Novel Applications for a SDN-enabled Internet eXchange Point

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

SDX Workshop, Washington DC

June, 5 2014

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 while simplifying operations

SDX = SDN + IXP

Augment the IXP data-plane with SDN capabilities

keeping default forwarding and routing behavior

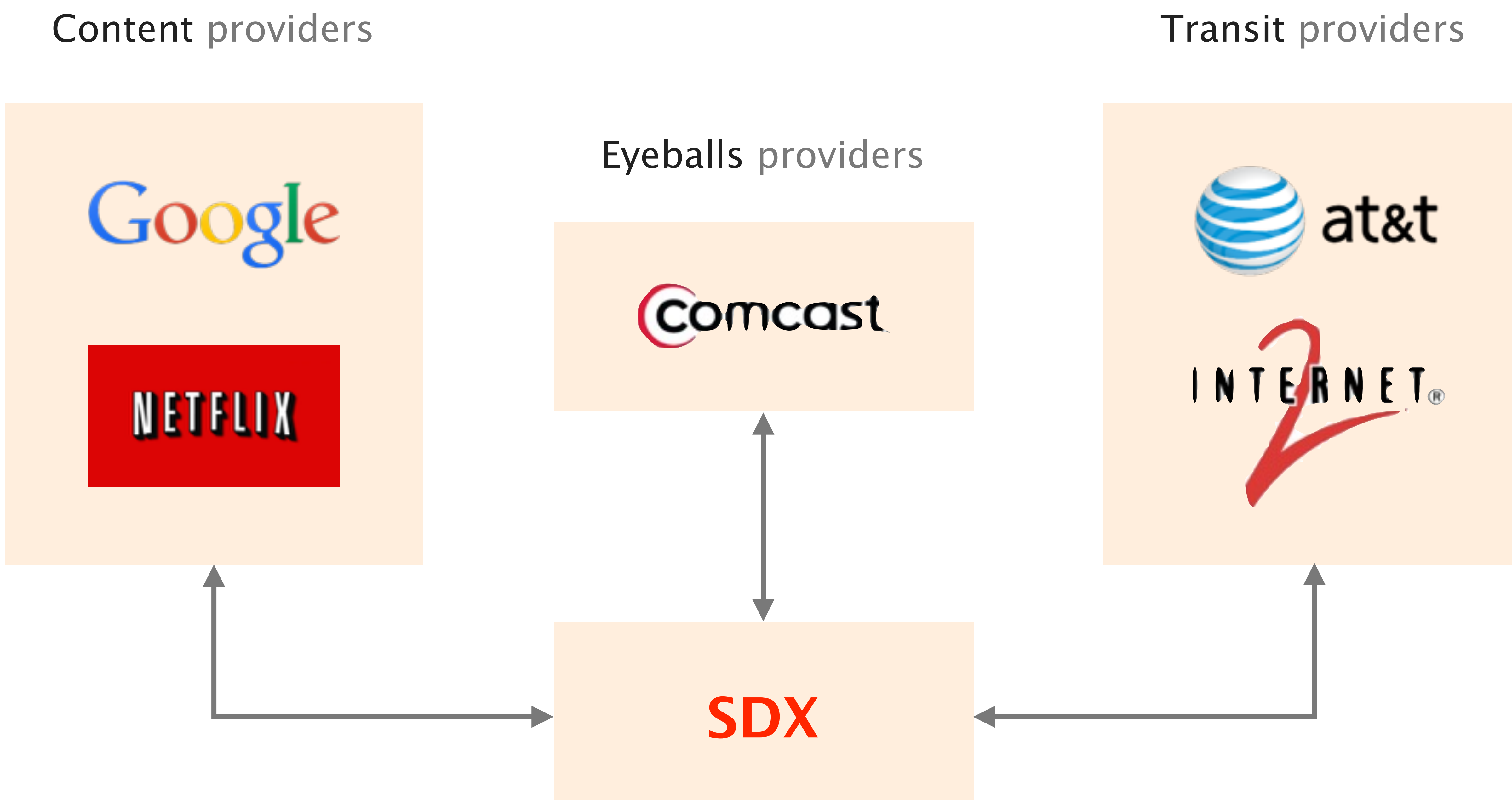
Enable fine-grained inter domain policies

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

Novel Applications for a SDN-enabled Internet eXchange Point

1 **Architecture**
programming model

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

3 **Applications**
inter domain bonanza

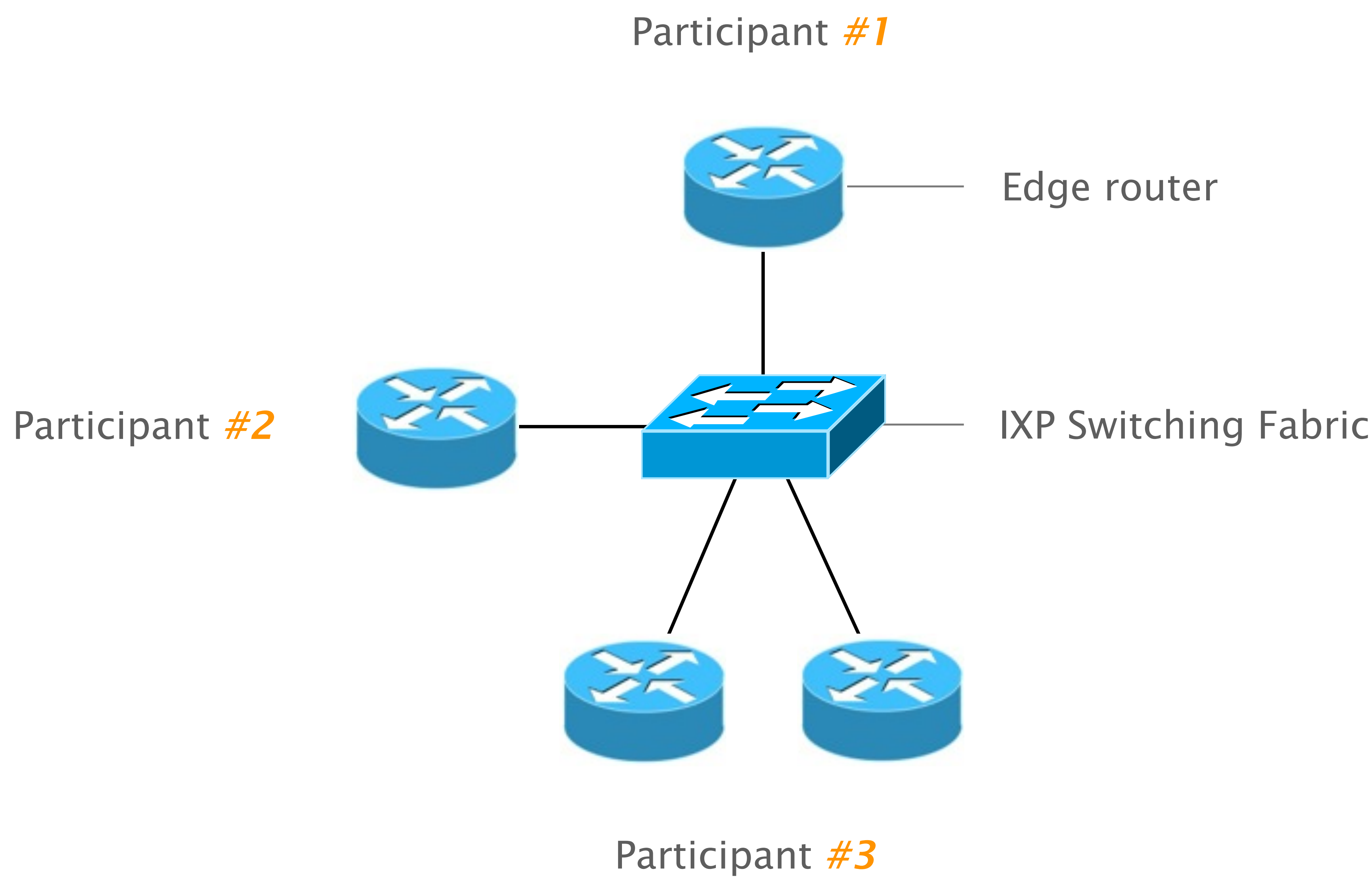
Novel Applications for a SDN-enabled Internet eXchange Point

1 **Architecture**
programming model

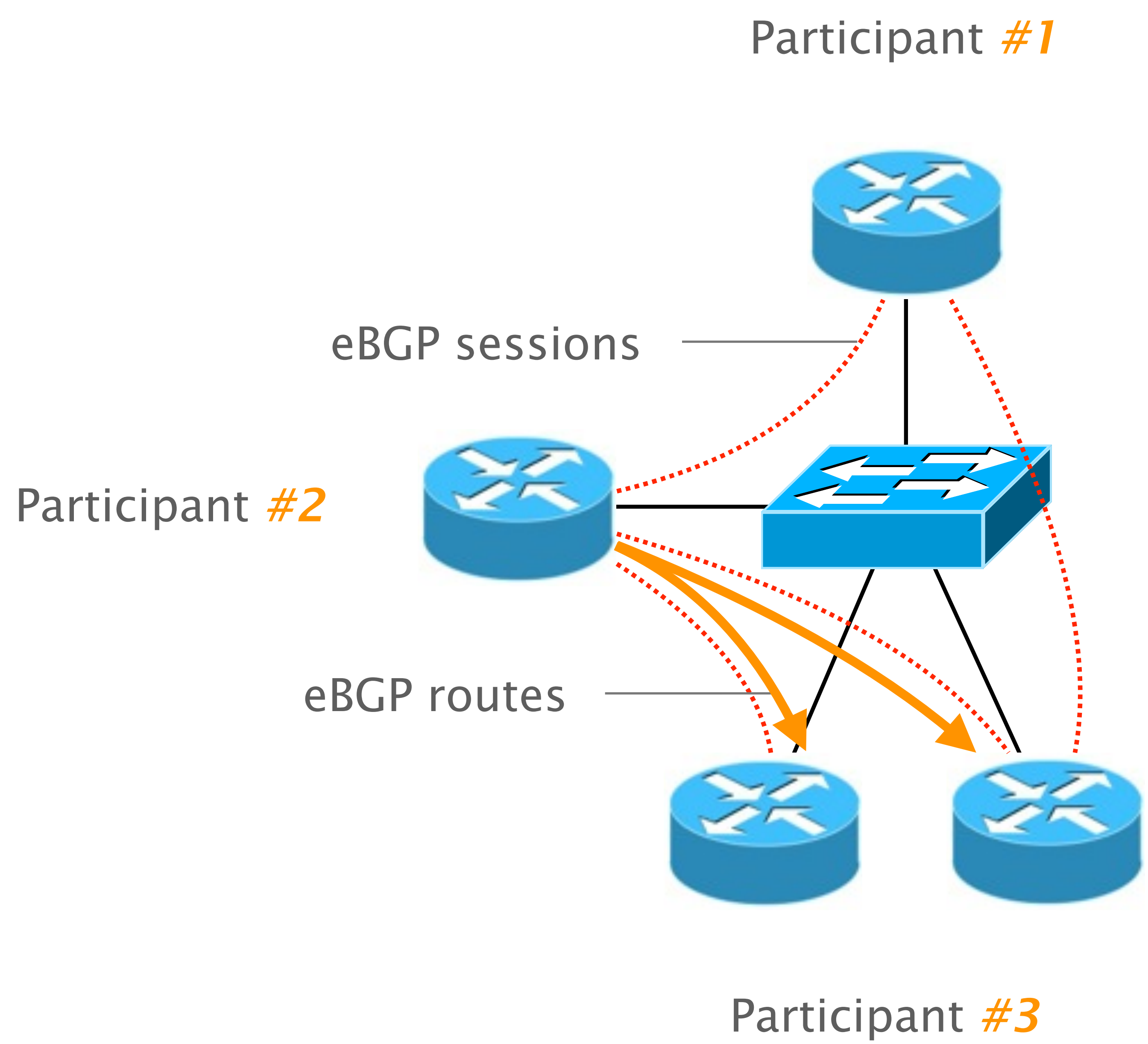
Scalability
control- & data-plane

Applications
inter domain bonanza

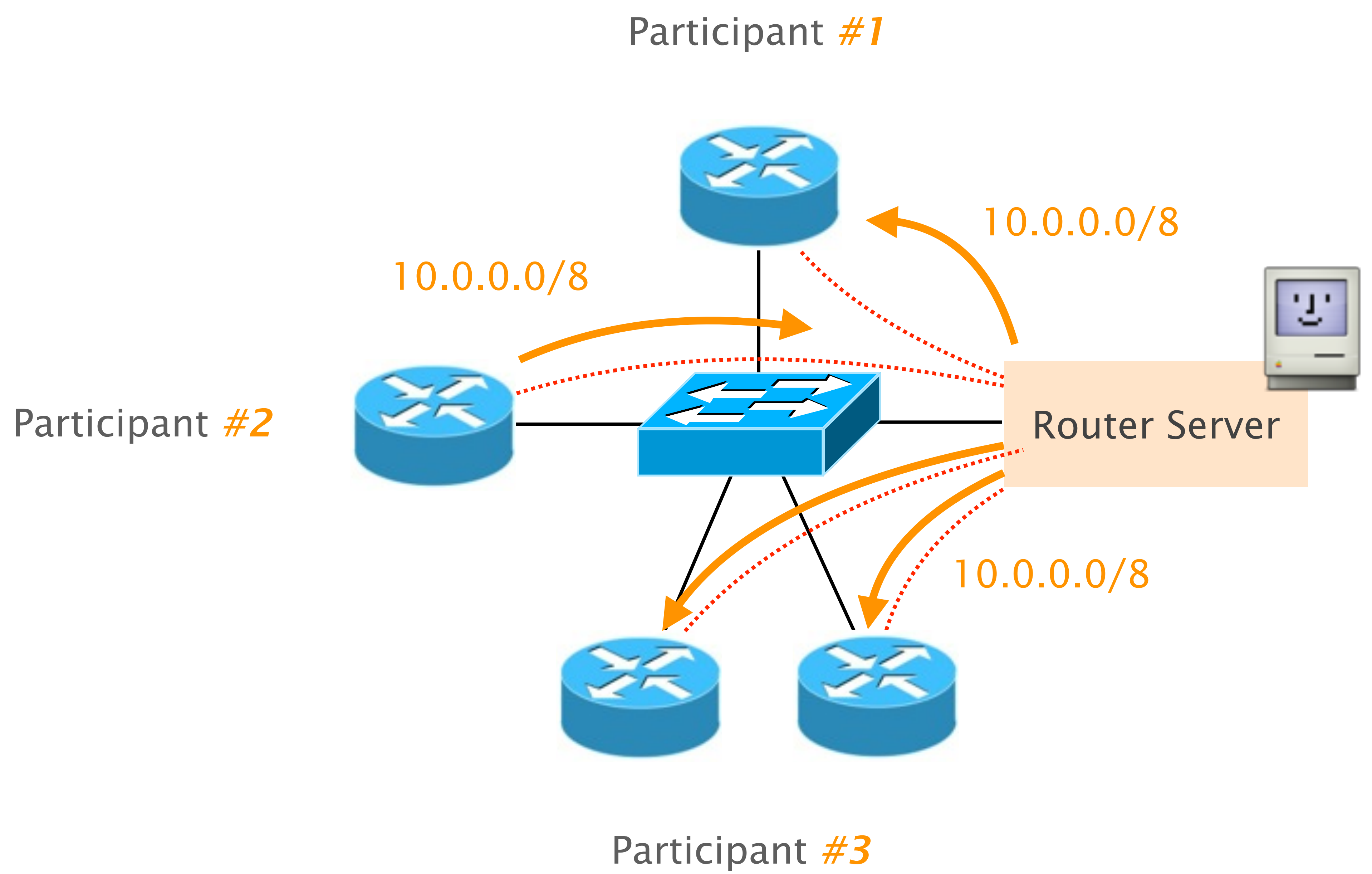
An IXP is a large layer-2 domain



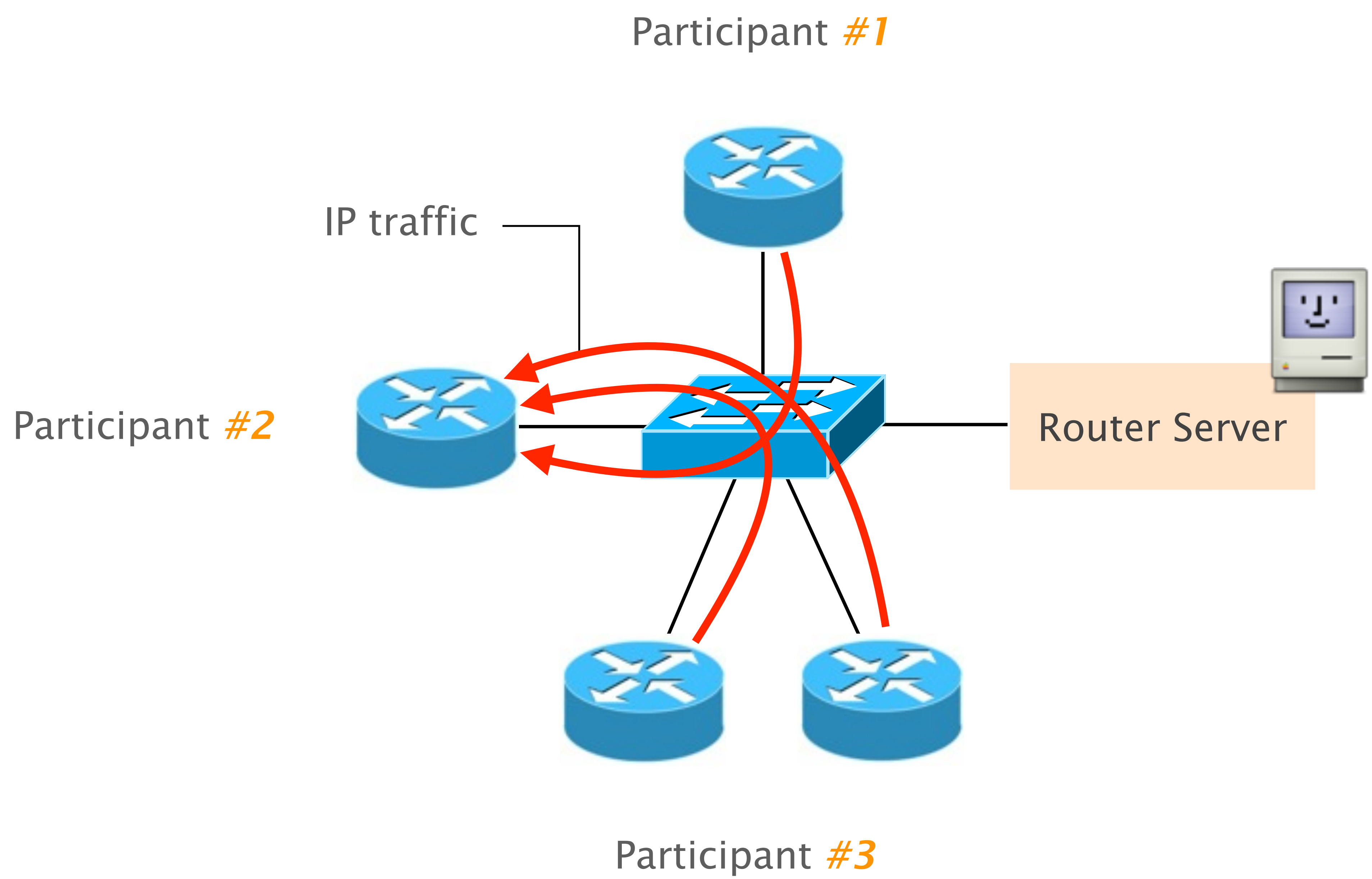
An IXP is a large layer-2 domain where participant routers exchange routes using BGP



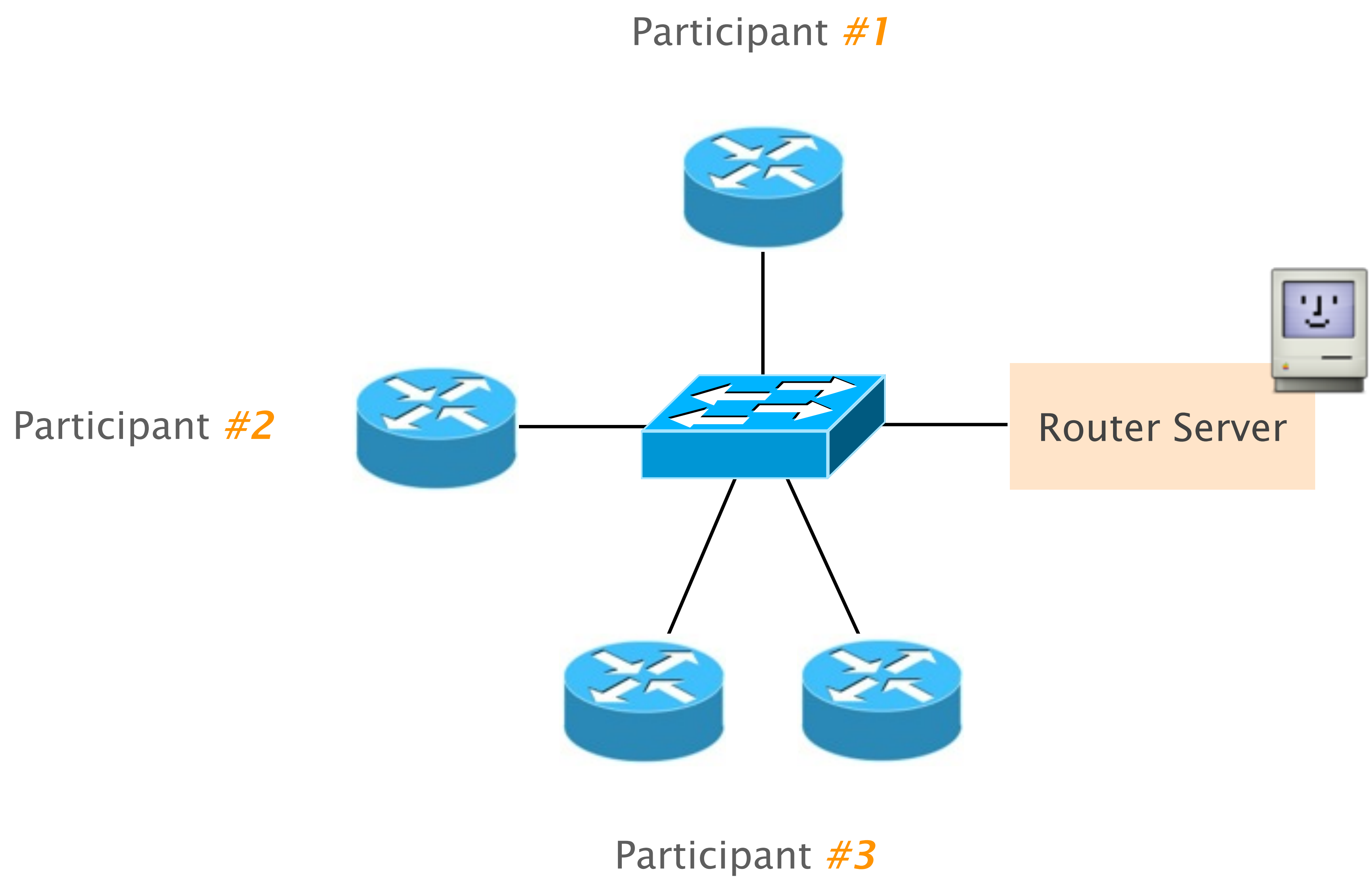
To alleviate the need of establishing eBGP sessions, IXP often provides a Route Server (route multiplexer)



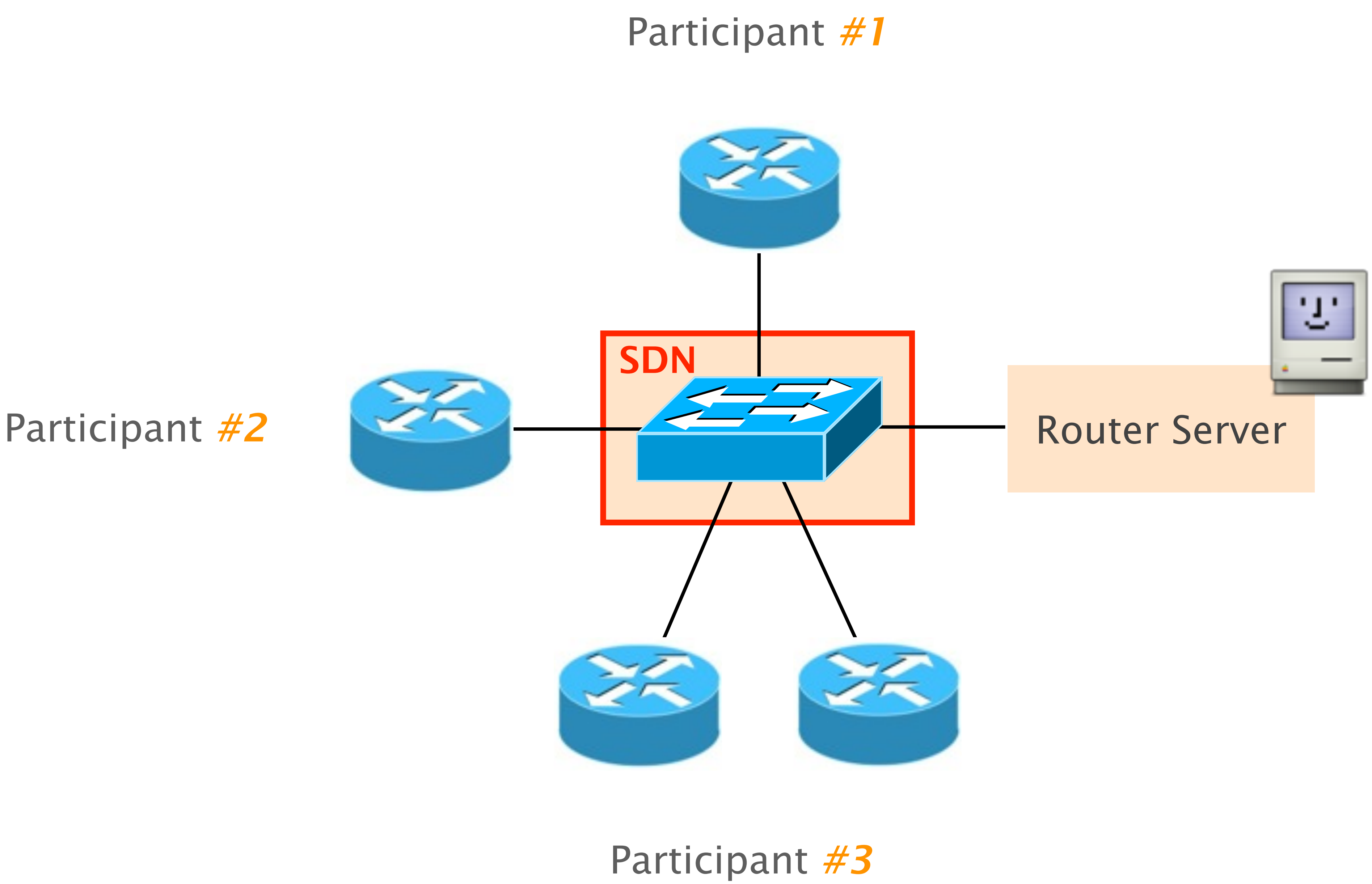
IP traffic is exchanged directly between participants—IXP is forwarding transparent



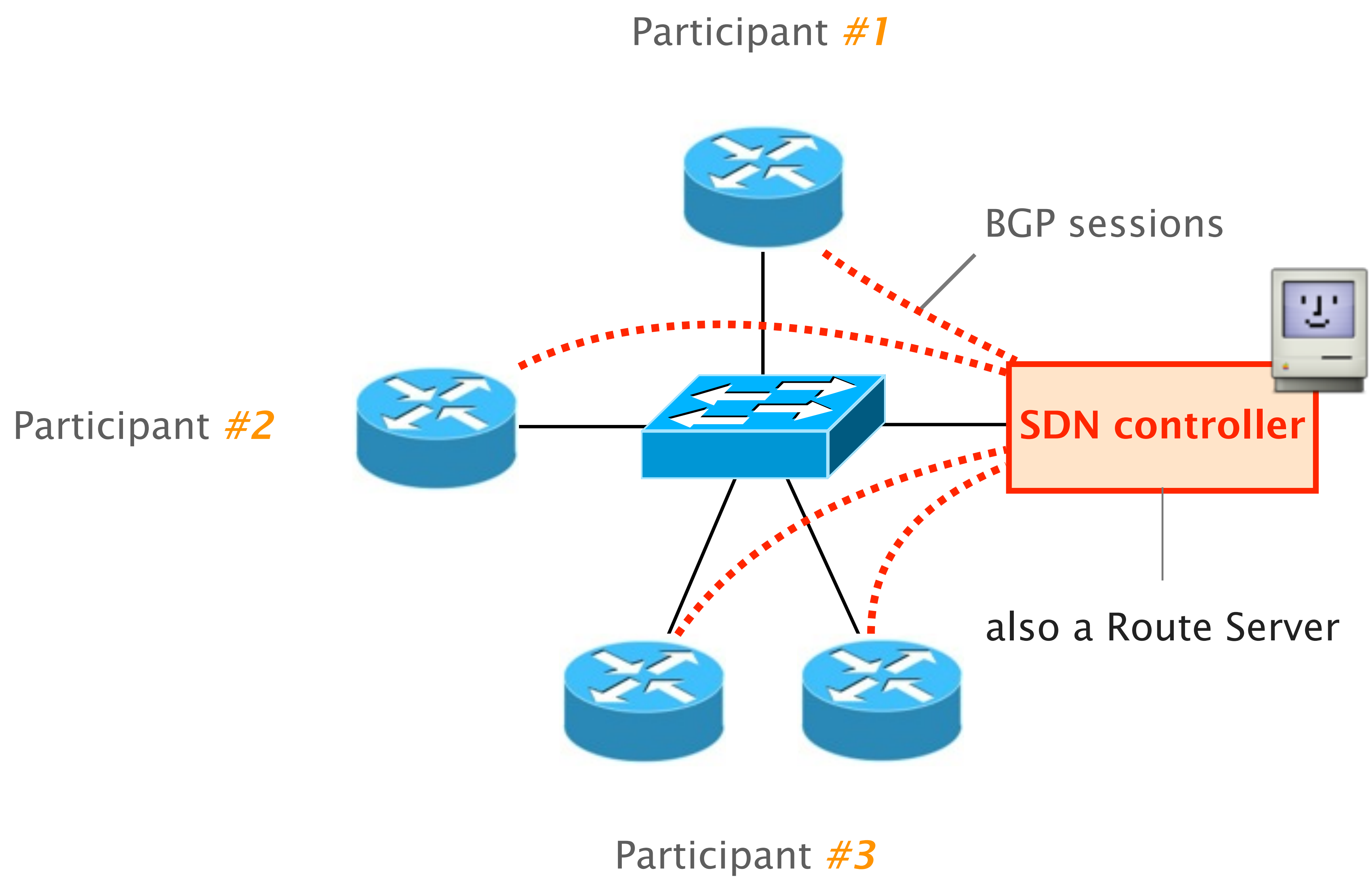
With respect to a traditional IXP, SDX…



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 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

```
           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

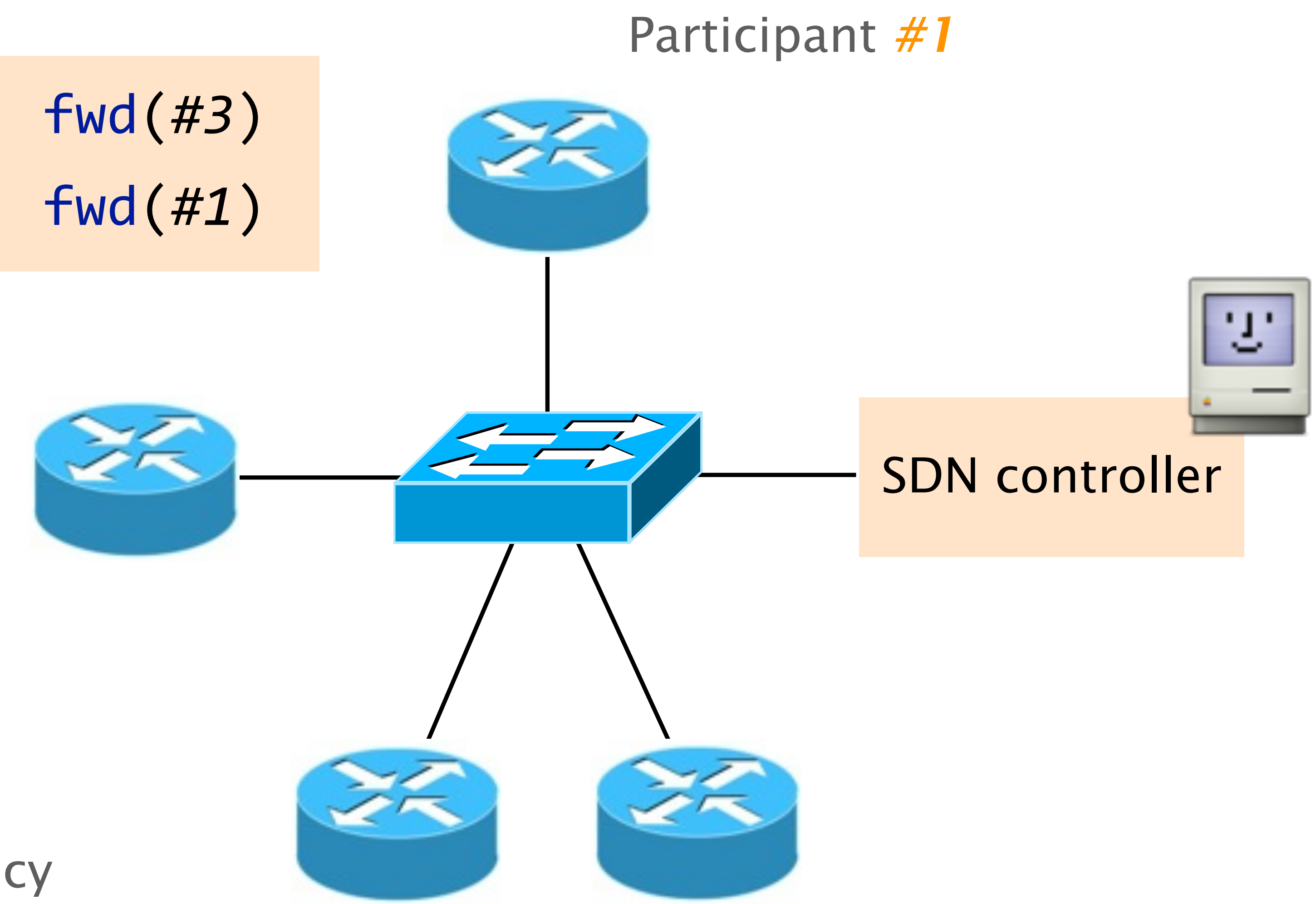
Actions

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



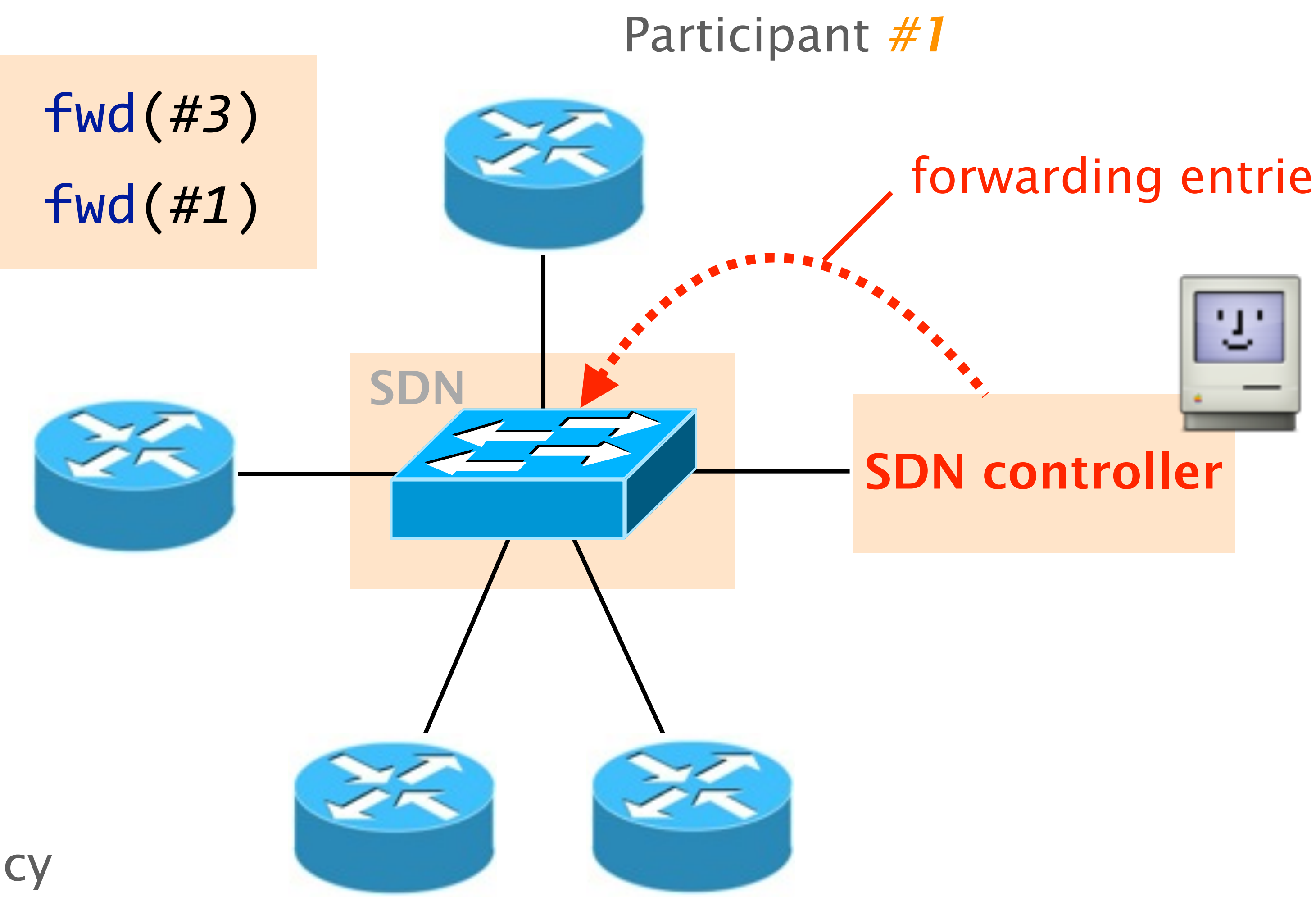
Participant *#3* policy

```
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

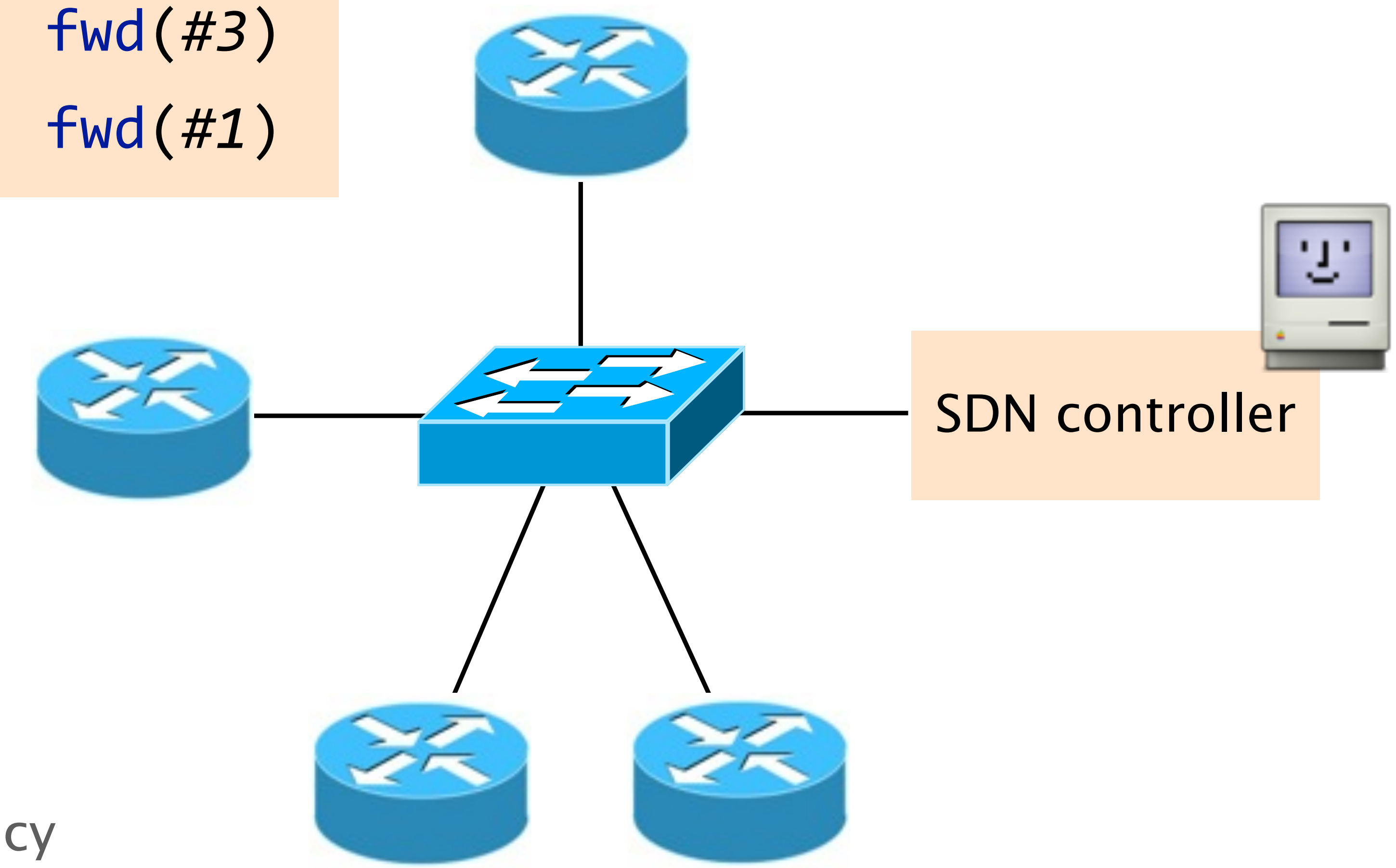
Listening to BGP is important to avoid correctness issues

#1 reachable prefixes: 11/24

Participant *#2* policy

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

Participant *#1*



Participant *#3* policy

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

#3 reachable prefixes: 10/24

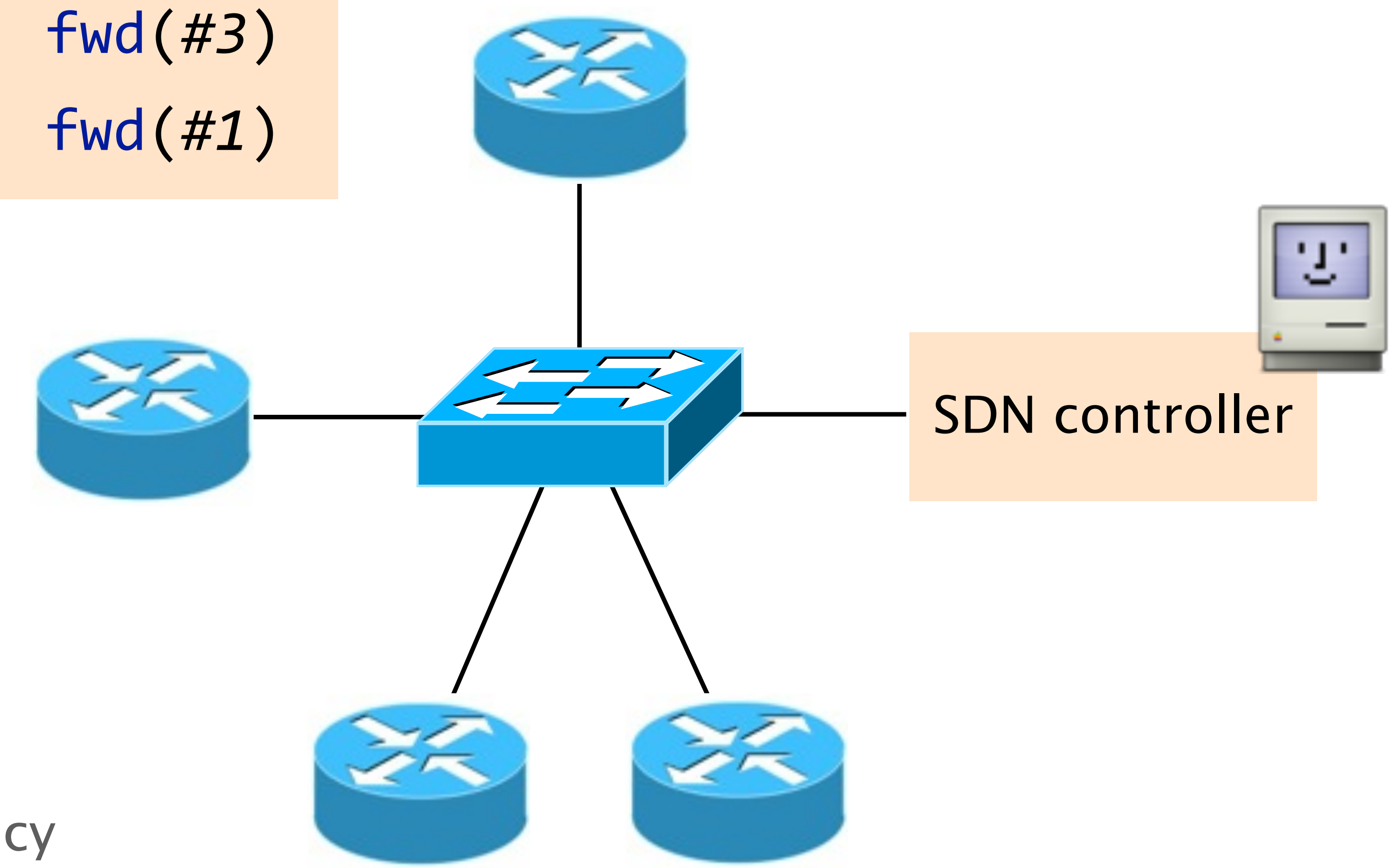
Traffic for 11/24, port 80 must be delivered to participant #1, not #3, to avoid blackhole

#1 reachable prefixes: 11/24

Participant #2 policy

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

Participant #1



Participant #3 policy

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

#3 reachable prefixes: 10/24

Novel Applications for a SDN-enabled Internet eXchange Point

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

500,000 prefixes, 500+ participants,
potentially *billions* of forwarding rules

control-plane

time

100s of policies that have to be updated
dynamically according to BGP

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

data-plane

space

leverage *existing routing platform*

control-plane

time

leverage *inherent policy structure*

data-plane

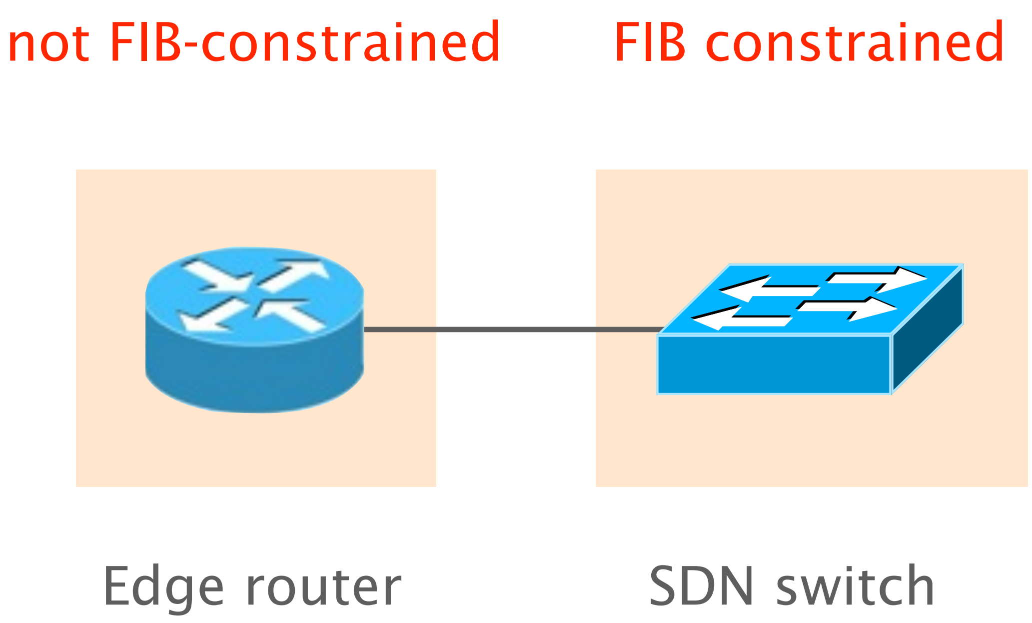
space

leverage *existing routing platform*

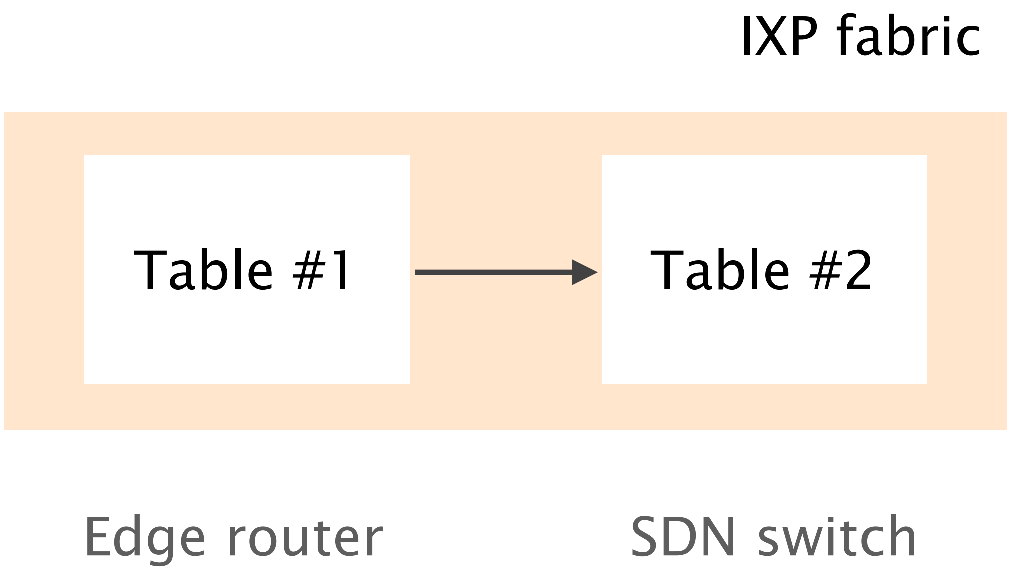
control-plane

time

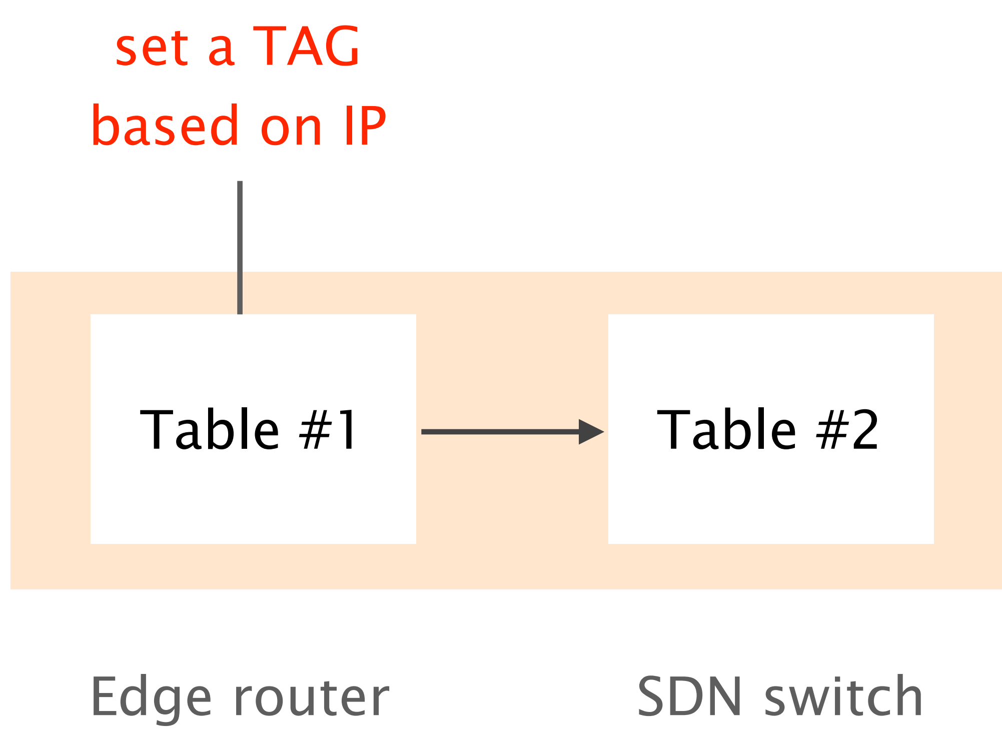
The edge routers, sitting next to the fabric, are tailored to match on numerous IP prefixes



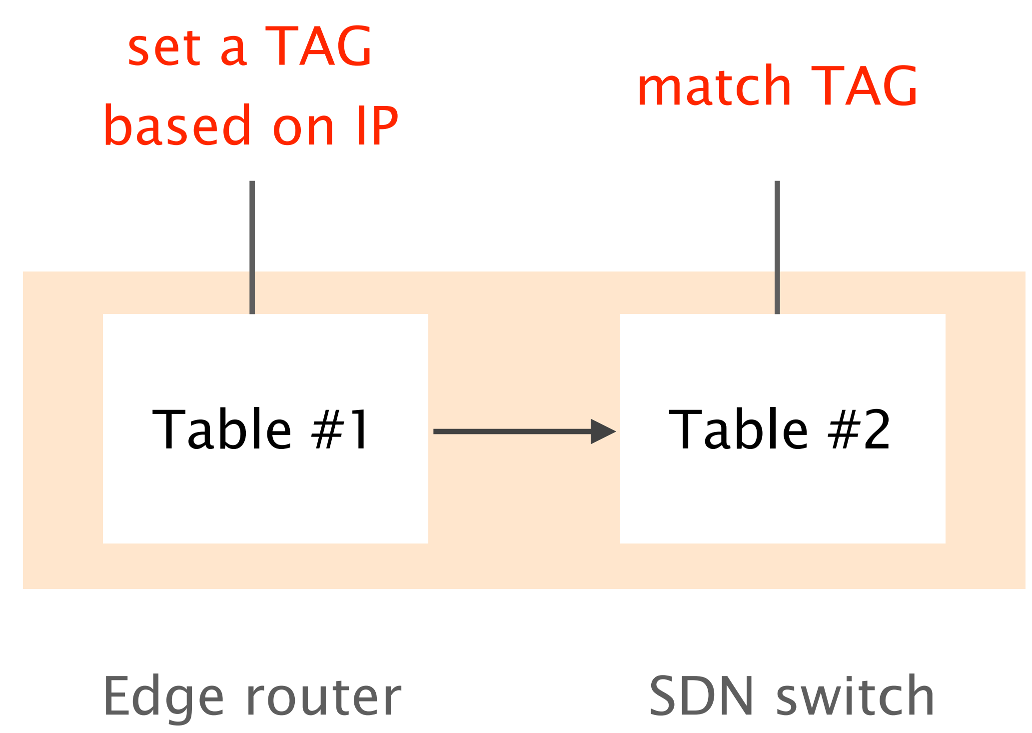
We consider routers FIB as the first stage of a multi-stage FIB



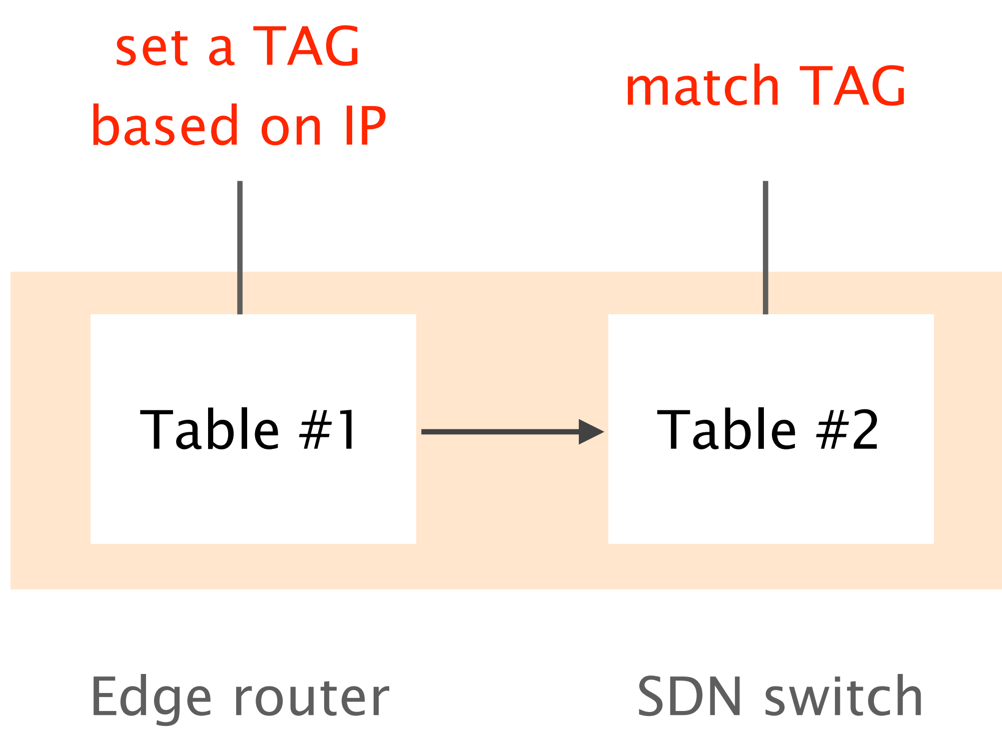
Routers FIB match on the destination prefix and set a tag accordingly



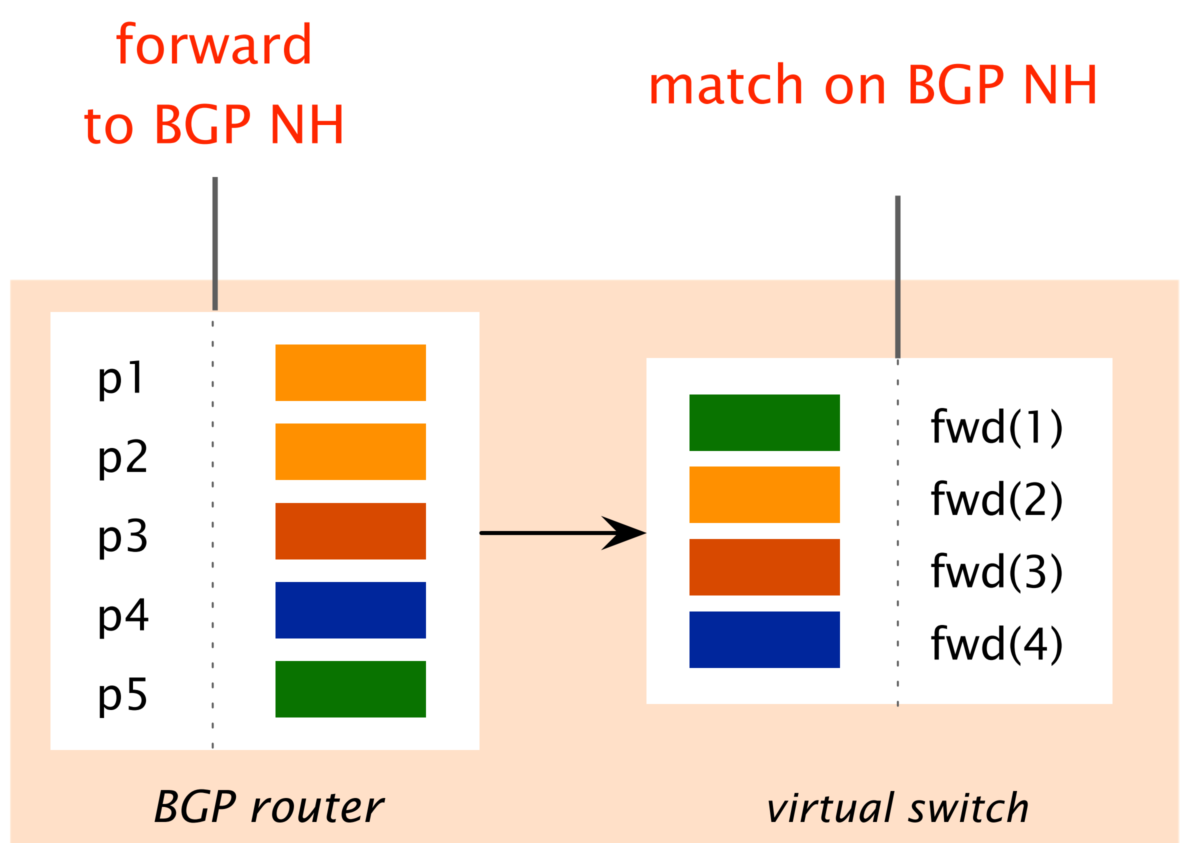
The SDN FIB matches on the tag, not on the IP prefixes



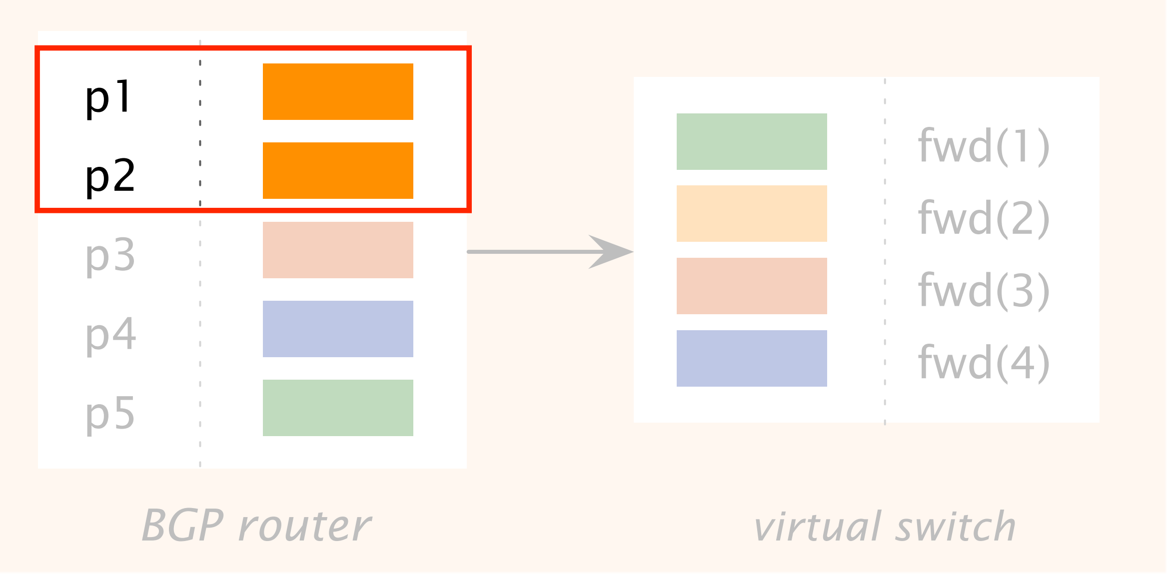
How do we provision tag entries in a router, and what are these tags?



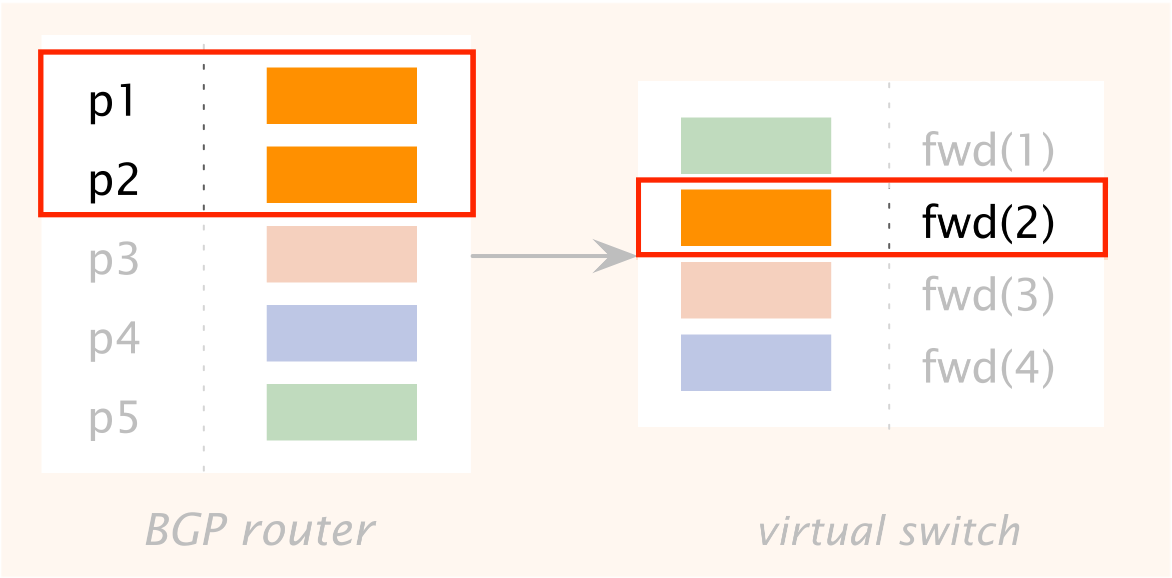
We use BGP as a provisioning interface and BGP next-hops as labels



All prefixes sharing the same forwarding behavior are grouped together using the same BGP next-hop

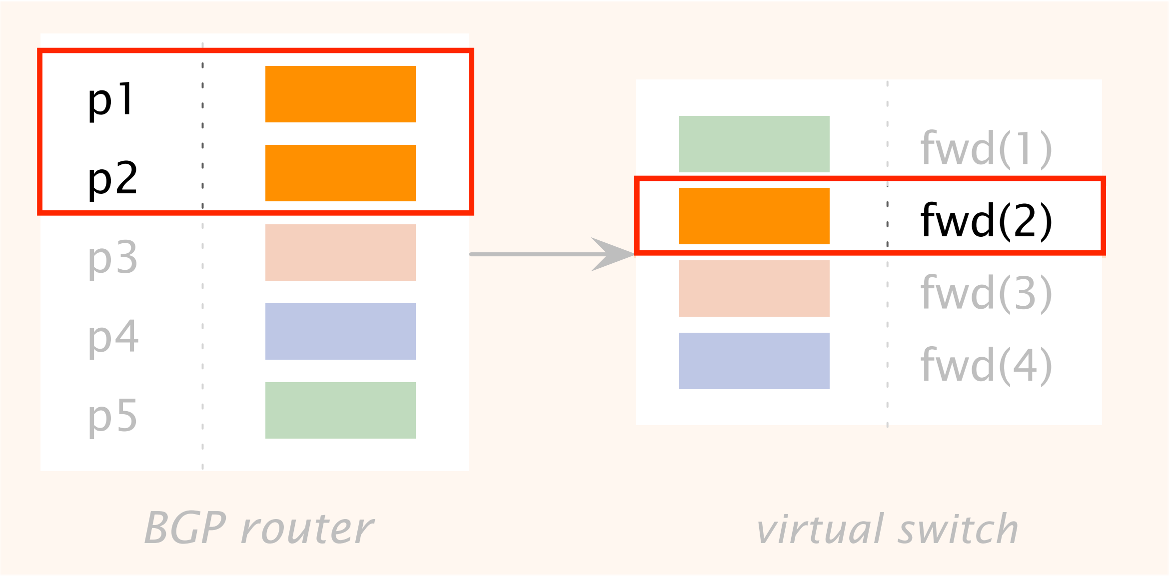


The SDX data-plane maintains one forwarding entry per prefix-group

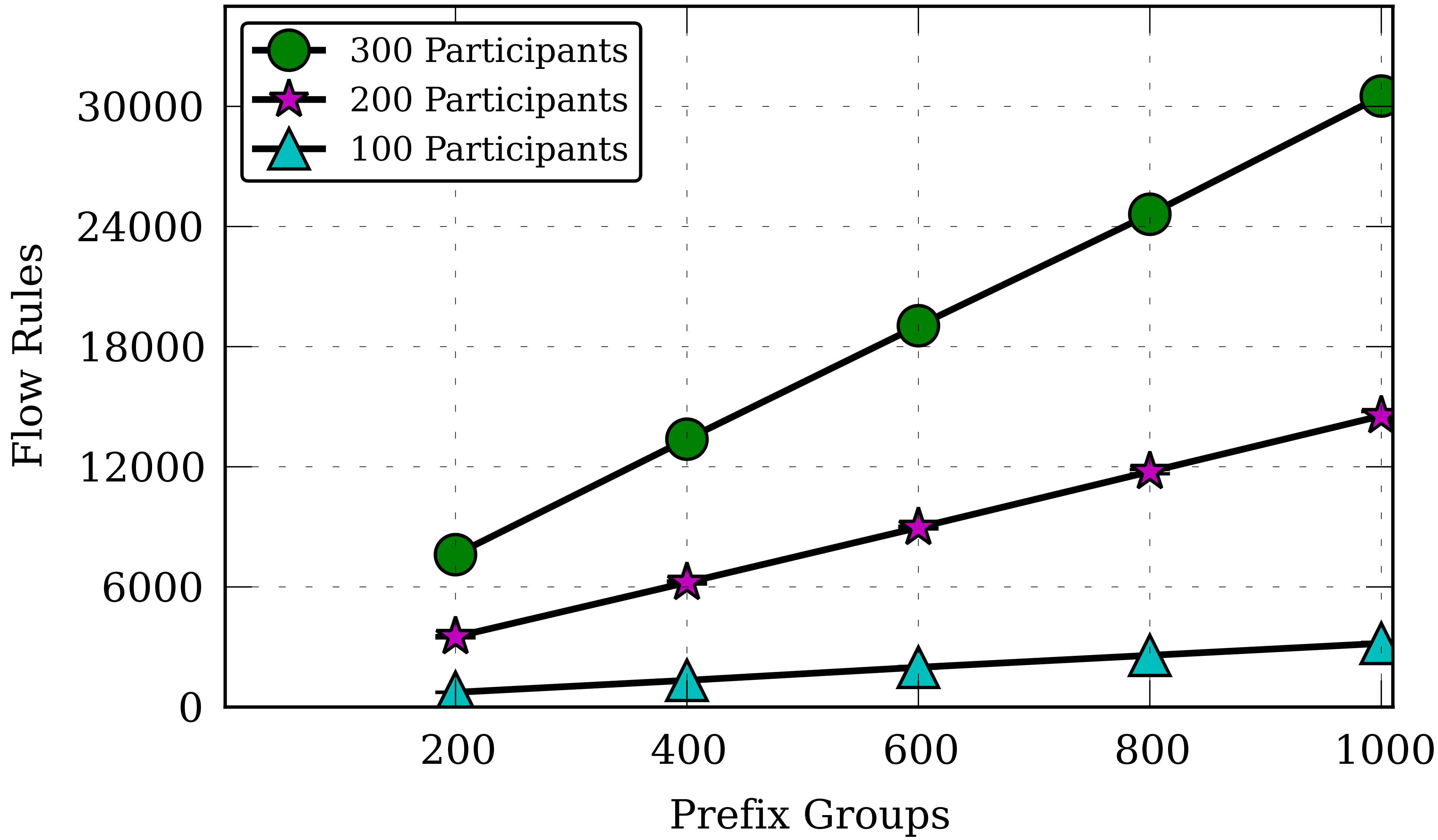


Data-plane utilization is reduced considerably as there are *way* more prefixes than prefixes groups

 # prefixes >> #prefixes groups



By leveraging BGP, the SDX can accommodate policies for hundreds of participants with less than 30k rules




data-plane
space
control-plane
time
leverage *inherent policy structure*

SDX policies exacerbate key characteristics that enable to speed-up compilation time considerably

Policies are often disjoint

Policy updates are local

Policy updates are bursty

SDX policies exacerbate key characteristics that enable to speed-up compilation time considerably

Policies are often disjoint

disjoint policy do not have to be composed together

significant gain as composing policies is time consuming

Policy updates are local

Policy updates are bursty

SDX policies exacerbate key characteristics that enable to speed-up compilation time considerably

Policies are often disjoint

Policy updates are local

Policy updates usually impact a few prefix-groups

75% of the updates affect no more than 3 prefixes

Policy updates are bursty

SDX policies exacerbate key characteristics that enable to speed-up compilation time considerably

Policies are often disjoint

Policy updates are local

Policy updates are bursty

policy changes are separated of large periode of inactivity

75% of the time, inter-arrival time between updates is at least 10s

The SDX controller adopts a two-staged compilation algorithm

Fast, but non-optimal algorithm upon updates

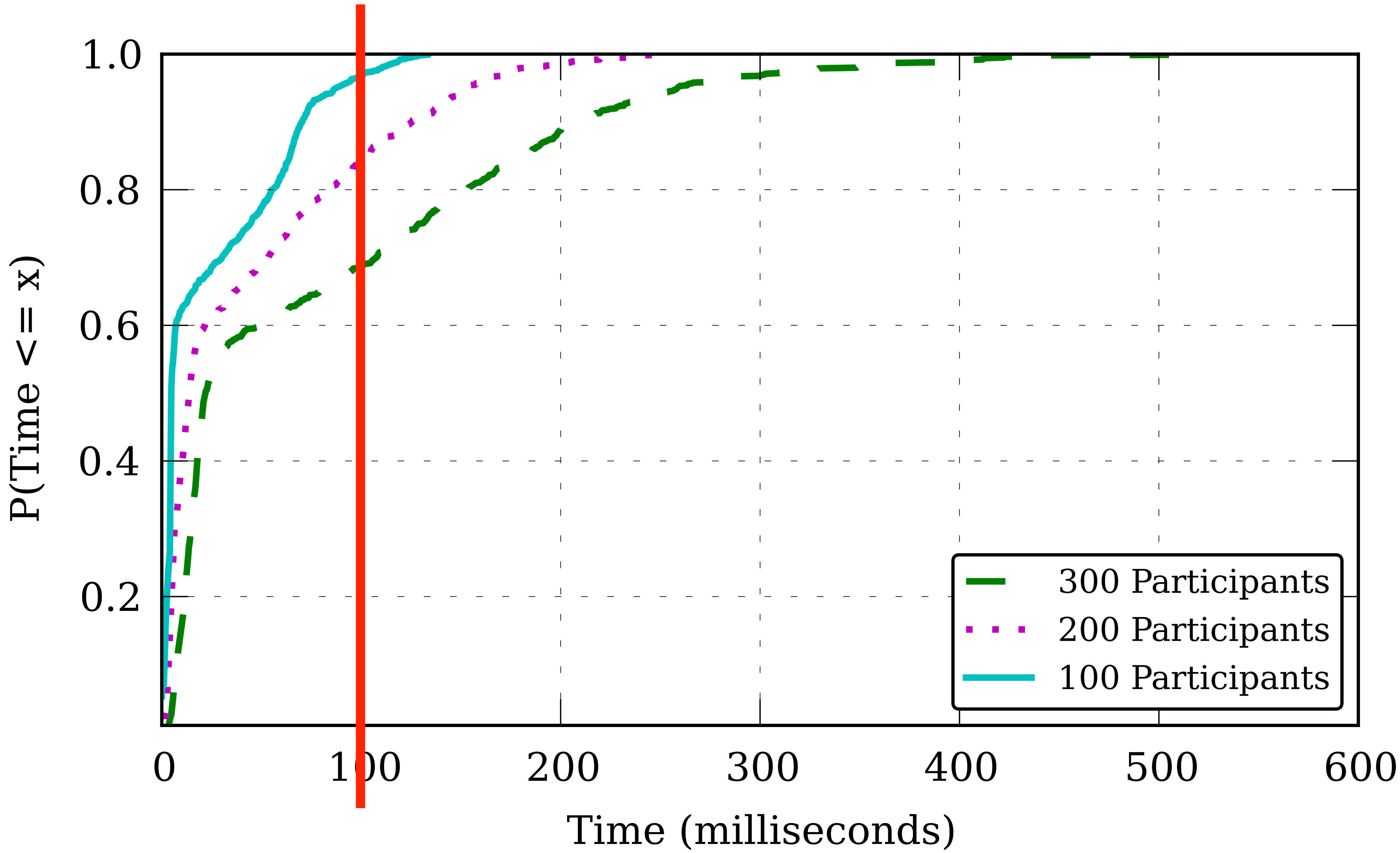
can create more rules than required

Slow, but optimal algorithm in background

recompute prefix groups

Time *vs* Space trade-off

In most cases, the SDX takes <100 ms
to recompute the global policy upon a BGP event



Novel Applications for a SDN-enabled Internet eXchange Point

Architecture
programming model

Scalability
control- & data-plane

3 Applications
inter domain bonanza

SDX enables a wide range of novel applications

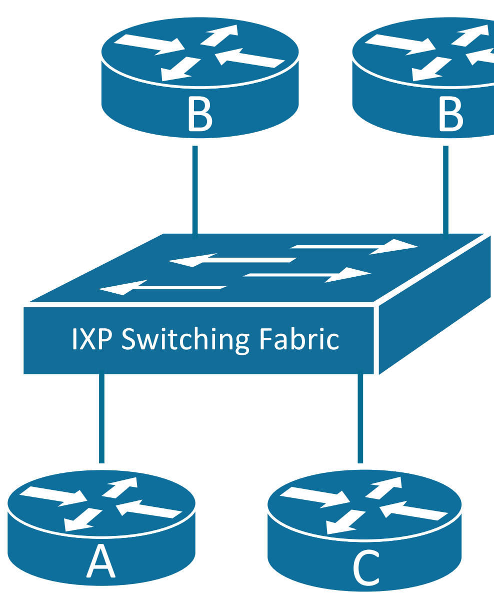
security	Prevent/block policy violation Prevent participants communication Upstream blocking of DoS attacks
forwarding optimization	Middlebox traffic steering Traffic offloading Inbound Traffic Engineering Fast convergence
peering	Application-specific peering
remote-control	Influence BGP path selection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

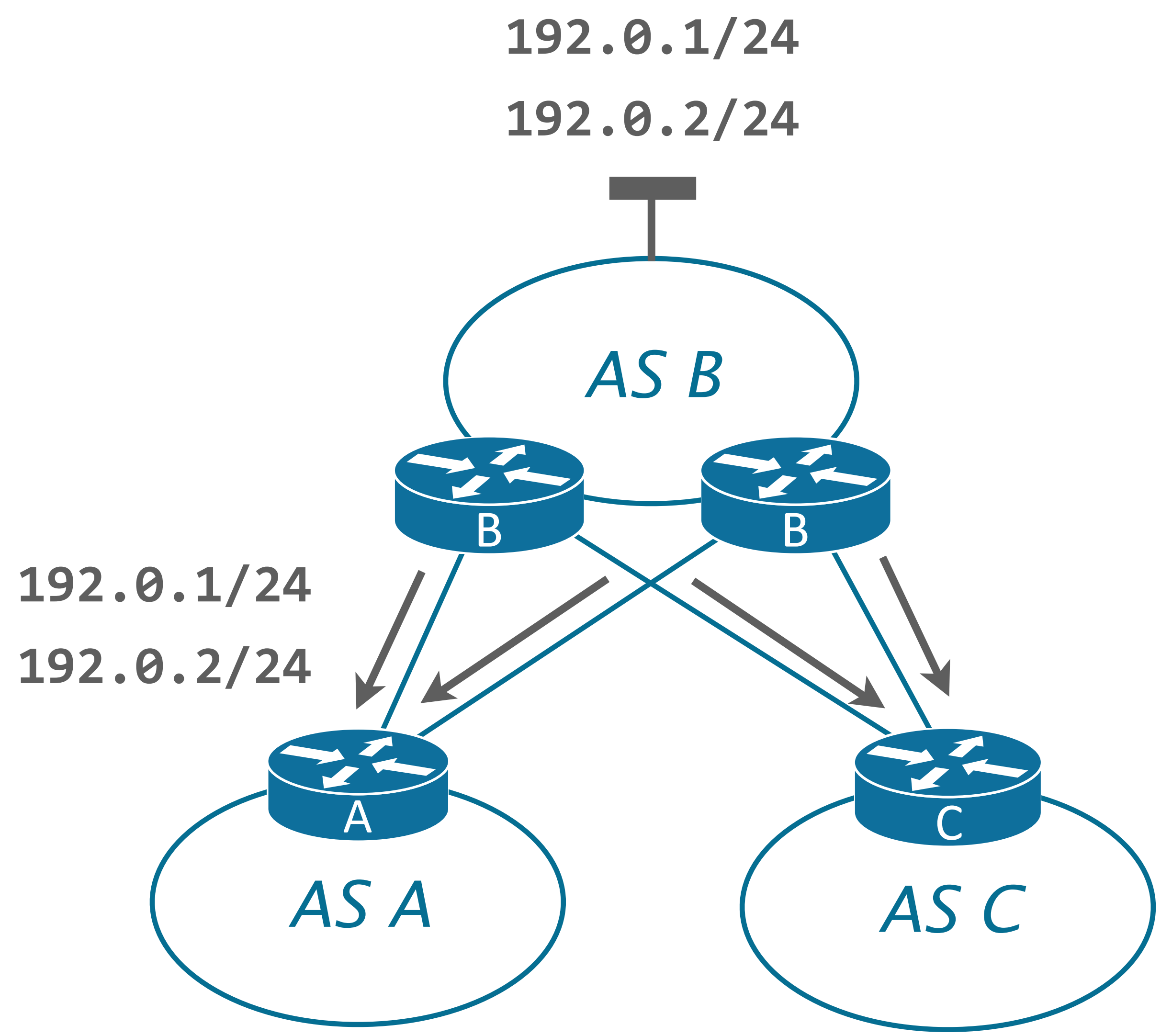
SDX enables a wide range of novel applications

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 improve inbound traffic engineering

Given an IXP Physical Topology and a BGP topology,

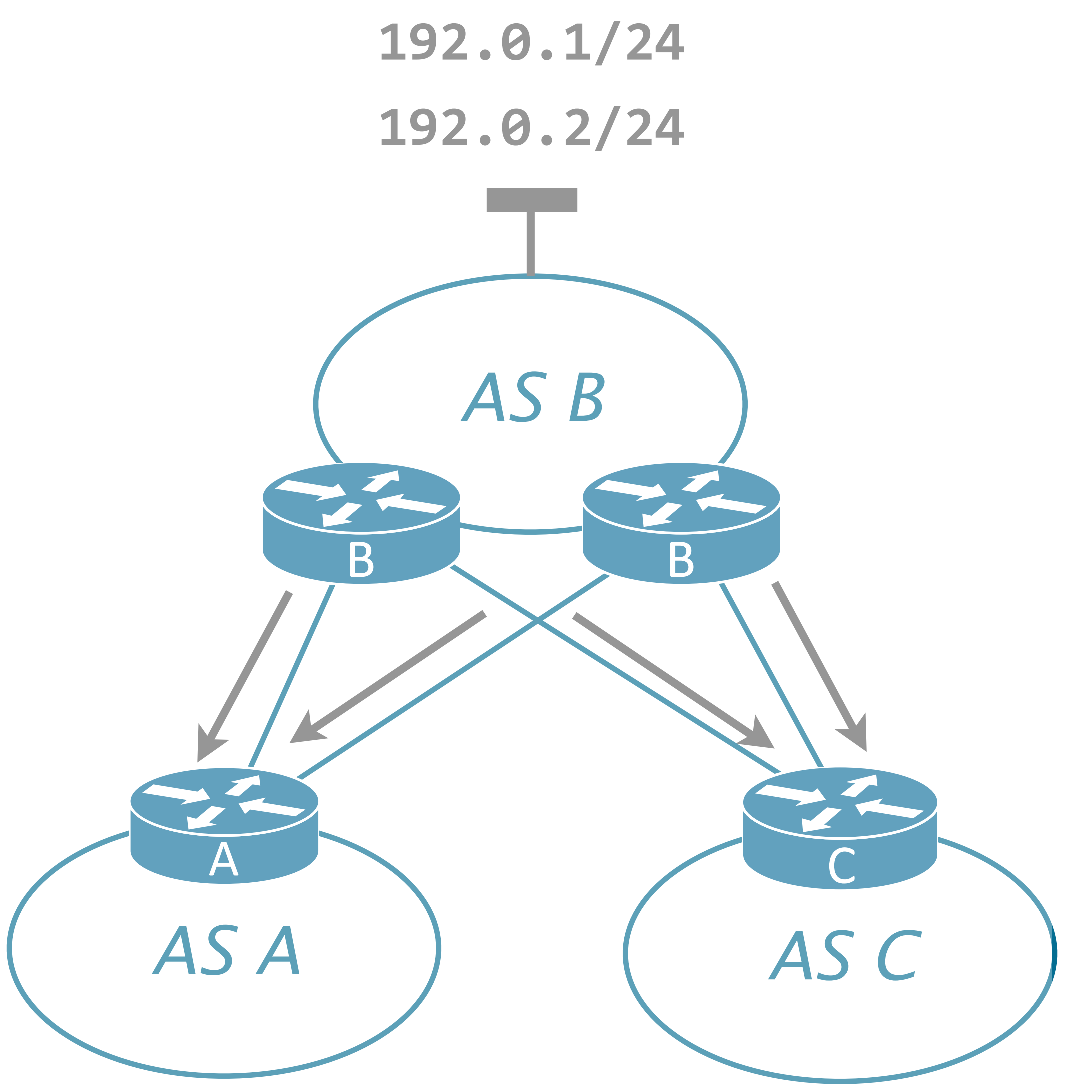




Given an IXP Physical Topology and a BGP topology, Implement B's inbound policies

B's inbound policies

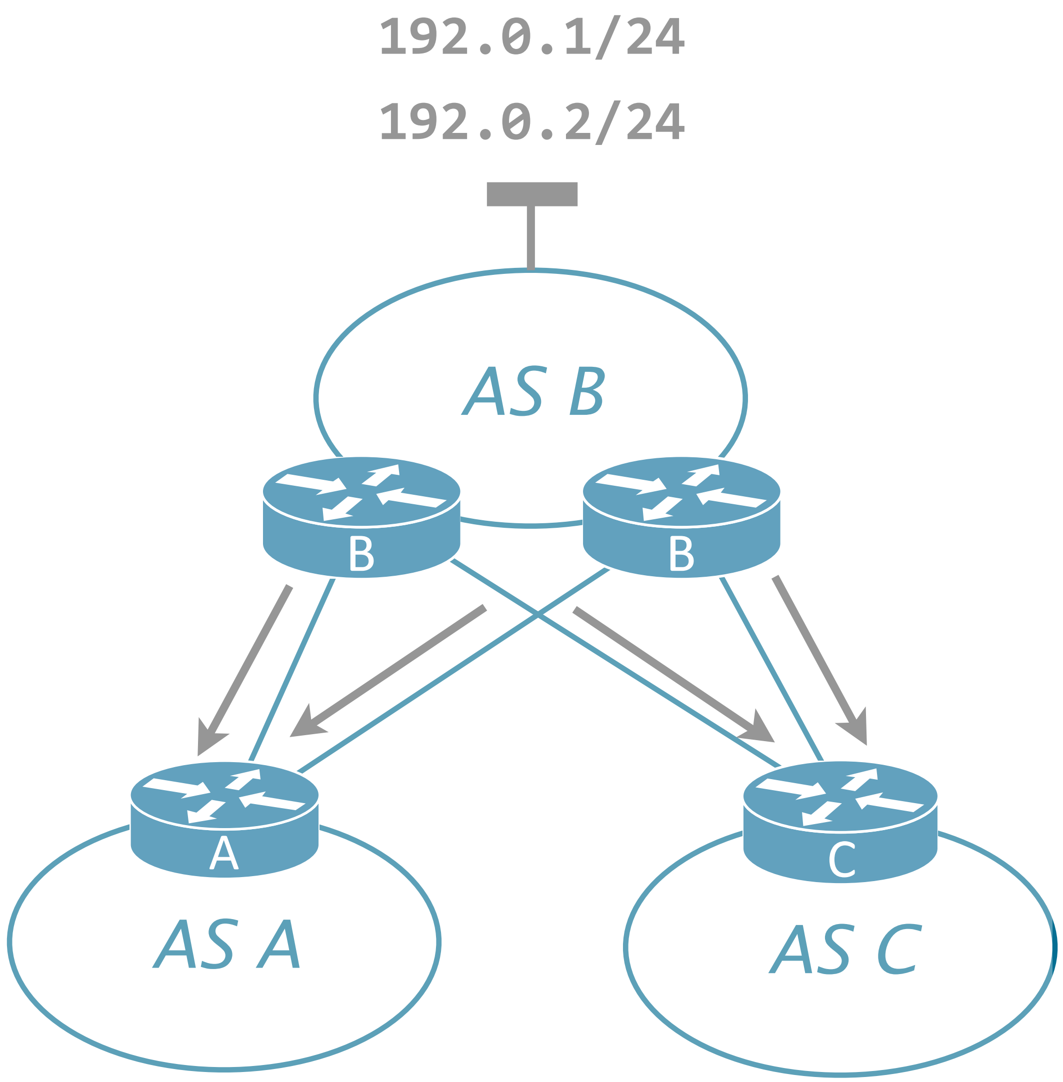
to	from	*receive on*
192.0.1/24	A	left
192.0.2/24	C	right
192.0.2/24	ATT_IP	right
192.0.1/24	*	right
192.0.2/24	*	left



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)
```

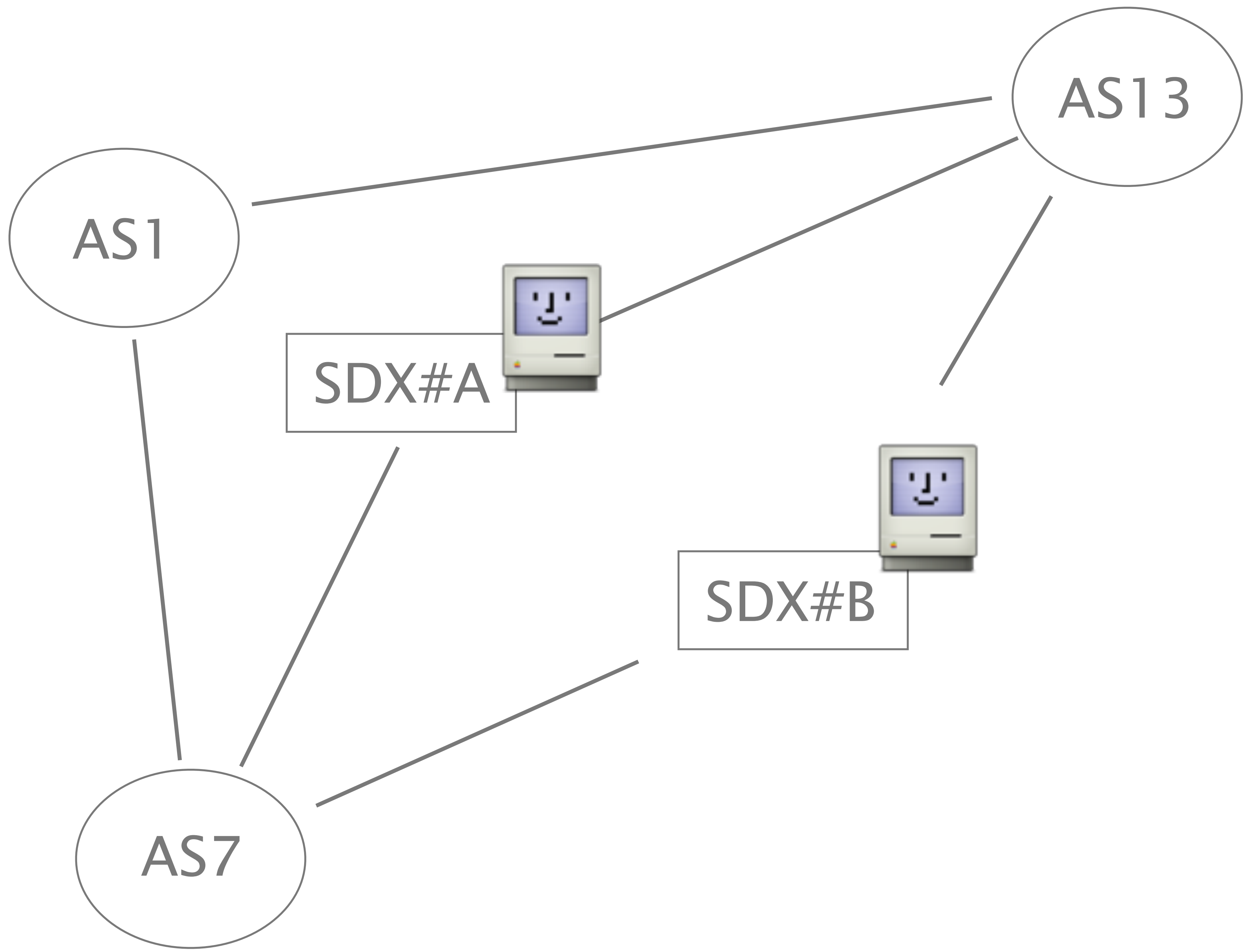
SDX enables a wide range of novel applications

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

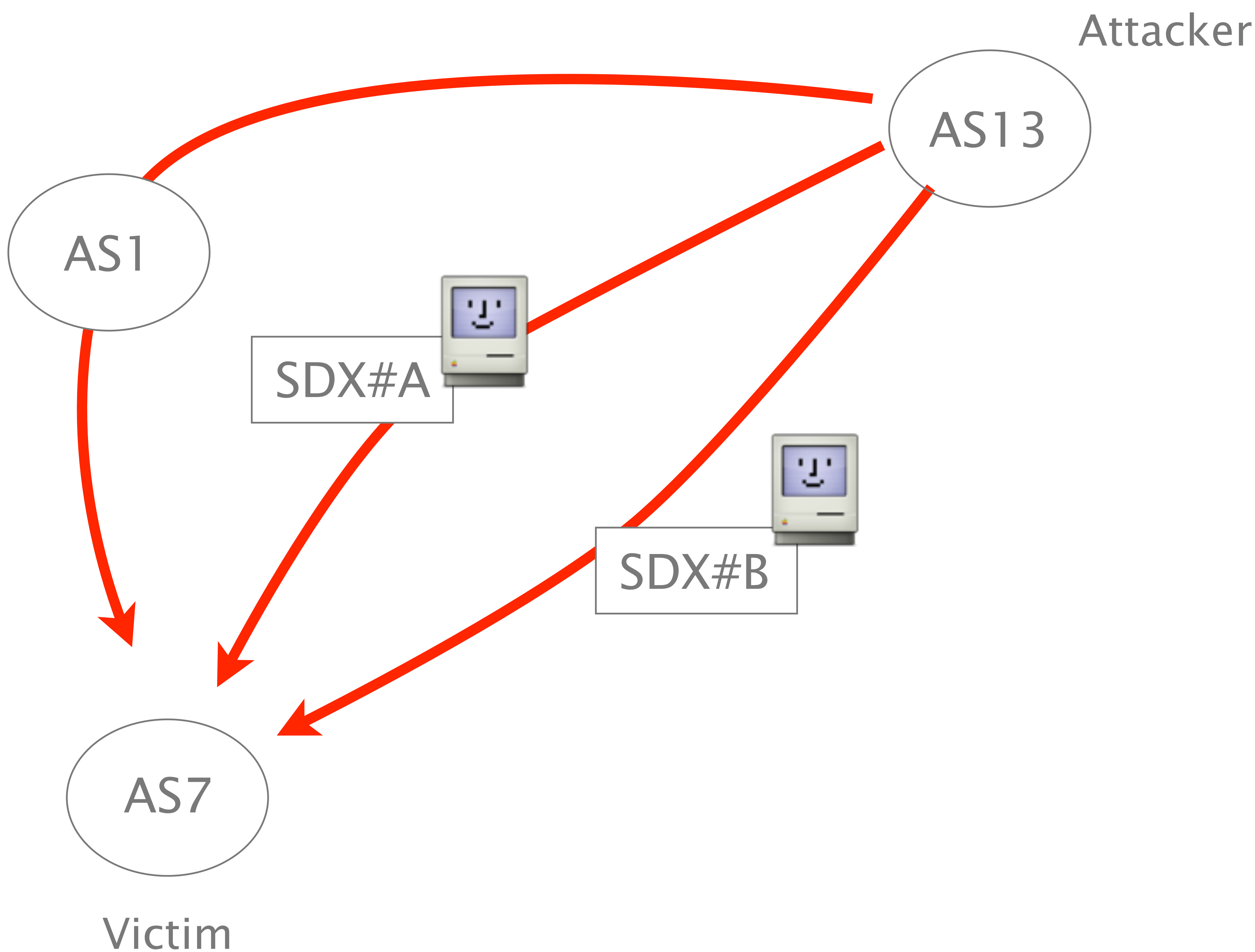
SDX enables a wide range of novel applications

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

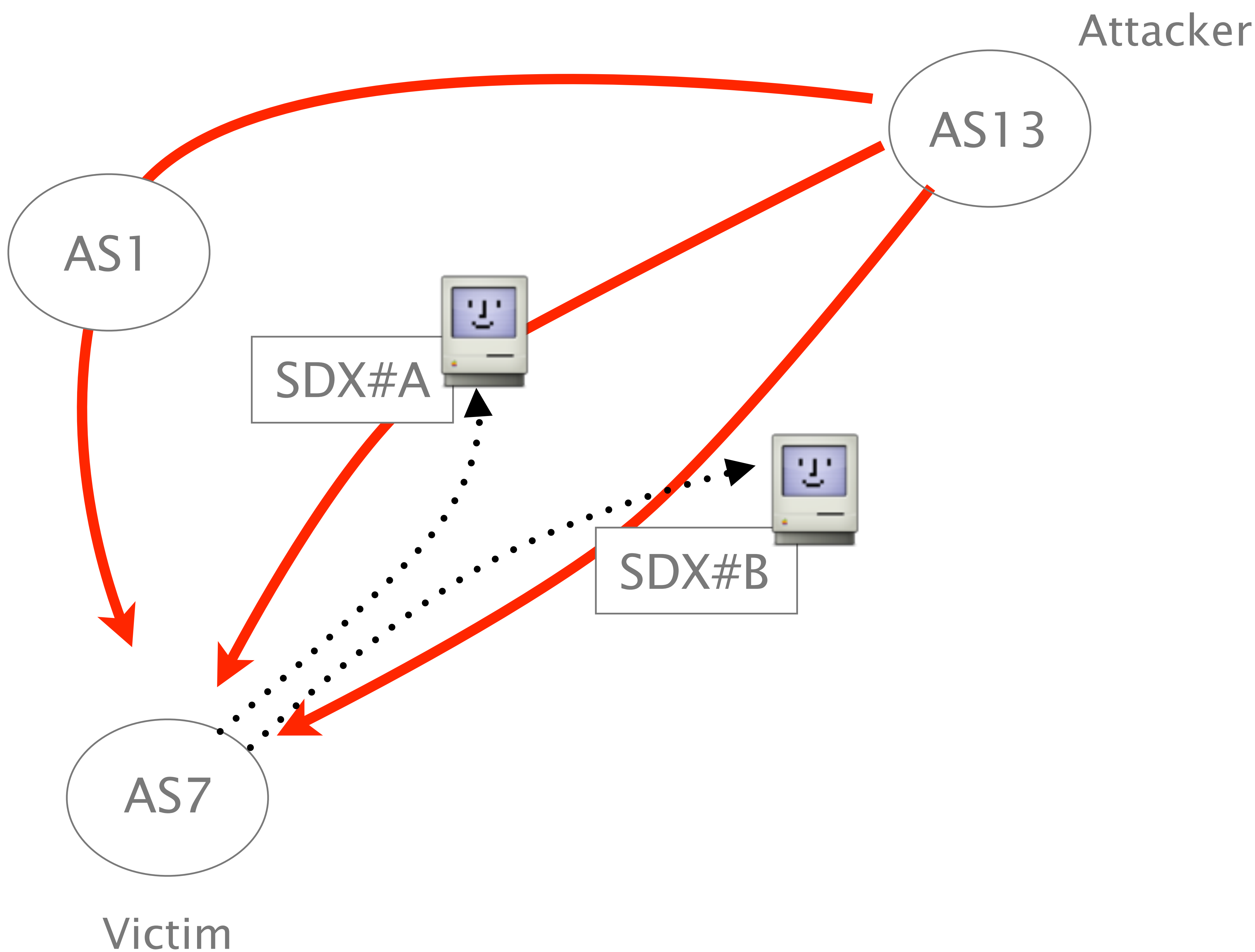
SDX can help in blocking DDoS attacks closer to the source



AS7 is victim of a DDoS attack originated from AS13



AS7 can remotely install *drop()* rule in the SDX platforms



```
match(srcip=Attacker/24, dstip=Victim/32) >> drop()
```

Novel Applications for a SDN-enabled Internet eXchange Point

Architecture

programming model

Scalability

control- & data-plane

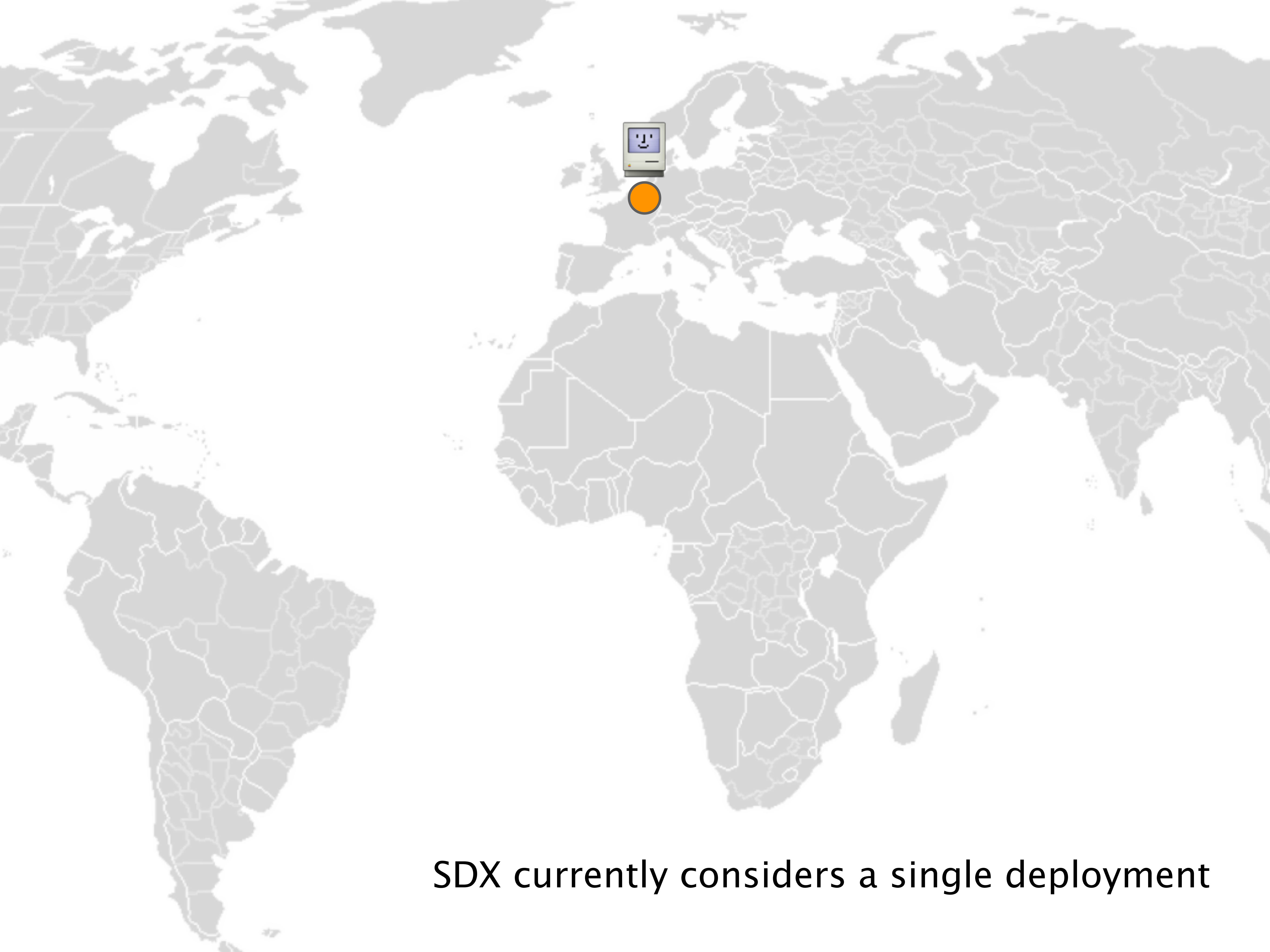
Applications

inter domain bonanza

What's next?

Internet ♥ SDX

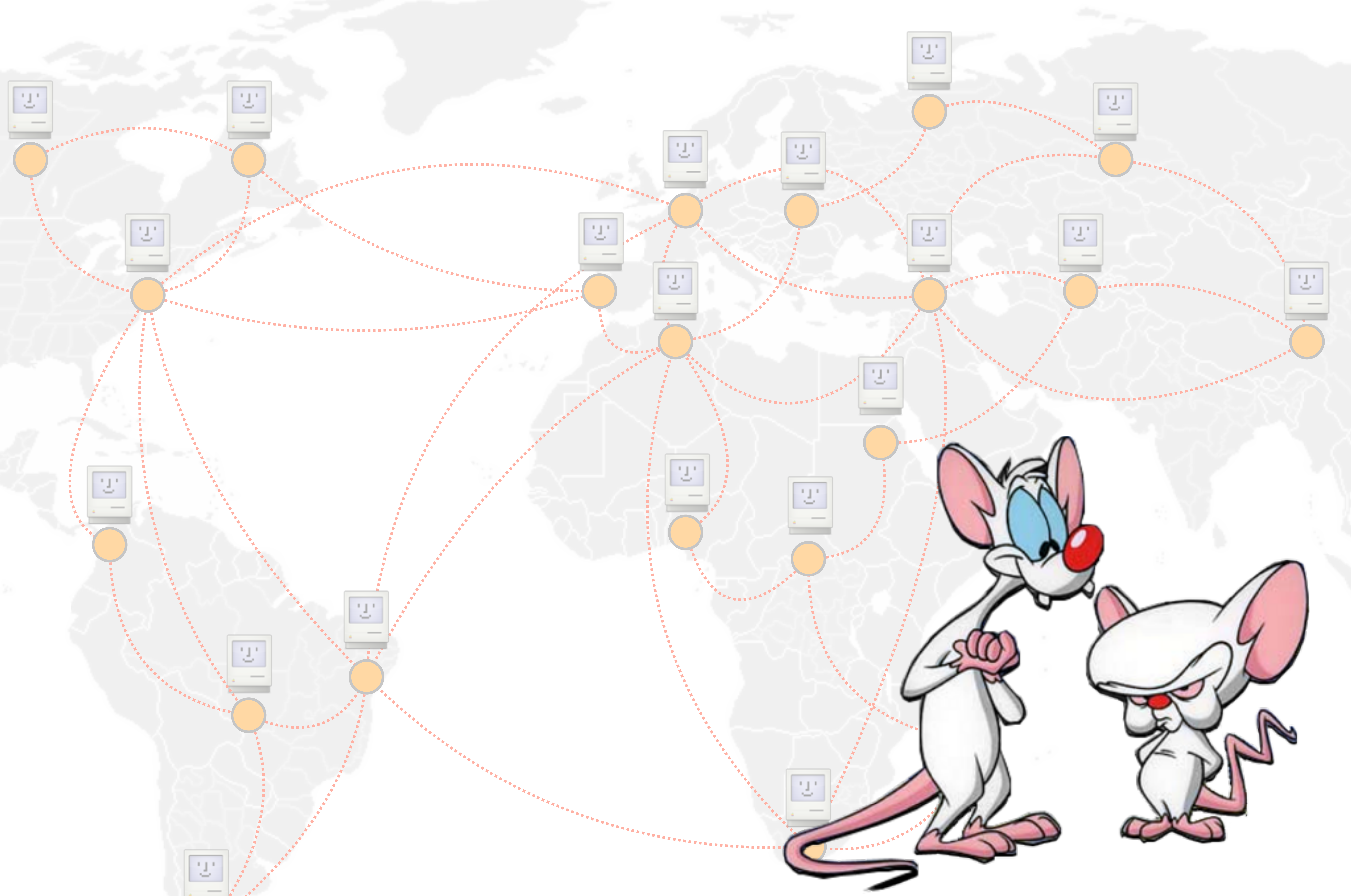
Building a SDX-mediated Internet



SDX currently considers a single deployment

Step 1: interconnecting SDX platforms

Step 2: completely replacing BGP with a SDX-mediated Internet



“Let’s take over the world”

How can our platform
benefit future efforts?

Our SDX platform can serve as skeleton for a SDX ecosystem

We have running code (*)

with full BGP integration, check out our tutorial

We are in the process of having a first deployment

SNAP @ ColoATL, planned deployment with GENI

Many interested parties already

important potential for impact

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

Novel Applications for a SDN-enabled Internet eXchange Point

Laurent Vanbever

www.vanbever.eu

SDX Workshop, Washington DC

June, 5 2014