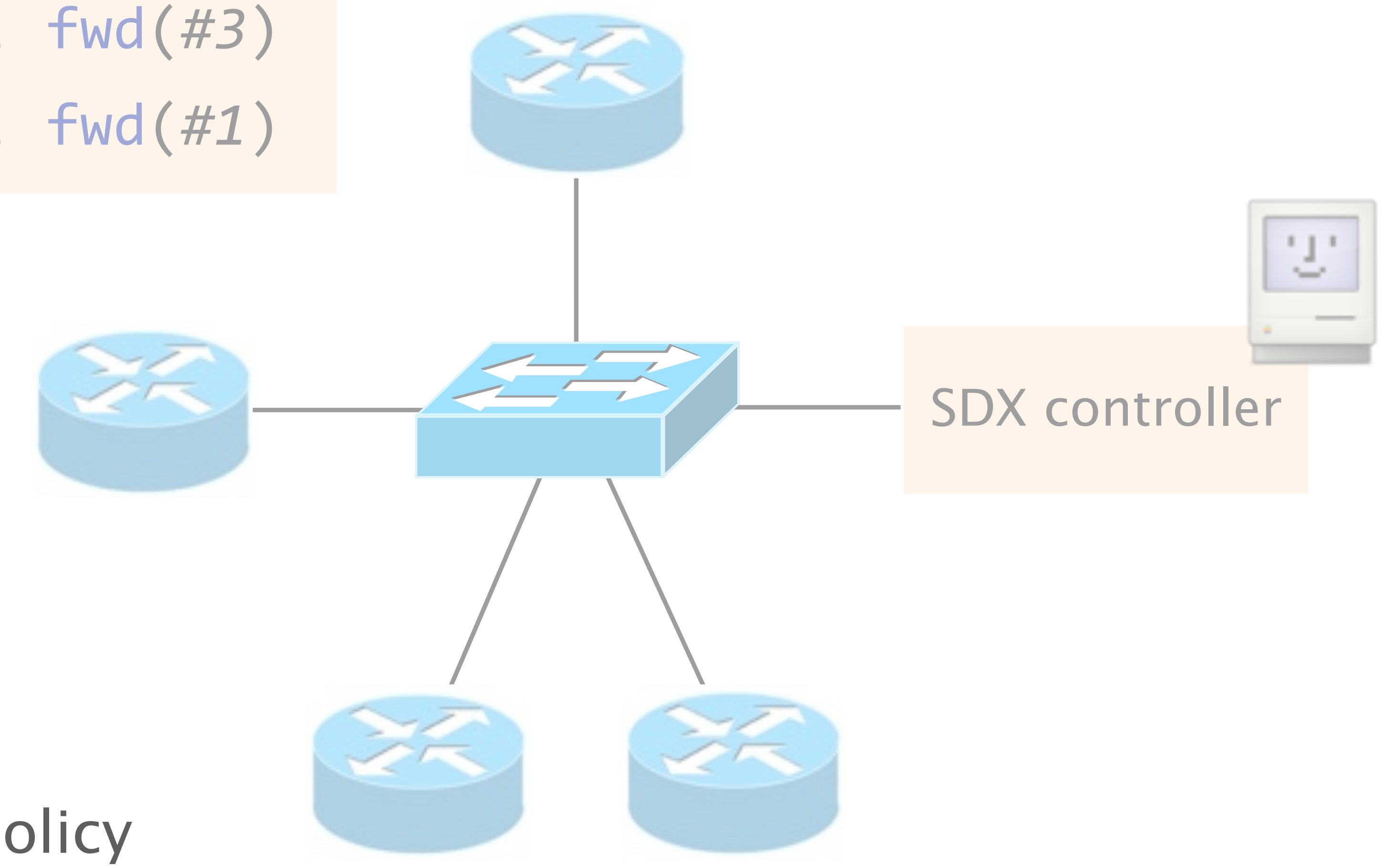
```
match(dstport=80), fwd(#3)
match(dstport=22), fwd(#1)
```



Each SDX participant writes her policies independently

Participant *#2* policy

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



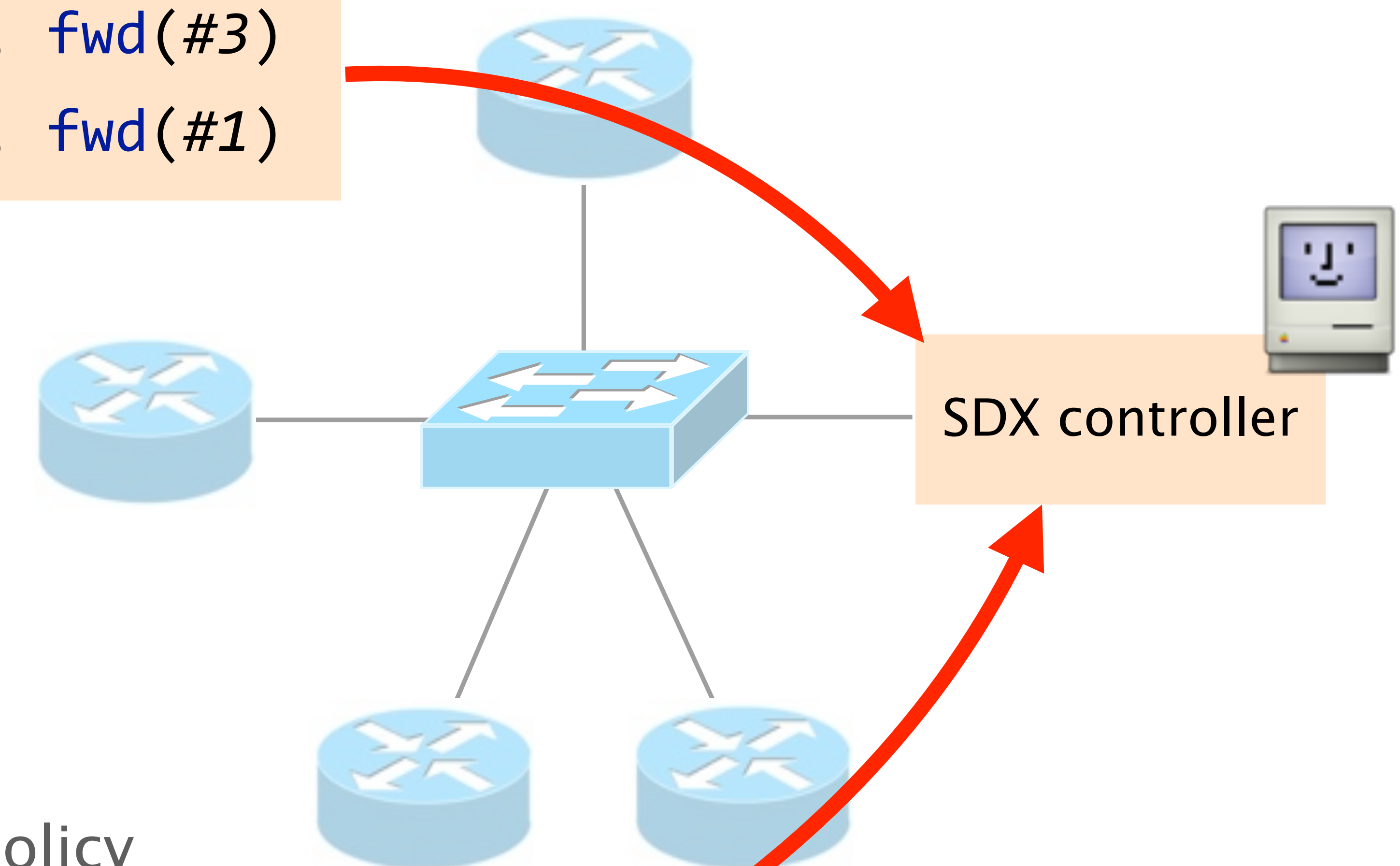
Participant *#3* policy

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

… and transmit them to the SDX controller

Participant #2 policy

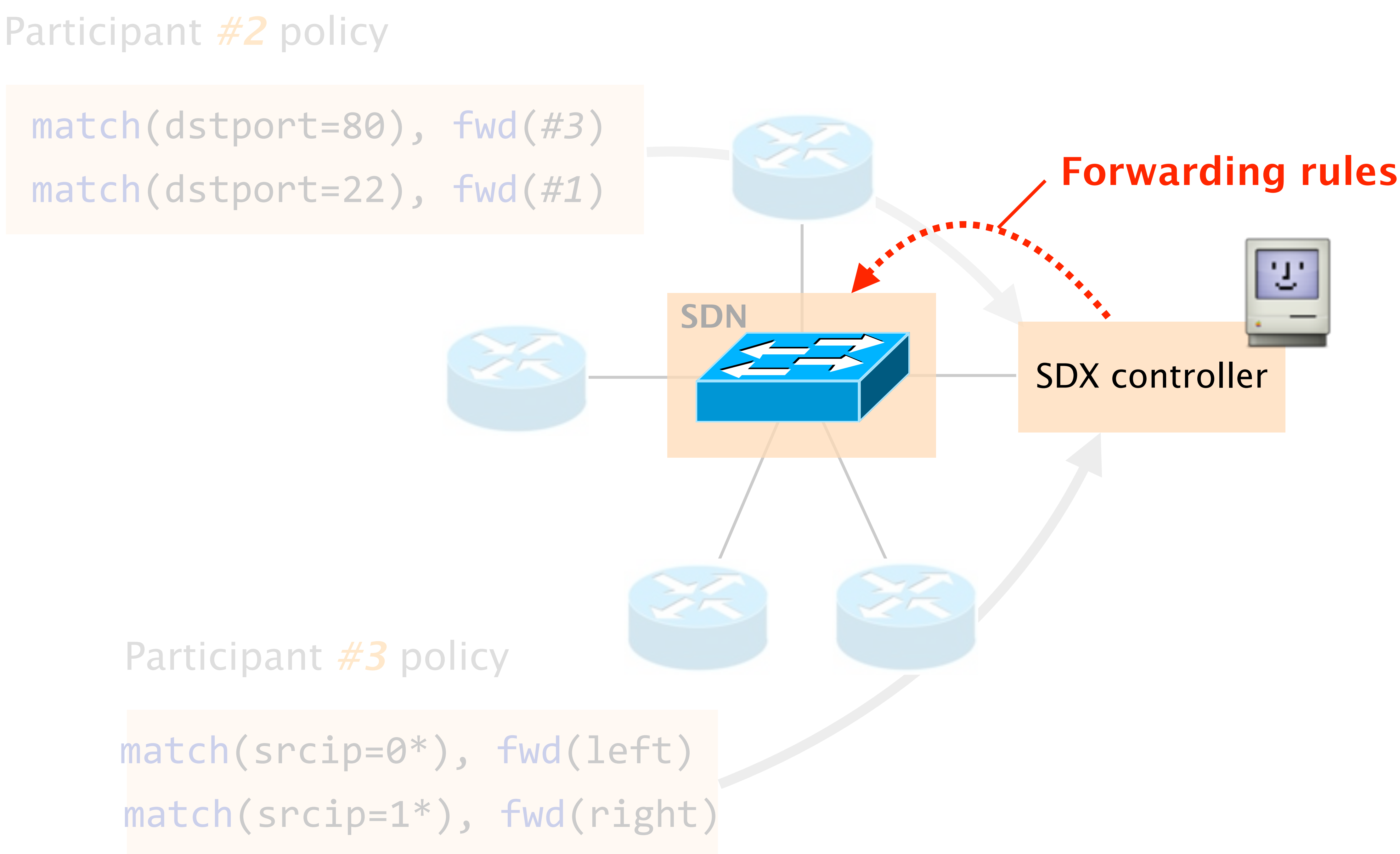
```
match(dstport=80), fwd(#3)
match(dstport=22), fwd(#1)
```



Participant #3 policy

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

The controller compiles all the policies into SDN forwarding rules



SDX compilation stage implements *each* participant policy in the data-plane

Ensuring isolation

Resolving conflict

Considering BGP

SDX compilation stage implements *each* participant policy in the data-plane

Ensuring isolation

Each participant controls one “virtual” switch

connected to participants it can communicate with

Resolving conflict

Considering BGP

SDX compilation stage implements *each* participant policy in the data-plane

Ensuring isolation

Resolving conflict

Considering BGP

Policies are composed

according to BGP
business relationships

SDX compilation stage implements *each* participant policy in the data-plane

Ensuring isolation

Resolving conflict

Considering BGP

Policies are augmented with BGP information

guarantee correctness and reachability

Bringing SDN to the Internet, one exchange point at the time

Architecture

programming model

2 **Scalability**

control- & data-plane

Applications

inter domain bonanza

The SDX platform faces scalability challenges in both the data- and in the control-plane

data-plane

space

control-plane

time

data-plane

space

512k prefixes, 500+ participants,
potentially *10^9* of forwarding rules

control-plane

time

forwarding rules must be updated
dynamically according to BGP

To scale, the SDX platform leverages *existing infrastructure & domain-specific knowledge*

data-plane

space

aggregate rules,
on *existing* routers

control-plane

time

leverage
policy structure

data-plane

space

aggregate rules,
on *existing* routers

control-plane

time

SDX groups IP prefixes according to their behavior through the fabric

- policies are prefix-based

  just the way the Internet works

- forwarding actions are shared for a lot of prefixes

  e.g., *all* prefixes advertised by *X*

SDX groups IP prefixes according to their behavior through the fabric

- policies are prefix-based

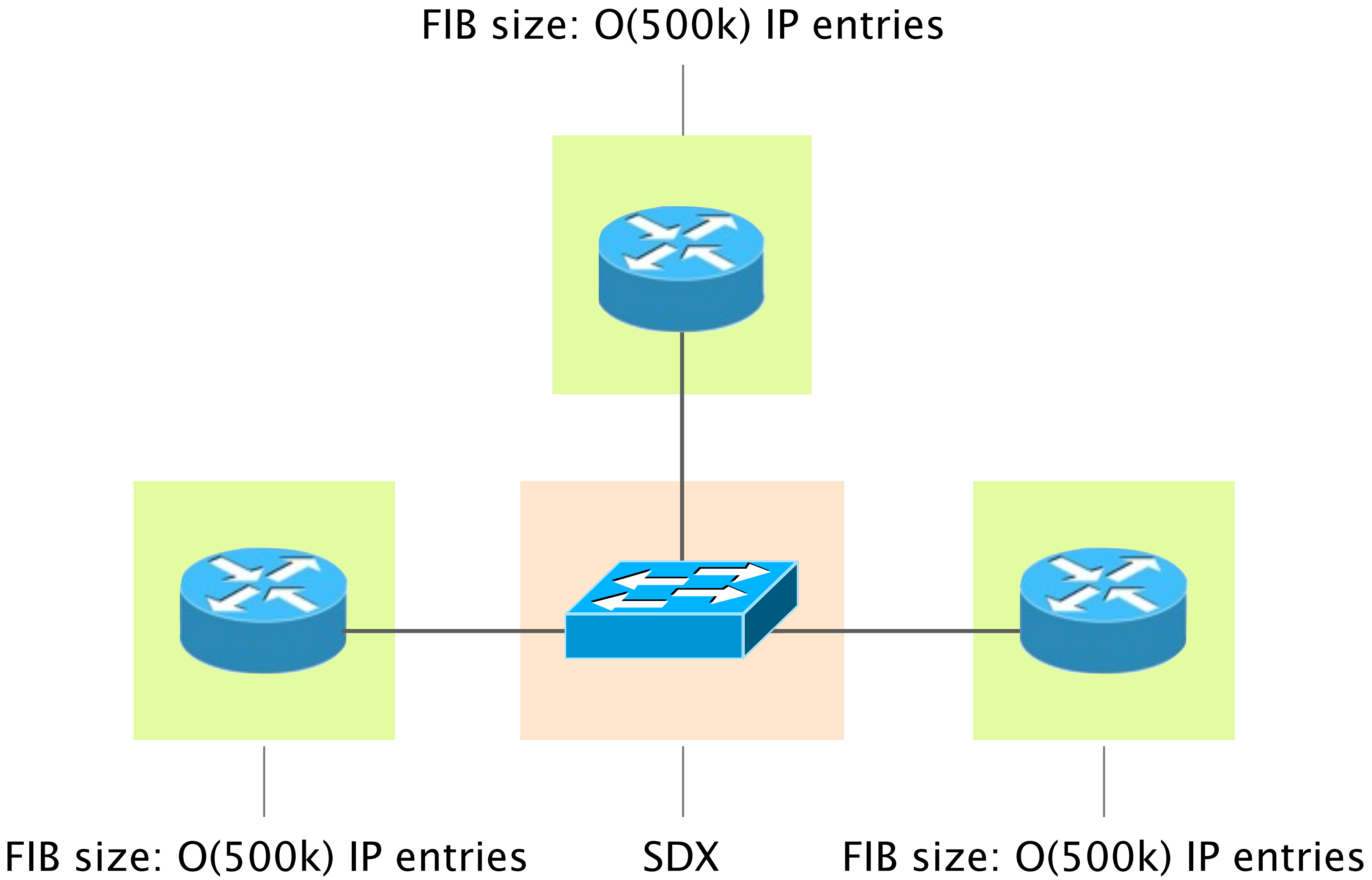
  just the way the Internet works

- forwarding actions are shared for a lot of prefixes

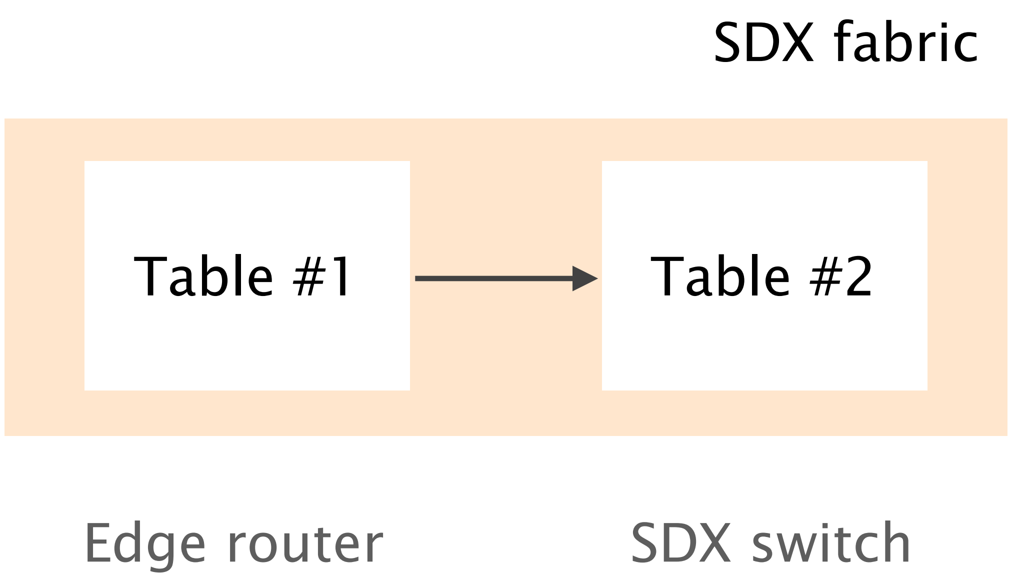
  e.g., all prefixes advertised by *X*

- group prefixes by equivalence class

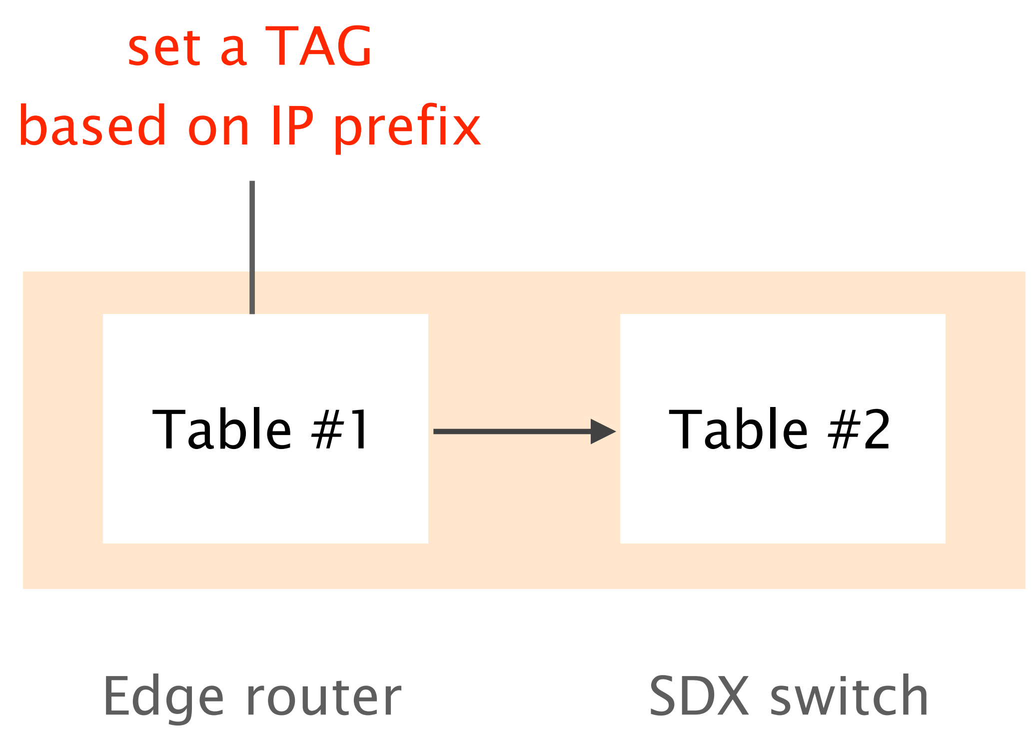
SDX leverages edge routers to map packets to their equivalence class



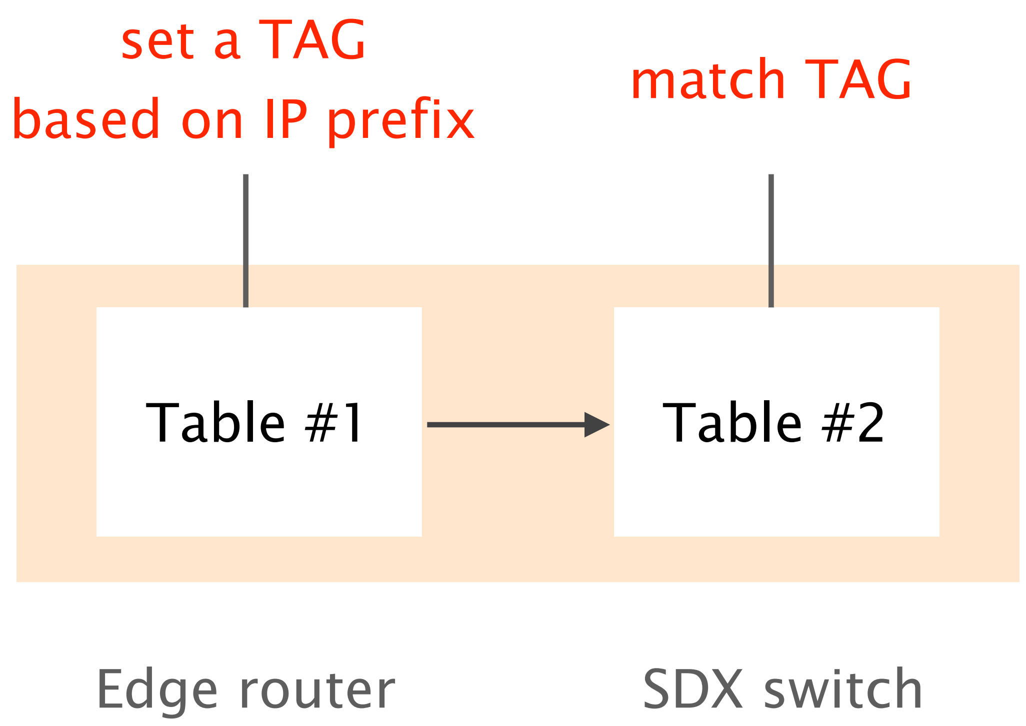
SDX considers edge routers' FIB as the first stage of a multi-stage FIB



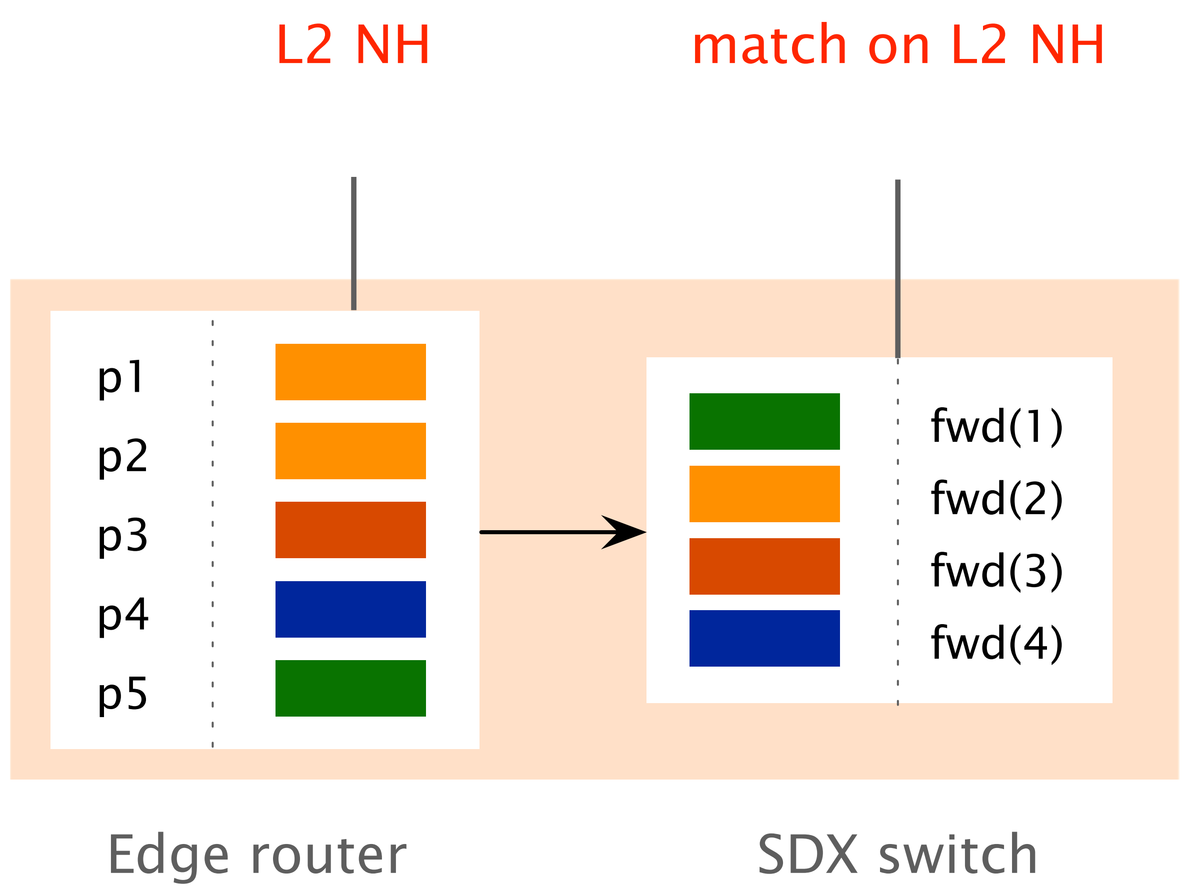
Routers FIB match on the destination prefix and set a tag accordingly



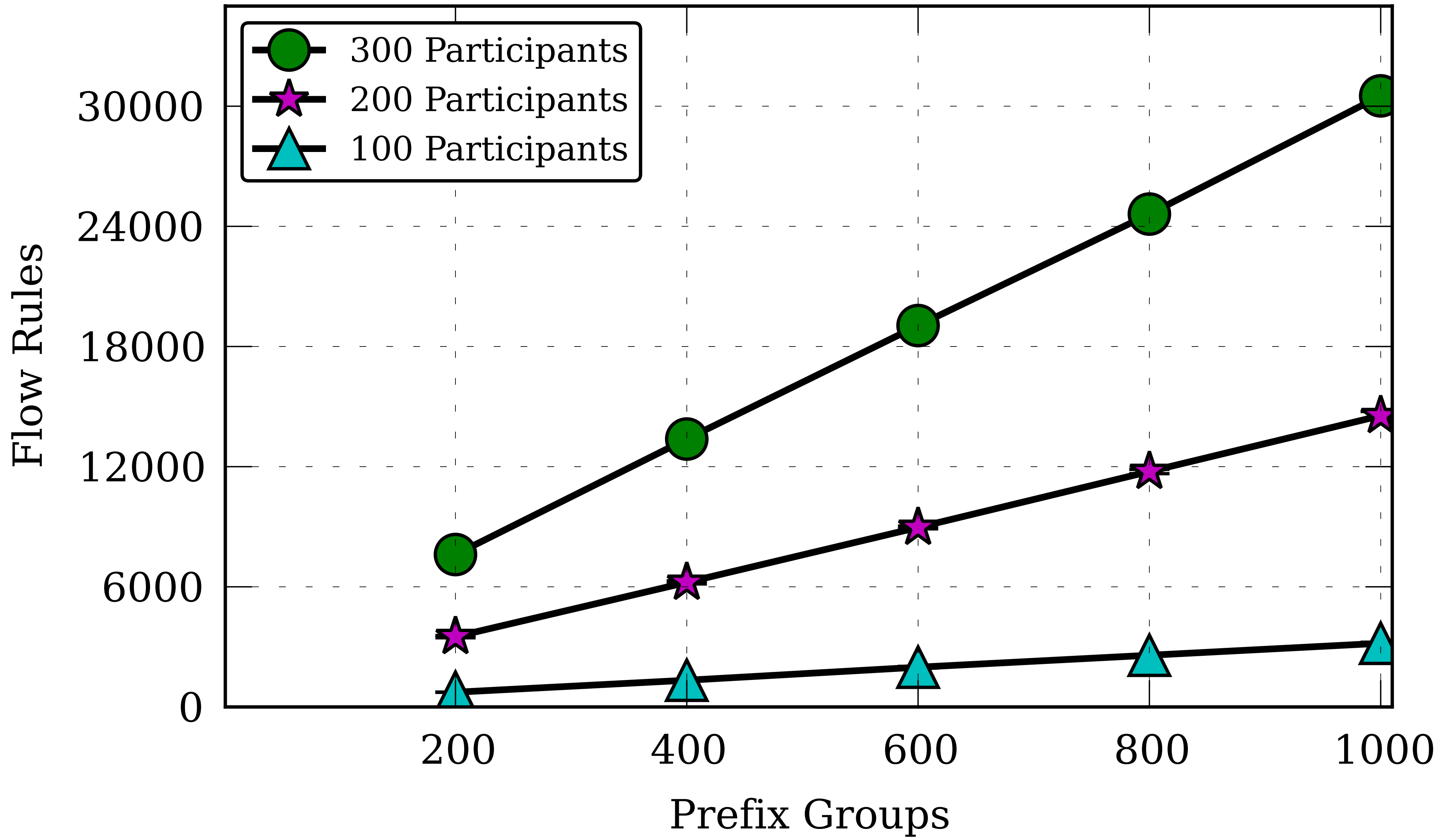
SDX FIB matches on the tag



SDX uses BGP NH as a provisioning interface and MAC addresses as tag in the data-plane



SDX accommodates policies
for 100+ participants, with less than 30k rules



data-plane

space

control-plane

time

leverage

policy structure

SDX policies share key characteristics

Static	disjointness
Dynamic	locality
	burstiness

SDX policies share key characteristics

Static

disjointness

disjoint policies don't need to be composed

significant gain as composition is costly

Dynamic

locality

burstiness

SDX policies share key characteristics

Static disjointness

Dynamic **locality**

burstiness

policy updates usually impact few prefixes

75% of the updates affect no more than 3 prefixes

SDX policies share key characteristics

Static disjointness

Dynamic locality

burstiness

policy updates are separated by large periods of inactivity

In 75% of the case, updates are separated by 10s or more

These characteristics enable an efficient, 2-stage compilation algorithm

Stage 1 — *Fast*, non-optimal algorithm upon updates

can install more forwarding rules than required

Stage 2 — *Slow*, but optimal algorithm in background

regroup rules according to forwarding behavior

These characteristics enable an efficient, 2-stage compilation algorithm

Fast, non-optimal algorithm upon updates

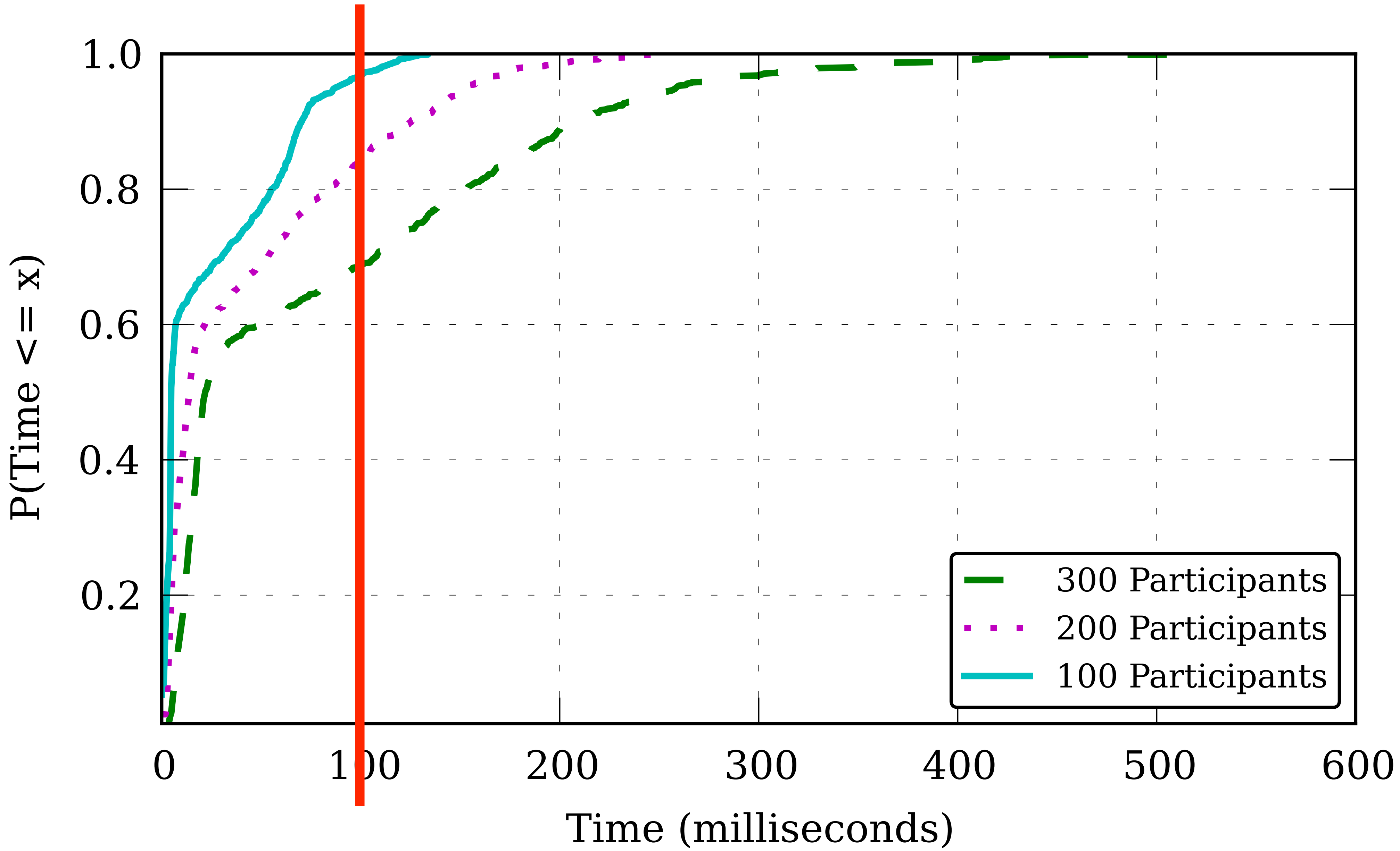
can install more forwarding rules than required

Slow, but optimal algorithm in background

regroup rules according to forwarding behavior

Time *vs* Space trade-off

In most cases, the SDX takes <100 ms
to recompute the entire policy



Bringing SDN to the Internet, one exchange point at the time

Architecture

programming model

Scalability

control- & data-plane

3 Applications

inter domain bonanza

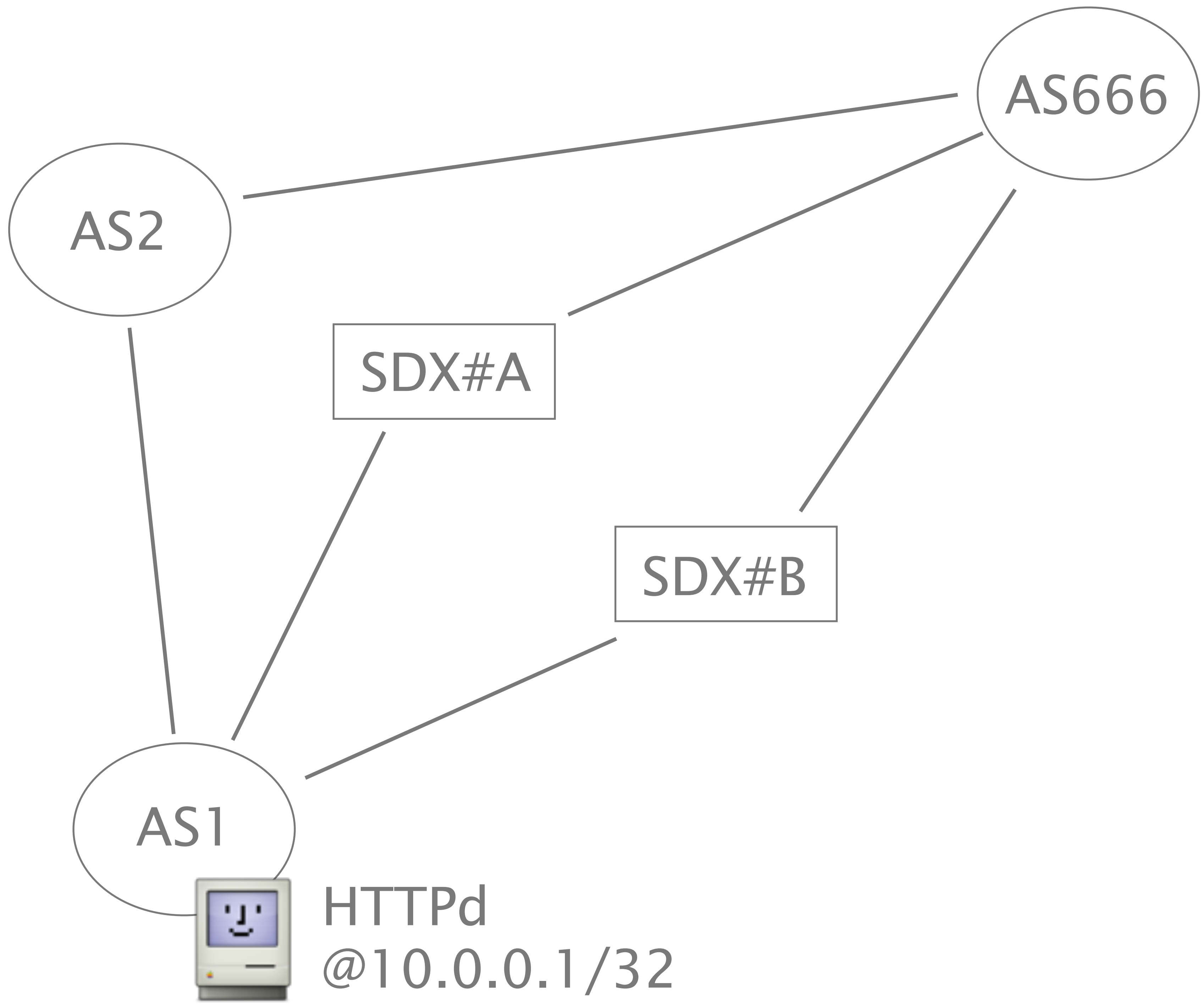
SDX enables a wide range of novel applications

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

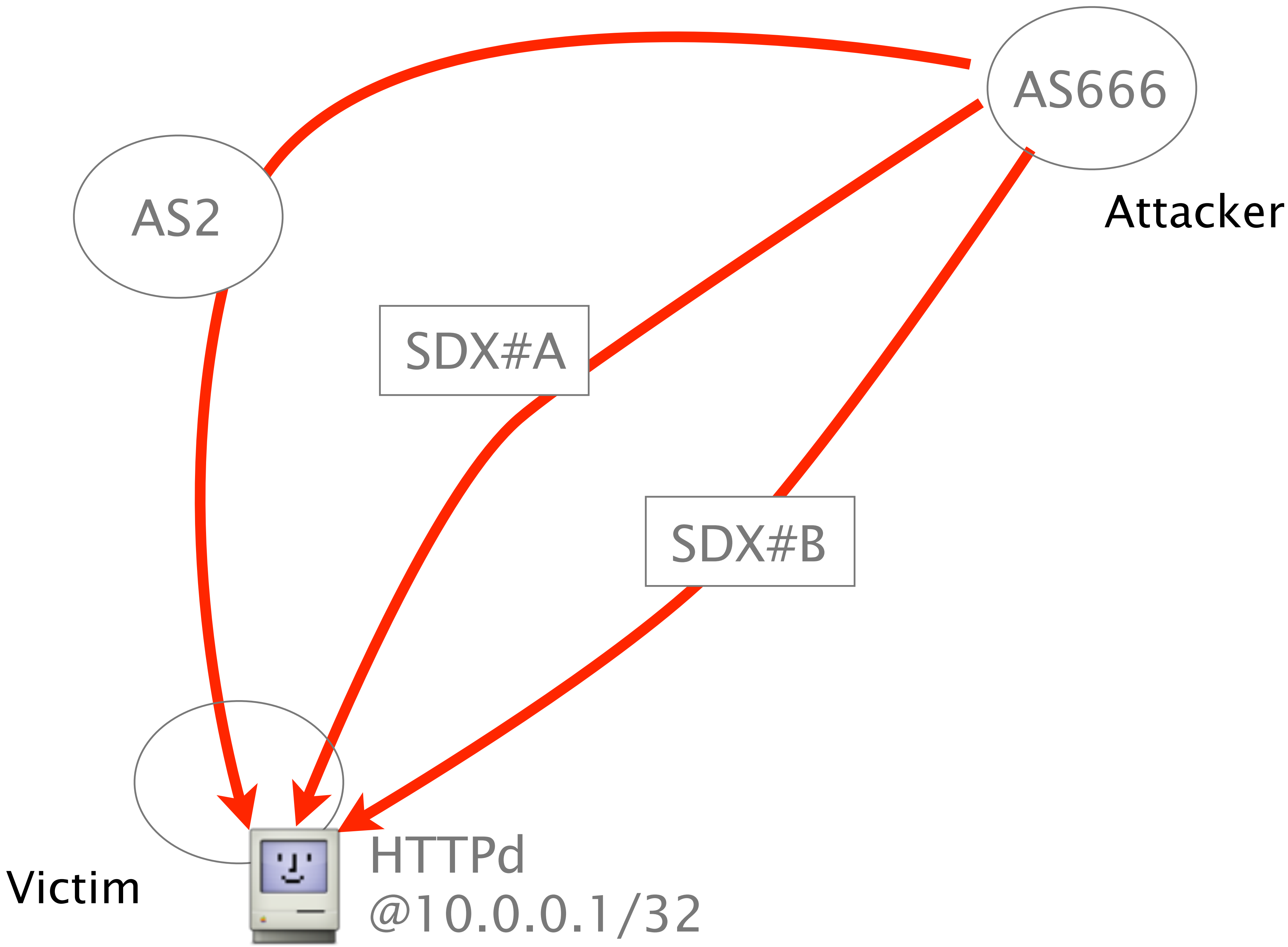
SDX enables a wide range of novel applications

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

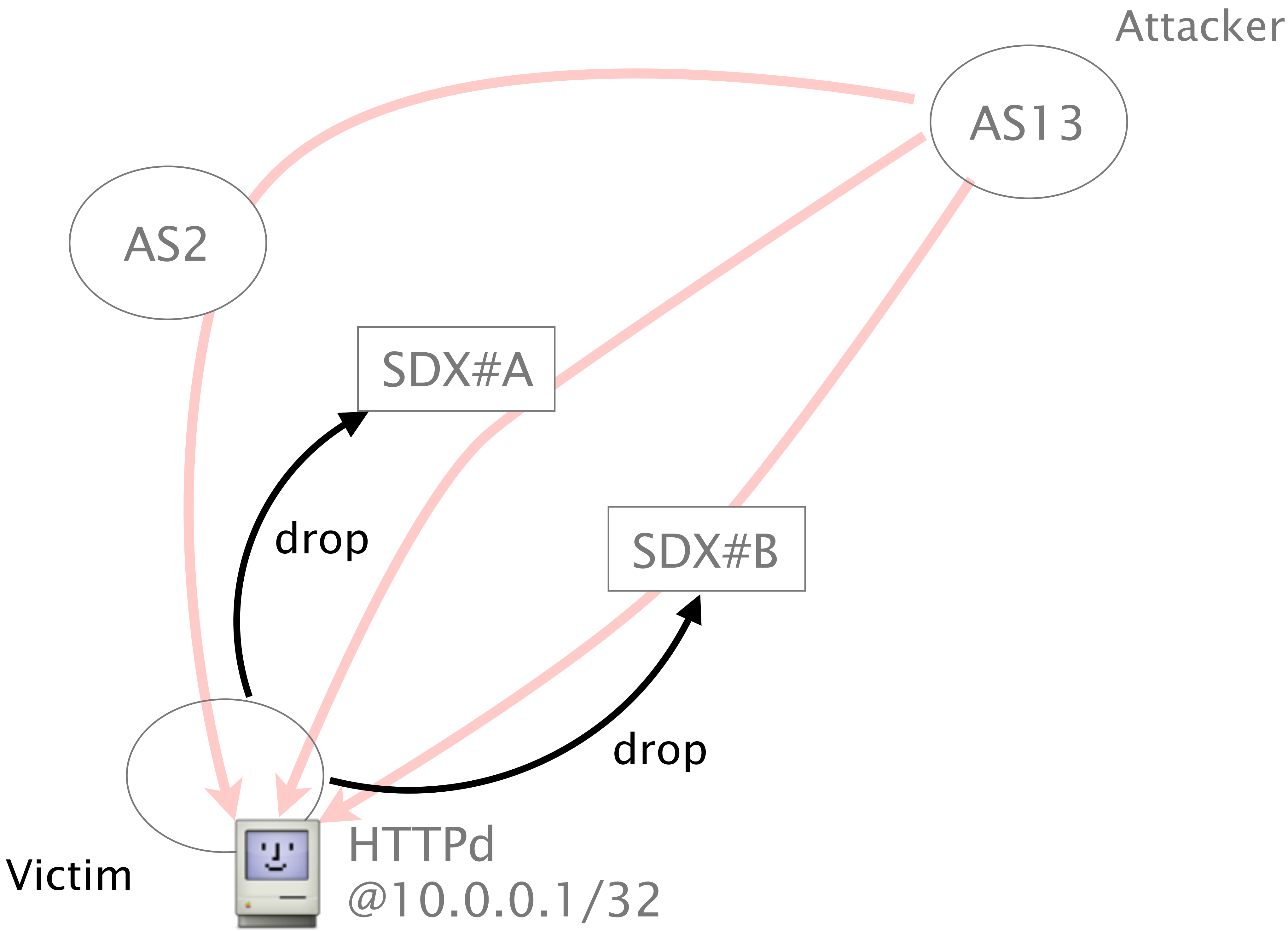
SDX can help mitigating DDoS attacks, closer to the source



AS1 is victim of a DDoS attack targeting its web server



AS1 remotely installs
drop policies in all SDXes



AS1 remotely installs *drop* policies in all SDXes

AS1 policy

```
match(srcip=*, dstip=10.0.01/32, dstport=80) >> drop()
```

SDX policies are targeted, hence other services stay reachable

AS1 policy

```
match(srcip=*, dstip=10.0.01/32, dstport=80) >> drop()
```

single IP

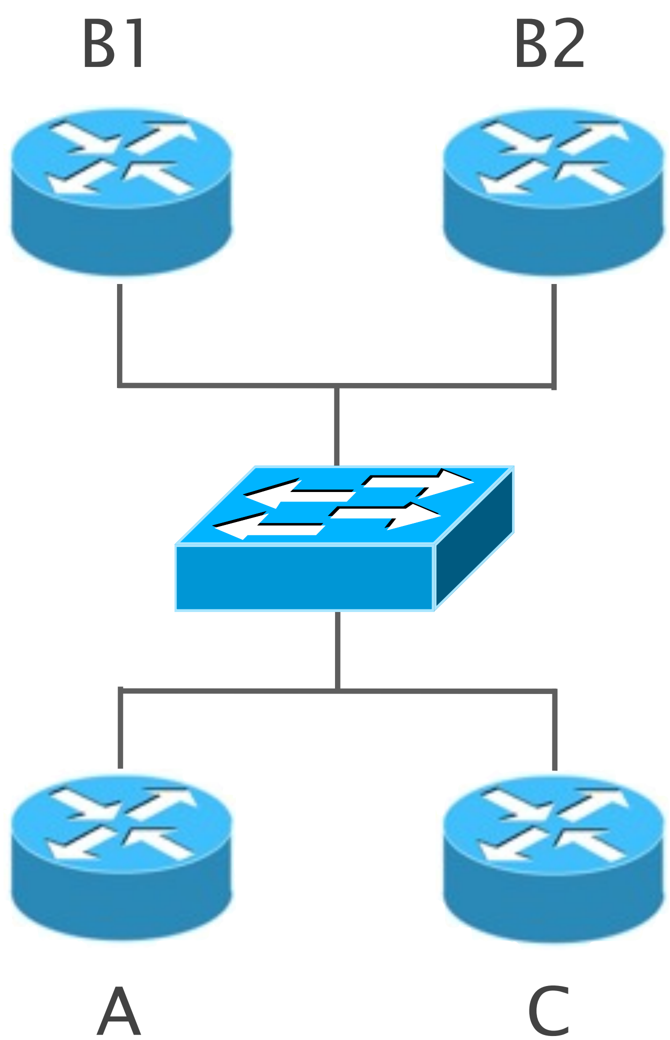
single service

SDX enables a wide range of novel applications

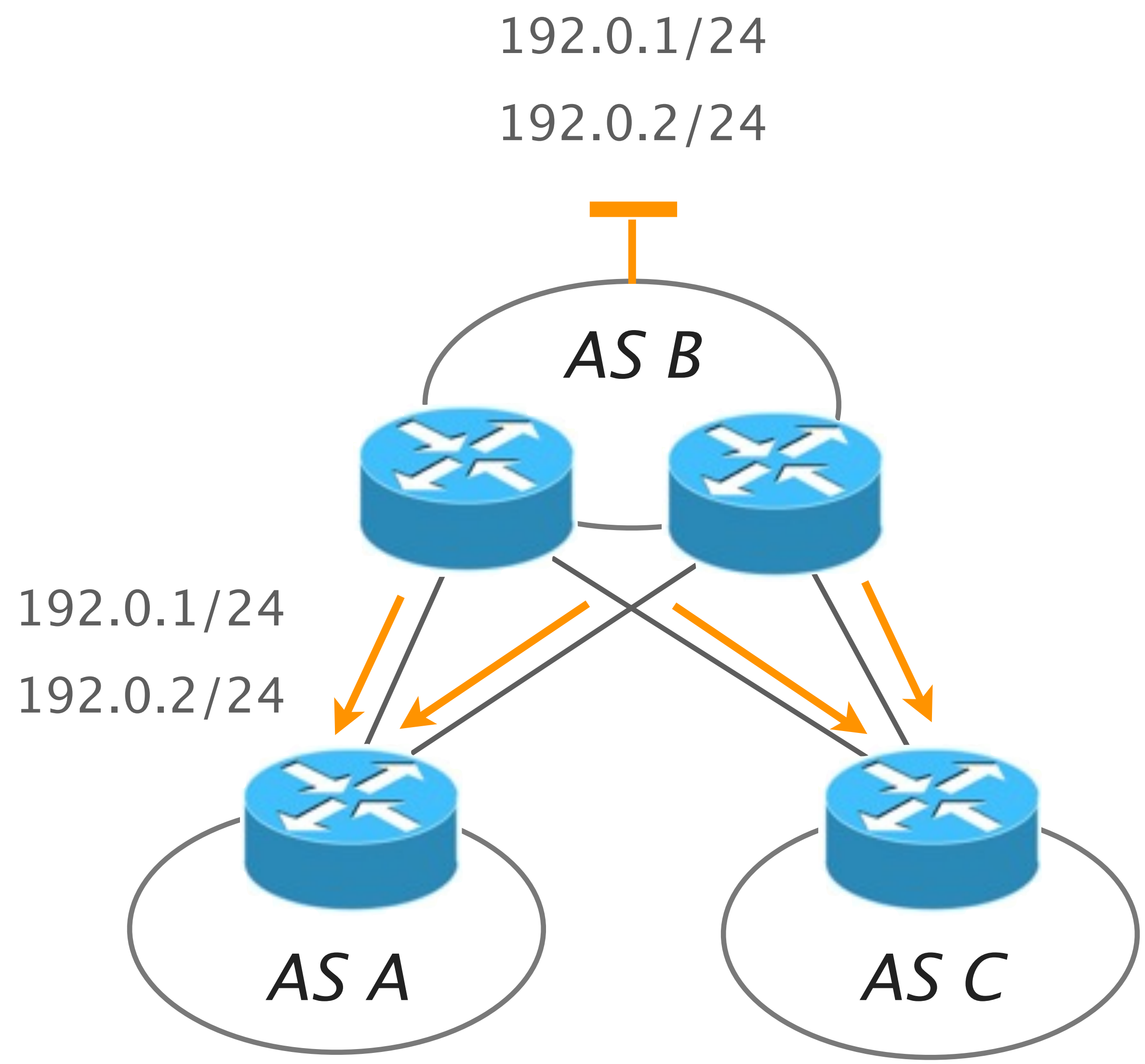
security	Prevent/block policy violation Prevent participants communication Upstream blocking of DoS attacks
forwarding optimization	Middlebox traffic steering Traffic offloading **Inbound Traffic Engineering** Fast convergence
peering	Application-specific peering
remote-control	Influence BGP path selection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

SDX can improve inbound traffic engineering

Given an IXP Physical Topology and a BGP topology,



IXP Fabric

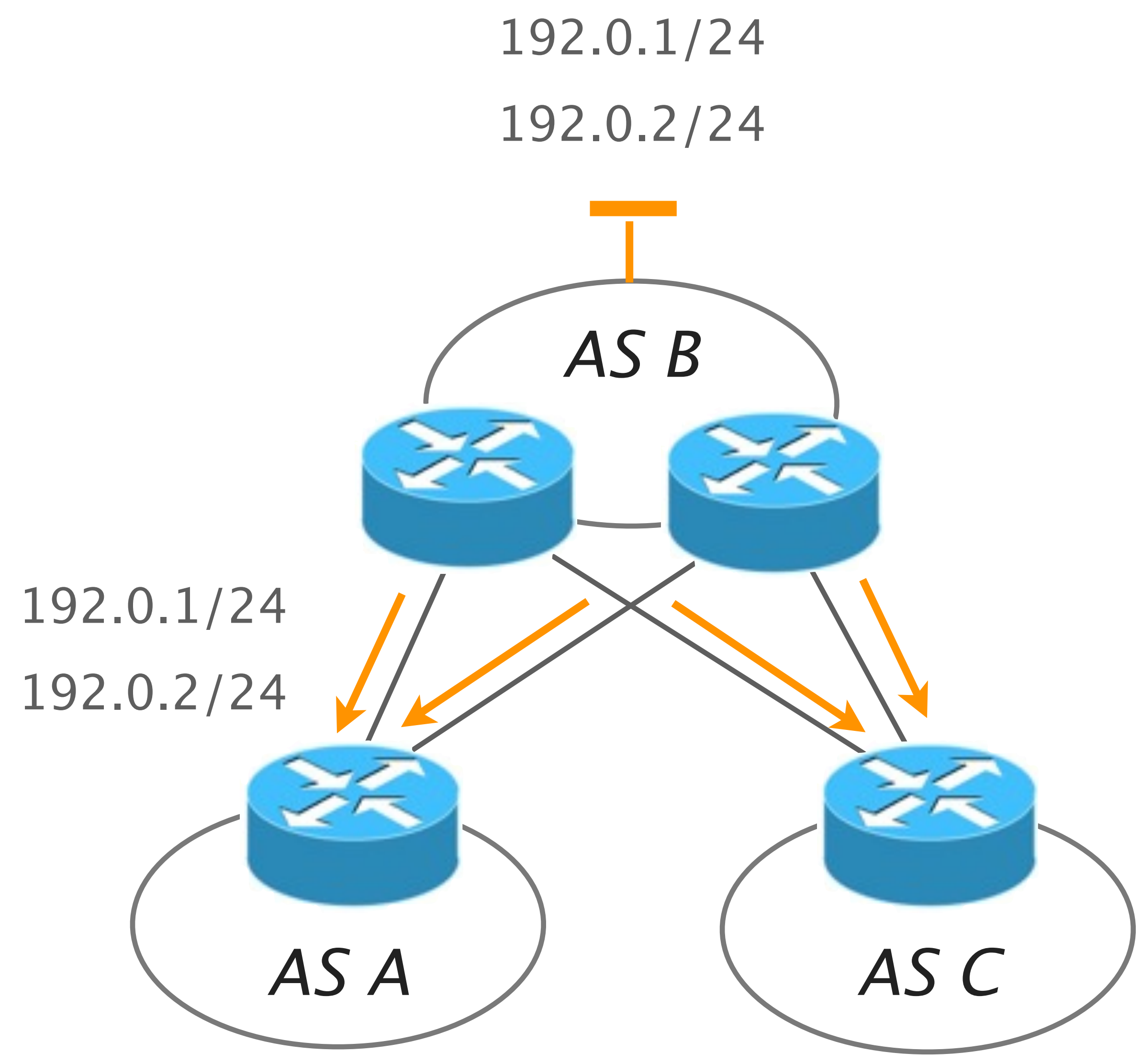


BGP topology

Given an IXP Physical Topology and a BGP topology, Implement B's inbound policies

B's inbound policies

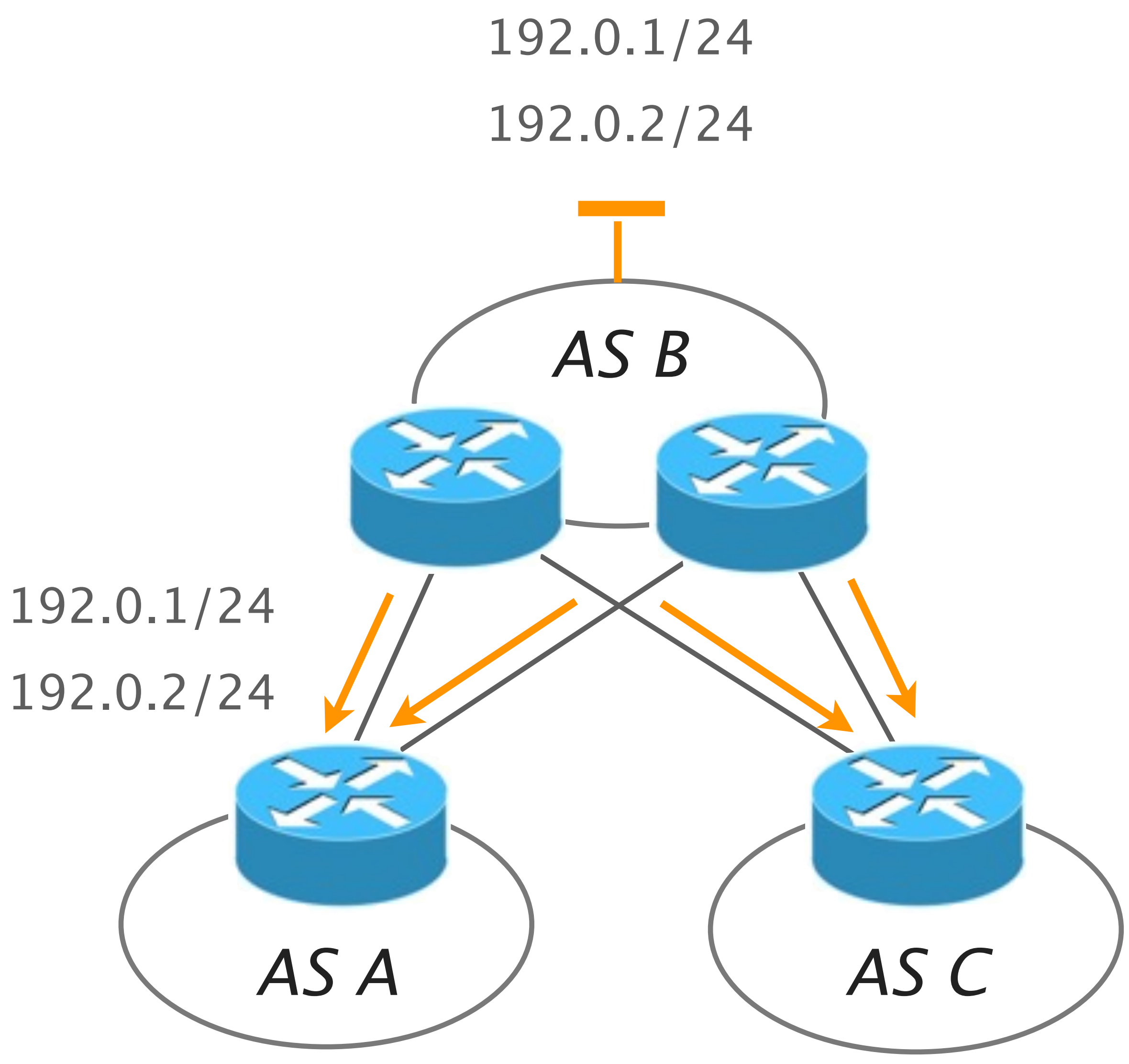
to	from	*receive on*
192.0.1/24	A	left
192.0.2/24	C	right
192.0.2/24	ATT_IP	right
192.0.1/24	*	right
192.0.2/24	*	left



How do you that with BGP?

B’s inbound policies

to	from	*receive on*
`192.0.1/24`	`A`	left
`192.0.2/24`	`C`	right
`192.0.2/24`	`ATT_IP`	right
`192.0.1/24`	`*`	right
`192.0.2/24`	`*`	left



It is hard

BGP provides few knobs to influence remote decisions

Implementing such a policy is configuration-intensive

using AS-Path prepend, MED, community tagging, etc.

... and even impossible for some requirements

BGP policies **cannot** influence remote decisions based on source addresses

to	**from**	receive on
`192.0.2.0/24`	**`ATT_IP`**	right

In any case, the outcome is unpredictable

Implementing such a policy is configuration-intensive

using AS-Path prepend, MED, community tagging, etc.

There is *no guarantee* that remote parties will comply

one can only “influence” remote decisions

Networks engineers have no choice but to “try and see”

which makes it impossible to adapt to traffic pattern

With SDX, implement B's inbound policy is easy

SDX policies give any participant *direct* control on its forwarding paths

to	from	fwd
192.0.1/24	A	left
192.0.2/24	B	right
192.0.2/24	ATT_IP	right
192.0.1/24	*	right
192.0.2/24	*	left

B's SDX Policy

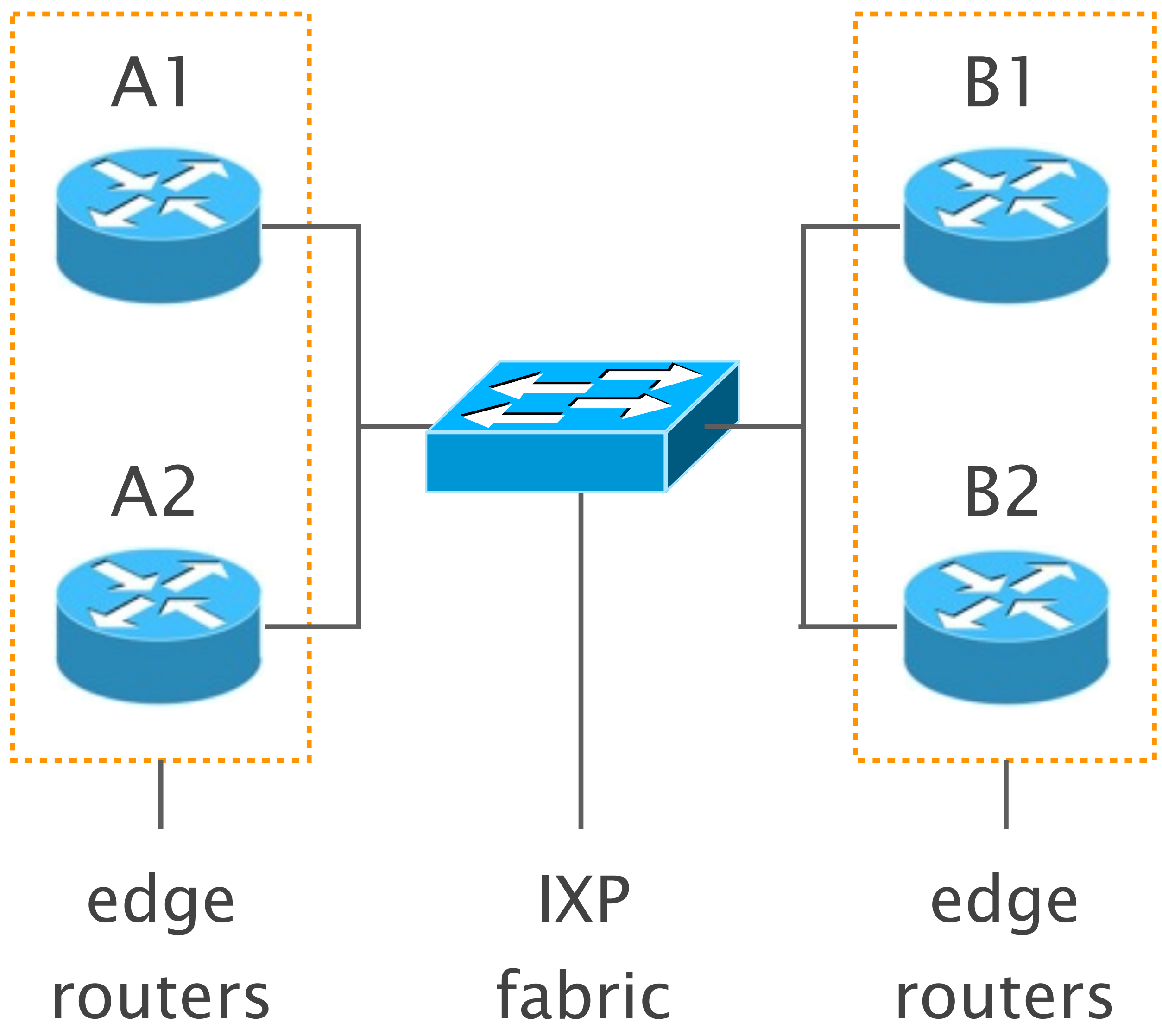
```
match(dstip=192.0.1/24, srcmac=A), fwd(L)
match(dstip=192.0.2/24, srcmac=B), fwd(R)
match(dstip=192.0.2/24, srcip=ATT), fwd(R)
match(dstip=192.0.1/24), fwd(R)
match(dstip=192.0.2/24), fwd(L)
```

SDX enables a wide range of novel applications

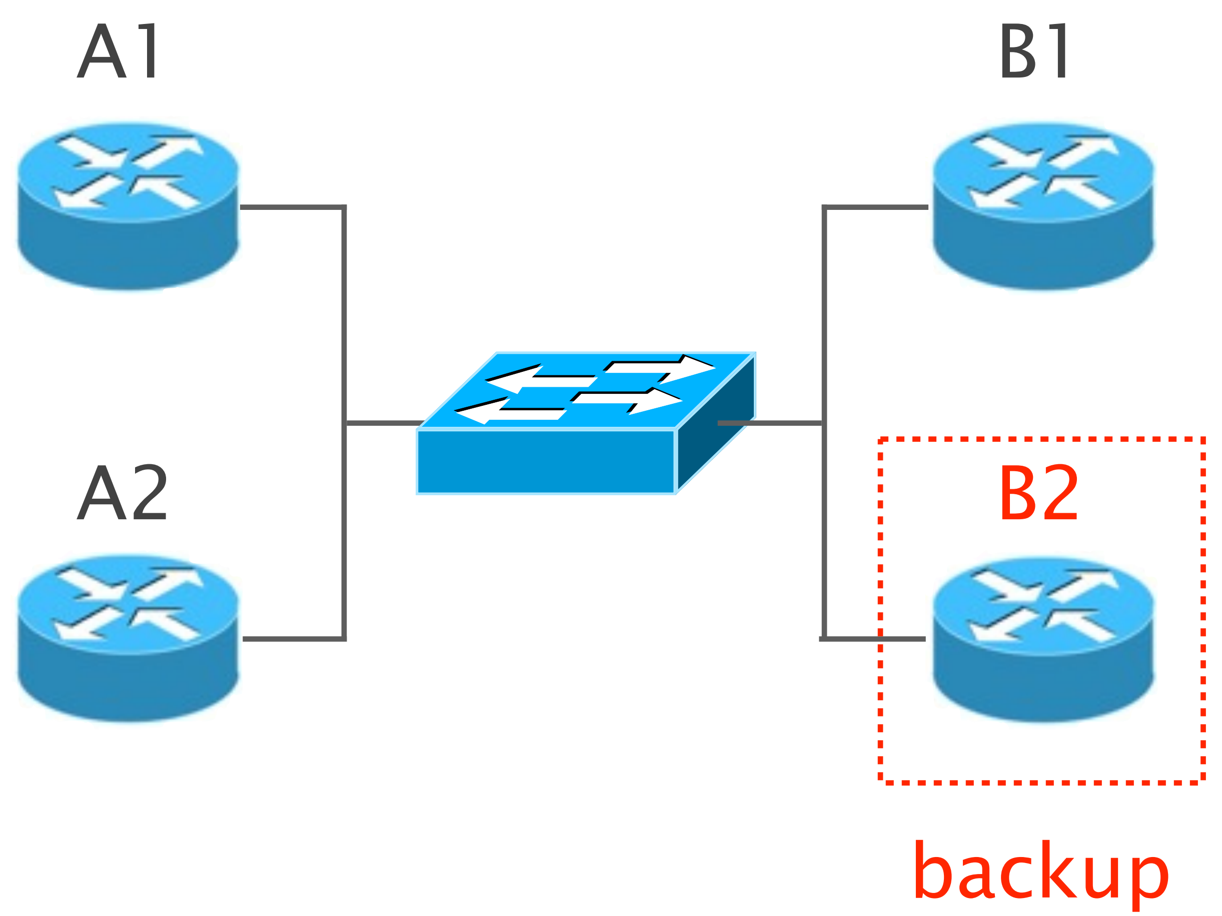
security	Prevent/block policy violation Prevent participants communication Upstream blocking of DoS attacks
forwarding optimization	Middlebox traffic steering Traffic offloading Inbound Traffic Engineering **Fast convergence**
peering	Application-specific peering
remote-control	Influence BGP path selection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

BGP is pretty slow to converge upon peering failure

Let's consider a example with 2 networks,
A and B, with B being the provider of A



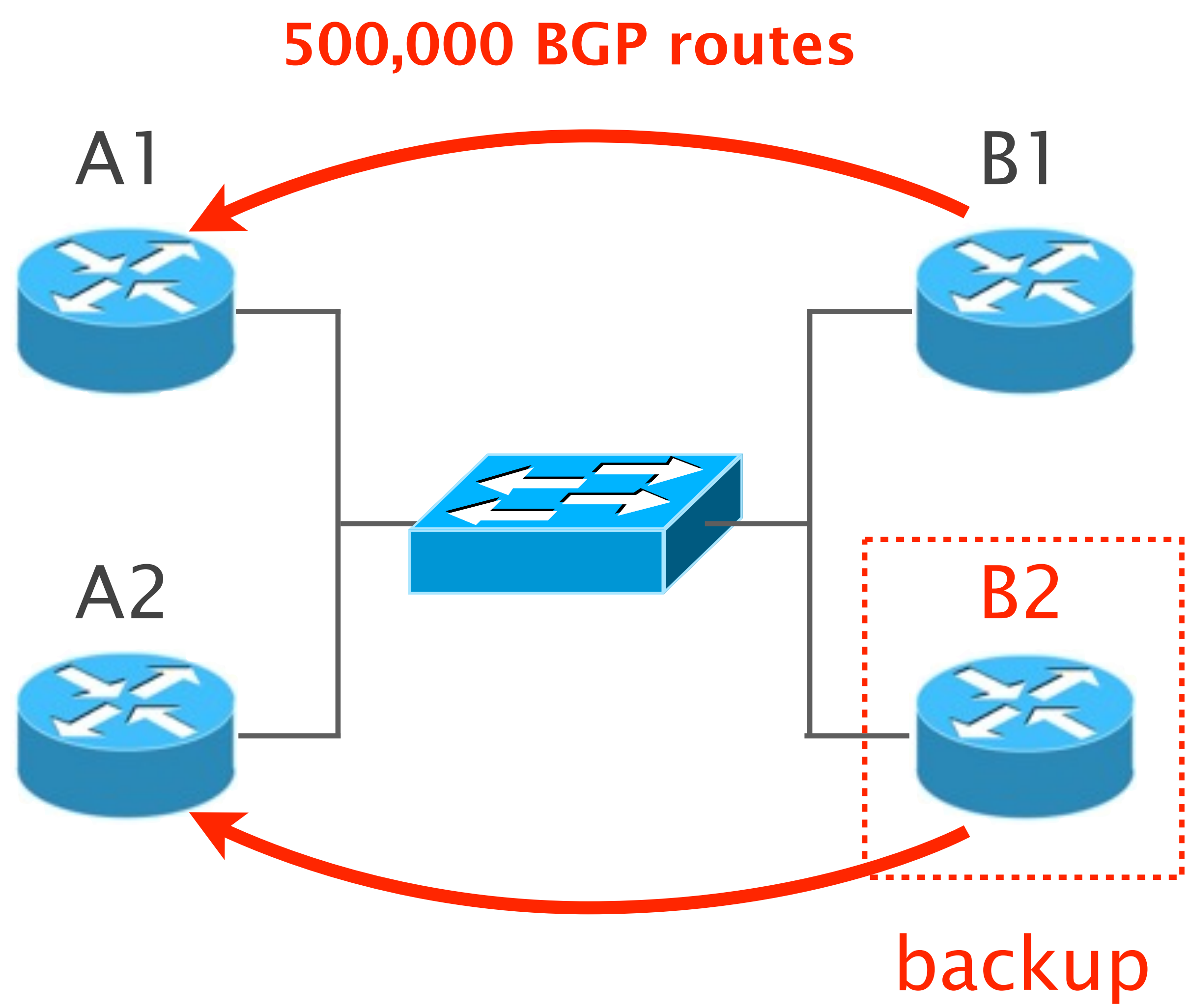
Router B2 is a backup router,
it may be used only upon B1's failure



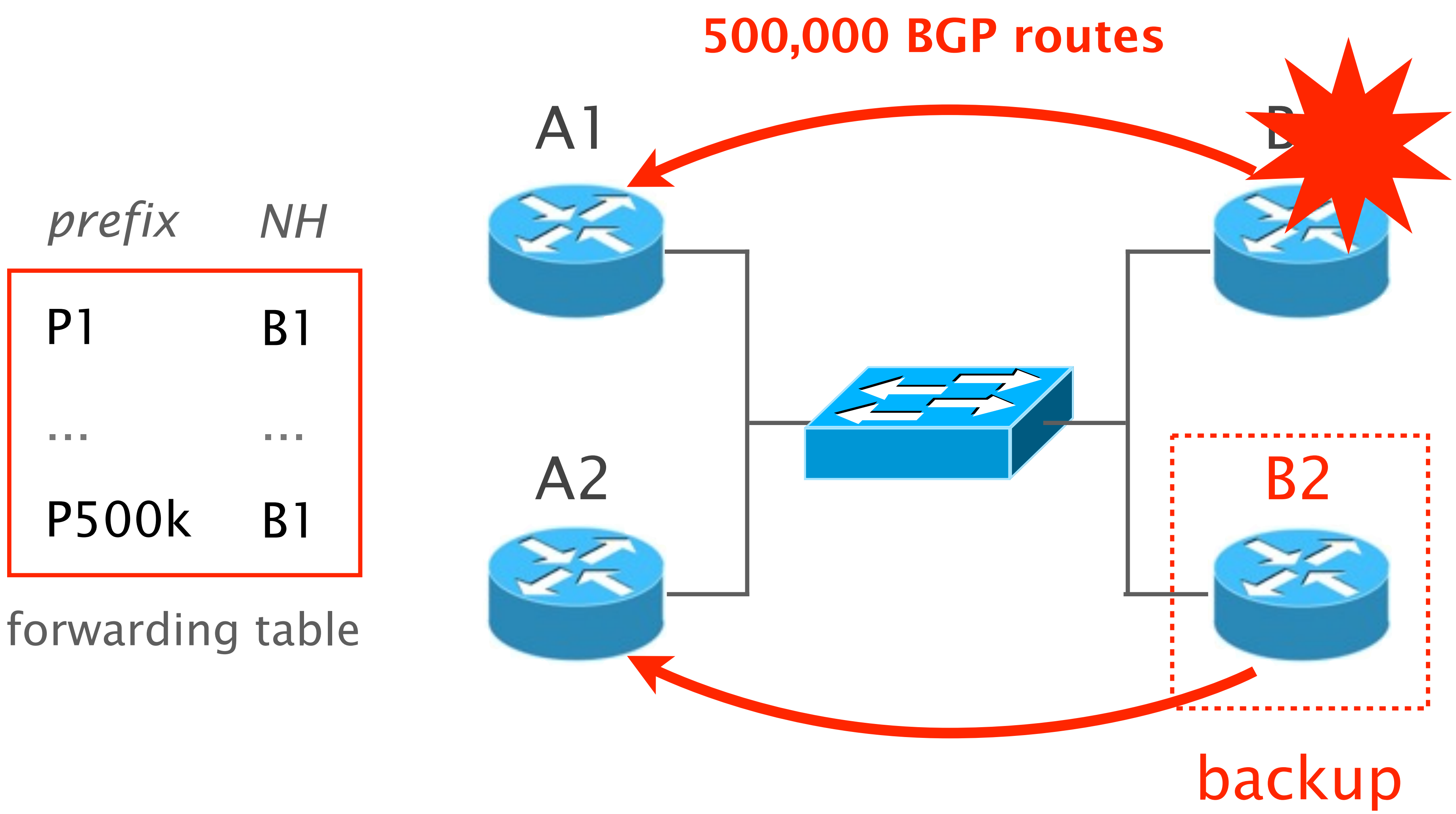
Both A1 and A2 prefer the routes received from B1 and install them in their FIB

prefix	*NH*
P1	B1
...	...
P500k	B1

forwarding table



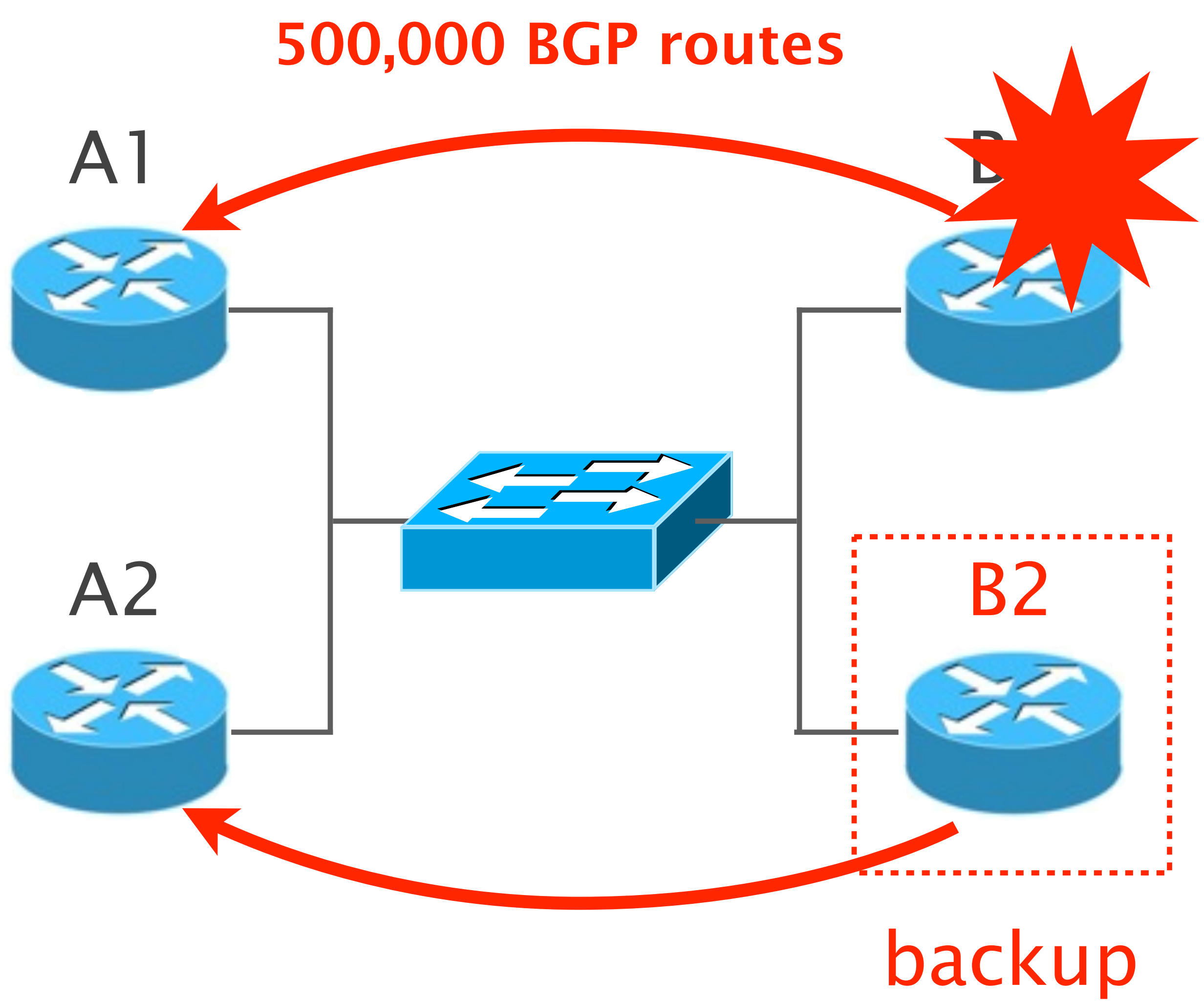
Upon B1's failure, A1 and A2 must update every single entry in their FIB (~500k entries)



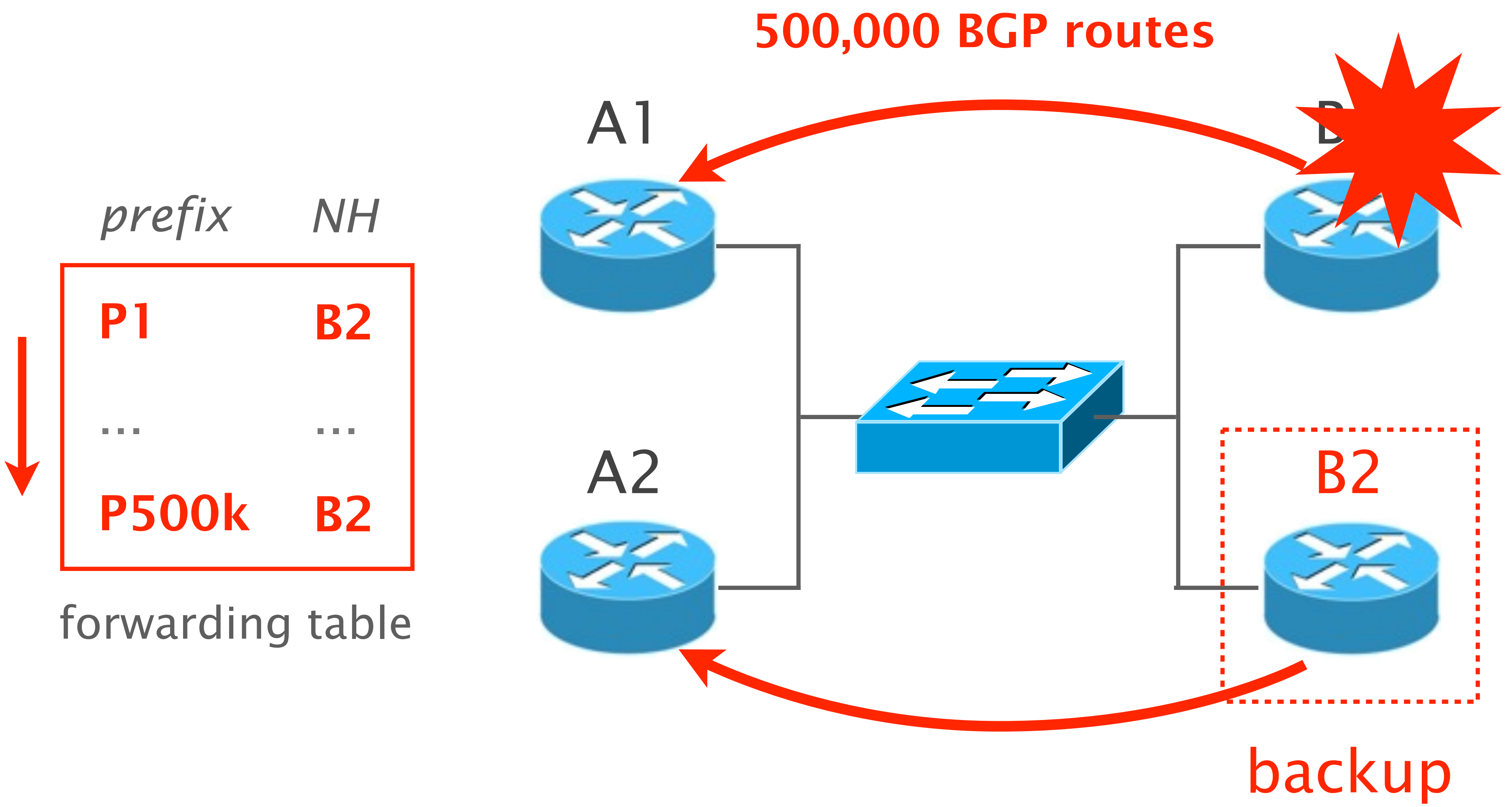
Upon B1's failure, A1 and A2 must update every single entry in their FIB (~500k entries)

prefix	*NH*
P1	B2
...	...
P500k	B1

forwarding table



Upon B1's failure, A1 and A2 must update every single entry in their FIB (~500k entries)



On most routers, FIB updates are performed linearly, entry-by-entry, leading to *slow* BGP convergence

convergence time   500k entries * 150 $\frac{\mu\text{secs}}{\text{entry}}$

average time
to update one entry

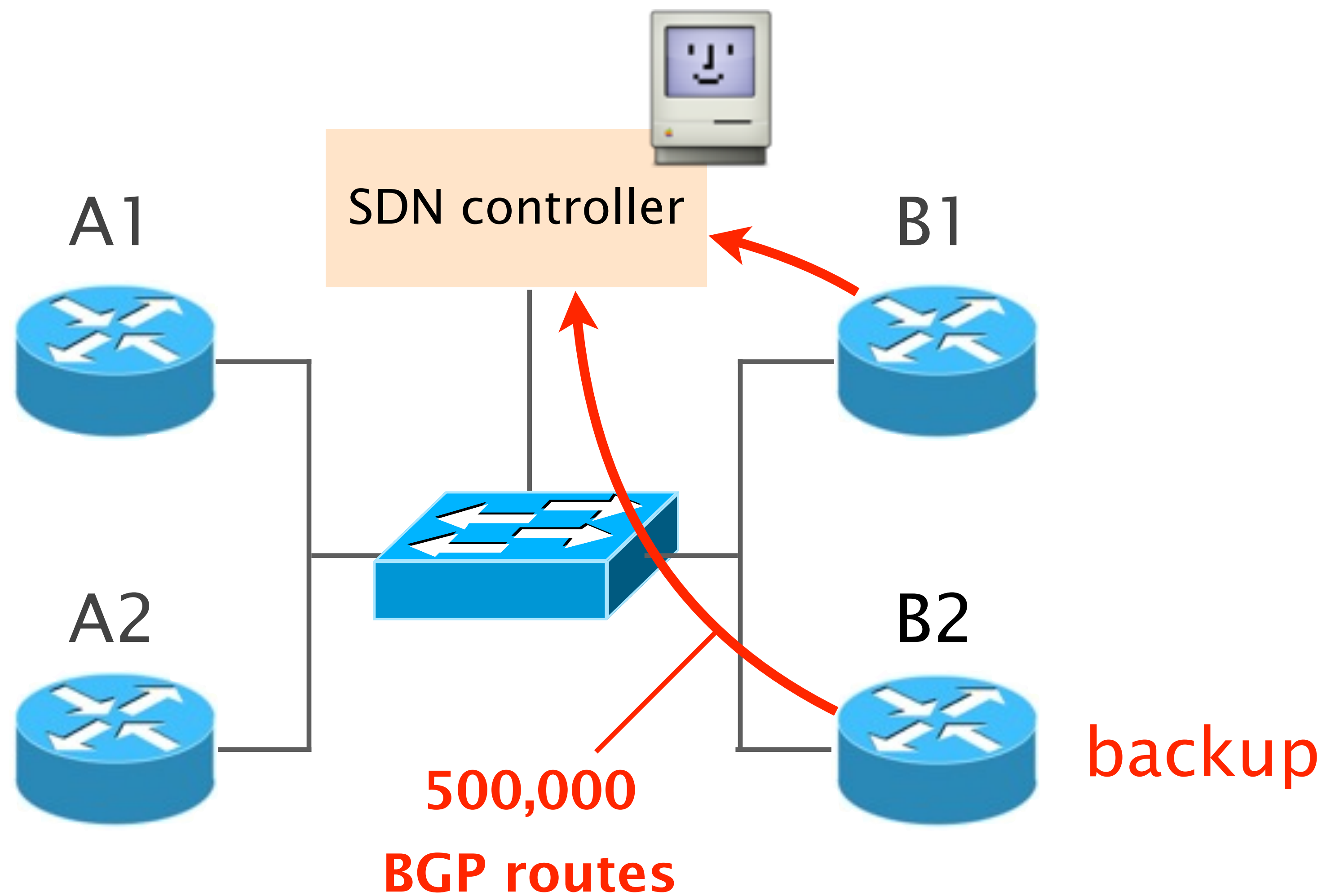
On most routers, FIB updates are performed linearly, entry-by-entry, leading to *slow* BGP convergence

convergence time $\quad$ 500k entries * 150 $\frac{\mu\text{secs}}{\text{entry}}$ = **O(75) seconds**

average time
to update one entry

With SDX, sub-second peering convergence can be achieved with any router

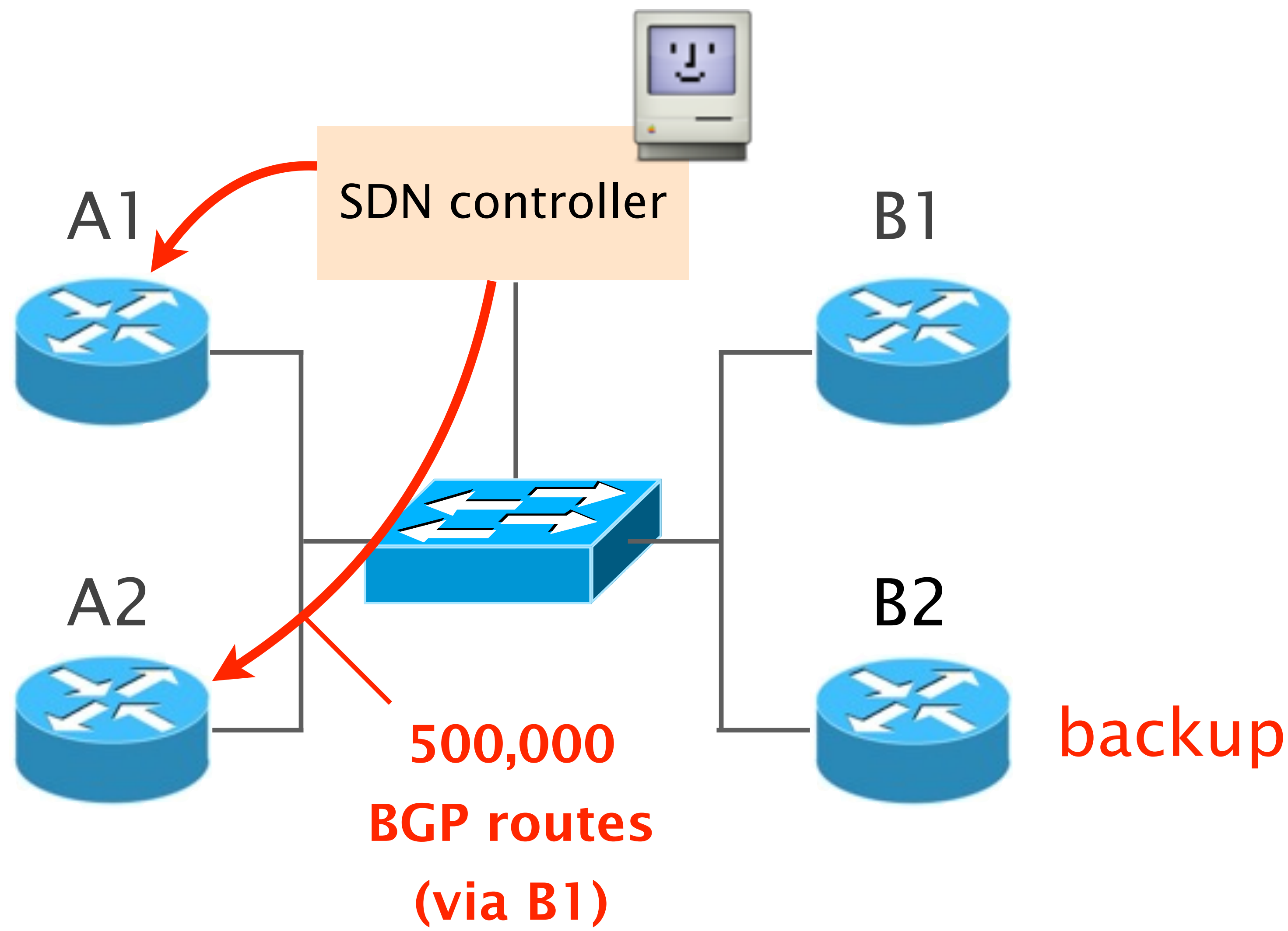
When receiving multiple routes, the SDX controller pre-computes a backup NH for each prefix



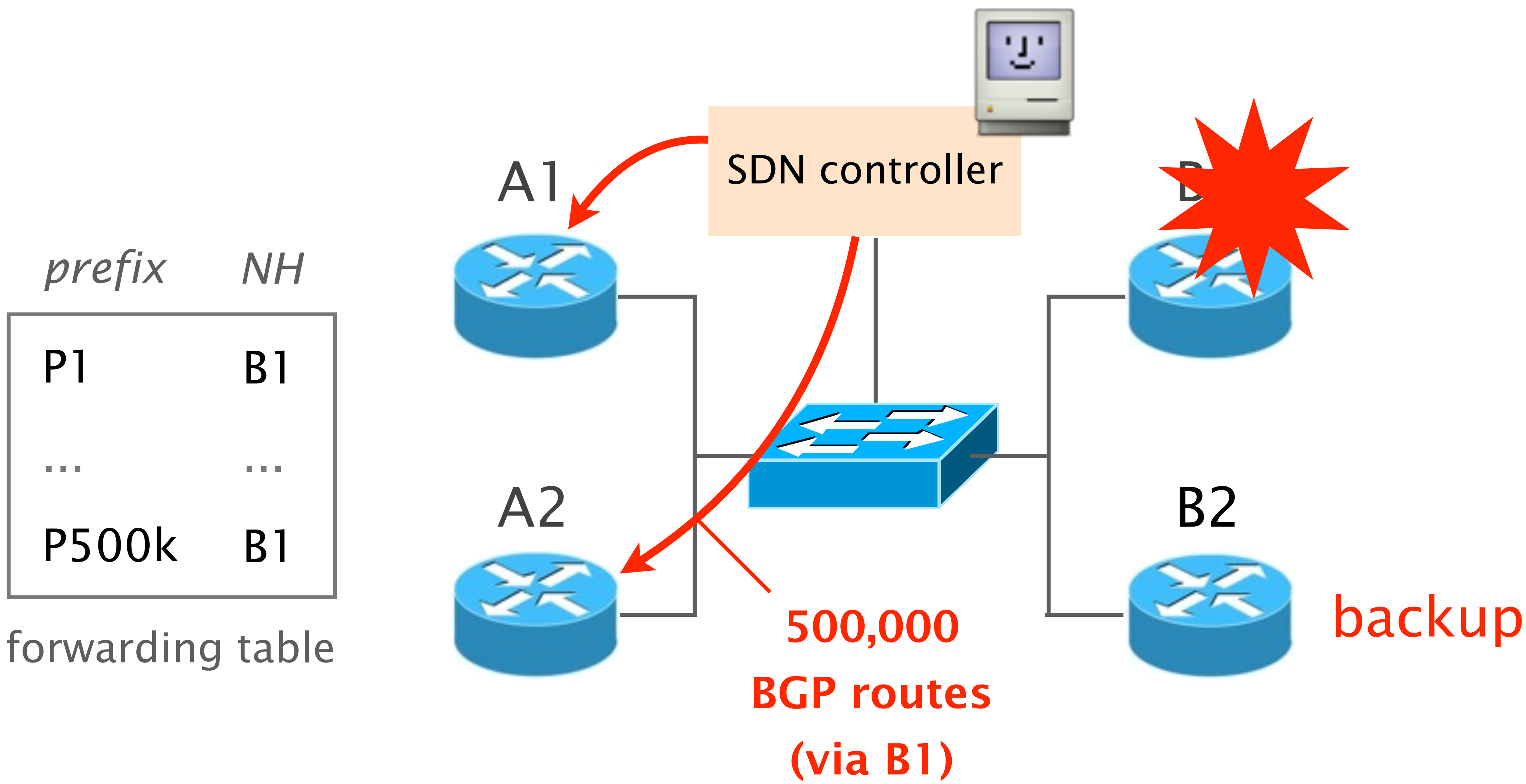
When receiving multiple routes, the SDX controller pre-computes a backup NH for each prefix

prefix	*NH*
P1	B1
...	...
P500k	B1

forwarding table



Upon a peer failure, the SDX controller directly pushes next-hop rewrite rules

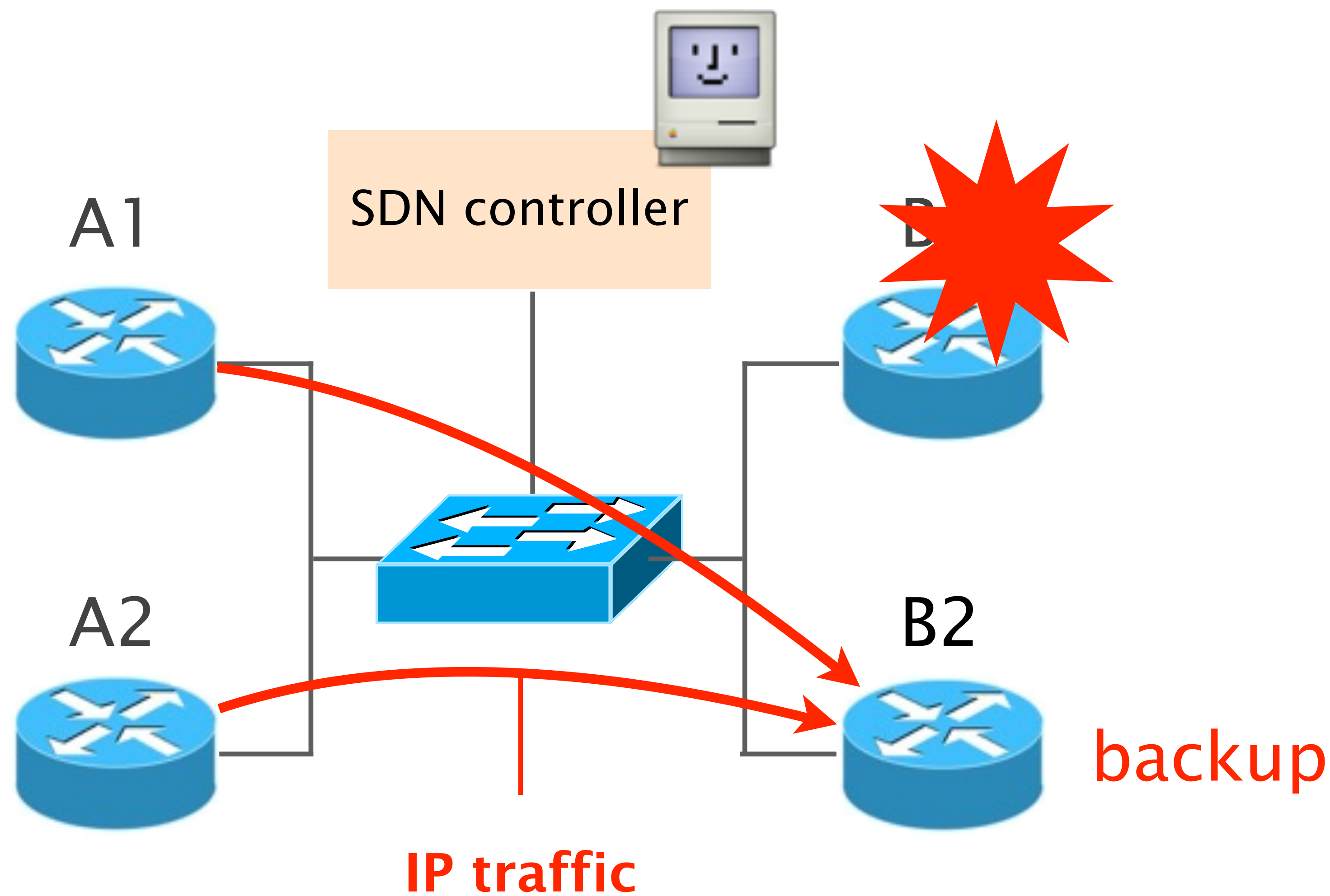


```
match(srcmac:A1, dstmac:B1), rewrite(dstmac:B2), fwd(B2)
match(srcmac:A2, dstmac:B1), rewrite(dstmac:B2), fwd(B2)
```

All IP traffic immediately moves from B1 to B2, independently of the number of FIB updates

prefix	*NH*
P1	B1
...	...
P500k	B1

forwarding table



SDX data-plane can enable sub-second, prefix-independent BGP convergence

controller
communication time

convergence time   # edge entries * 150 $\frac{\mu secs}{entry}$ + 30~50 ms

average update time per entry

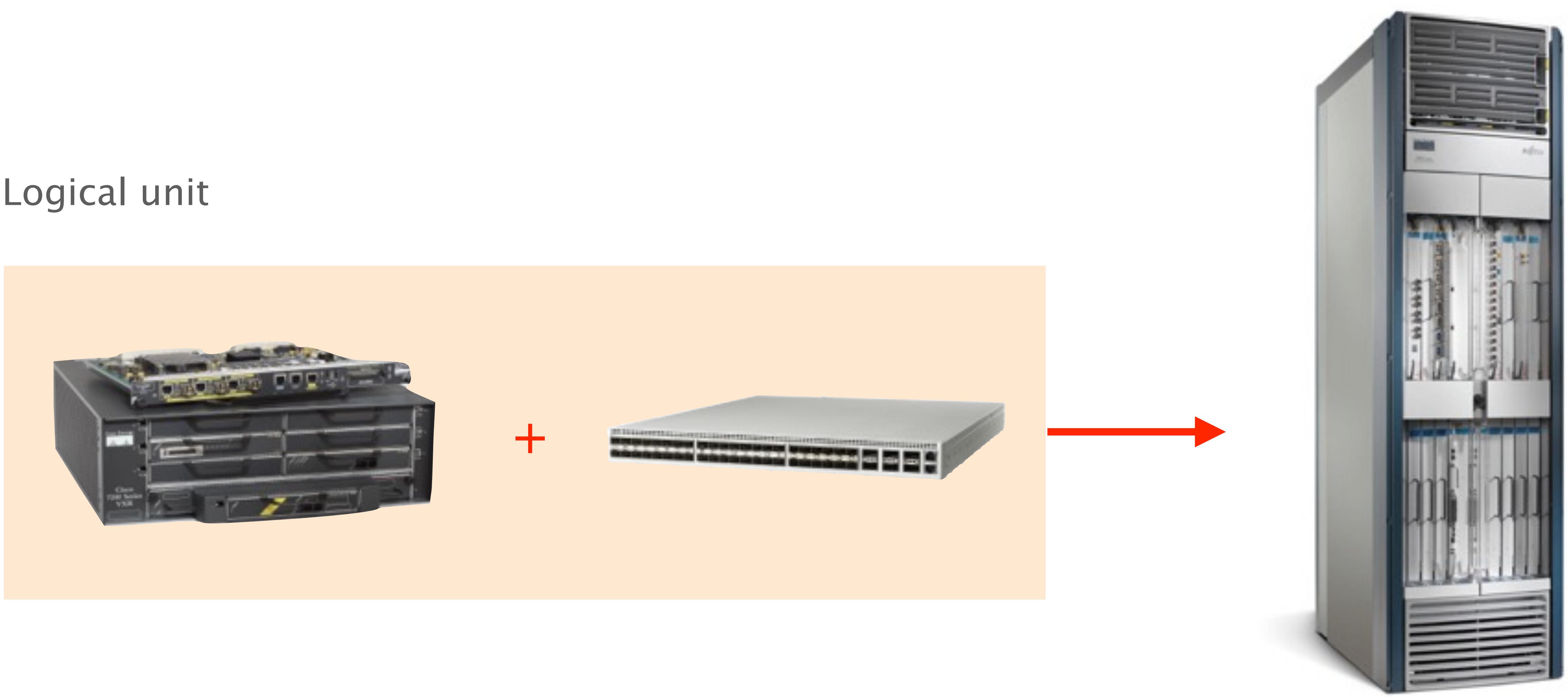
SDX data-plane can enable sub-second, prefix-independent BGP convergence

$$\text{convergence time} \quad \#\,\text{edge entries} * 150 \, \frac{\mu\text{secs}}{\text{entry}} + 30\sim50 \text{ ms}$$

$$= \quad O(30\sim50) \text{ ms}$$

SDN devices can boost the performance of traditional devices. Prototype under way!

Logical unit



old router
(Cisco 7200)

cheap
SDN switch

high-end router
(Cisco CRS 12000)

Bringing SDN to the Internet, one exchange point at the time

Architecture

programming model

Scalability

control- & data-plane

Applications

inter domain bonanza

SDX is a promising first step towards fixing Internet routing

Enable declarative, fine-grained inter-domain policies

many of which are not possible Today

Scale to hundreds of participants

both in the control- and in the data-plane

Running code (*) and deployment under way

important potential for impact

(*) https://github.com/sdn-ixp/sdx-platform

Bringing SDN to the Internet, one exchange point at the time

Laurent Vanbever

www.vanbever.eu

COS 561

November, 11 2014