lecture 13:
more SDN challenges

5590: software defined networking

anduo wang, Temple University

TTLMAN 401B, R 17:30-20:00

challenges

security

orchestration

middleboxes

security challenge

security challenge

SDN control platform and programming abstraction

- simplify management and programming

malicious user?

- lacks security enforcement mechanism

P Porras., et al. "A security enforcement kernel for OpenFlow networks"

bypassing firewall

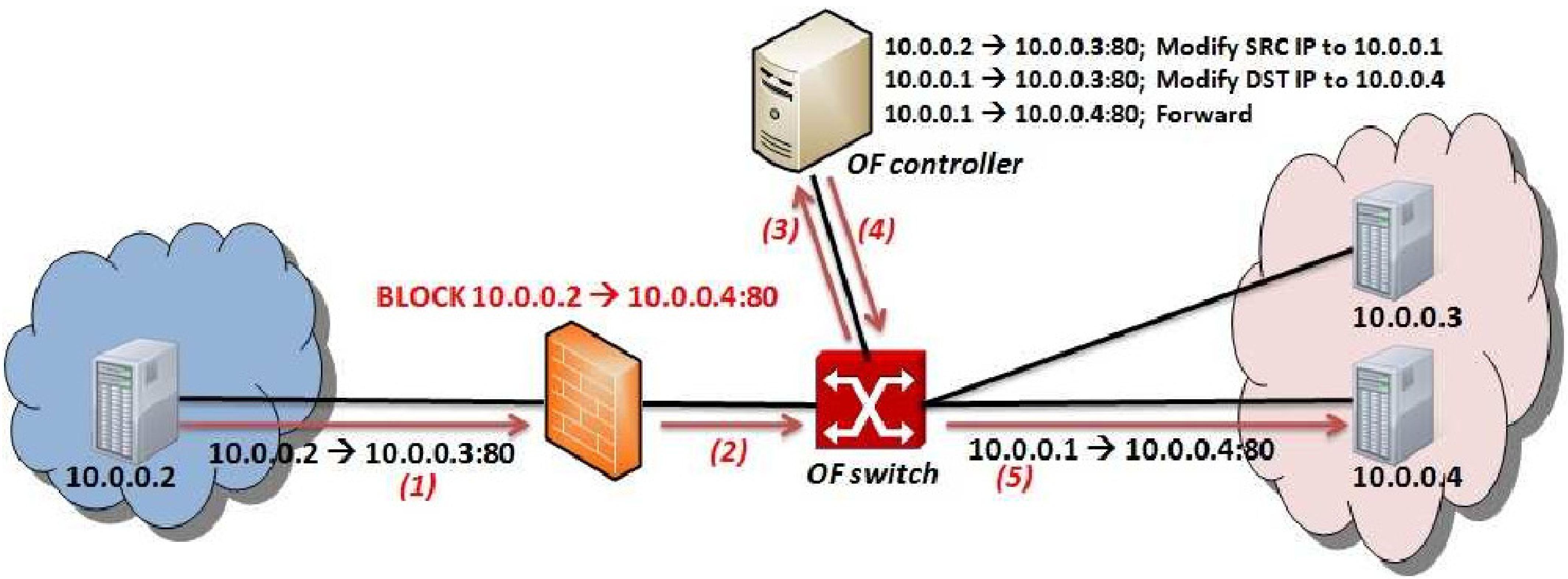


Figure 1: Dynamic Flow Tunneling Scenario

P Porras., et al. "A security enforcement kernel for OpenFlow networks"

bypassing firewall

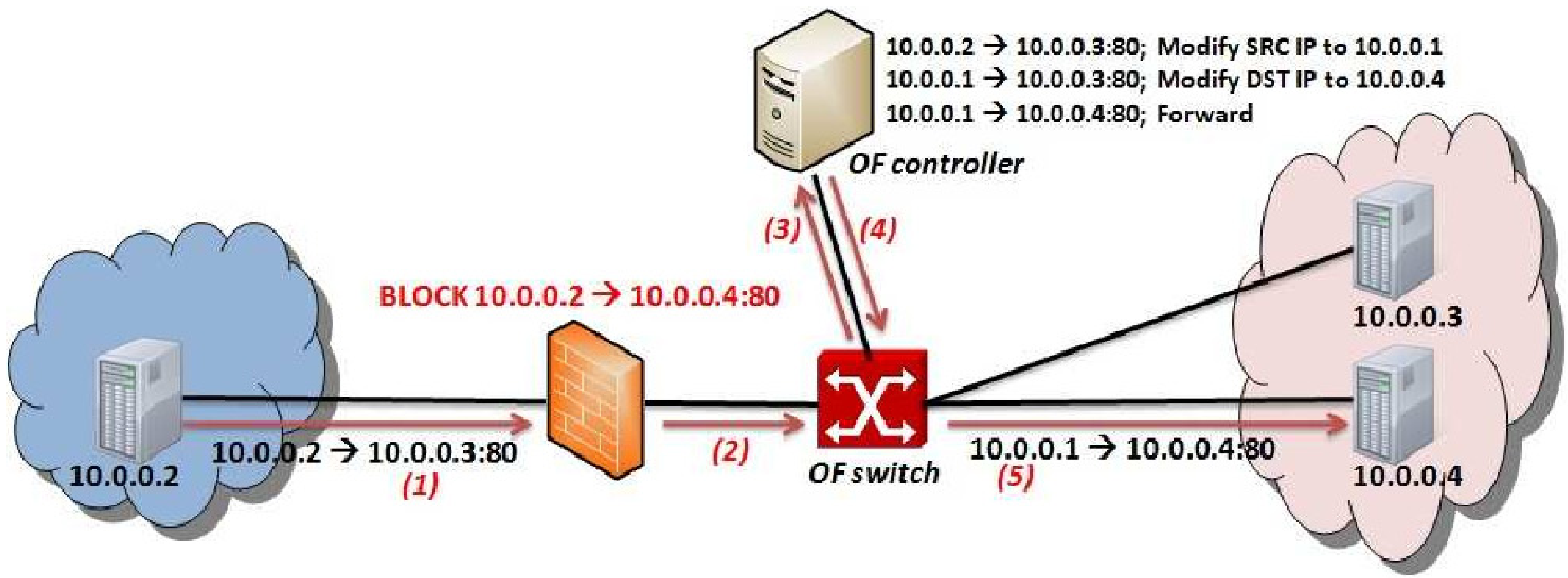


Figure 1: Dynamic Flow Tunneling Scenario

rule conflict

P Porras., et al. "A security enforcement kernel for OpenFlow networks"

FortNOX goal

- security constraint enforcement
  - a live rule conflict detection engine
- rule conflict
  - the candidate OP rule enables or disables a network flow that is otherwise inversely prohibited (allowed) by existing rules

P Porras., et al. "A security enforcement kernel for OpenFlow networks"

FortNOX solution — alias set

alias reduced rules (ARR)

- a rewrite creates an "alias"

OF rules		ARR
$a \longrightarrow b$; action	$\longrightarrow$	$\{a\} \longrightarrow \{b\}$; action
$a \longrightarrow b$; set $(a \Longrightarrow a')$		$\{a,a'\} \longrightarrow \{b\}$
$a' \longrightarrow b$; set $(b \Longrightarrow b')$		$\{a,a'\} \longrightarrow \{b,b'\}$

P Porras., et al. "A security enforcement kernel for OpenFlow networks"

bypassing firewall

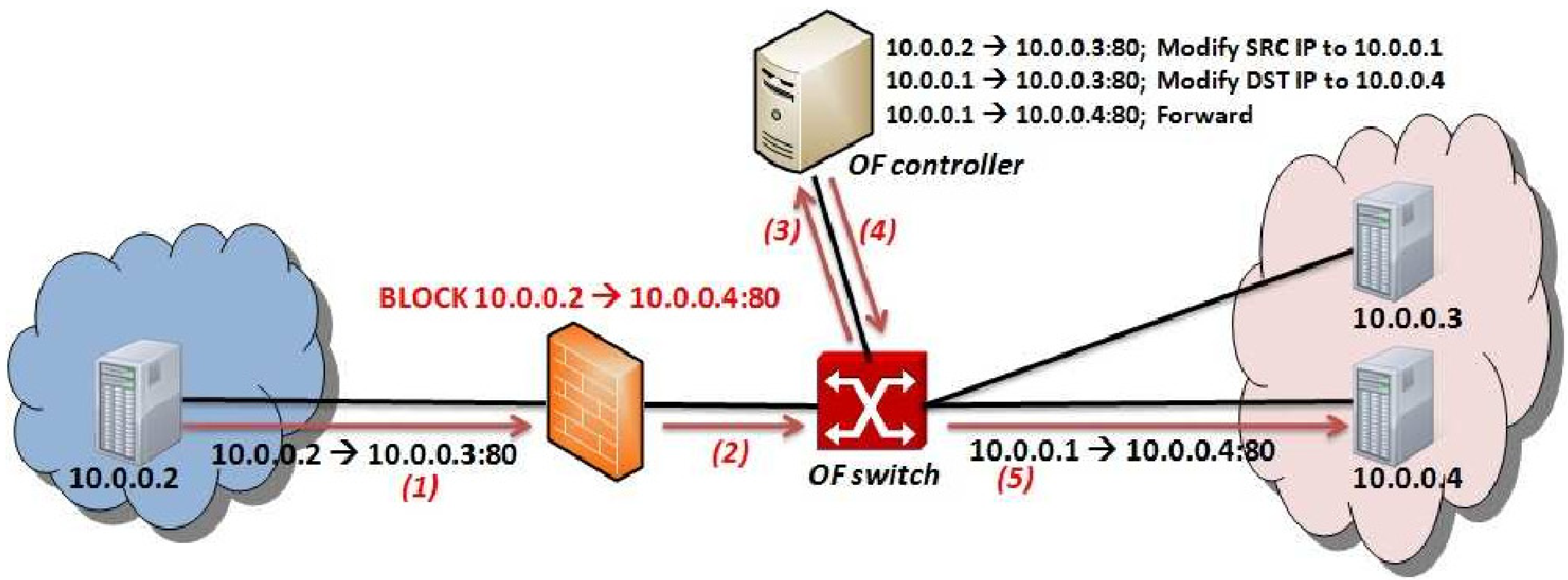


Figure 1: Dynamic Flow Tunneling Scenario

10.2 ⟶ 10.3; set 10.2 to 10.1
10.1 ⟶ 10.3; set 10.3 to 10.4
10.1 ⟶ 10.4; forward

⟶

{10.2, 10.1} ⟶ {10.3}
{10.2, 10.1} ⟶ {10.3, 10.4}
{10.2, 10.1} ⟶ {10.3, 10.4}; forward

P Porras., et al. "A security enforcement kernel for OpenFlow networks"

bypassing firewall

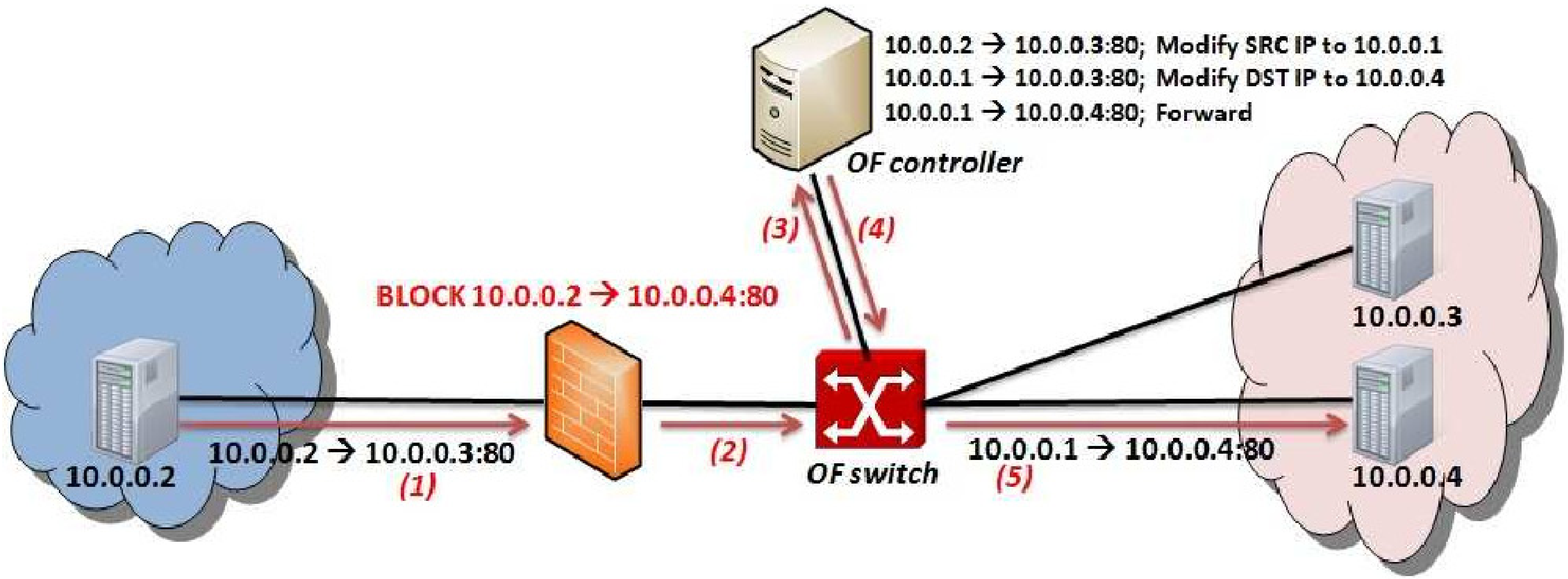


Figure 1: Dynamic Flow Tunneling Scenario

10.2 ⟶ 10.3; set 10.2 to 10.1
10.1 ⟶ 10.3; set 10.3 to 10.4
10.1 ⟶ 10.4; forward

→

{10.2, 10.1} ⟶ {10.3}
{10.2, 10.1} ⟶ {10.3, 10.4}
{10.2, 10.1} ⟶ {10.3, 10.4}; forward

P Porras., et al. "A security enforcement kernel for OpenFlow networks"

orchestration challenge

orchestration challenge

network operation is complex

- early days: data pipes that enable communication among computers
- the present
  - securing resources, performance, reliability, value-added services

Z Cai., et al. "The Preliminary Design and Implementation of the Maestro Network Control Platform"

fundamental problem

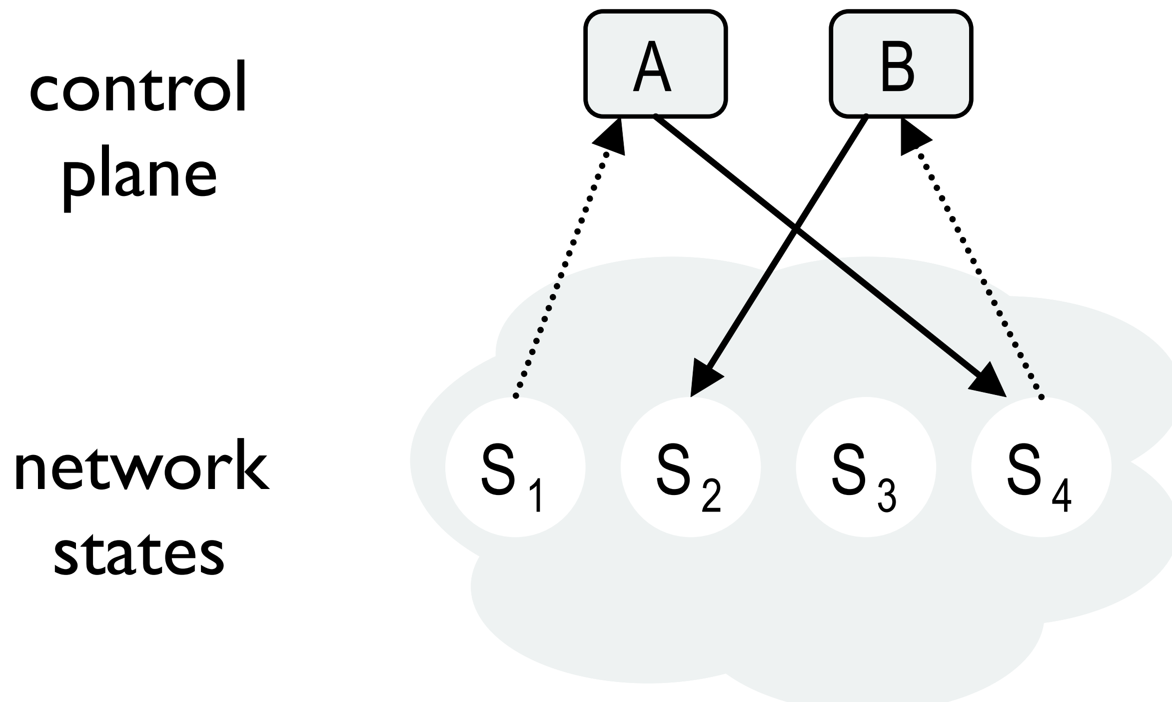
modular approach decomposes the complexity into more manageable pieces

BUT

- the modules concurrently modify the behavior of the shared network

Z Cai., et al. "The Preliminary Design and Implementation of the Maestro Network Control Platform"

orchestration

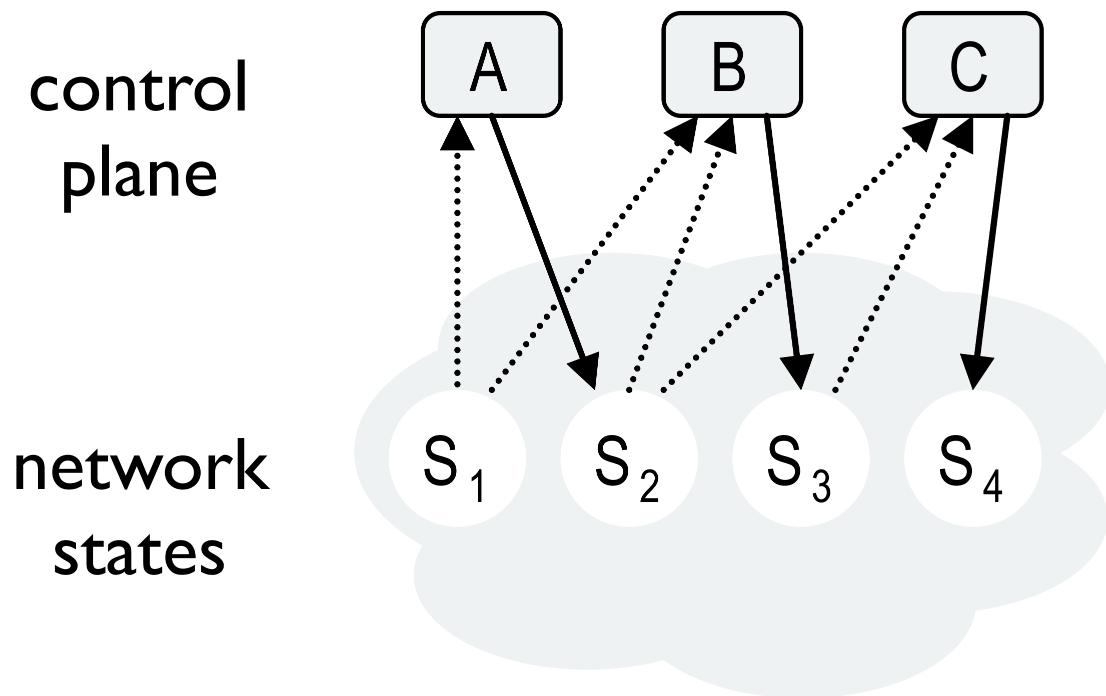


communication

- decision of one component can depend on another
- A communicates to B

Z Cai., et al. "The Preliminary Design and Implementation of the Maestro Network Control Platform"

orchestration

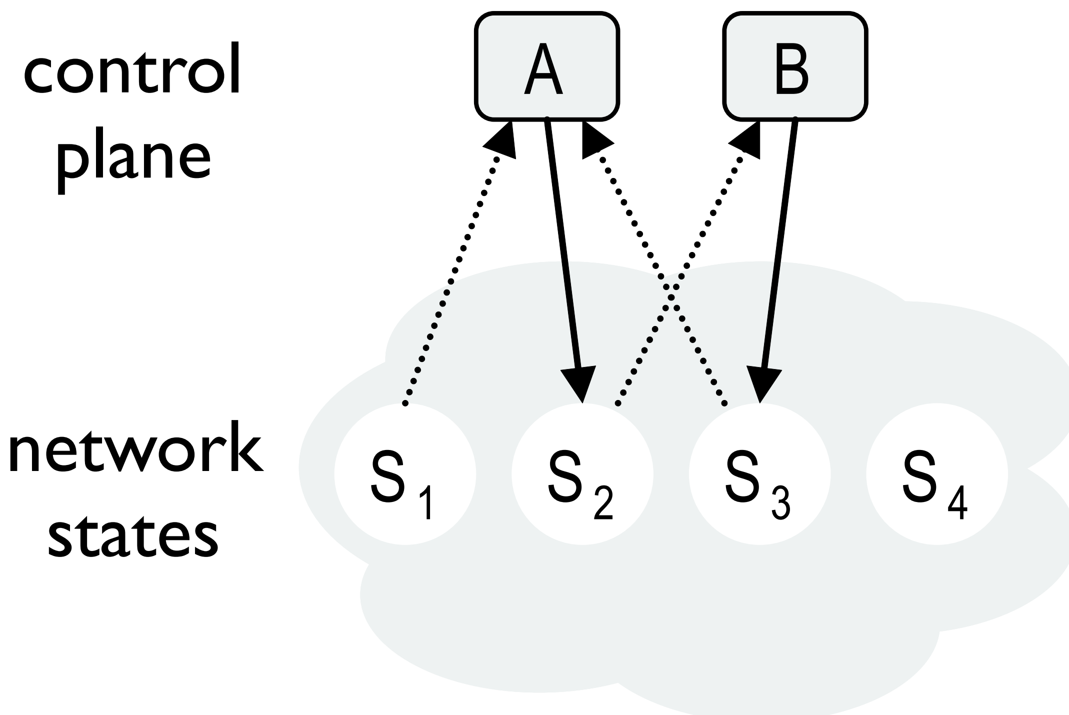


scheduling

- components need to communicate their decisions, and to schedule their executions
  - communication not enough
  - explicit scheduling of the execution order is needed

Z Cai., et al. "The Preliminary Design and Implementation of the Maestro Network Control Platform"

orchestration

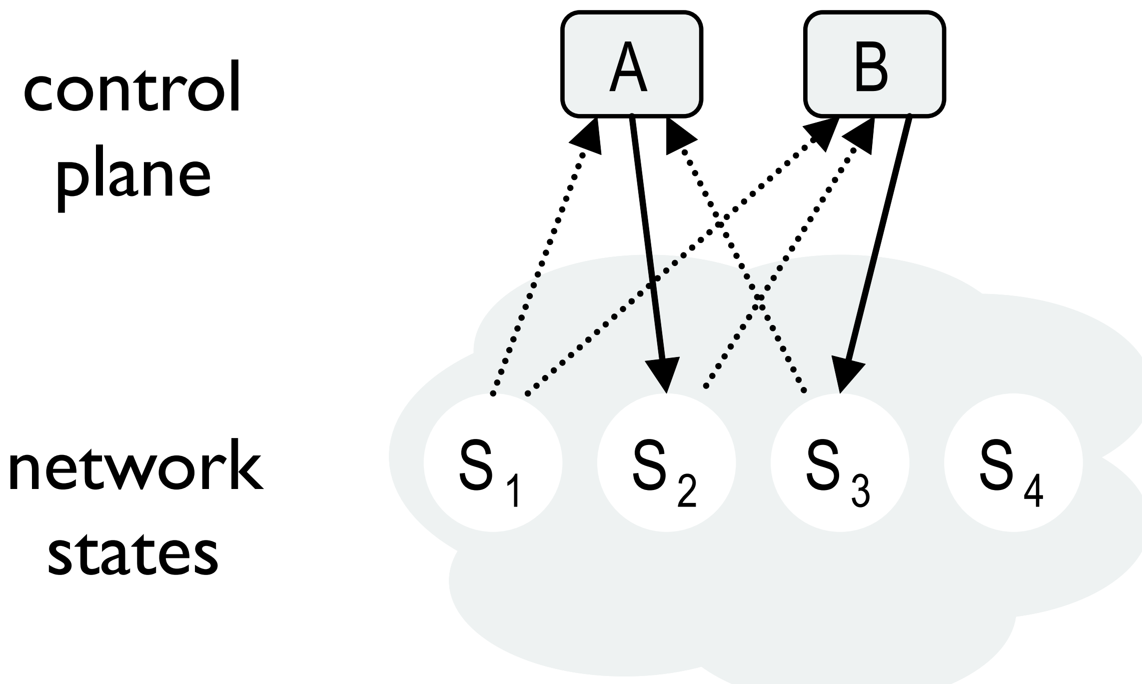


feedback

- network behavior caused by one component may inadvertently change the input condition of another
  - no guarantee that S_2 and S_3 will stabilize

Z Cai., et al. "The Preliminary Design and Implementation of the Maestro Network Control Platform"

orchestration

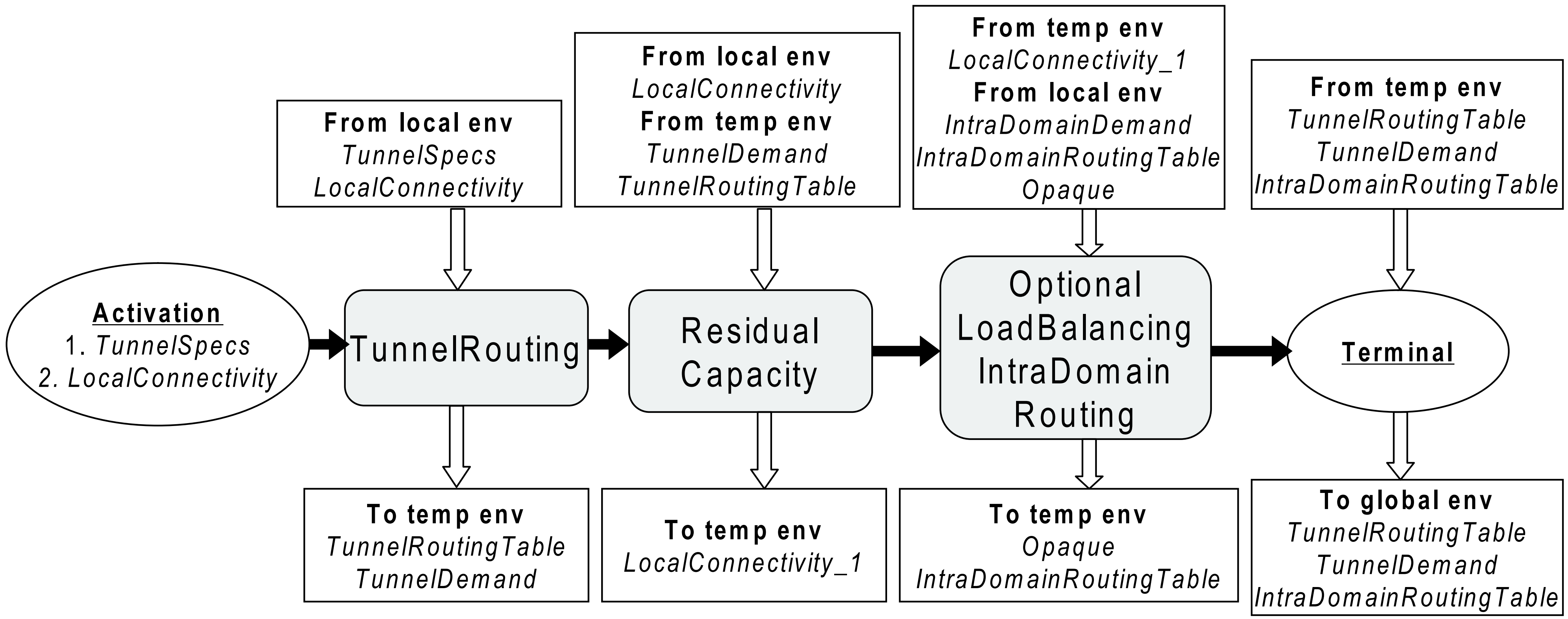


concurrency

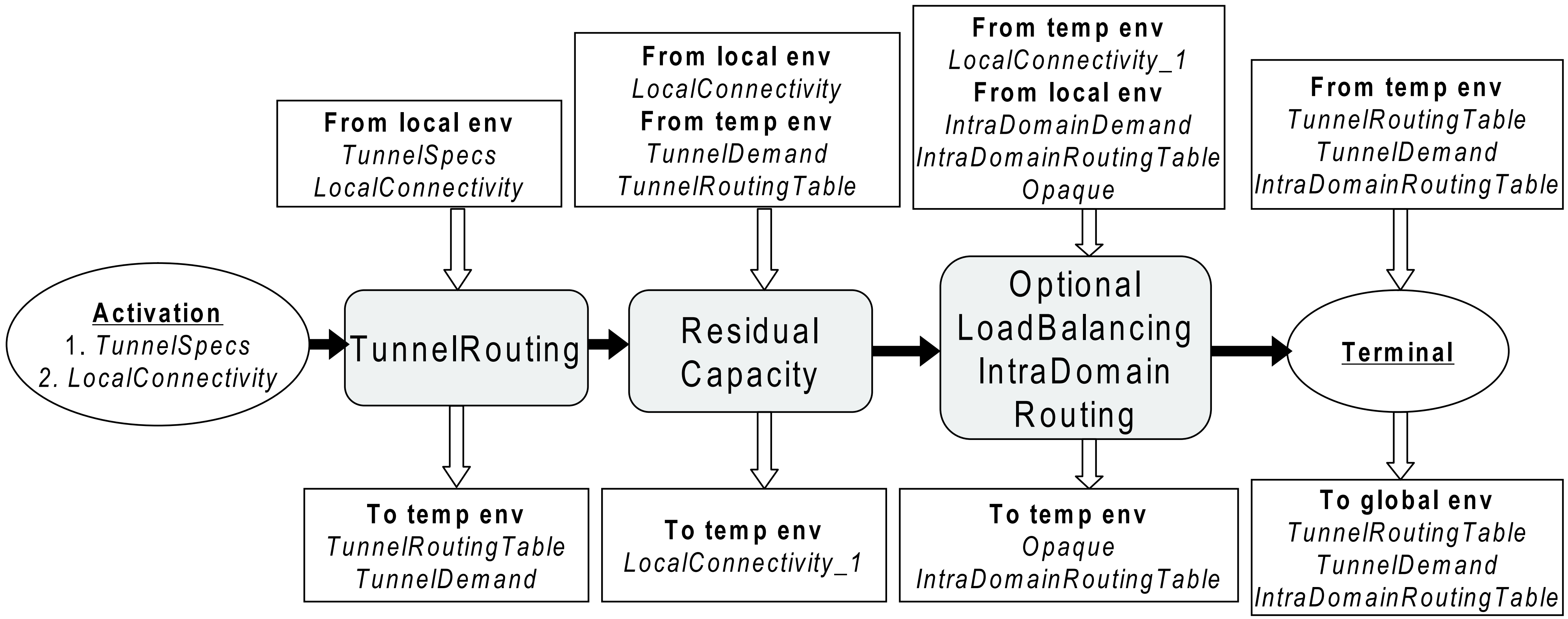
- network behavior caused by one component may inadvertently change the input condition of another
  - concurrent actions of inter-dependent control components may lead to a inconsistent network state

Z Cai., et al. “The Preliminary Design and Implementation of the Maestro Network Control Platform”

Maestro solution for communication & scheduling



Maestro solution for communication & scheduling



DAG abstraction

- specifies the composition of three applications
- communication
  - binding input and output views (a network state abstraction)
- scheduling
  - explicit ordering of executing control components

middlebox challenge

Seyed Kaveh Fayazbakhsh., et al. “Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags”

middleboxes and SDN

key security and performance guarantees

BUT, introduces new challenges

- difficult to ensure the "service-chaining" policies
  - e.g., web traffic processed by a proxy and then a firewall
- hinder performance debugging & forensics
- exacerbated with virtualized/multi-tenant deployments

root causes

traffic is modified by dynamic and opaque middle box behaviors

—violating two SDN principles

- *origin binding:* between a packet and its "origin"
- *path follow policy:* policy explicitly determine the path packets follow

origin-binding violation — I

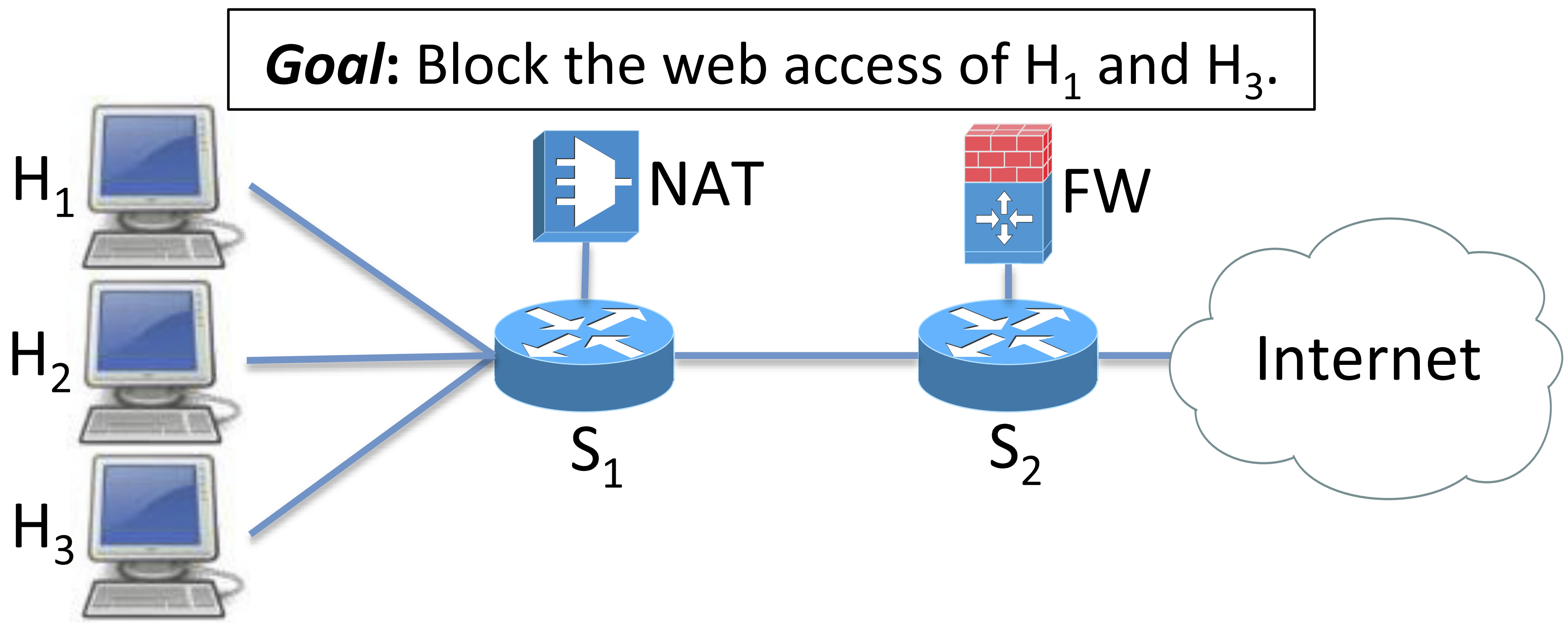


Figure 1: Applying the blocking policy is challenging, as the NAT hides the true packet sources.

origin-binding violation — 2

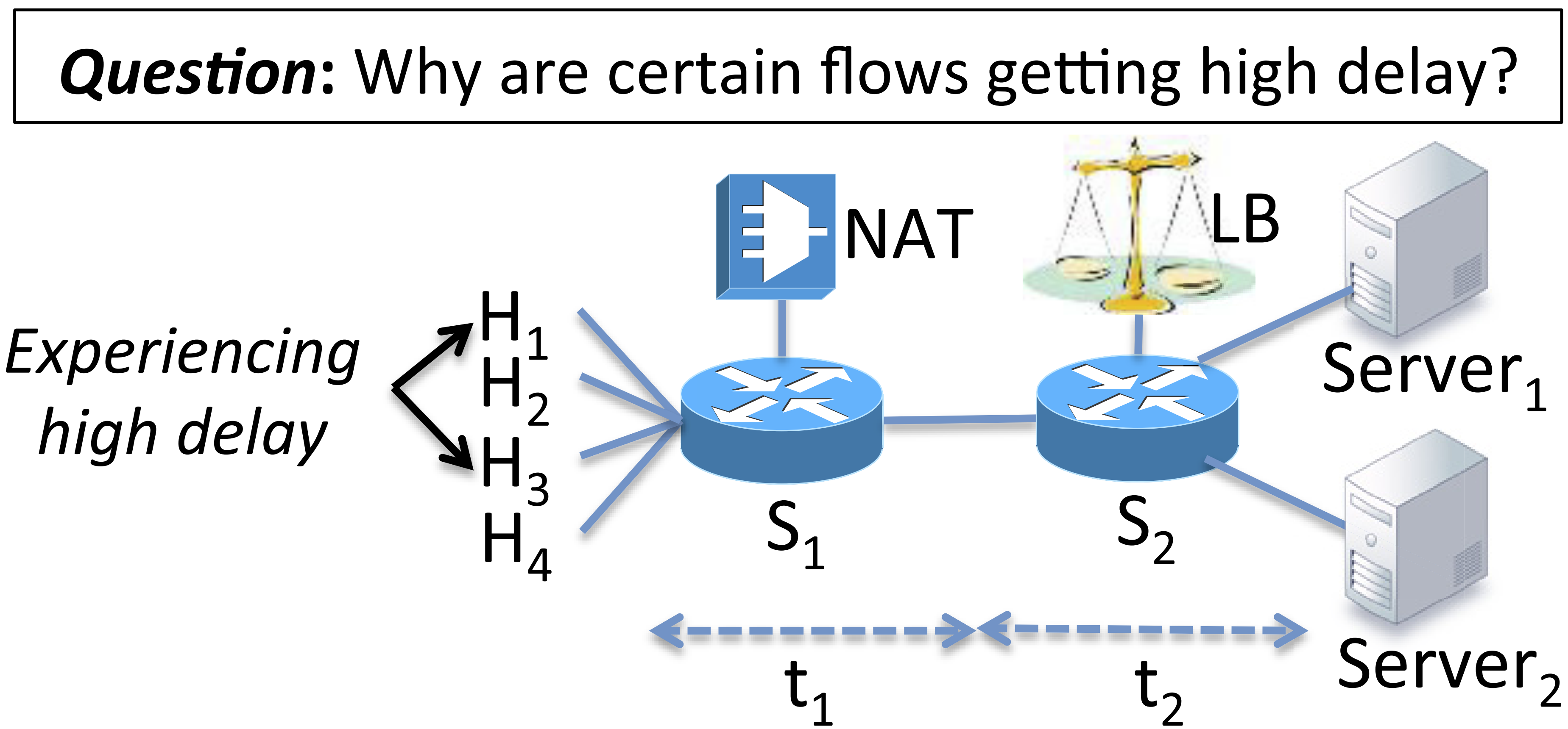


Figure 2: Middlebox modifications make it difficult to consistently correlate network logs for diagnosis.

path-follow-policy violation

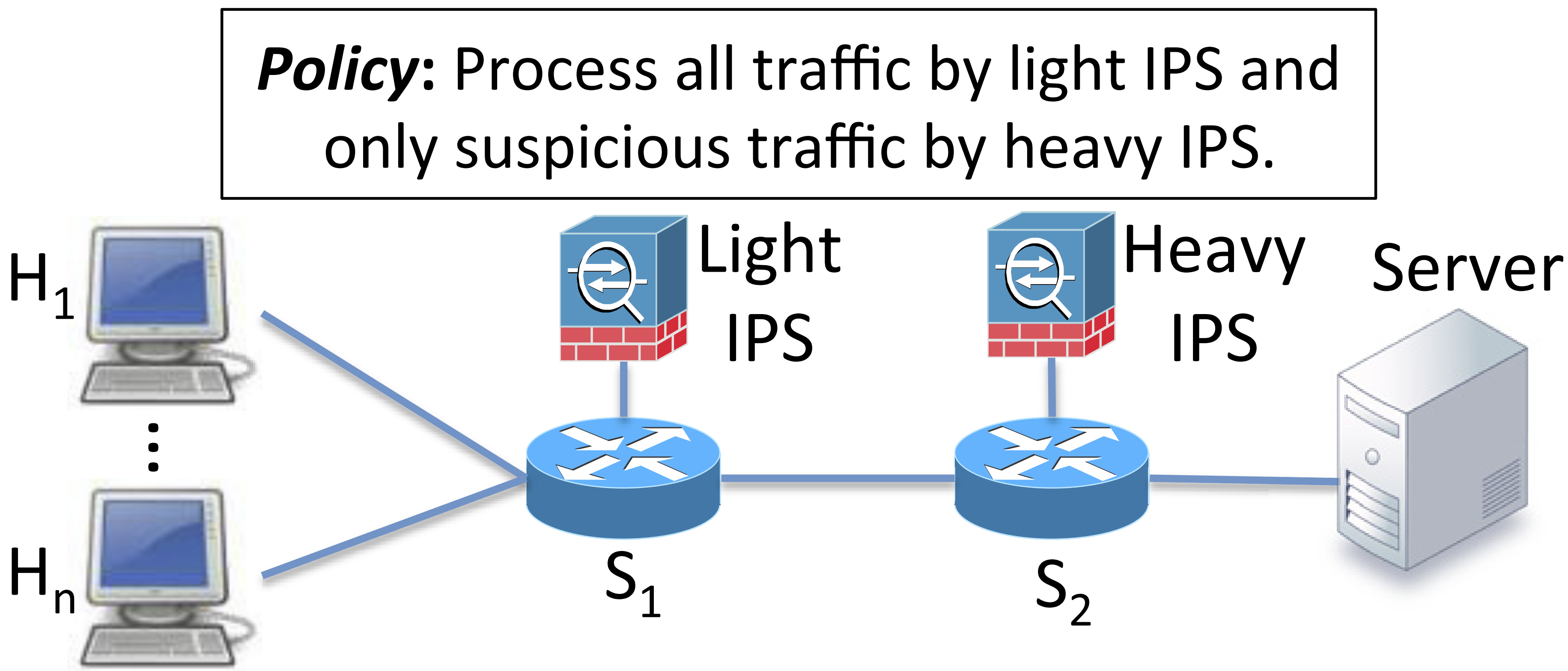


Figure 3: S_2 cannot decide if an incoming packet should be sent to the heavy IPS or the server.

FlowTags solution

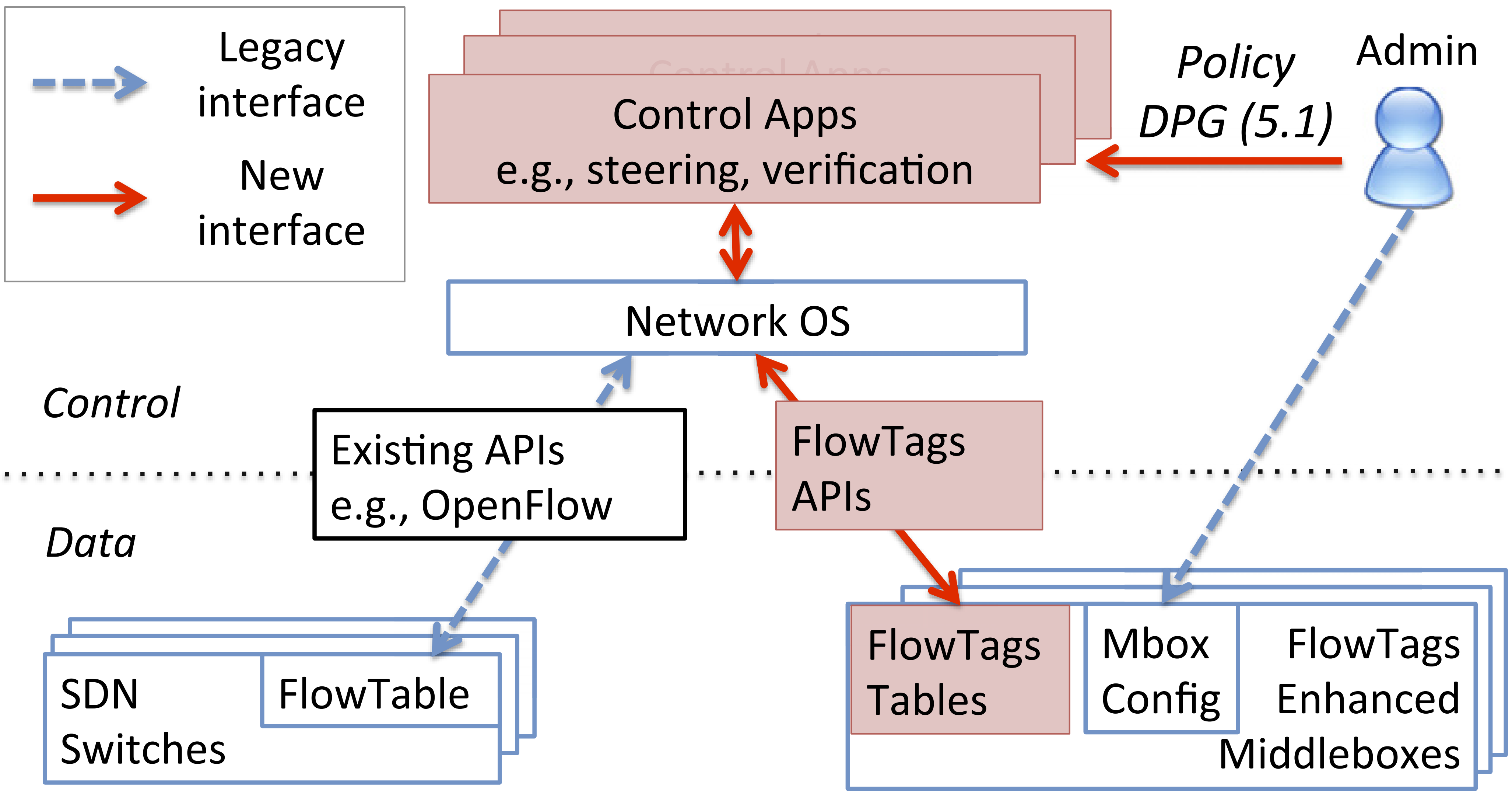
pragmatic stance

- integrate middleboxes into SDN fold as "cleanly" as possible

(re)enforce orgin-binding and policy-follow-path

- flow tracking

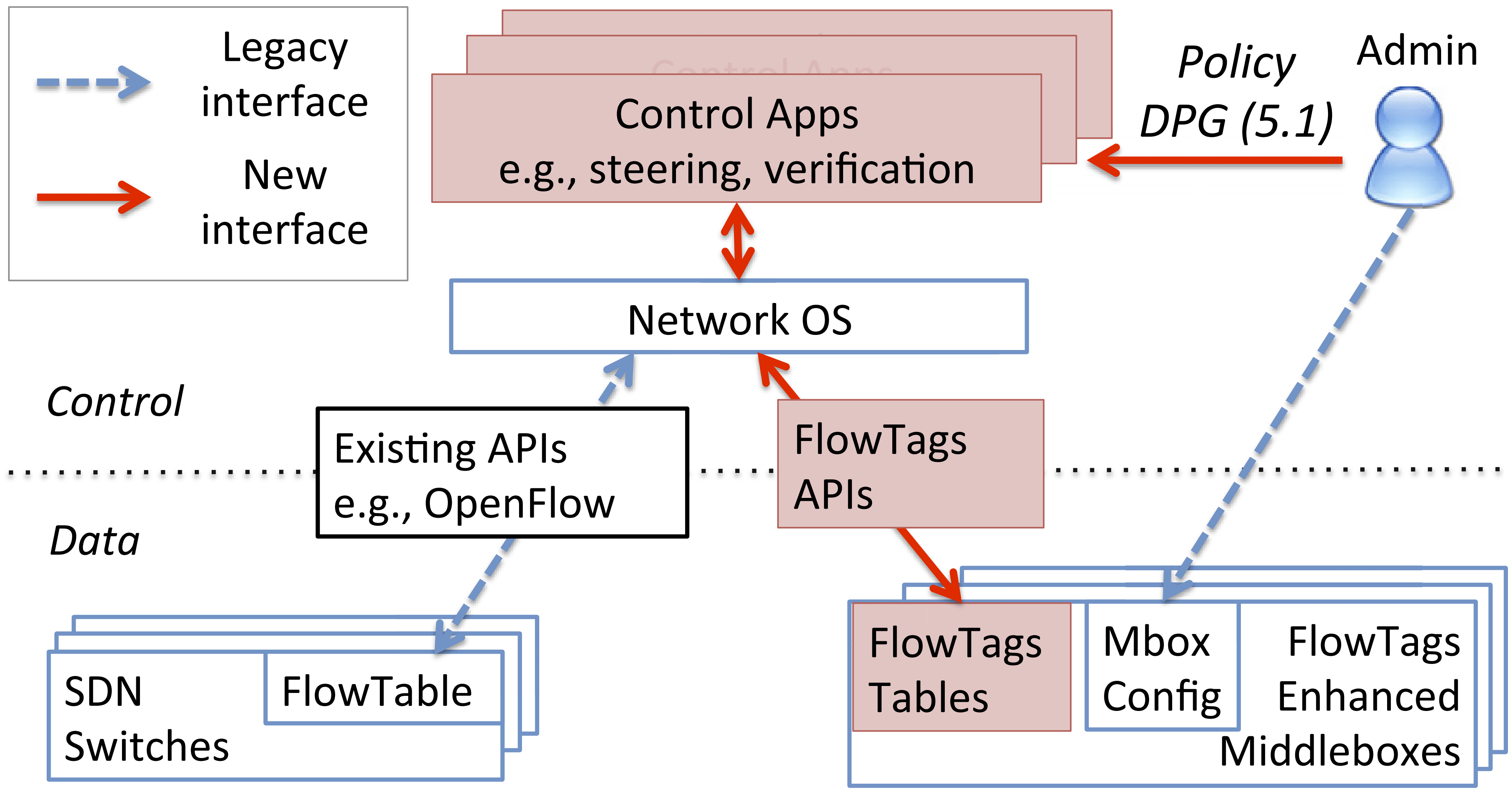
FlowTags overview



tags

- generated by middle boxes, carried in packet headers

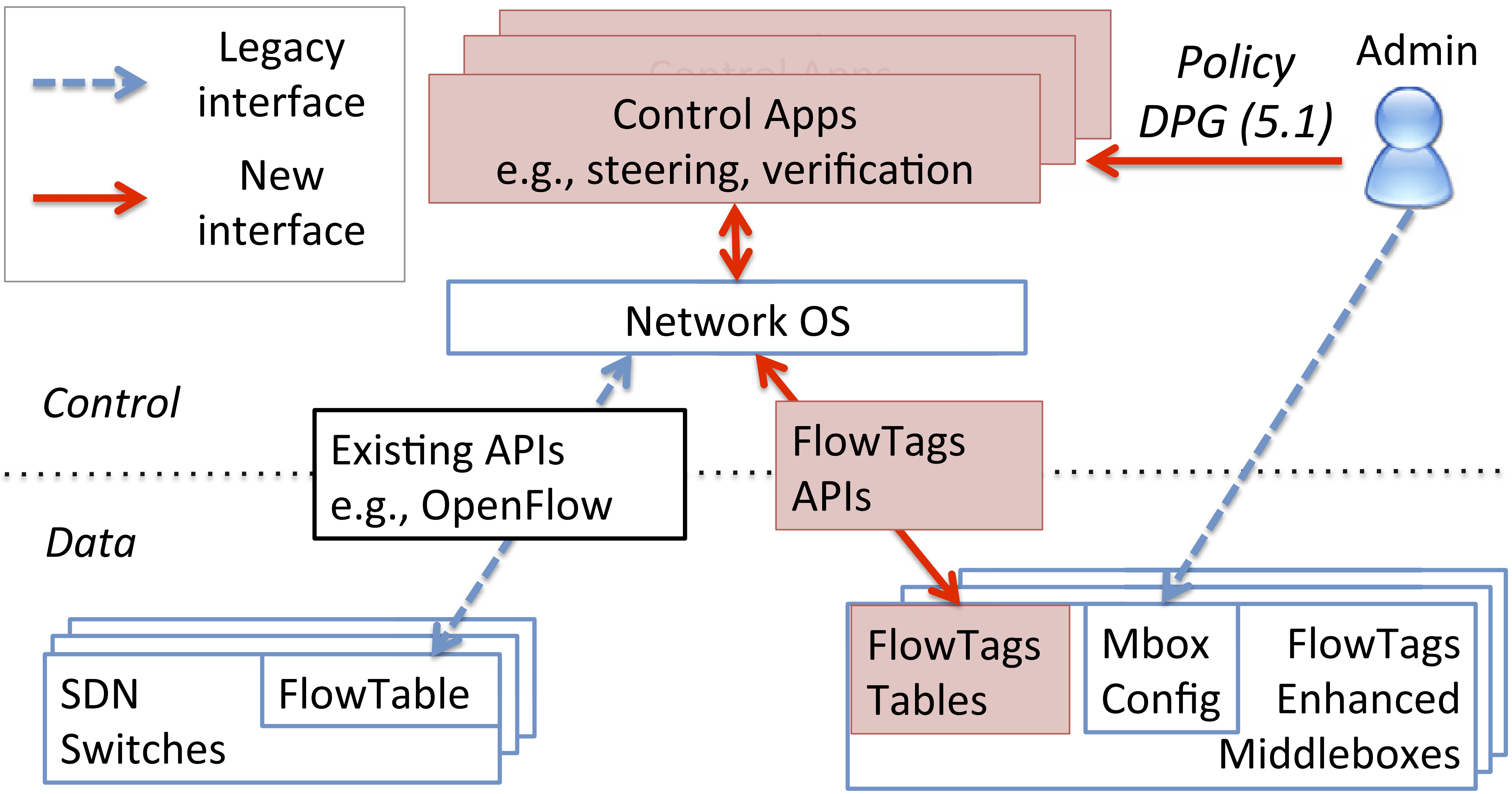
FlowTags overview



SDN switches

- use tags in flow matching

FlowTags overview



downstream middleboxes

- use the tags in packet processing

FlowTags example

NAT Generate Tags

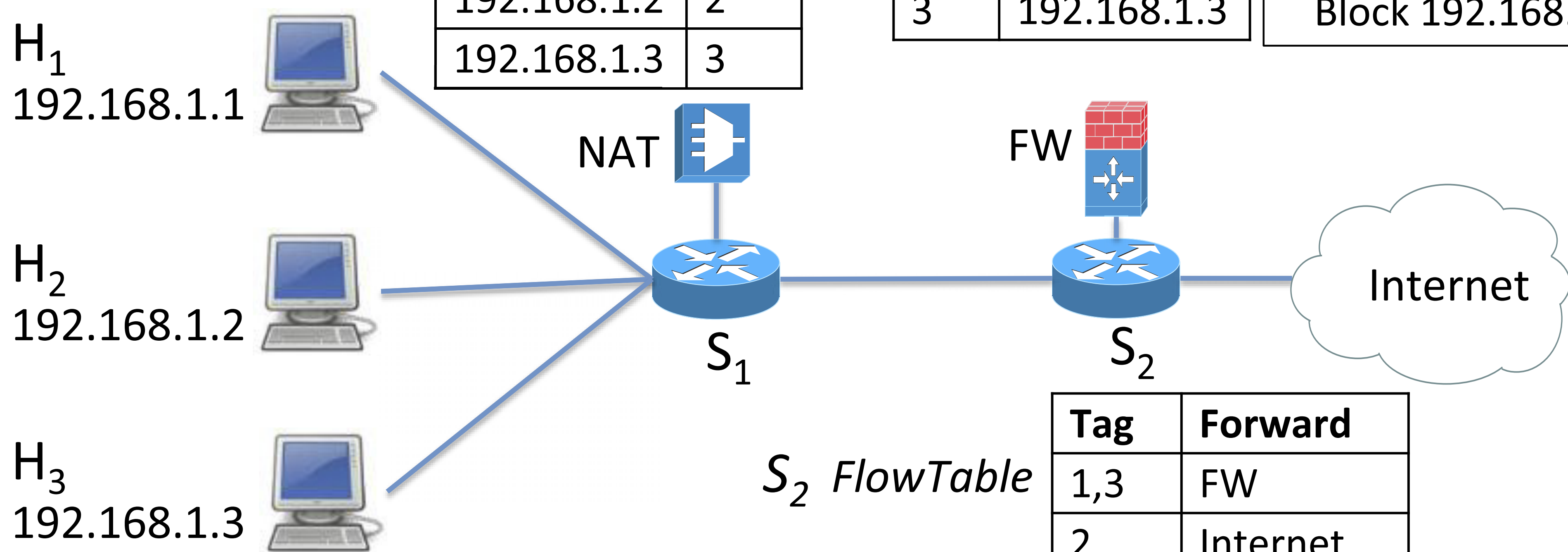
SrcIP	Tag
192.168.1.1	1
192.168.1.2	2
192.168.1.3	3

FW Consume Tags

Tag	OrigSrcIP
1	192.168.1.1
3	192.168.1.3

FW Config

Block 192.168.1.1 Block 192.168.1.3



S_2 FlowTable

Tag	Forward
1,3	FW
2	Internet