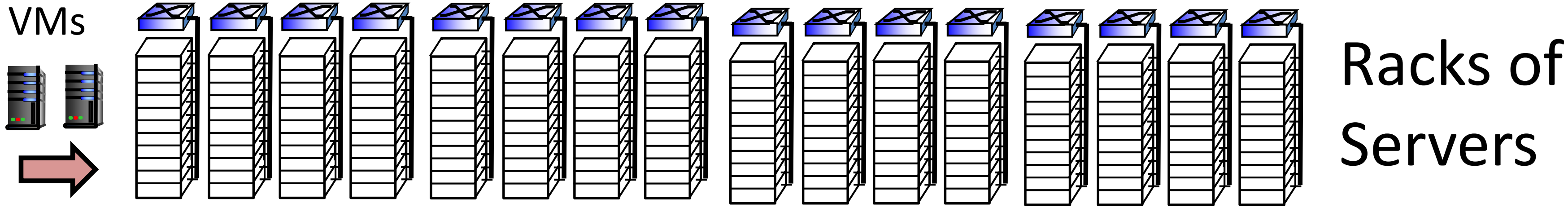
Emerging paradigms in networking: Software-defined networks, programmable dataplanes, and network virtualization

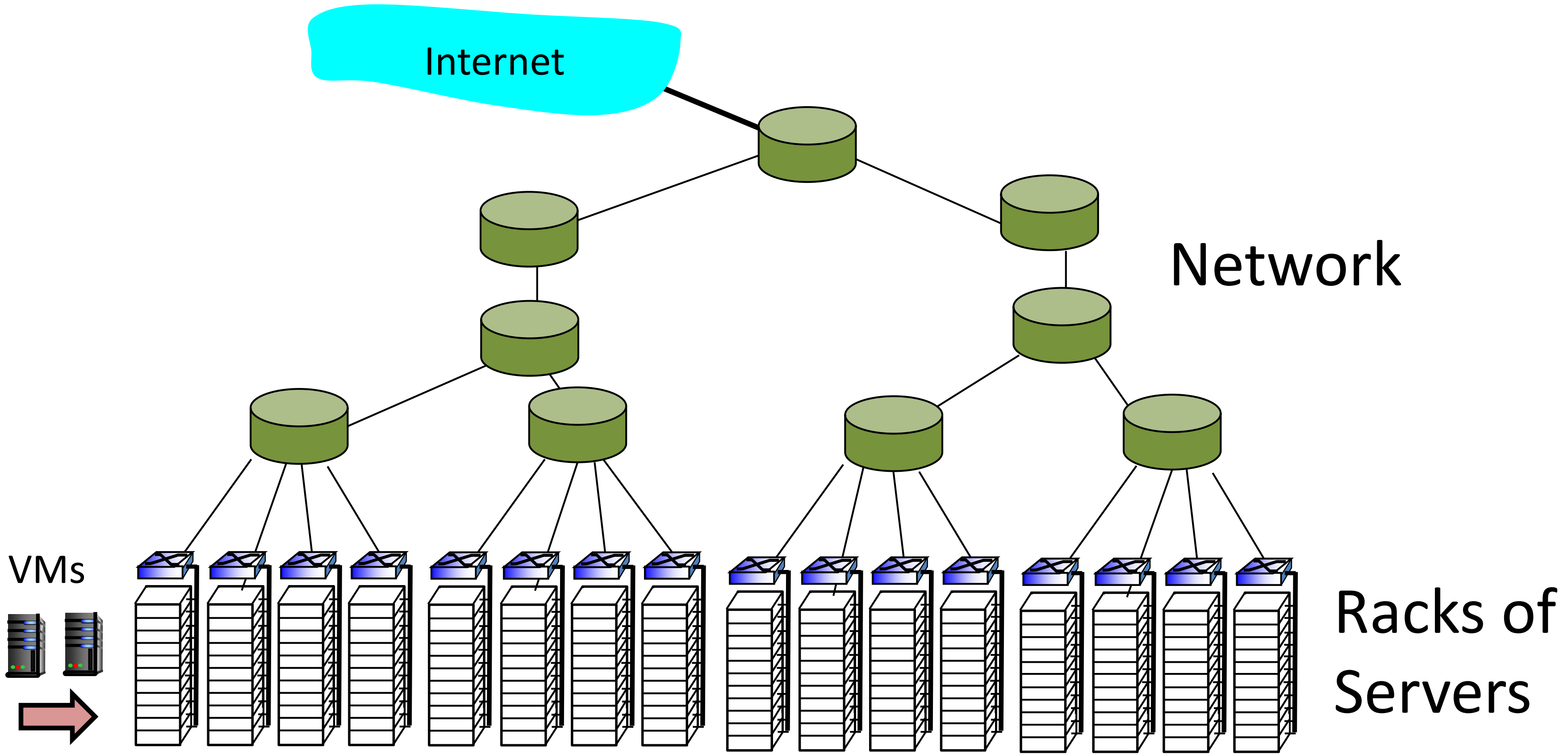
Stefan Schmid

A warmup:
How to design a datacenter network?

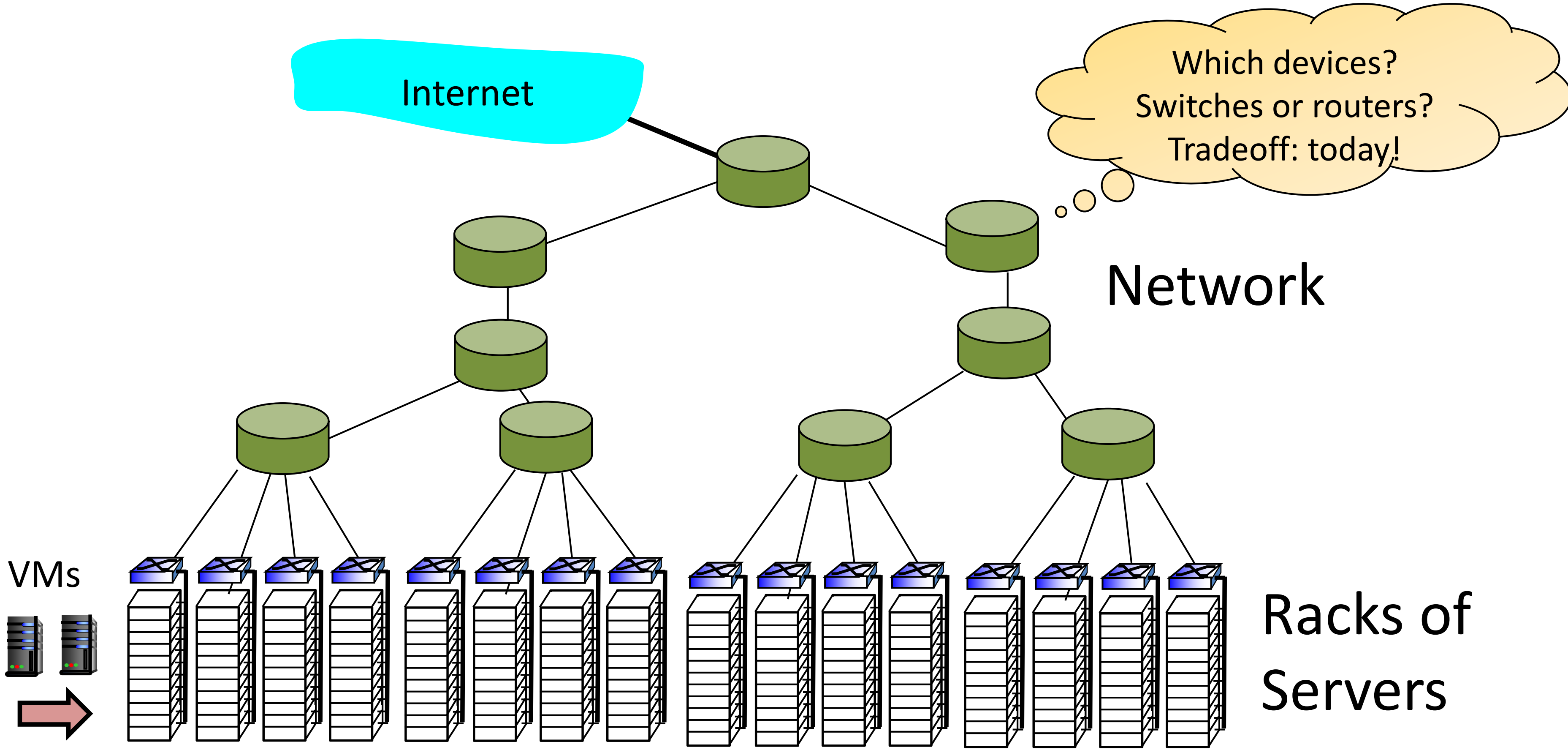
How to design a datacentre network?



How to design a datacentre network?



How to design a datacentre network?



Refresher: Layer-2 Networks

- **Layer-2** networks are very ***flexible***: location-independent addresses, plug&play, self-learning, etc.: devices (and virtual machines!) can move (migrate)

- But: Layer-2 networks do ***not scale***: despite caching, LAN-wide broadcasts needed once in a while (ARP, MAC learning, DHCP, etc.)!

? How large should a LAN be?

Refresher: Layer-2 Networks

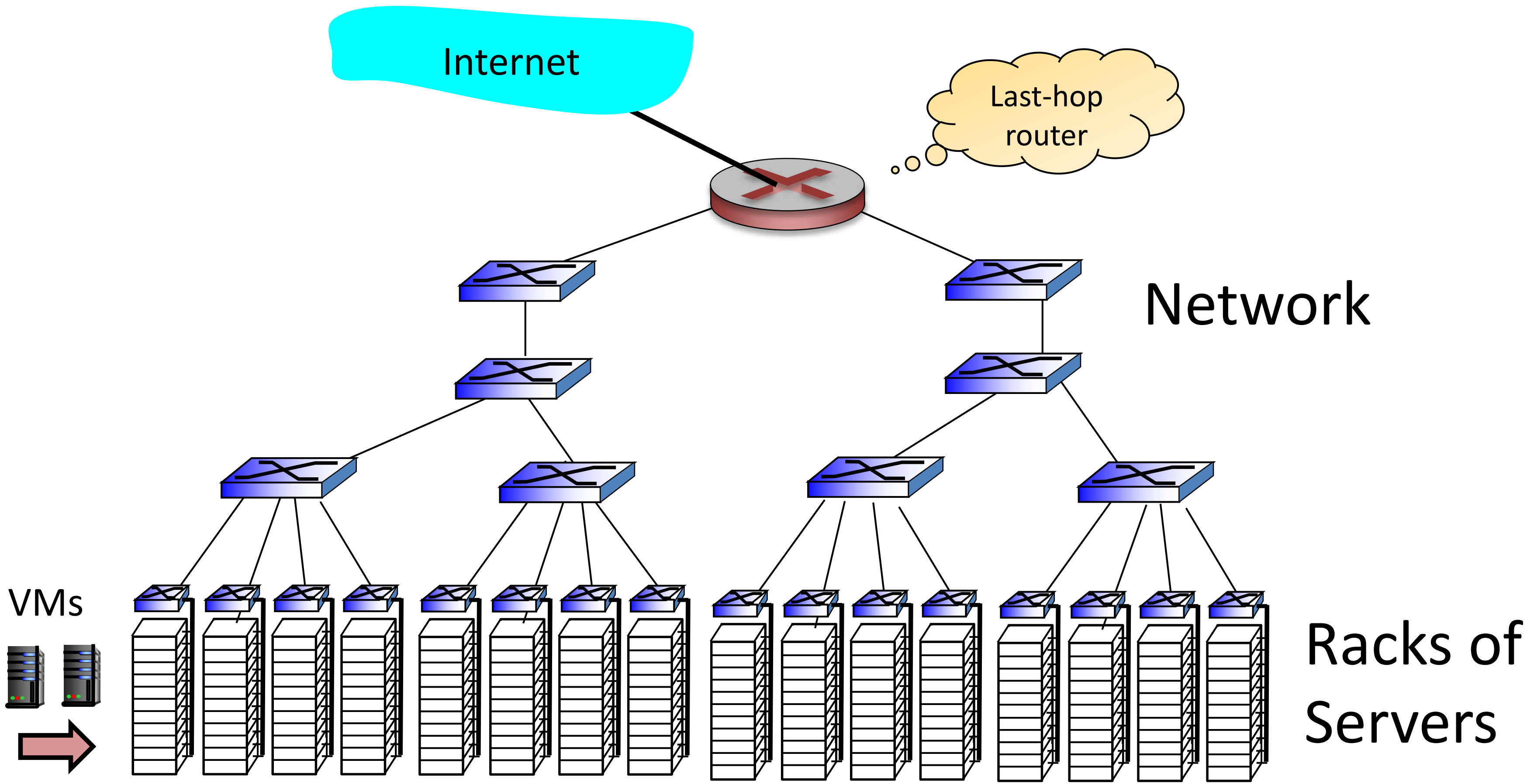
- **Layer-2** networks are very ***flexible***: location-independent addresses, plug&play, self-learning, etc.: devices (and virtual machines!) can move (migrate)

- But: Layer-2 networks do ***not scale***: despite caching, LAN-wide broadcasts needed once in a while (ARP, MAC learning, DHCP, etc.)!

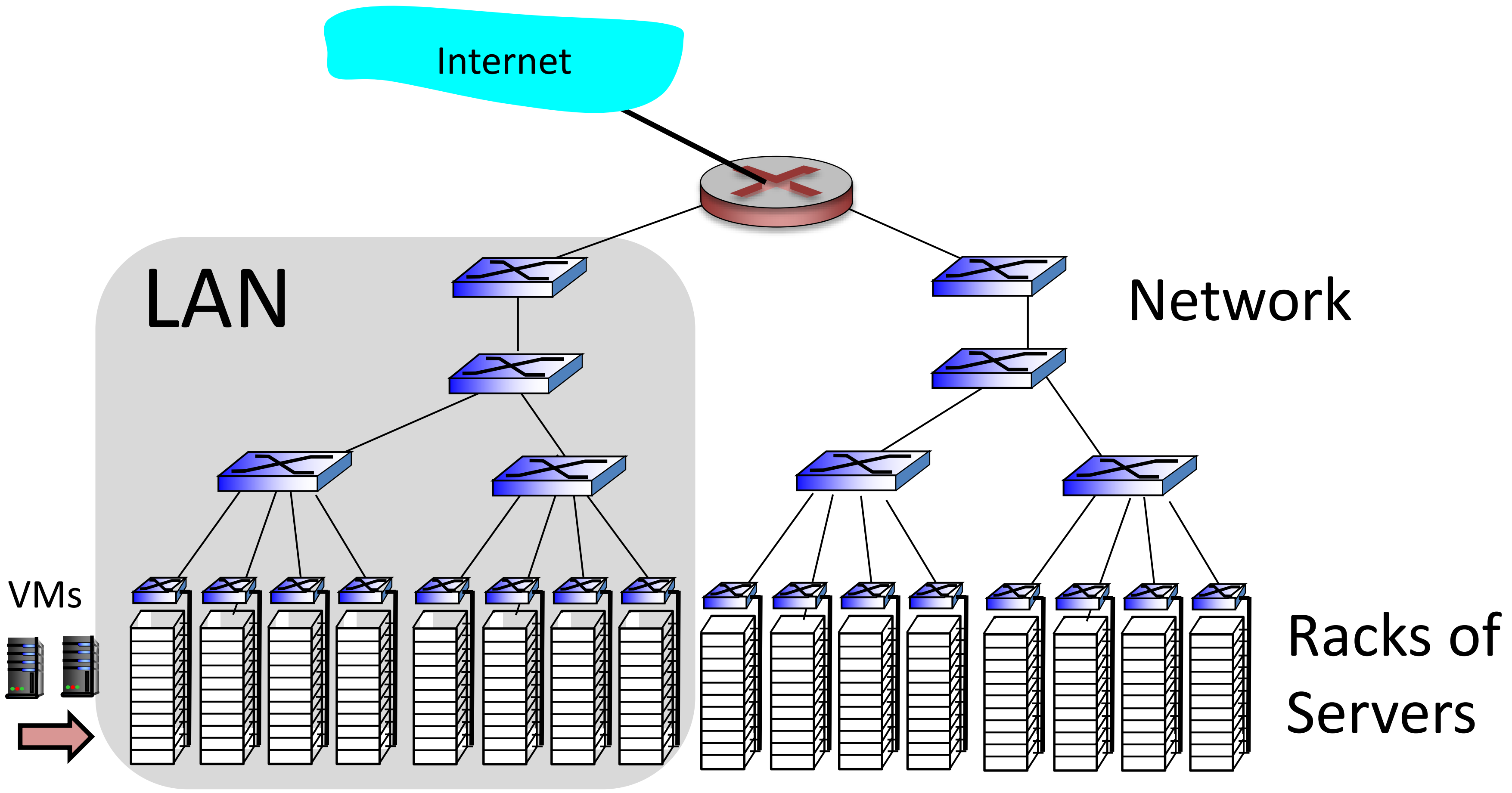
? How large should a LAN be?

! Flexibility vs Scalability tradeoff!

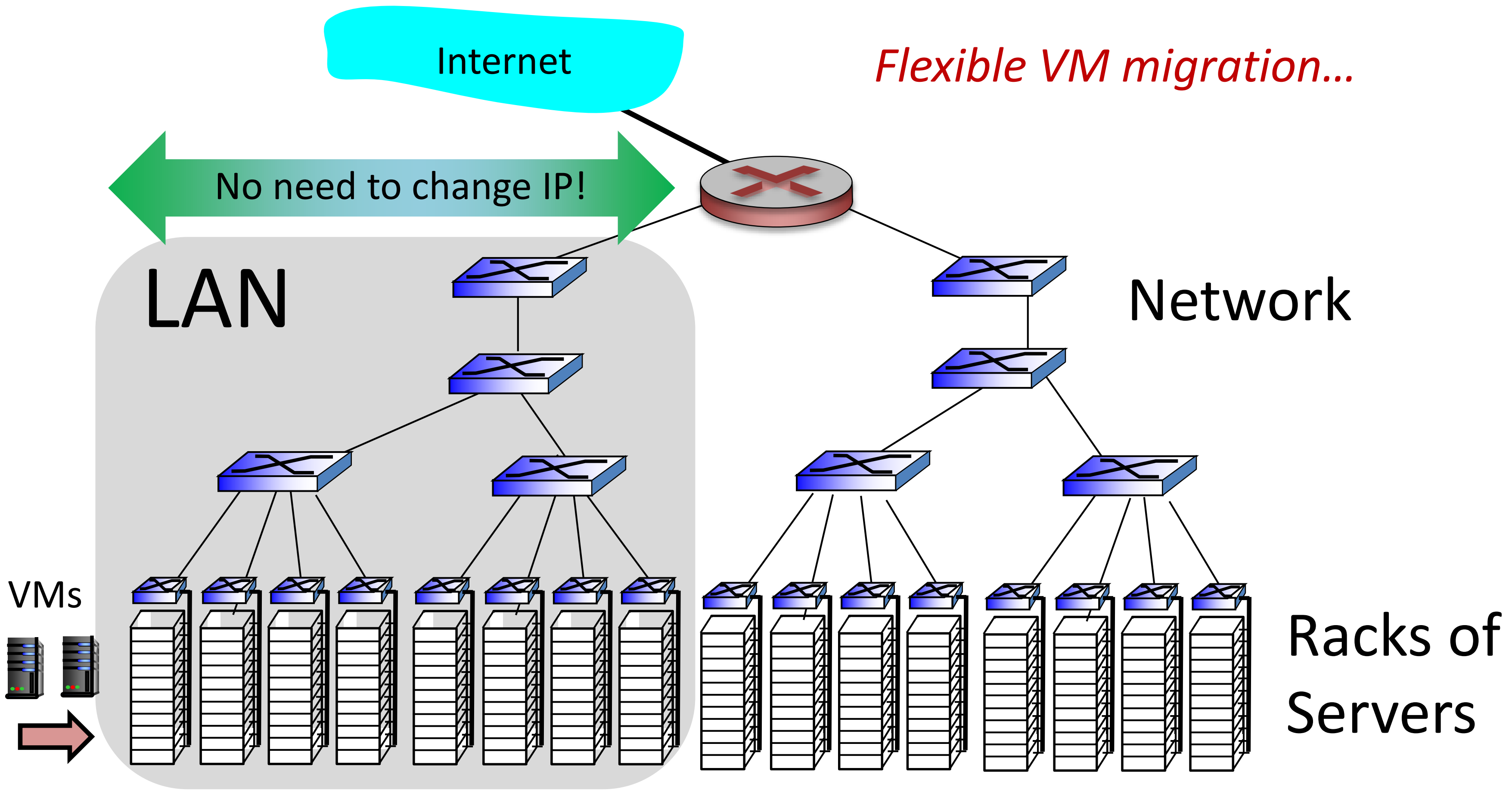
Datacenter Network Design: Proposal #1



Datacenter Network Design: Proposal #1

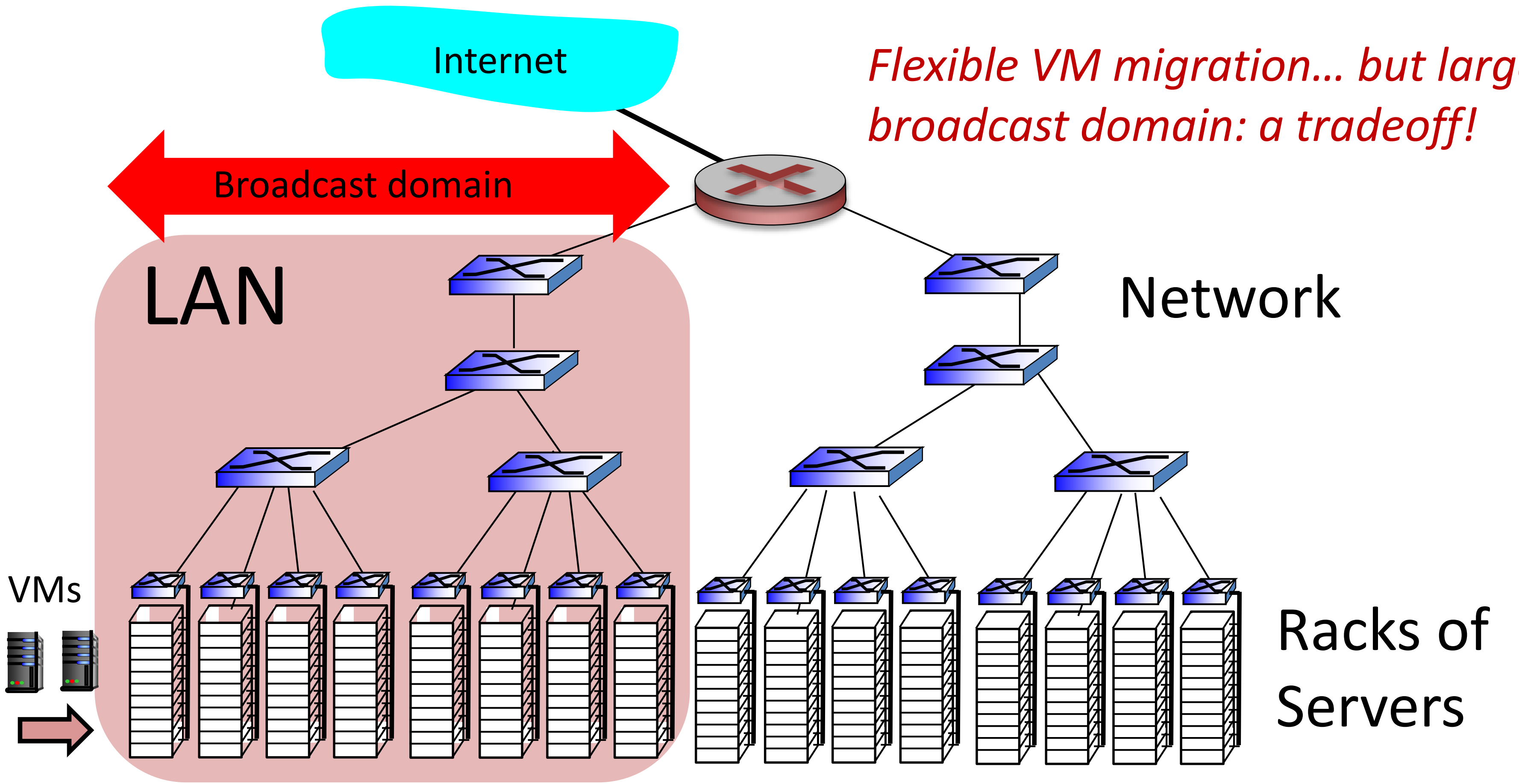


Datacenter Network Design: Proposal #1

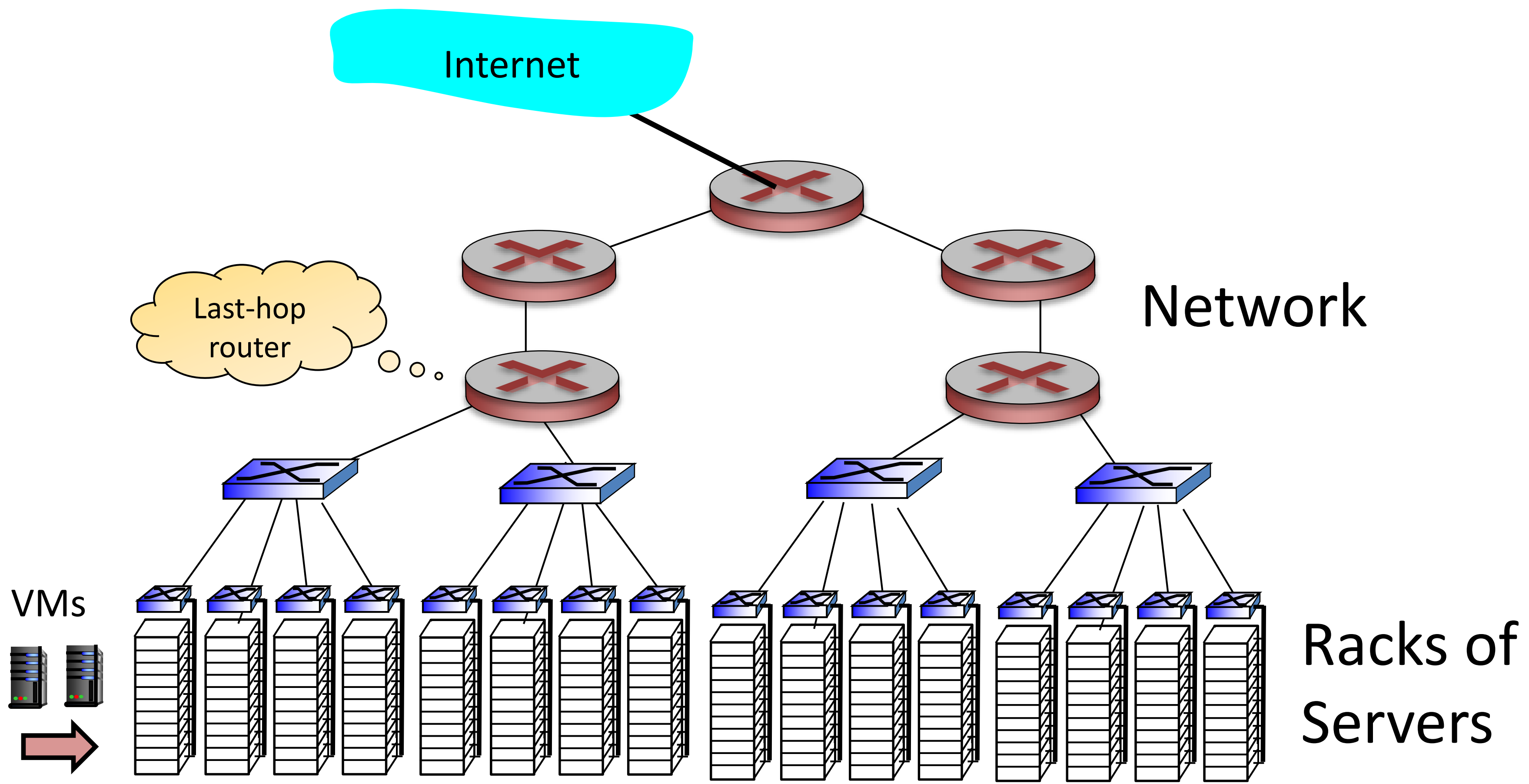


Datacenter Network Design: Proposal #1

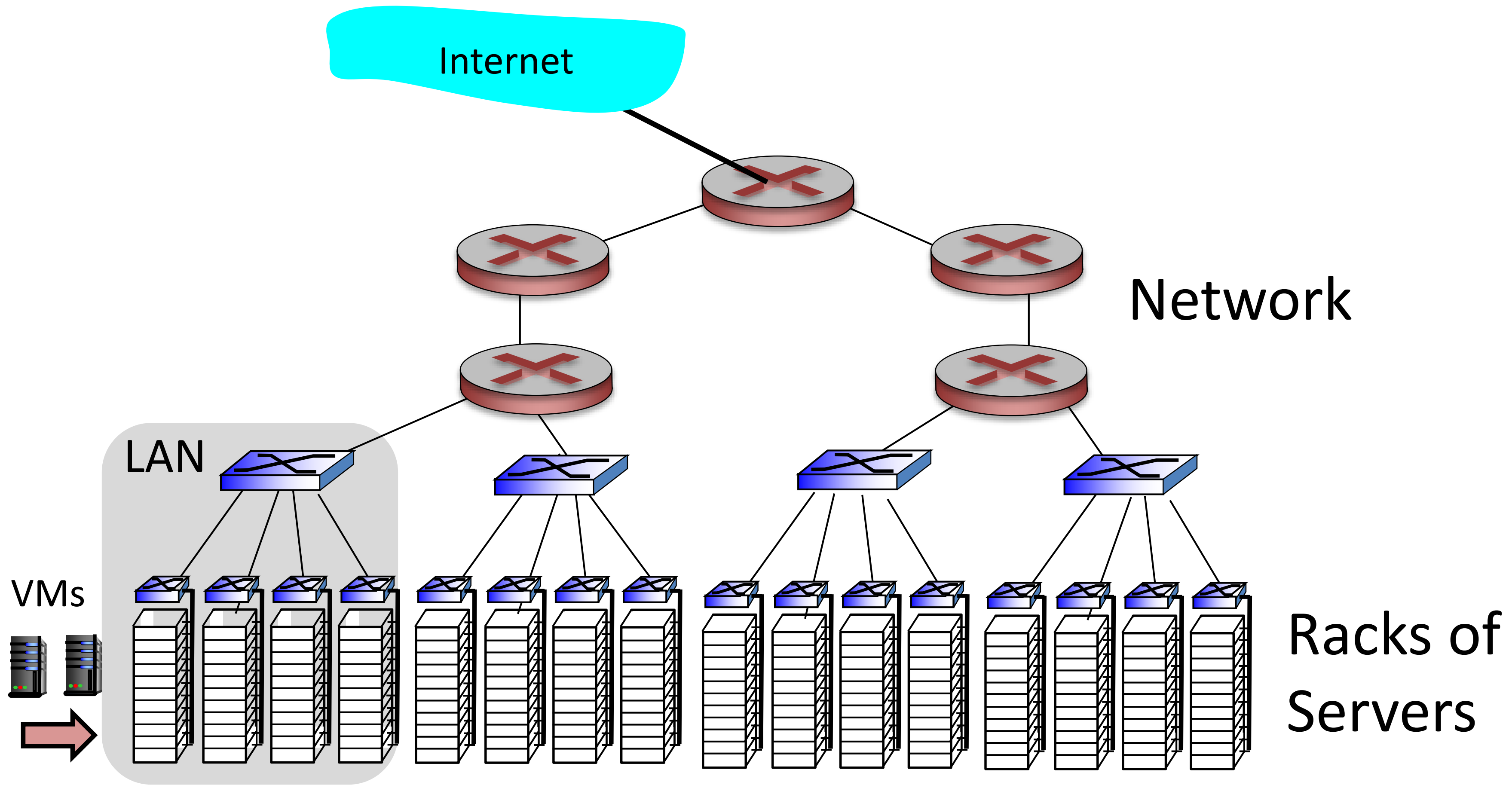
Flexible VM migration… but large broadcast domain: a tradeoff!



Datacenter Network Design: Proposal #2

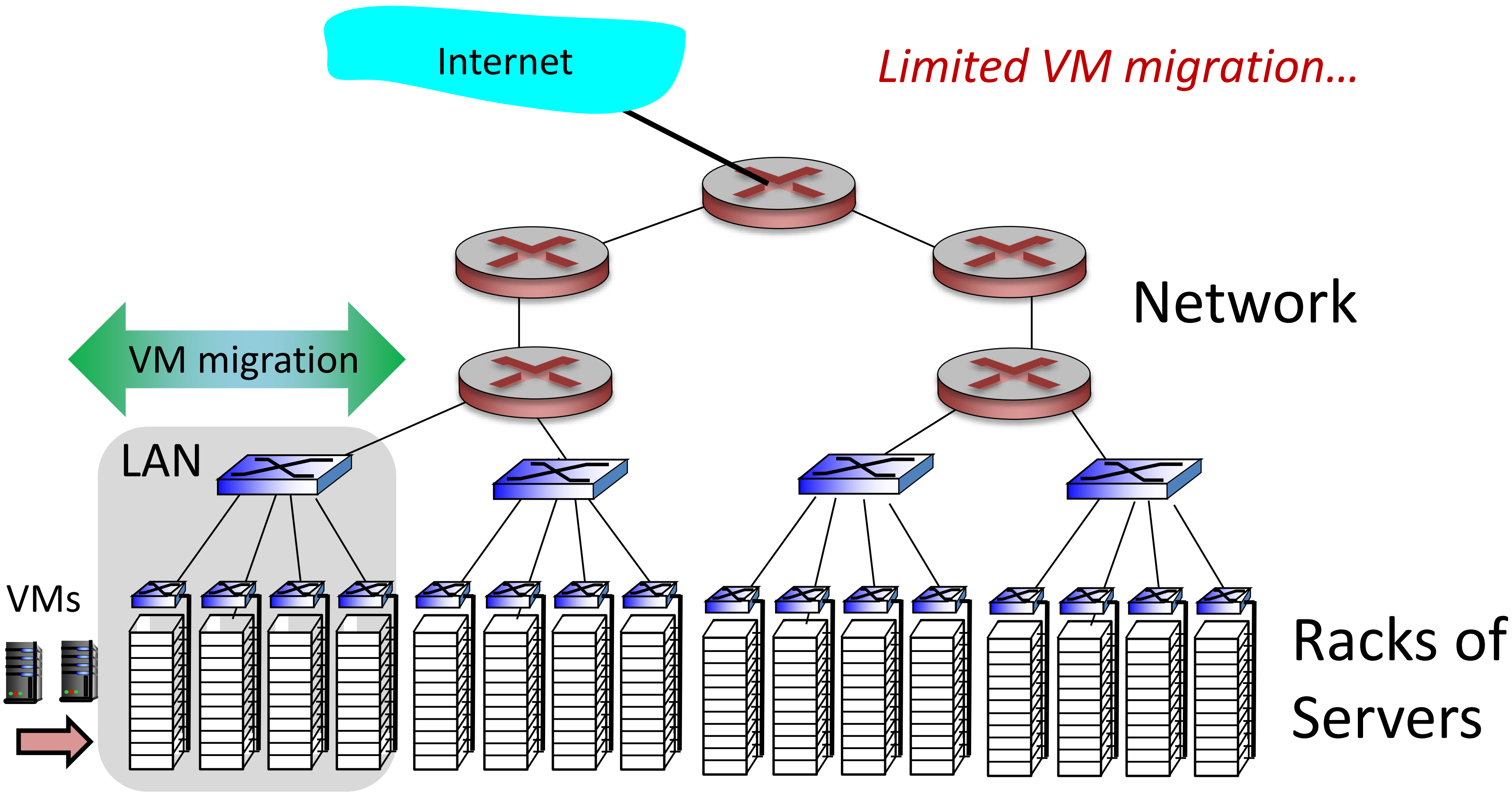


Datacenter Network Design: Proposal #2



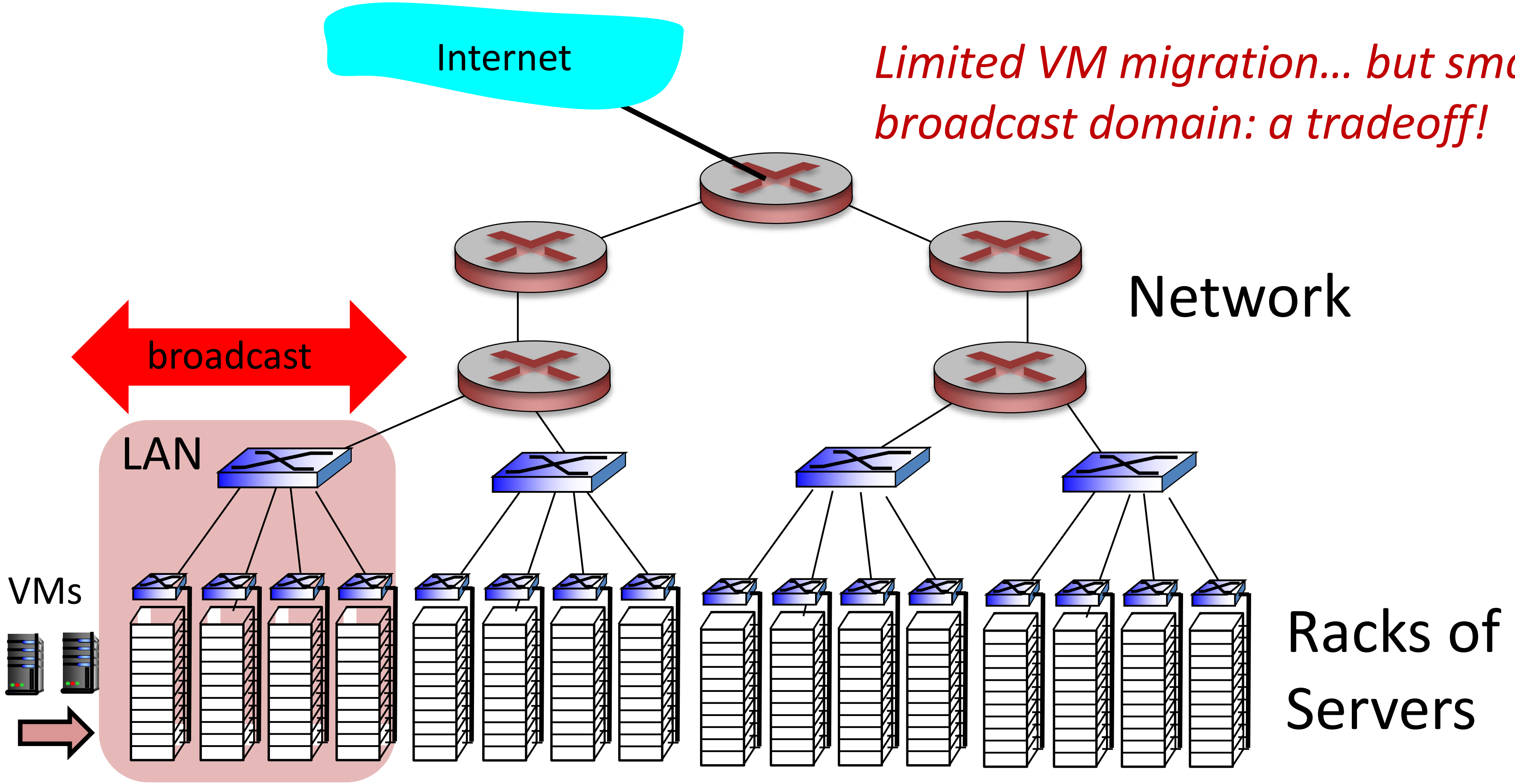
Datacenter Network Design: Proposal #2

Limited VM migration…



Datacenter Network Design: Proposal #2

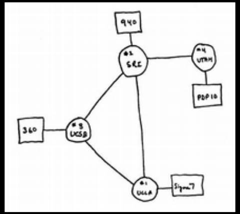
Limited VM migration… but small broadcast domain: a tradeoff!



Motivation: Why networks still require research and innovation

Slide credit:
Pedro Casas

The Internet 50 Years Ago



- Connectivity between fixed locations / "super computers"
- For researchers : Simple applications like email and file transfer

Kudos to: Pedro Casas

The Internet Is A Huge Success Story

Today:

- Supports connectivity between **diverse "users"** : humans, machines, datacenters, or even **things**
- Also supports wireless and **mobile** endpoints
- **Heterogeneous** applications: e-commerce, Internet telephony, VoD, gaming, etc.
- "One of the complex artefacts created by mankind" (Christos H. Papadimitriou)

Yet:

- ***Technology hardly changed! But now: mission-critical infrastructure***

But how secure are our networks?

The Internet at first sight:

- Monumental
- Passed the “Test-of-Time”
- Should not and cannot be changed

But how secure are our networks?

The Internet at first sight:

- Monumental
- Passed the “Test-of-Time”
- Should not and cannot be changed

The Internet at second sight:

- Antique
- Brittle
- More and more successful attacks

Slide credit: Adrian Perrig

A 1st Issue with Today's Networks: Trust Assumptions

- Internet in 80s: based on **trust**
- Danny Hillis, TED talk, Feb. 2013, "There were two Dannys. *I knew both.* Not everyone knew everyone, but there was an atmosphere of trust."

Slide credit: Adrian Perrig

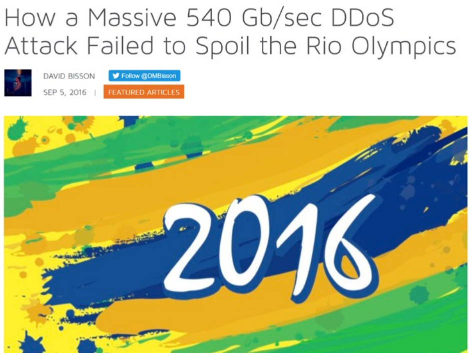
More exploits in the news…

Vulnerabilities in **VPNs**

Vulnerabilities in **IoT**

DDoS attacks often in the news

(e.g. "babyphone attack", **Olympics**)



A 2nd Issue with Today's Networks: Complexity

Many outages due to **misconfigurations** and **human errors**.

Entire countries disconnected…

Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

By Richard Chirgwin 27 Aug 2017 at 22:35 40 SHARE ▼

Last Friday, someone in Google fat-thumbed a border gateway protocol (BGP) advertisement and sent Japanese Internet traffic into a black hole.

The trouble began when The Chocolate Factory "leaked" a big route table to Verizon, the result of which was traffic from Japanese giants like NTT and KDDI was sent to Google on the expectation it would be treated as transit.

… 1000s passengers stranded…

British Airways' latest Total Inability To Support Upwardness of Planes* caused by Amadeus system outage

Stuck on the ground awaiting a load sheet? Here's why

By Gareth Corfield 19 Jul 2018 at 11:16 109 SHARE ▼

… even 911 services affected!

Officials: Human error to blame in Minn. 911 outage

According to a press release, CenturyLink told department of public safety that human error by an employee of a third party vendor was to blame for the outage

Aug 16, 2018

Duluth News Tribune

SAINT PAUL, Minn. — The Minnesota Department of Public Safety Emergency Communication Networks division was told by its 911 provider that an Aug. 1 outage was caused by human error.

Even Tech-Savvy Companies Struggle to Provide Reliable Networks

We discovered a misconfiguration on this pair of switches that caused what's called a "bridge loop" in the network.

A network change was […] executed incorrectly […] more "stuck" volumes and added more requests to the re-mirroring storm

Service outage was due to a series of internal network events that corrupted router data tables

Experienced a network connectivity issue […] interrupted the airline's flight departures, airport processing and reservations systems

A 3rd Issue: *Lack of Tools*
Anecdote "Wall Street Bank"

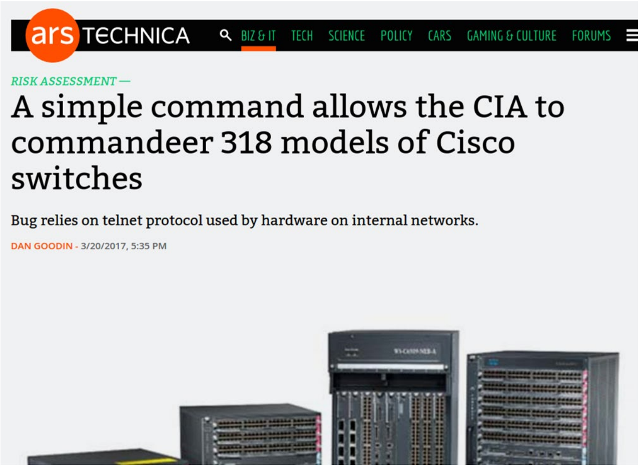
- Outage of a data center of a Wall Street investment bank
- Lost revenue measured in USD 10^6 / min
- Quickly, an emergency team was assembled with experts in compute, storage and networking:
  - **The compute team:** soon came armed with **reams of logs**, showing how and when the applications failed, and had already written experiments to reproduce and **isolate the error**, along with candidate prototype programs to workaround the failure.
  - **The storage team:** similarly equipped, showing which file **system logs** were affected, and already progressing with **workaround programs**.
  - "All the **networking team** had were **two tools invented over 20y ago** to merely test end-to-end connectivity. Neither tool could reveal **problems with switches**, the **congestion** experienced by individual packets, or provide any means to create experiments to identify, quarantine and resolve the problem. Whether or not the problem was in the network, the **networking team would be blamed** since they were unable to demonstrate otherwise."

Source: «The world's fastest and most programmable networks»
White Paper Barefoot Networks

Also: How much can we trust *technology*?

(TS//SI//NF) Such operations involving **supply-chain interdiction** are some of the most productive operations in TAO, because they pre-position access points into hard target networks around the world.

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon



- **Hardware backdoors** and exploits
- The problem seems fundamental: how can we *hope to build a secure network* if the underlying hardware can be insecure?!
- E.g., *secure cloud for the government*: no resources and expertise to build own "trustworthy" high-speed hardware

Takeaway

Complexity and human errors: networks should be operated in a *less manual* but more **automated** way. Hence: need to rely on **formal specifications**.

Another Takeaway

Our digital society relies on *all sorts of networks*, e.g., increasingly on the networks to, from, and in **datacenters**, but also more “exotic” networks such as **in-cabin** and car **networks**, **cryptocurrency** networks, etc.

Source: Facebook

Roadmap

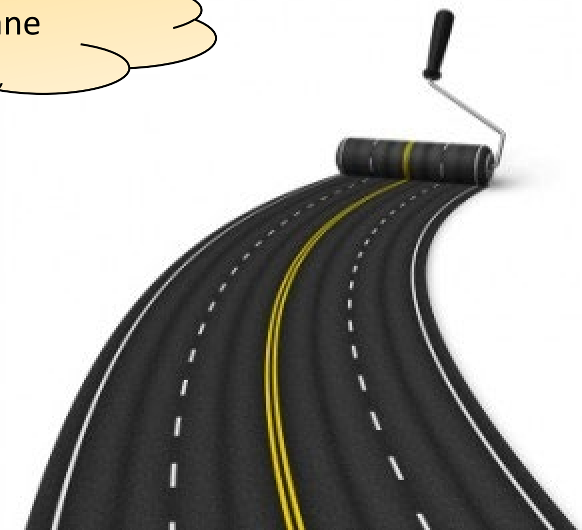
- Software-defined networks
- Programmable dataplanes
- Network virtualization

Roadmap

- Software-defined networks
- Programmable dataplanes
- Network virtualization

Making the control plane programmable

Making the data plane programmable



Control Plane vs Data Plane

Recall: two network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output

data plane

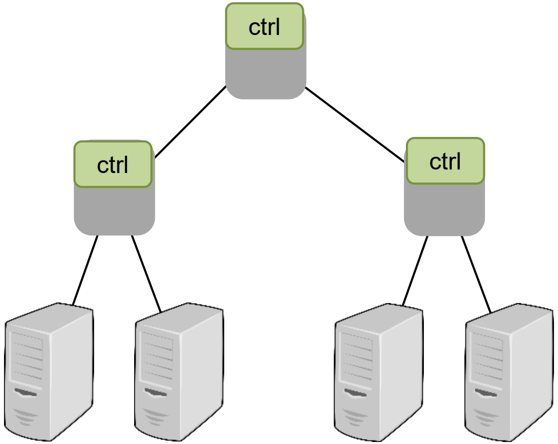
- *routing:* determine route taken by packets from source to destination

control plane

Roadmap

- **Software-defined networks**
- Programmable dataplanes
- Network virtualization

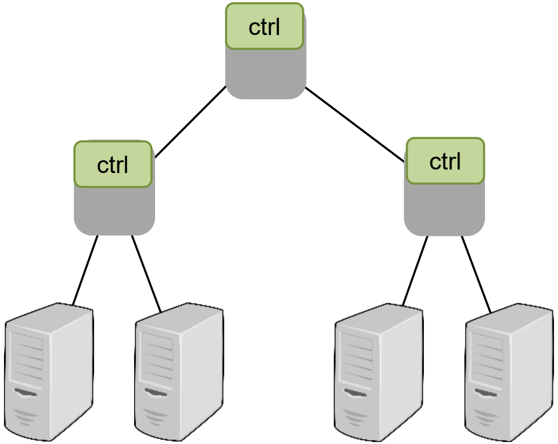
Control Plane

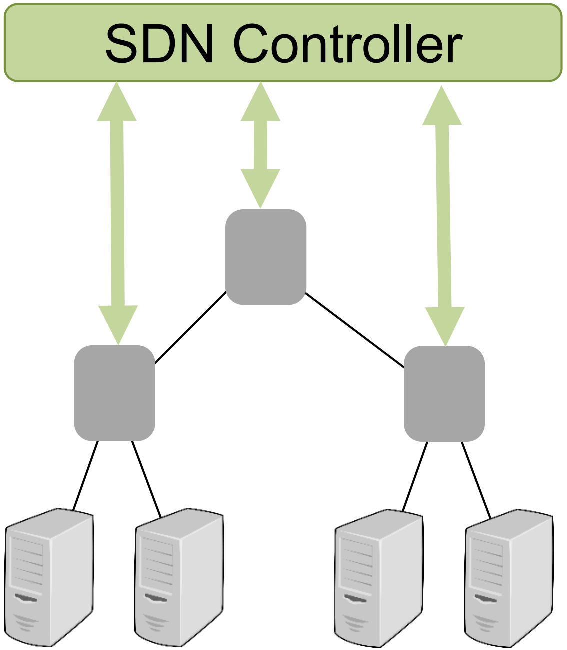


Traditionally:

- Distributed control plane
- Blackbox, not programmable

Control Plane





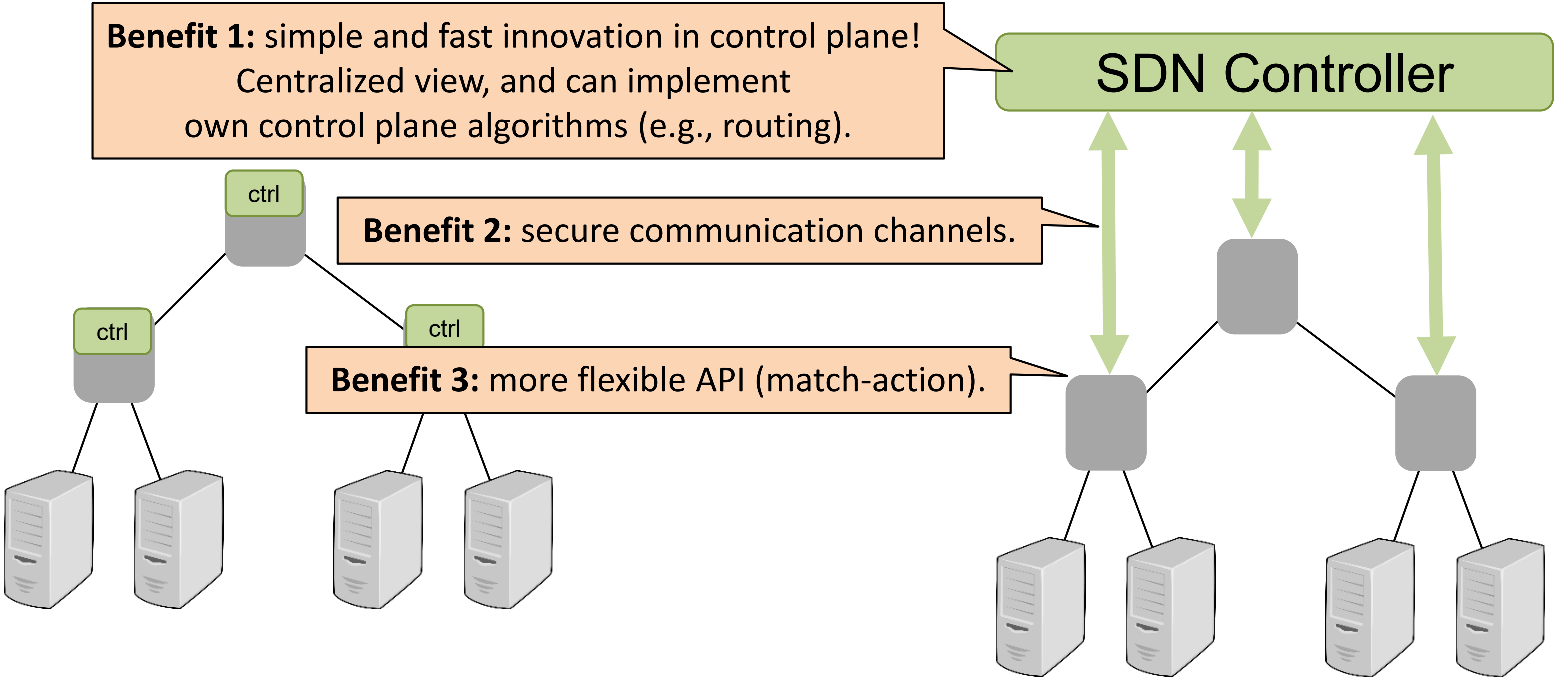
Traditionally:

- Distributed control plane
- Blackbox, not programmable

Software-defined Networs (SDN):

- Logically centralized control
- Programmable, match-action

Control Plane



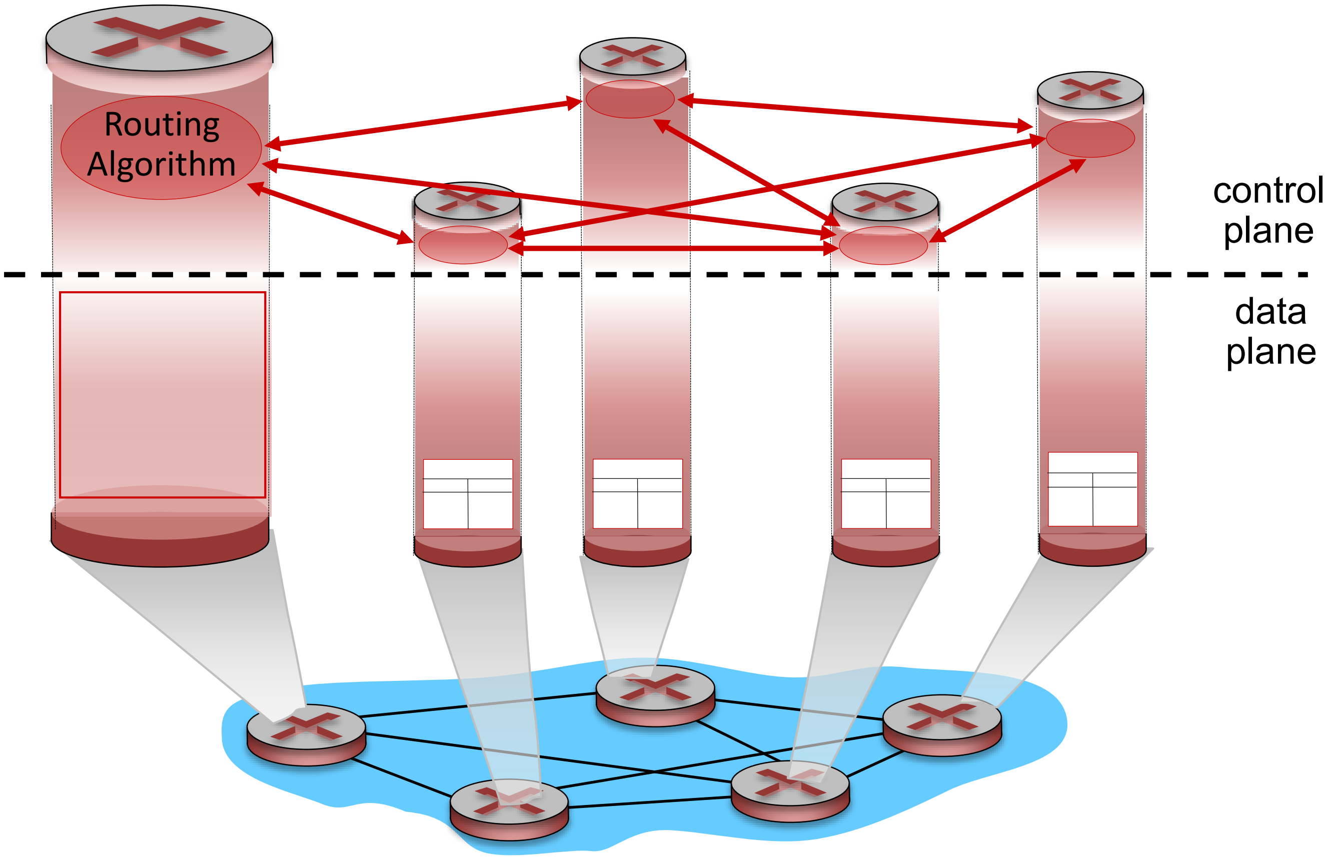
Traditionally:

- Distributed control plane
- Blackbox, not programmable

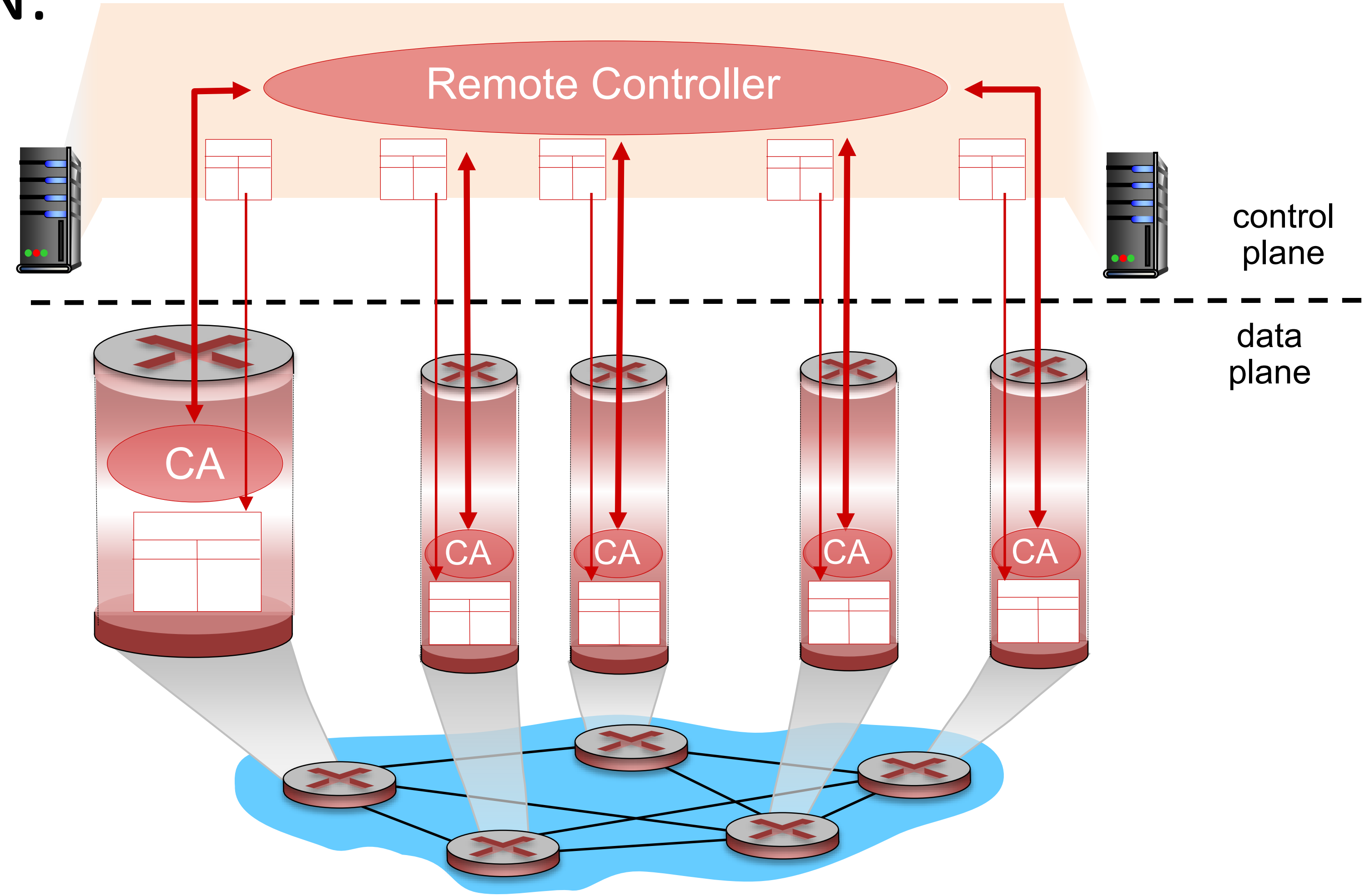
Software-defined Networs (SDN):

- Logically centralized control
- Programmable, match-action

In more details: Traditionally…



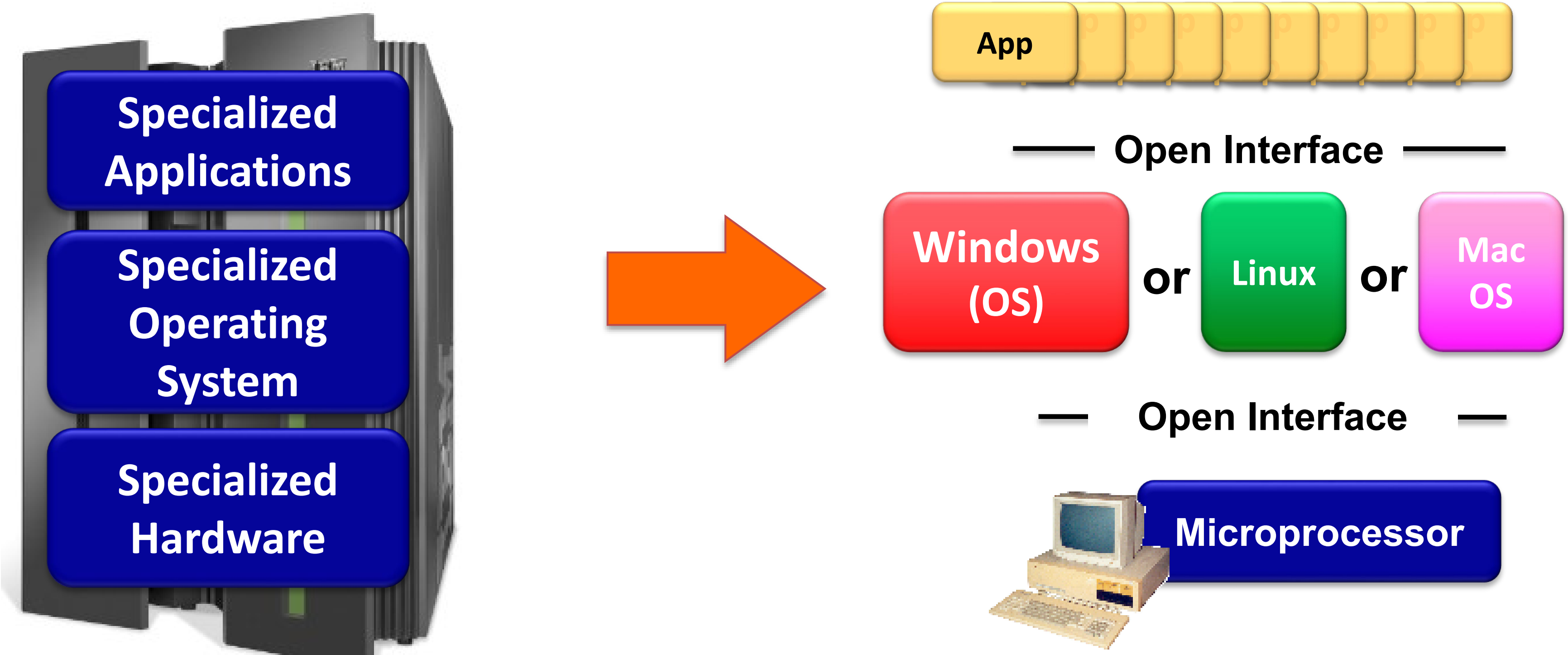
… and SDN:



Why logically centralized control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (recall OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming: more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- open (non-proprietary) implementation of control plane

Analogy: Mainframe to PC Evolution

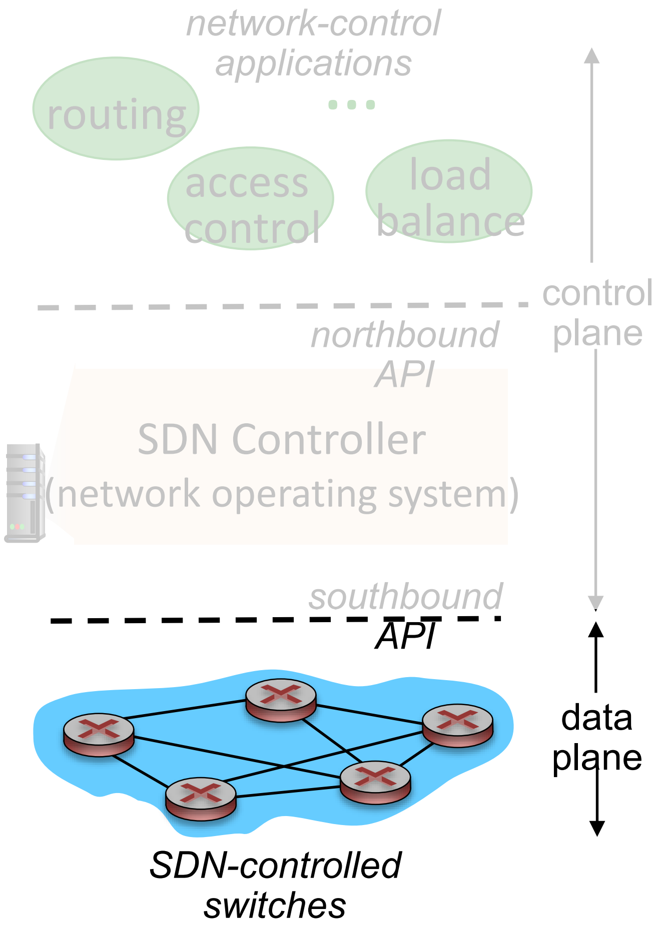


Horizontal Open interfaces
Rapid innovation Huge industry

The SDN Perspective

Data plane switches

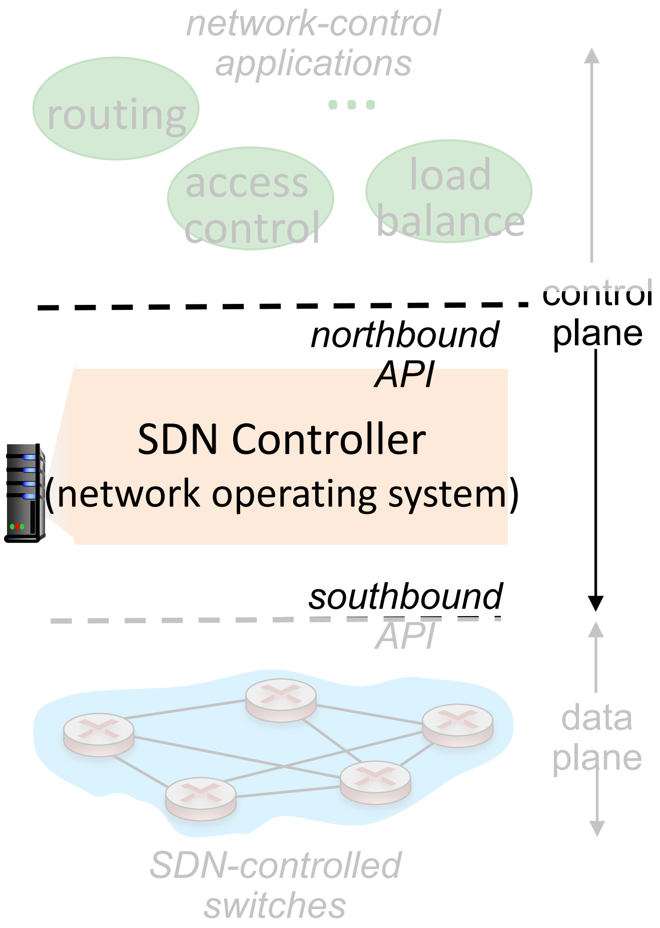
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable and what is not
- protocol for communicating with controller (e.g., OpenFlow)



The SDN Perspective

SDN controller (network OS):

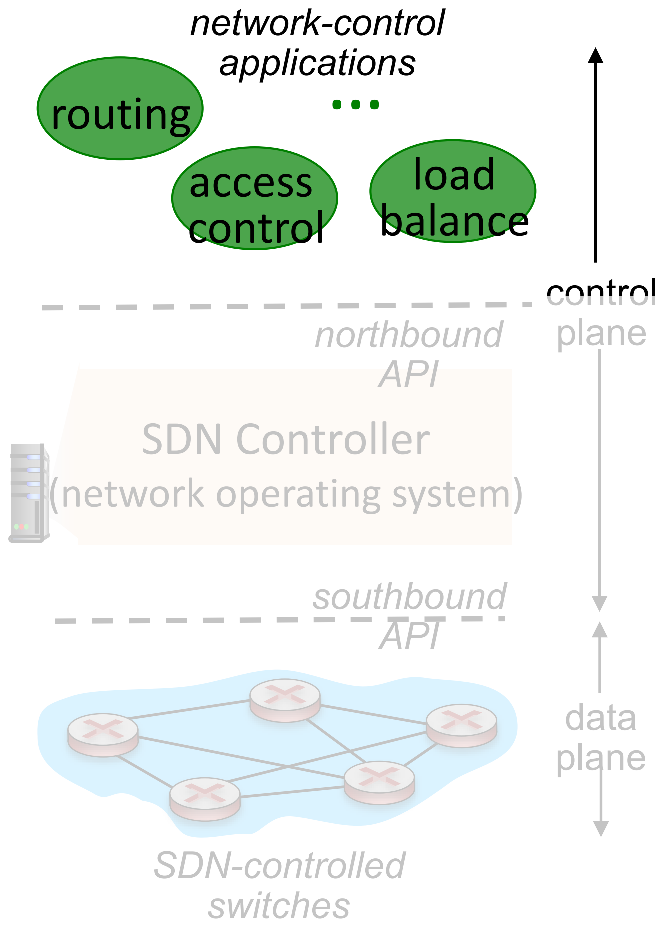
- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



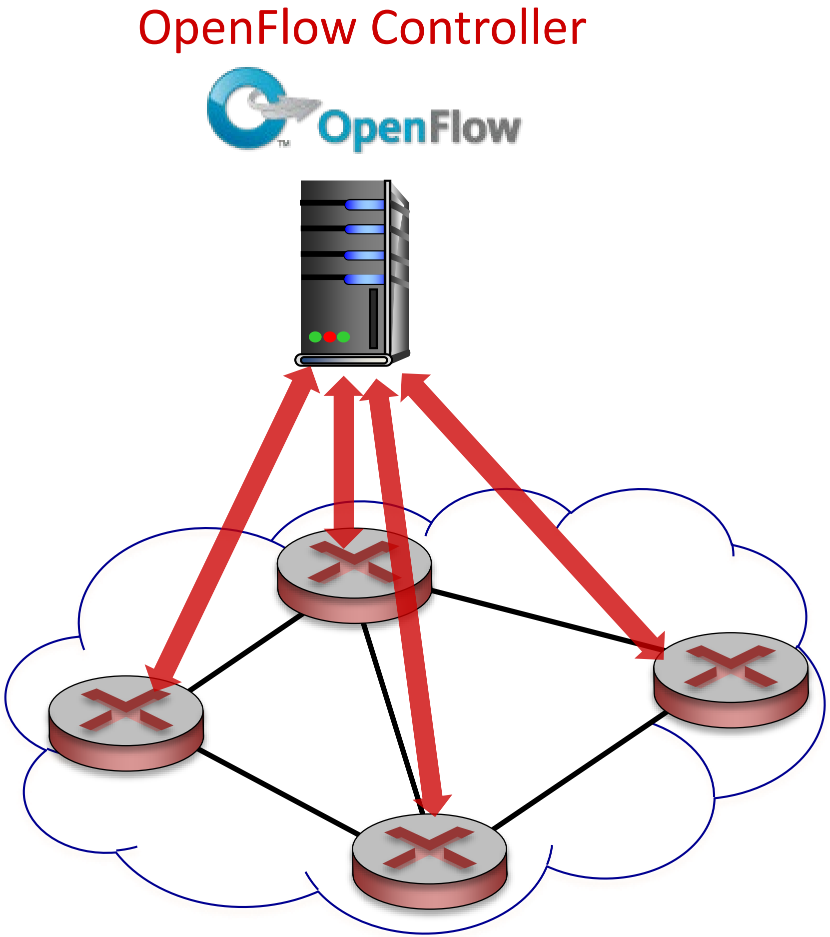
The SDN Perspective

network-control apps:

- "brains" of control: implement control functions using lower-level services, API provided by SND controller
- *unbundled:* can be provided by 3rd party: distinct from routing vendor, or SDN controller



The OpenFlow Protocol

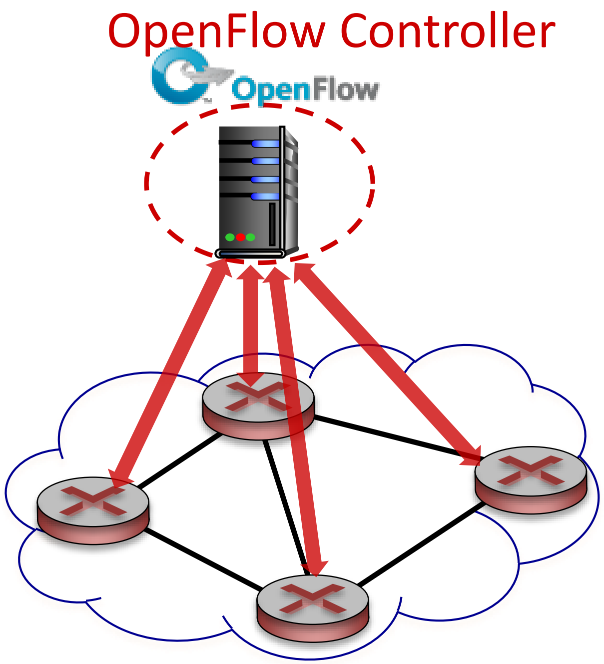


- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

Controller-to-Switch Messages

Key controller-to-switch messages

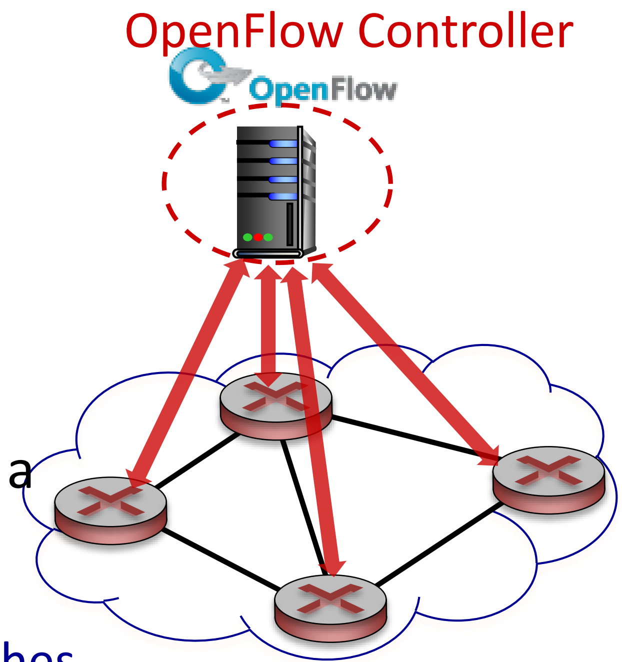
- *features:* controller queries switch features, switch replies
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *packet-out:* controller can send this packet out of specific switch port



Switch-to-Controller Messages

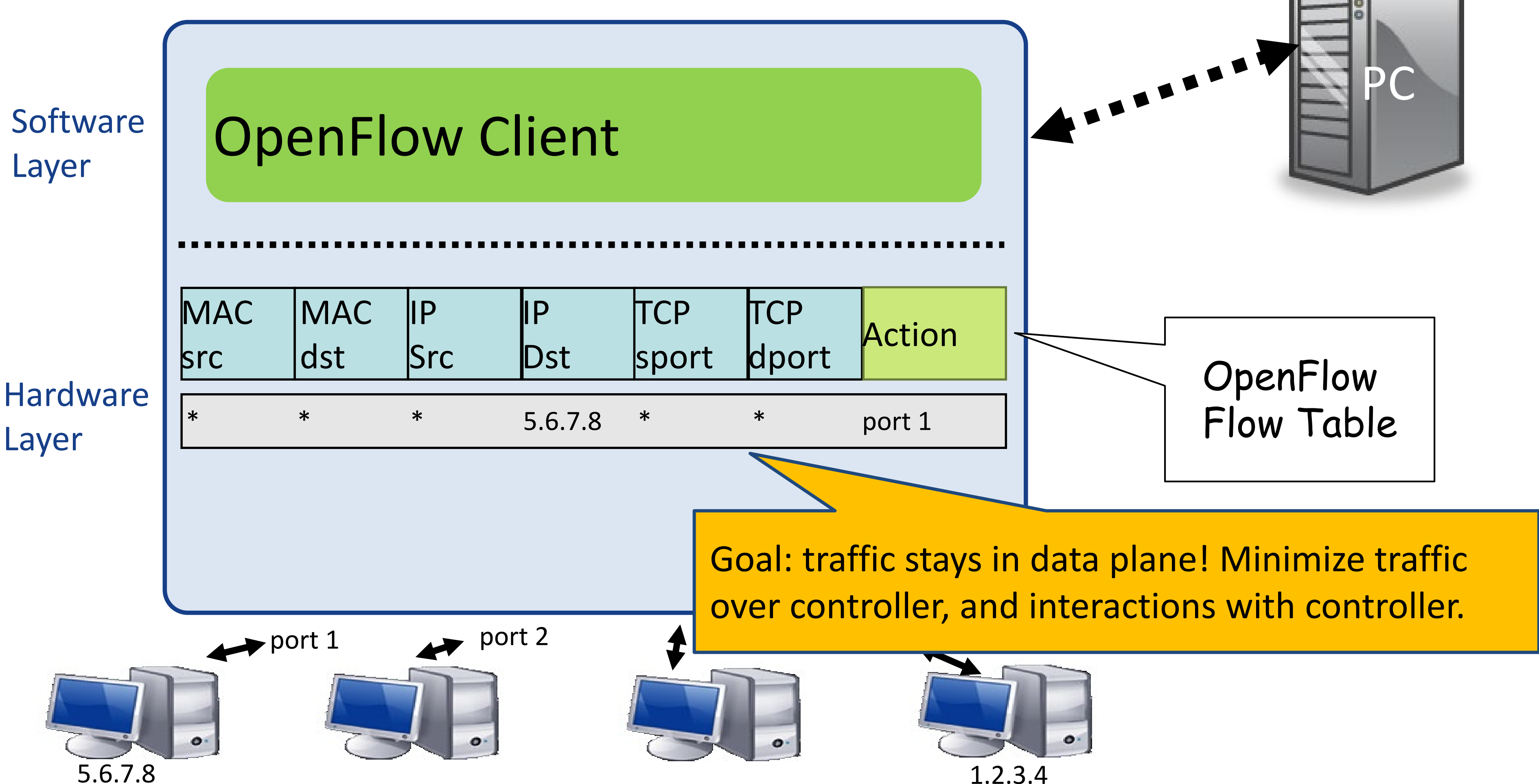
Key switch-to-controller messages

- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* flow table entry deleted at switch
- *port status:* inform controller of a change on a port.



Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

OpenFlow


Controller
PC
Software Layer
OpenFlow Client
Hardware Layer
MAC src
MAC dst
IP Src
IP Dst
TCP sport
TCP dport
Action
* * * 5.6.7.8 * * port 1
OpenFlow Flow Table
Goal: traffic stays in data plane! Minimize traffic over controller, and interactions with controller.
port 1
port 2
5.6.7.8
1.2.3.4

OpenFlow: Flow Table Entries

Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. ….

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

+ mask

Examples

L2: Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

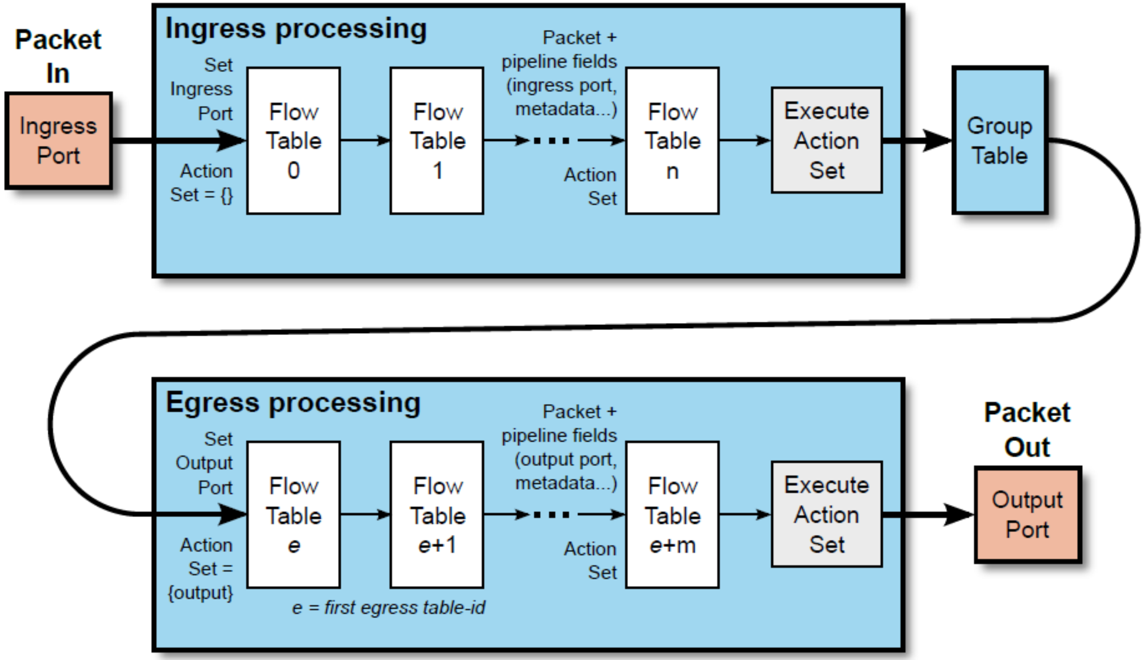
L3: Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

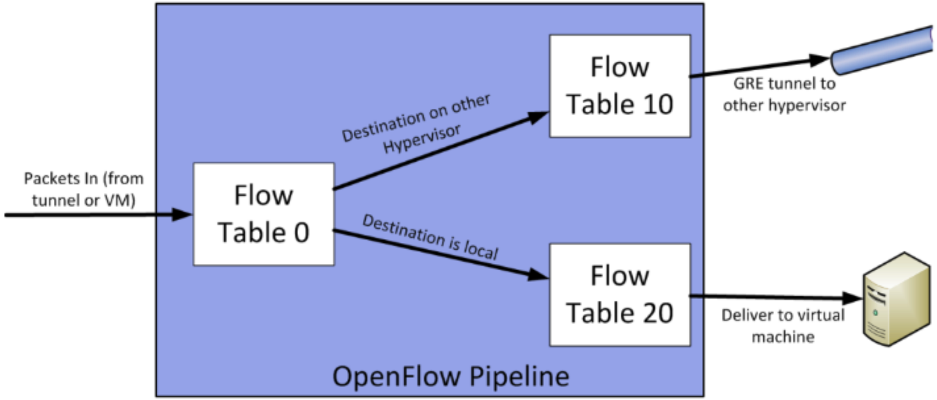
L4: Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

OpenFlow 1.5 Switch Model



Example

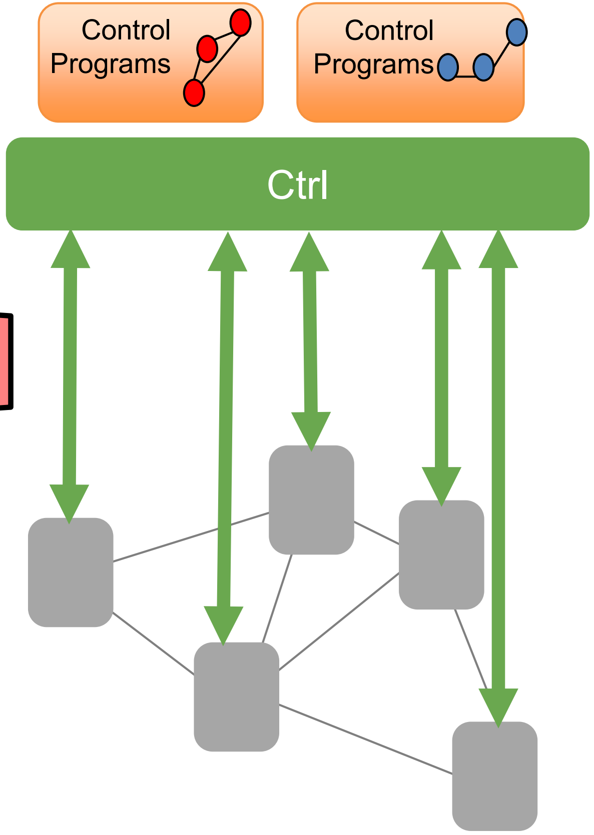


Fedora example: Packets enter the (virtual) switch either from outside the hypervisor (e.g., an overlay tunnel) or from a virtual machine. Either way, all packets first go to table 0: e.g., decides if traffic is destined for another hypervisor (forwards to table 10), or to a virtual machine local to this switch (forwards to table 20). Both table 10 and 20 will apply any actions required (e.g., NAT then send to interface)

https://keepingitclassless.net/2014/07/sdn-protocols-2-openflow-deep-dive/

Example: MAC Learning With SDN

A First (Algorithmic) Challenge: Decoupling



Challenge: centralization and decoupling!

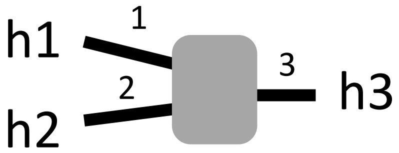
Despite centralization: SDN stays a distributed system!

Recall: Networking 101

- Networking «Hello World»: MAC learning

- Principle: for packet (***src***,***dst***) arriving at port ***p***
  - If ***dst*** unknown: broadcast packets to all ports
    - Otherwise forward directly to known port
  - Also: if ***src*** unknown, switch learns: ***src*** is behind ***p***

- Example



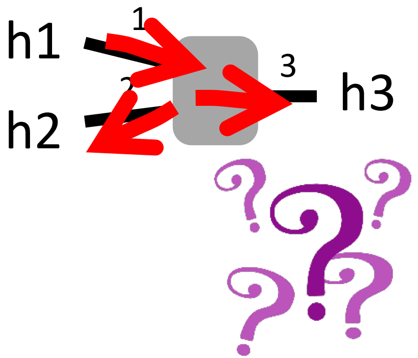
Credits: Jennifer Rexford

Recall: Networking 101

- Networking «Hello World»: MAC learning

- Principle: for packet (***src***,***dst***) arriving at port ***p***
  - If ***dst*** unknown: broadcast packets to all ports
    - Otherwise forward directly to known port
  - Also: if ***src*** unknown, switch learns: ***src*** is behind ***p***

- Example
  - h1 sends to h2: **flood, learn (h1,p1)**

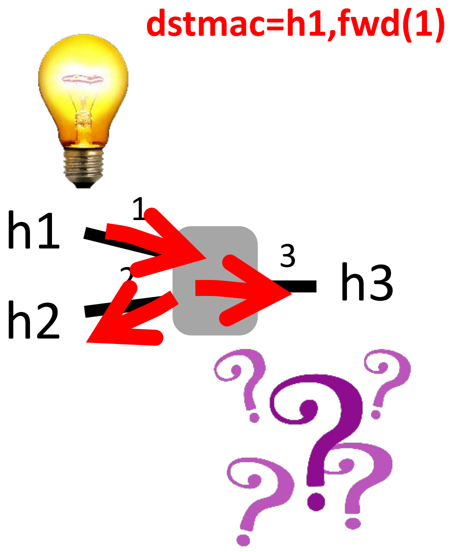


Recall: Networking 101

- Networking «Hello World»: MAC learning

- Principle: for packet (***src***,***dst***) arriving at port ***p***
  - If ***dst*** unknown: broadcast packets to all ports
    - Otherwise forward directly to known port
  - Also: if ***src*** unknown, switch learns: ***src*** is behind ***p***

- Example
  - h1 sends to h2: **flood, learn (h1,p1)**

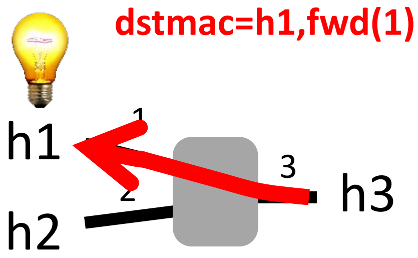


Recall: Networking 101

- Networking «Hello World»: MAC learning

- Principle: for packet (***src***,***dst***) arriving at port ***p***
  - If ***dst*** unknown: broadcast packets to all ports
    - Otherwise forward directly to known port
  - Also: if ***src*** unknown, switch learns: ***src*** is behind ***p***

- Example
  - h1 sends to h2: **flood, learn (h1,p1)**
  - h3 sends to h1: **forward to p1, learn (h3,p3)**

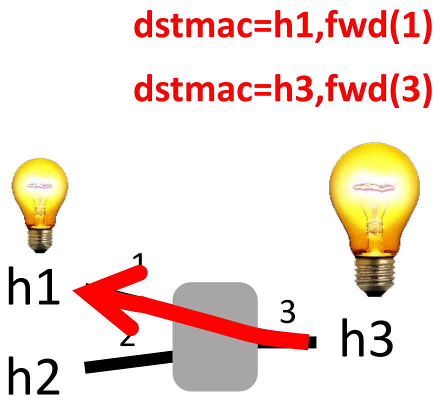


Recall: Networking 101

- Networking «Hello World»: MAC learning

- Principle: for packet (***src***,***dst***) arriving at port ***p***
  - If ***dst*** unknown: broadcast packets to all ports
    - Otherwise forward directly to known port
  - Also: if ***src*** unknown, switch learns: ***src*** is behind ***p***

- Example
  - h1 sends to h2: **flood, learn (h1,p1)**
  - h3 sends to h1: **forward to p1, learn (h3,p3)**



Recall: Networking 101

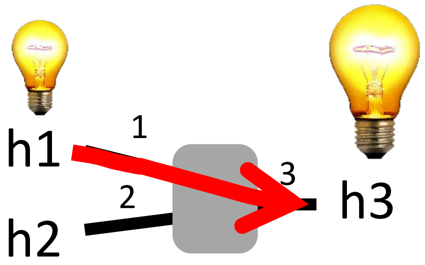
- Networking «Hello World»: MAC learning

- Principle: for packet (***src***,***dst***) arriving at port ***p***
  - If ***dst*** unknown: broadcast packets to all ports
    - Otherwise forward directly to known port
  - Also: if ***src*** unknown, switch learns: ***src*** is behind ***p***

- Example
  - h1 sends to h2: **flood, learn (h1,p1)**
  - h3 sends to h1: **forward to p1, learn (h3,p3)**
  - h1 sends to h3: **forward to p3**

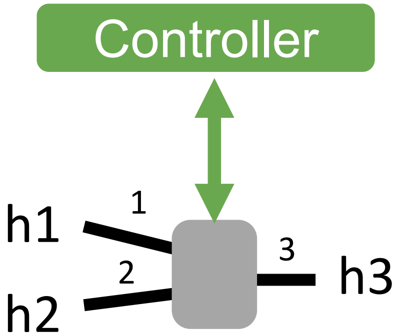
dstmac=h1,fwd(1)

dstmac=h3,fwd(3)

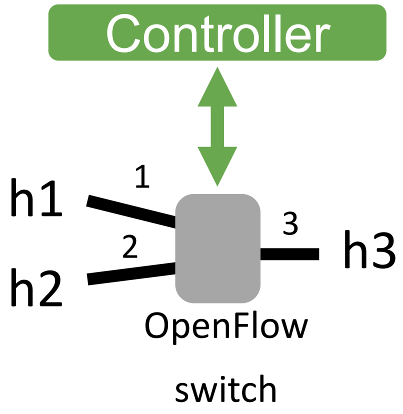


From Traditional Networks to SDN

How to implement this behavior in SDN?



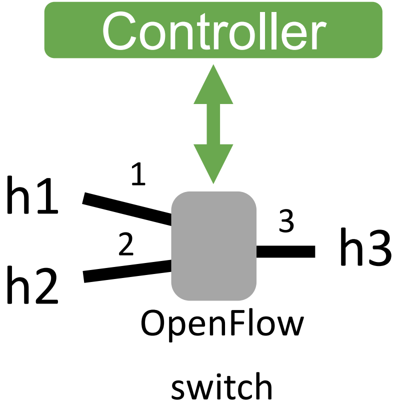
Example: SDN MAC Learning Done Wrong



- Initial table: Send everything to controller

Pattern	Action
*	send to controller

Example: SDN MAC Learning Done Wrong

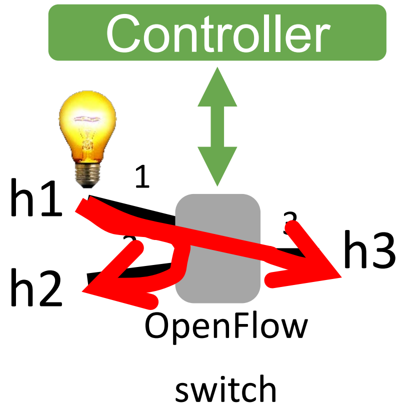


- Initial table: Send everything to controller

Pattern	Action
*	send to controller

- When h1 sends to h2:

Example: SDN MAC Learning Done Wrong



- Principle: only send to ctrl if destination unknown

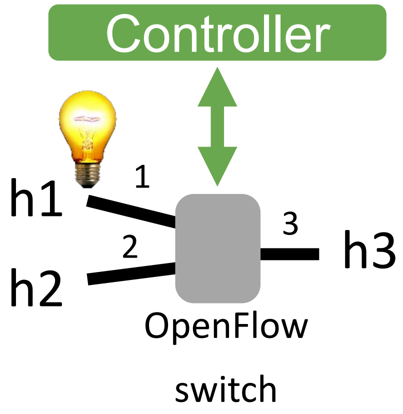
Pattern	Action
*	send to controller

h1 sends to h2

Pattern	Action
dstmac=h1	Forward(1)
*	send to controller

- When h1 sends to h2:
  - Controller learns that h1@p1, updates table, and floods

Example: SDN MAC Learning Done Wrong

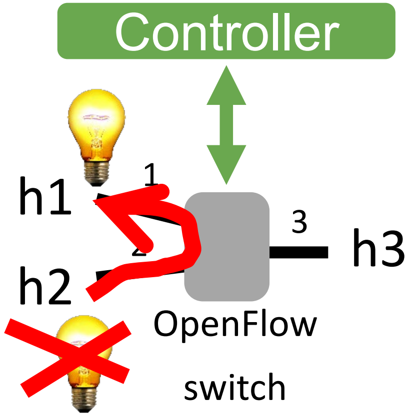


- Principle: only send to ctrl if destination unknown

Pattern	Action
dstmac=h1	Forward(1)
*	send to controller

- Now assume h2 sends to h1:

Example: SDN MAC Learning Done Wrong

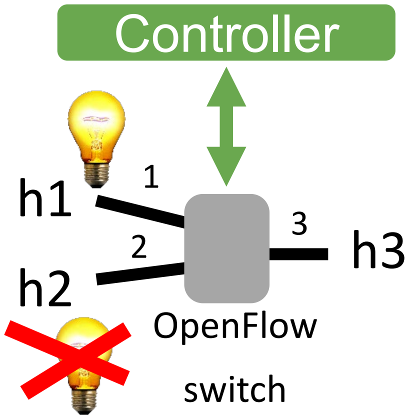


- Principle: only send to ctrl if destination unknown

Pattern	Action
dstmac=h1	Forward(1)
*	send to controller

- Now assume h2 sends to h1:
  - *Switch* knows destination: message forwarded to h1
  - ***BUT***: No controller interaction, does not learn about h2: no new rule for h2

Example: SDN MAC Learning Done Wrong



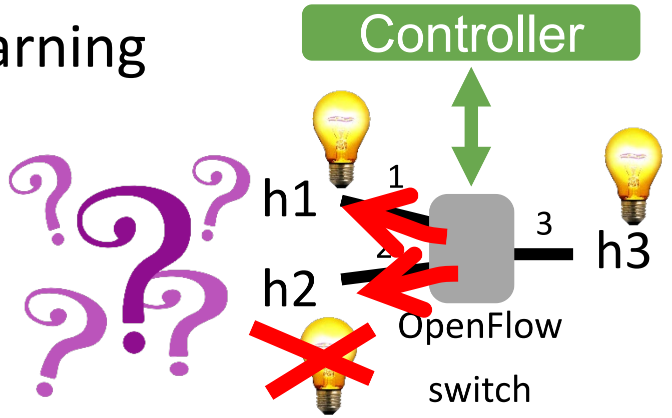
- Principle: only send to ctrl if destination unknown

Pattern	Action
dstmac=h1	Forward(1)
*	send to controller

h3 sends to h2

- Now, when h3 sends to h2:

Example: SDN MAC Learning Done Wrong



- Principle: only send to ctrl if destination unknown

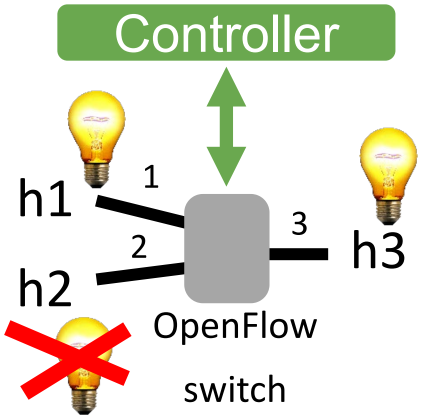
Pattern	Action
dstmac=h1	Forward(1)
*	send to controller

h3 sends to h2

Pattern	Action
dstmac=h3	Forward(3)
dstmac=h1	Forward(1)
*	send to controller

- Now, when h3 sends to h2:
  - Dest unknown: goes to controller which learns about h3
  - And then floods

Example: SDN MAC Learning Done Wrong

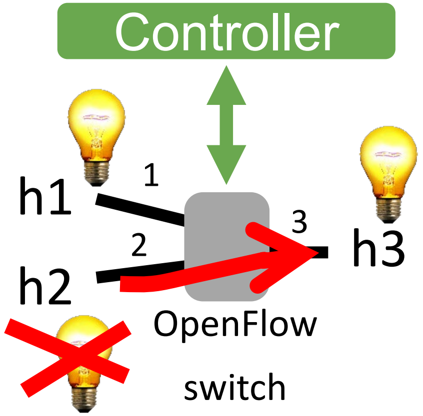


- Principle: only send to ctrl if destination unknown

Pattern	Action
dstmac=h3	Forward(3)
dstmac=h1	Forward(1)
*	send to controller

- Now, if h2 sends to h3 or h1:

Example: SDN MAC Learning Done Wrong

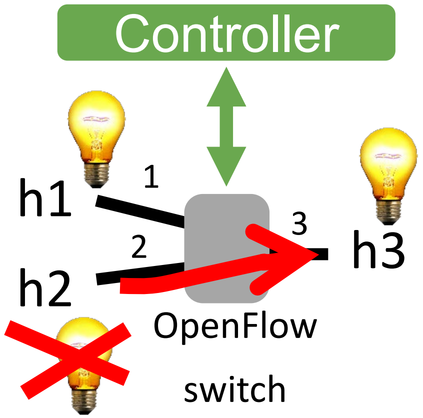


- Principle: only send to ctrl if destination unknown

Pattern	Action
dstmac=h3	Forward(3)
dstmac=h1	Forward(1)
*	send to controller

- Now, if h2 sends to h3 or h1:
  - Destinations known: controller does not learn about h2

Example: SDN MAC Learning Done Wrong

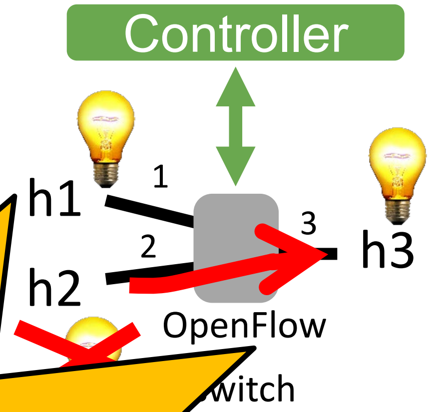


- Principle: only send to ctrl if destination unknown

Pattern	Action
dstmac=h3	Forward(3)
dstmac=h1	Forward(1)
*	send to controller

Ouch! Controller cannot learn about h2 anymore: whenever h2 is source, destination is known. All future requests to h2 will ***all be flooded***: inefficient!

Example: SDN MAC Learning Done Wrong



- Principle: only send to ctrl if destination unknown

How to efficiently detect such problems? And which rules to use to overcome them? An algorithmic problem!

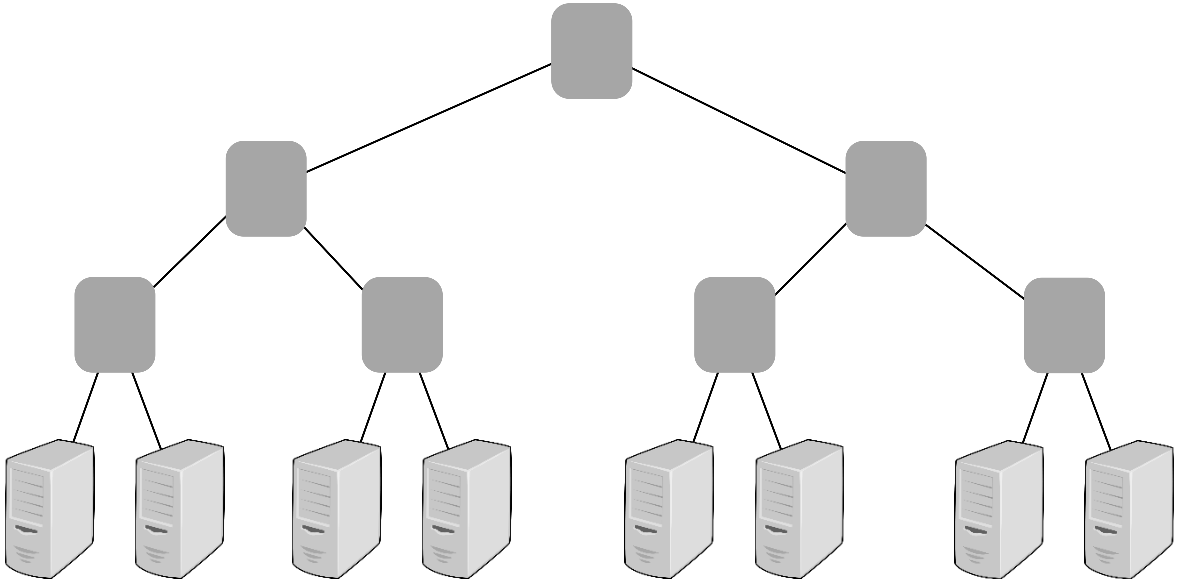
Ouch! Controller cannot learn about h2 anymore: whenever h2 is source, destination is known. All future requests to h2 will ***all be flooded***: inefficient!

Example Application for SDN: Detecting Misbehavior

Allows to Deal with New Threat Vectors: Secure Trajectory Sampling

Monitor packets, traditionally:
trajectory sampling

- *Globally* sample packets with ***hash(imm. header)∈[x,y]***
- See full routes *of some packets*

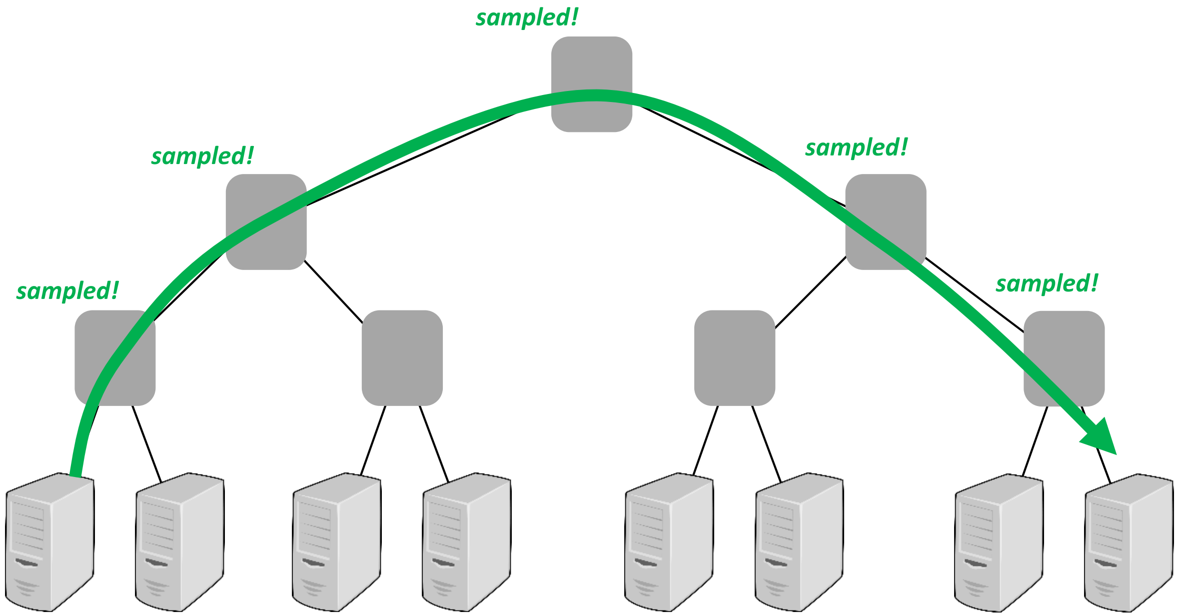


Allows to Deal with New Threat Vectors: Secure Trajectory Sampling

Monitor packets, traditionally:

trajectory sampling

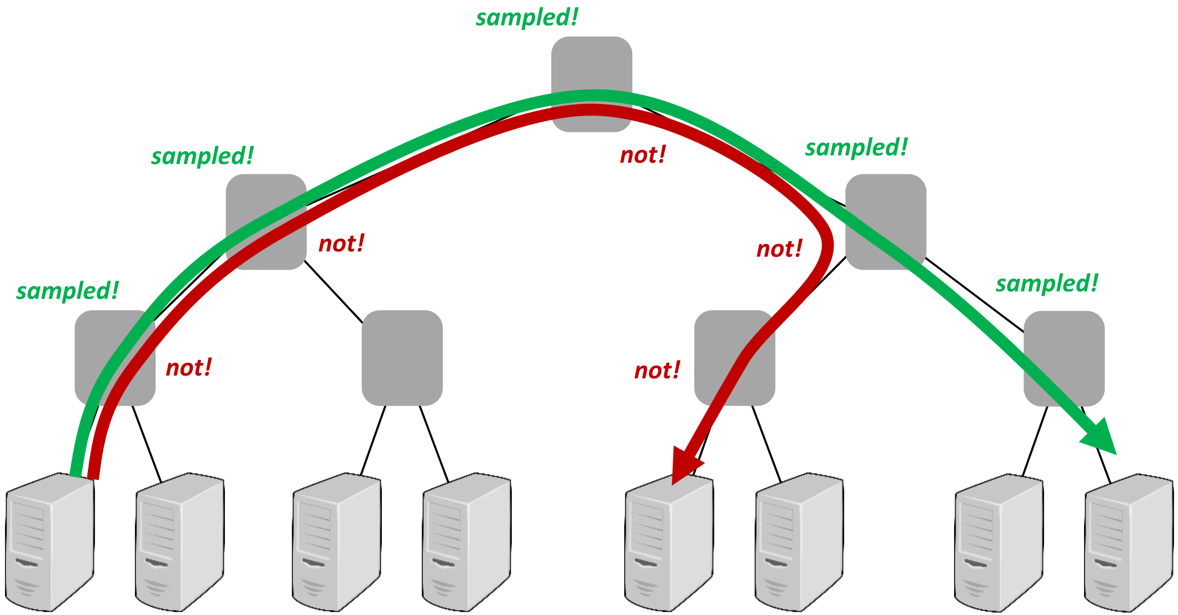
- *Globally* sample packets with ***hash(imm. header)∈[x,y]***
- See full routes *of some packets*



Allows to Deal with New Threat Vectors: Secure Trajectory Sampling

Monitor packets, traditionally:
trajectory sampling

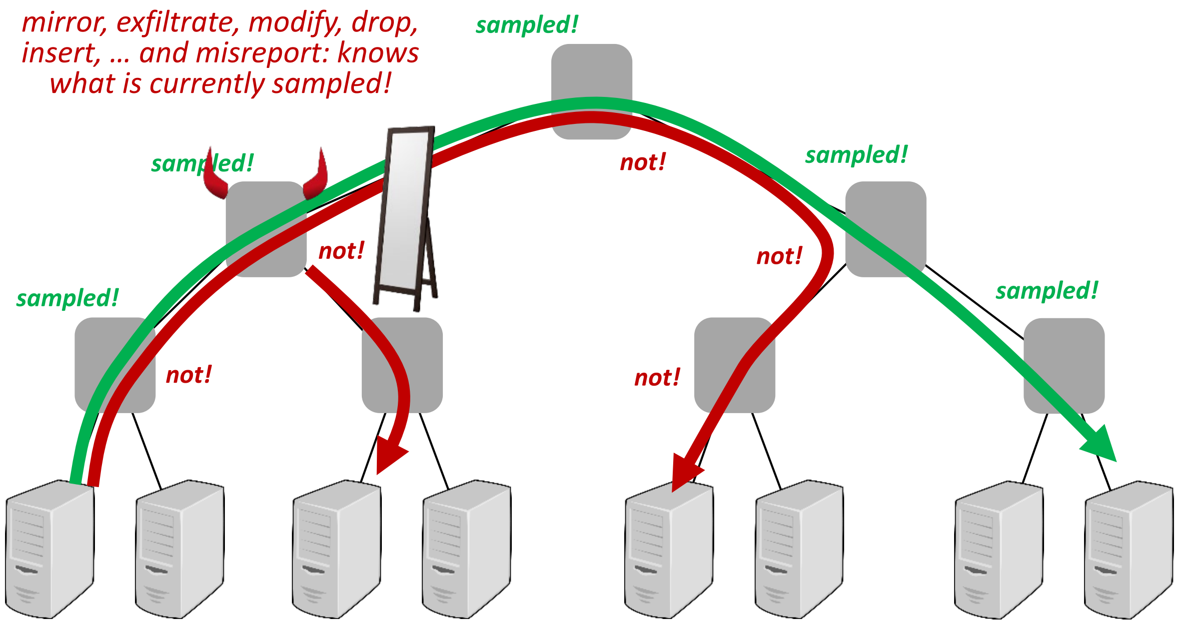
- *Globally* sample packets with **_hash(imm. header)∈[x,y]_**
- See full routes *of some packets*
- But *not others!* (resp. later)



Allows to Deal with New Threat Vectors: Secure Trajectory Sampling

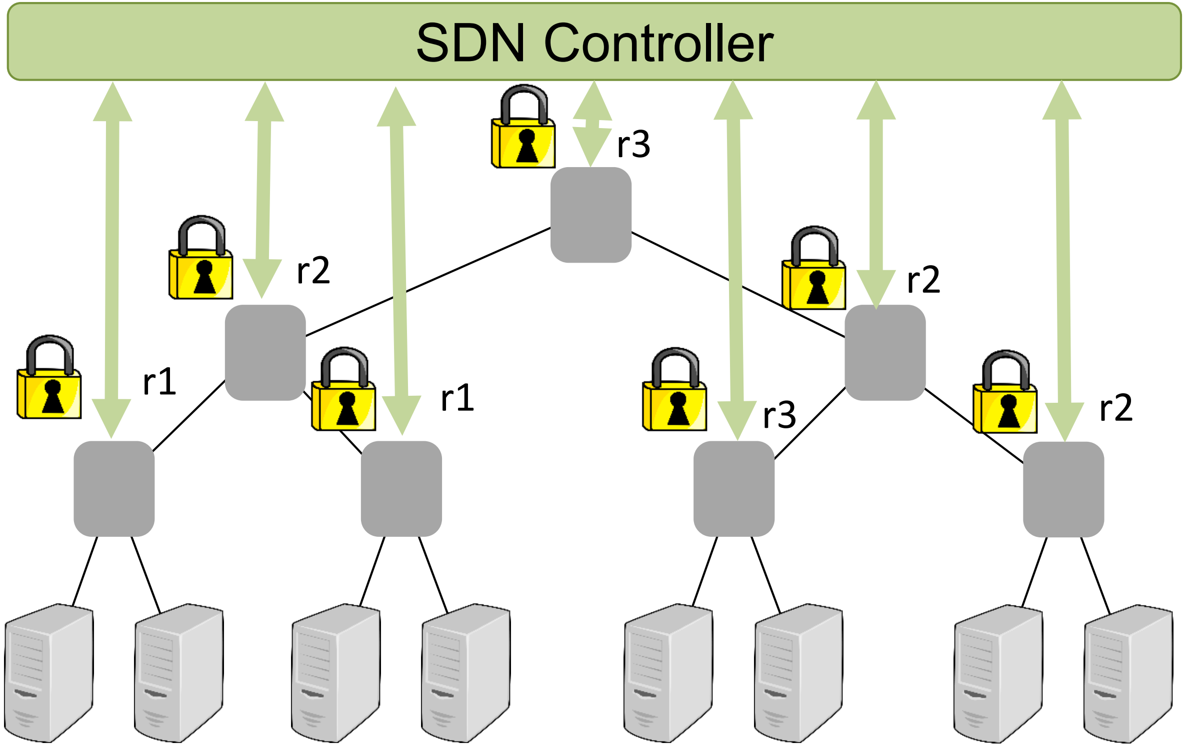
Monitor packets, traditionally:
trajectory sampling

- *Globally* sample packets with ***hash(imm. header)∈[x,y]***
- See full routes *of some packets*
- But *not others!* (resp. later)



Solution: Use SDN for *Secure* Trajectory Sampling

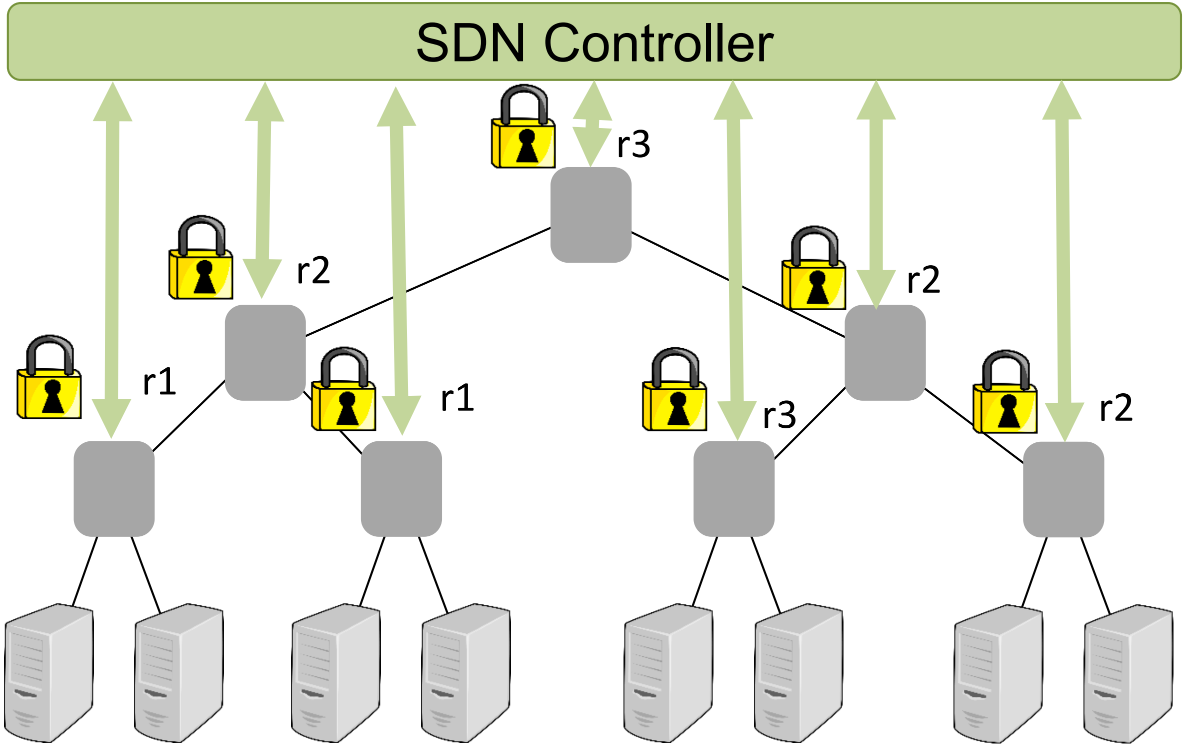
- Idea:
  - Use ***secure*** channels between controller and switches to distribute hash ranges
  - Give ***different hash ranges*** hash ranges to different switches, but add some ***redundancy***: risk of being caught!



Network Policy Checker for Adversarial Environments. Kashyap Thimmaraju, Liron Schiff, and S. SRDS 2019.

Solution: Use SDN for *Secure* Trajectory Sampling

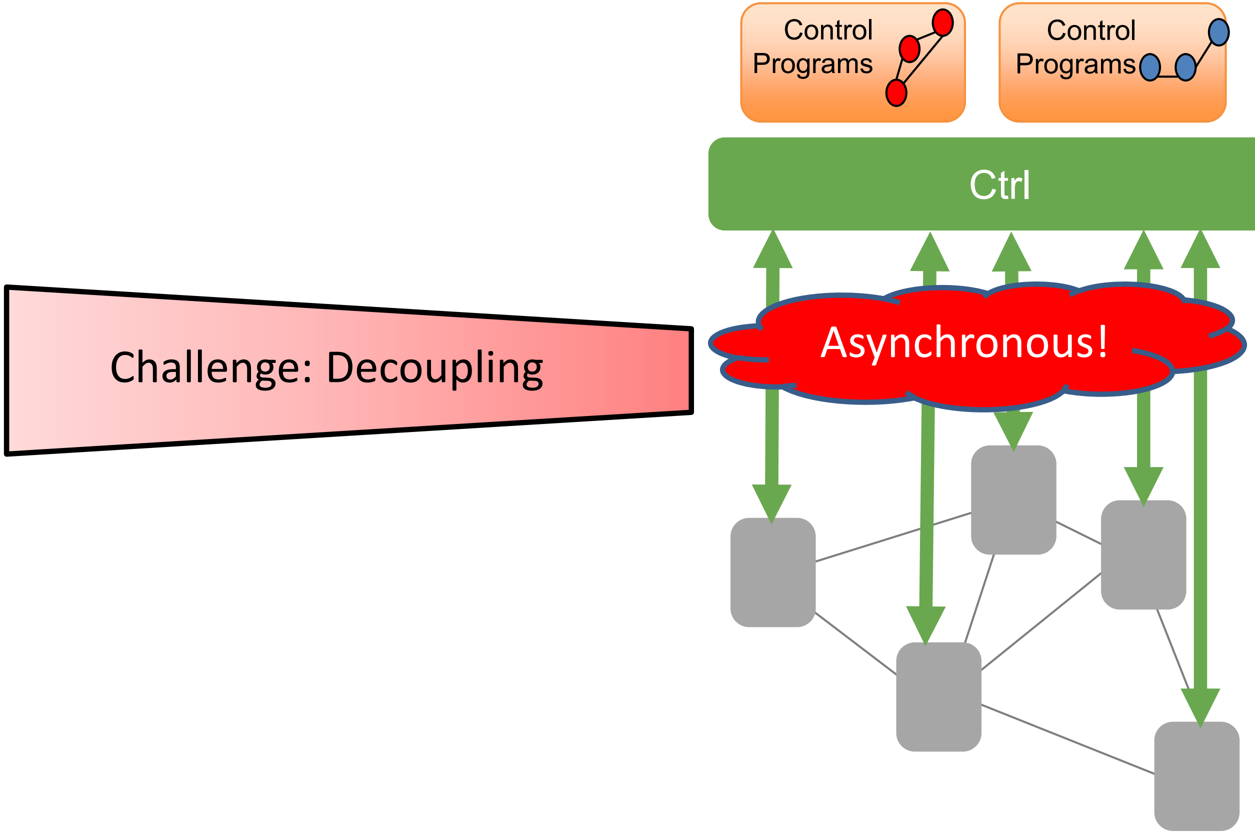
- Idea:
  - Use ***secure*** channels between controller and switches to distribute hash ranges
  - Give ***different hash ranges*** hash ranges to different switches, but add some ***redundancy***: risk of being caught!
- In general: obtaining live data from the network ***becomes easier!***



Network Policy Checker for Adversarial Environments. Kashyap Thimmaraju, Liron Schiff, and S. SRDS 2019.

Example: New Challenges

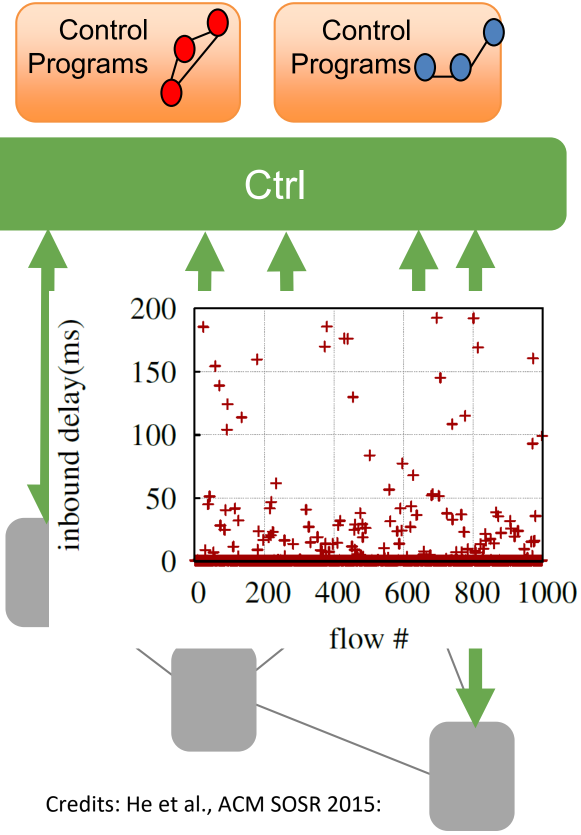
Recall: Our Mental Model



Recall: Our Mental Model

Challenge: Decoupling

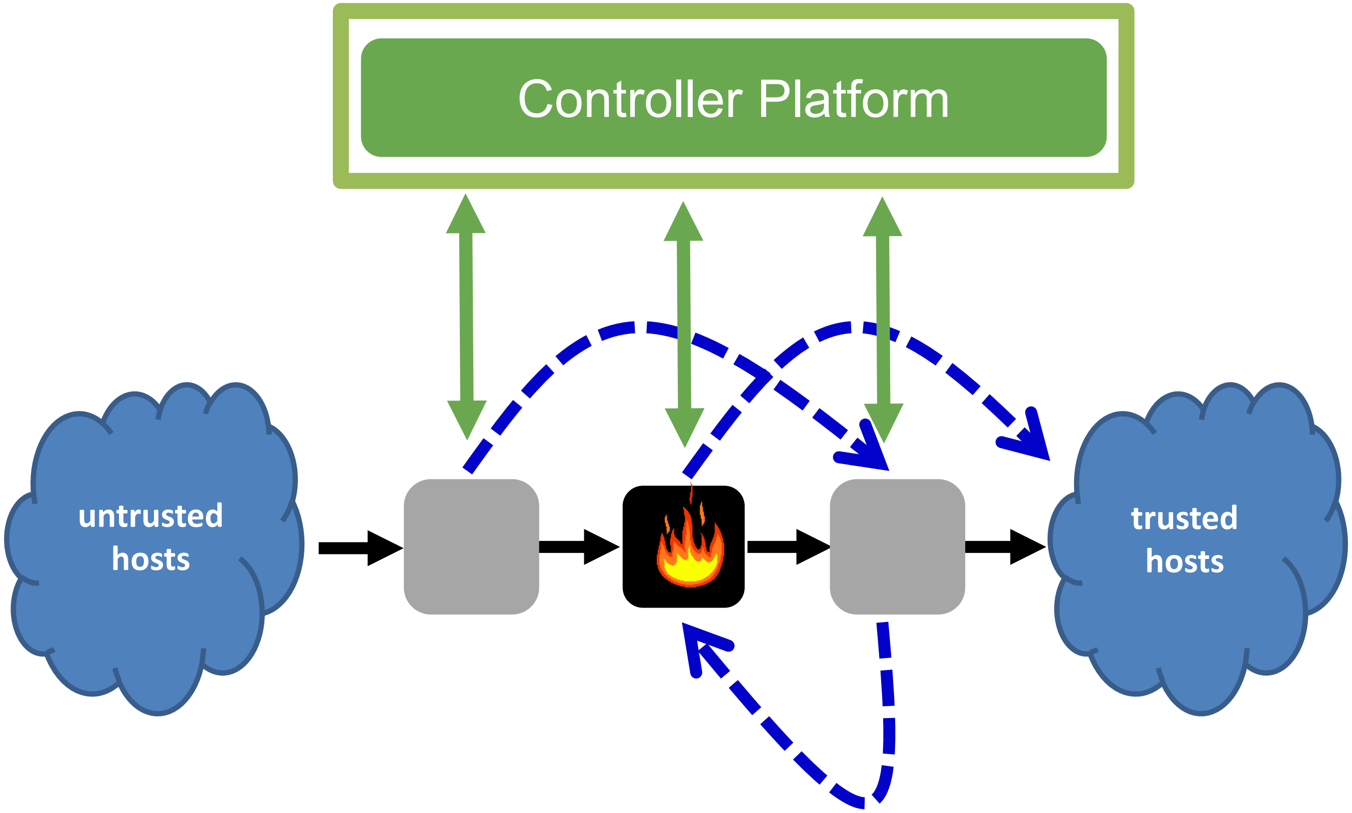
Despite centralization: SDN stays a distributed system!



Credits: He et al., ACM SOSR 2015:

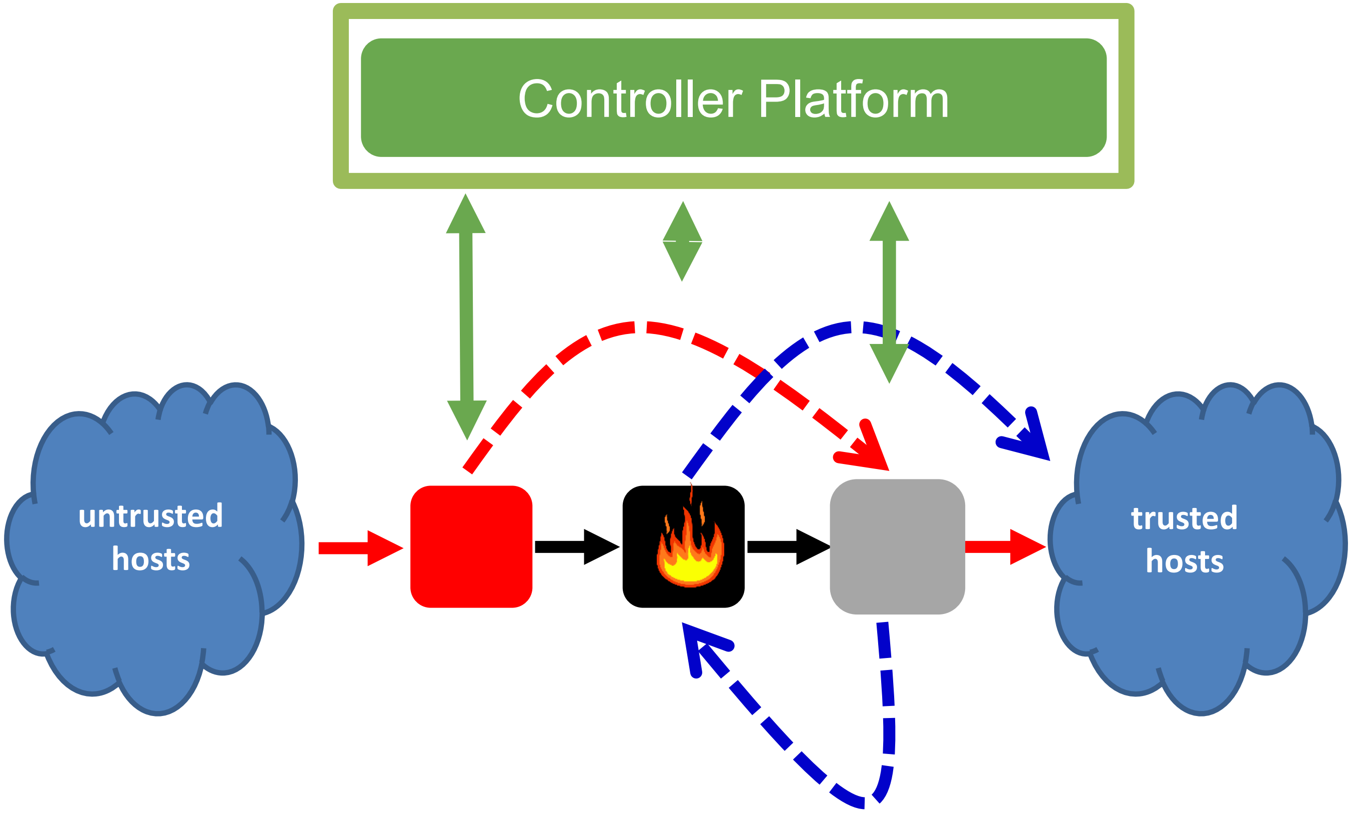
without network latency

Example “Route Updates”: *What can possibly go wrong?*



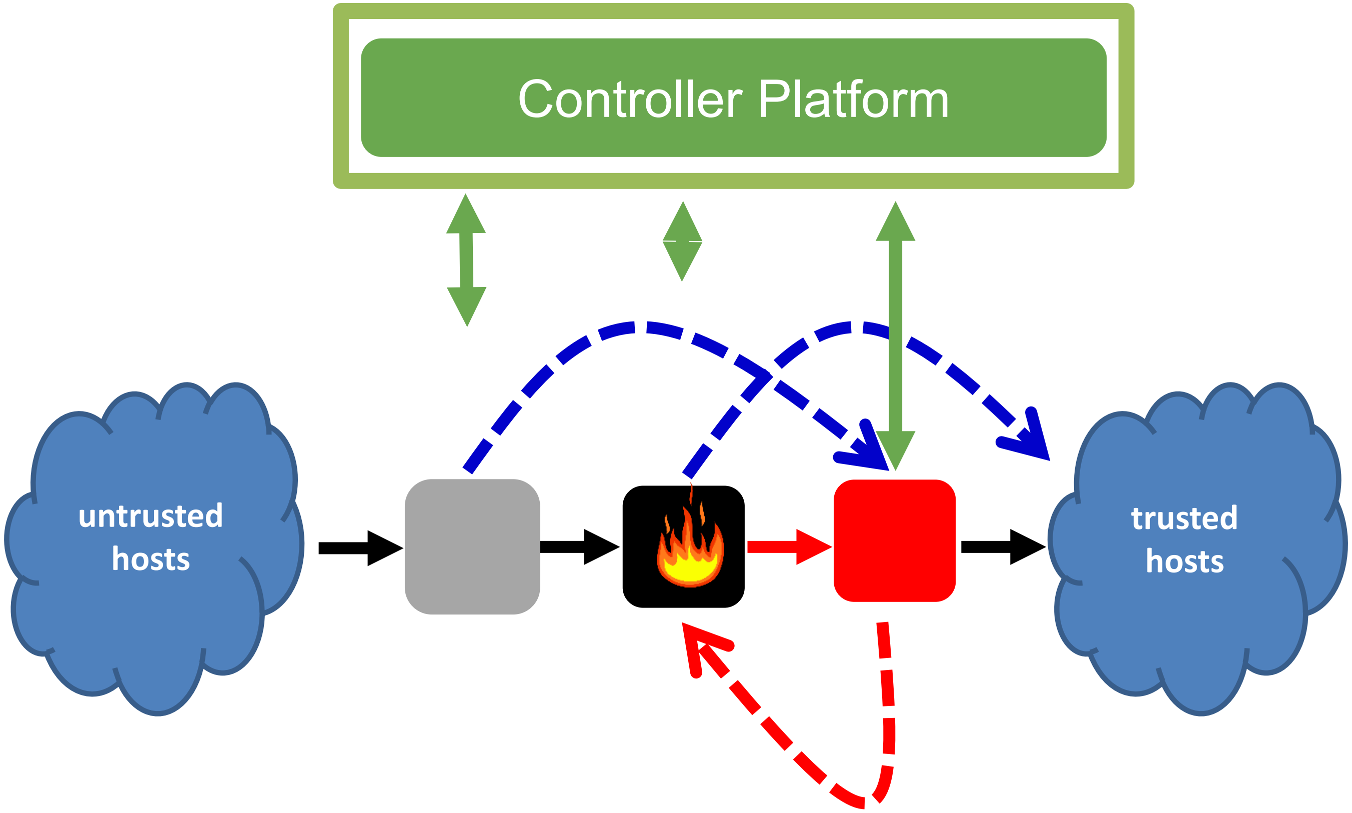
Invariant: Traffic from untrusted hosts to trusted hosts via firewall!

Problem 1: Bypassed Waypoint



Invariant: Traffic from untrusted hosts to trusted hosts via firewall!

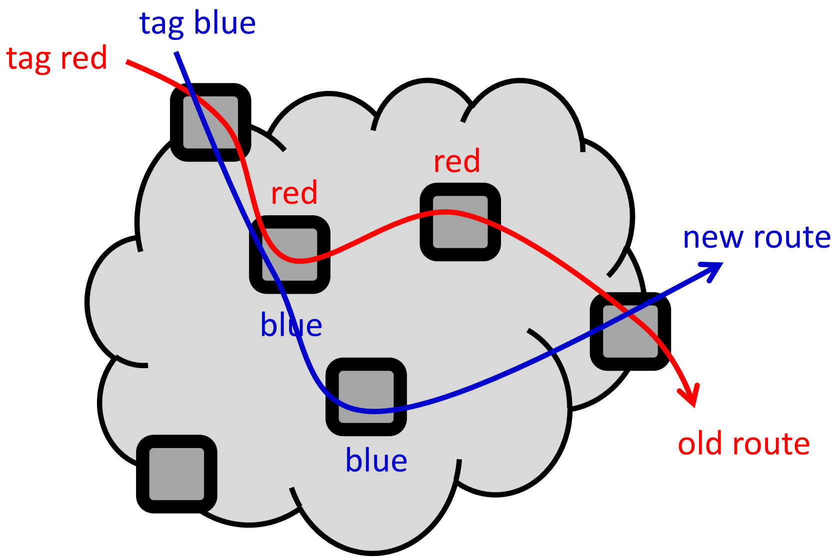
Problem 2: *Transient* Loop



Invariant: Traffic from untrusted hosts to trusted hosts via firewall!

Tagging: A Universal Solution?

- Old route: red
- New route: blue
- 2-Phase Update:
  - Install blue flow rules internally
  - Flip tag at ingress ports

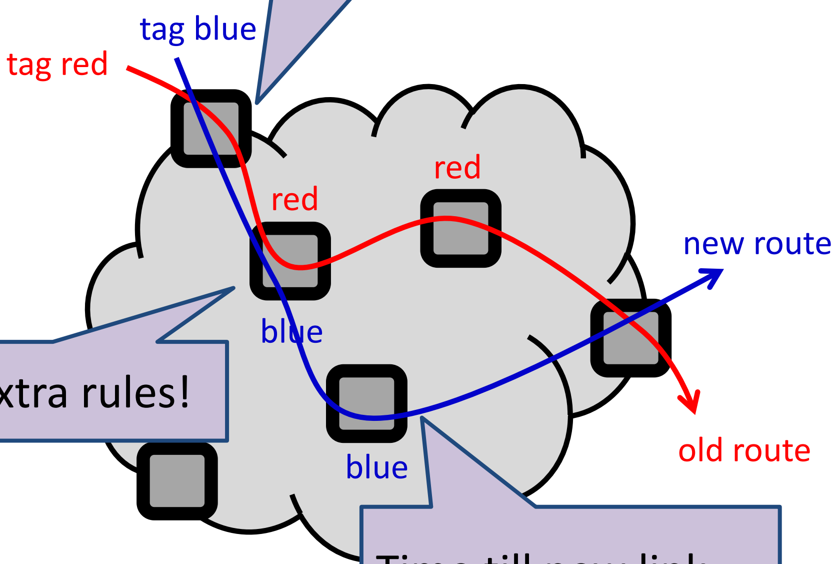


Reitblatt et al. Abstractions for Network Update, ACM SIGCOMM 2012.

Tagging: A Universal Solution?

- Old route: red
- New route: blue
- 2-Phase Update:
  - Install blue rules internally
  - Flip tag at ingress ports

Where to tag?
Header space?
Overhead!



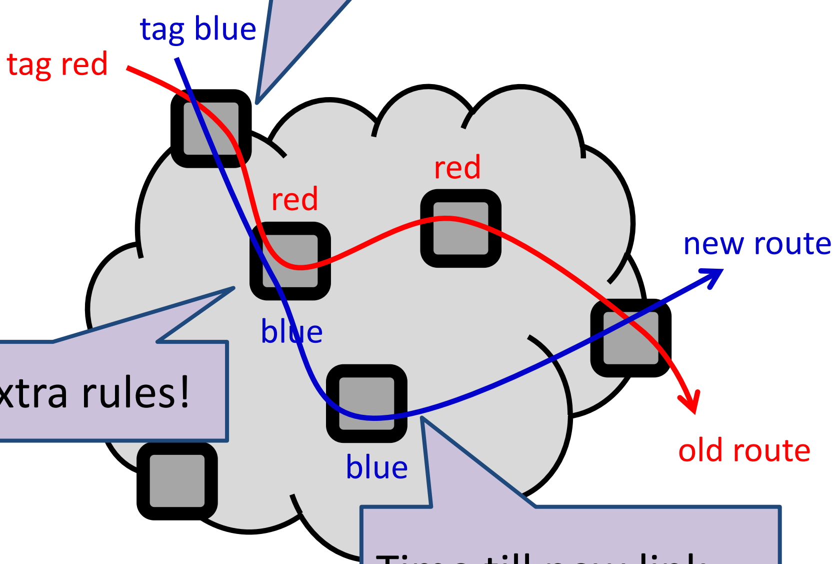
Cost of extra rules!

Time till new link becomes available!

Reitblatt et al. Abstractions for Network Update, ACM SIGCOMM 2012.

Tagging: A Universal Solution?

- Old route: red
- New route: blue
- 2-Phase Update:
  - Install blue rules internally
  - Flip tag at ingress ports



Where to tag?
Header space?
Overhead!

Cost of extra rules!

Time till new link becomes available!

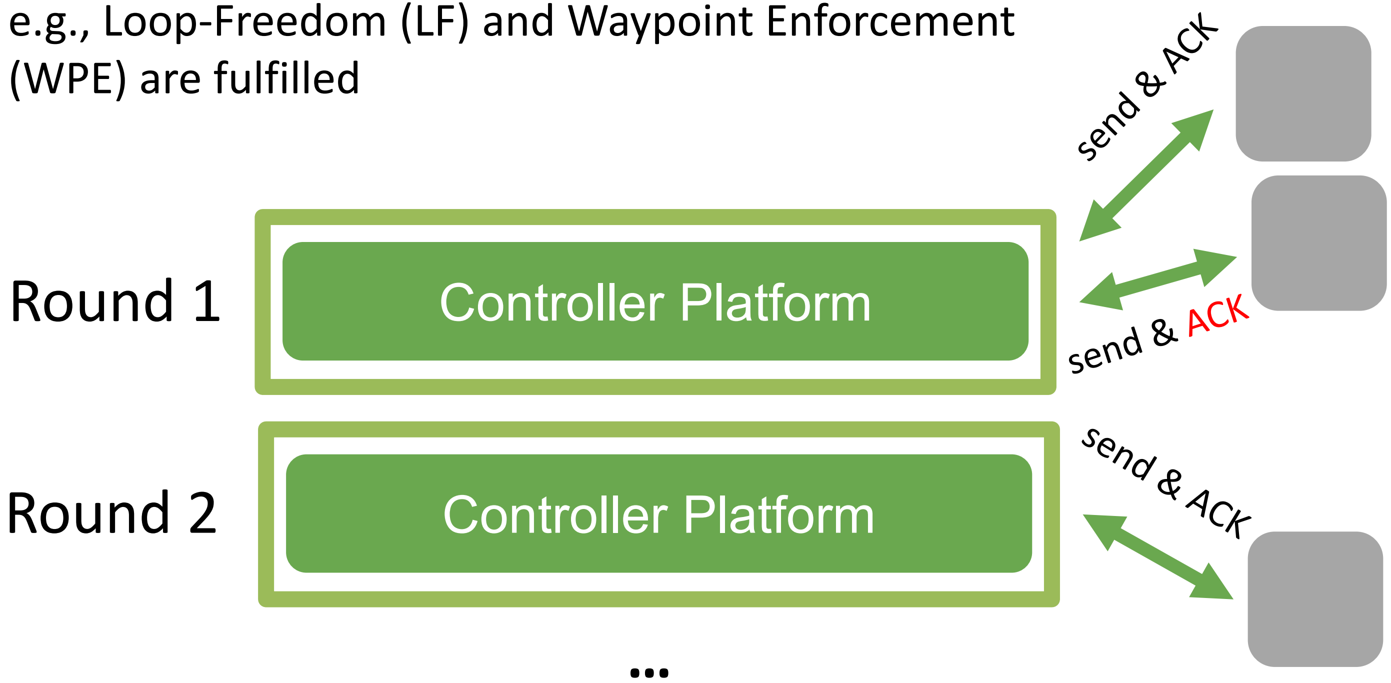
Possible solution without tagging, and at least preserve weaker consistency properties?

Reitblatt et al. Abstractions for Network Update, ACM SIGCOMM 2012.

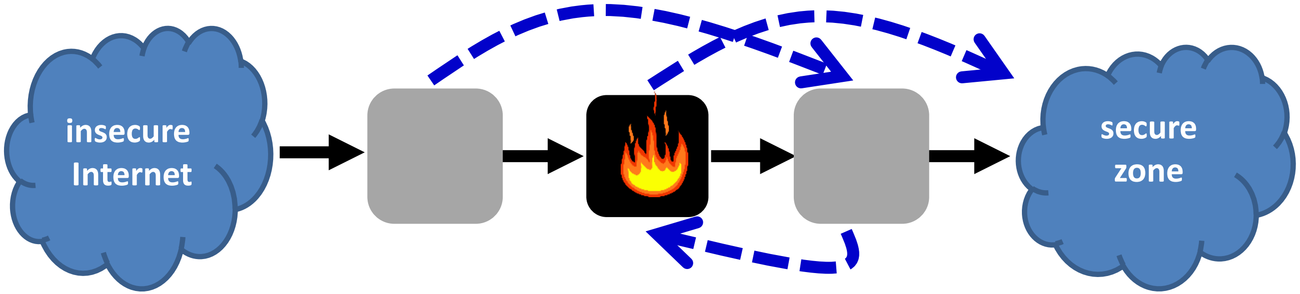
Idea: Schedule “Safe” Subsets of Nodes Only, Then Wait for ACK!

Idea: Schedule safe update subsets in multiple rounds!

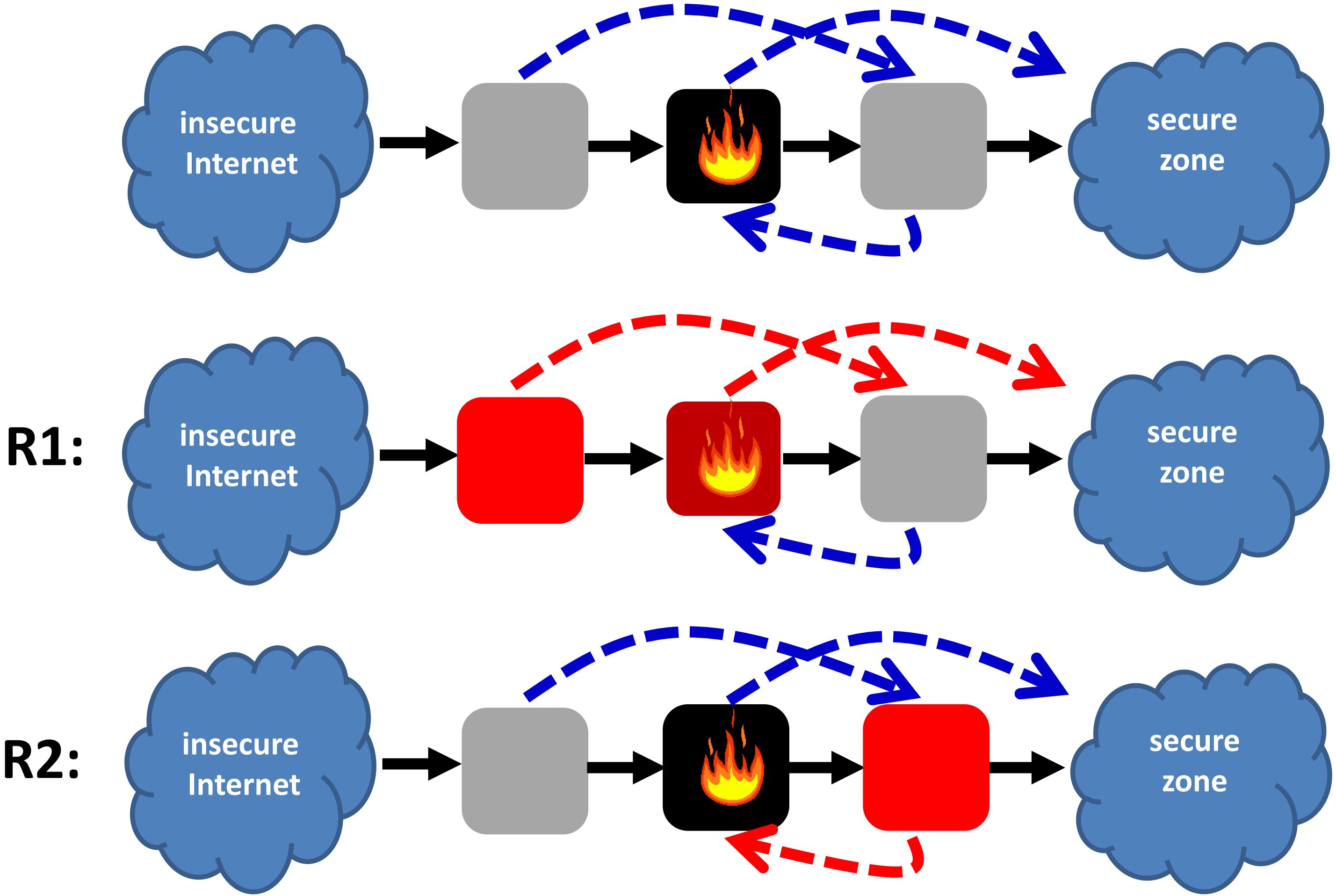
Packet may take a mix of old and new path, as long as, e.g., Loop-Freedom (LF) and Waypoint Enforcement (WPE) are fulfilled



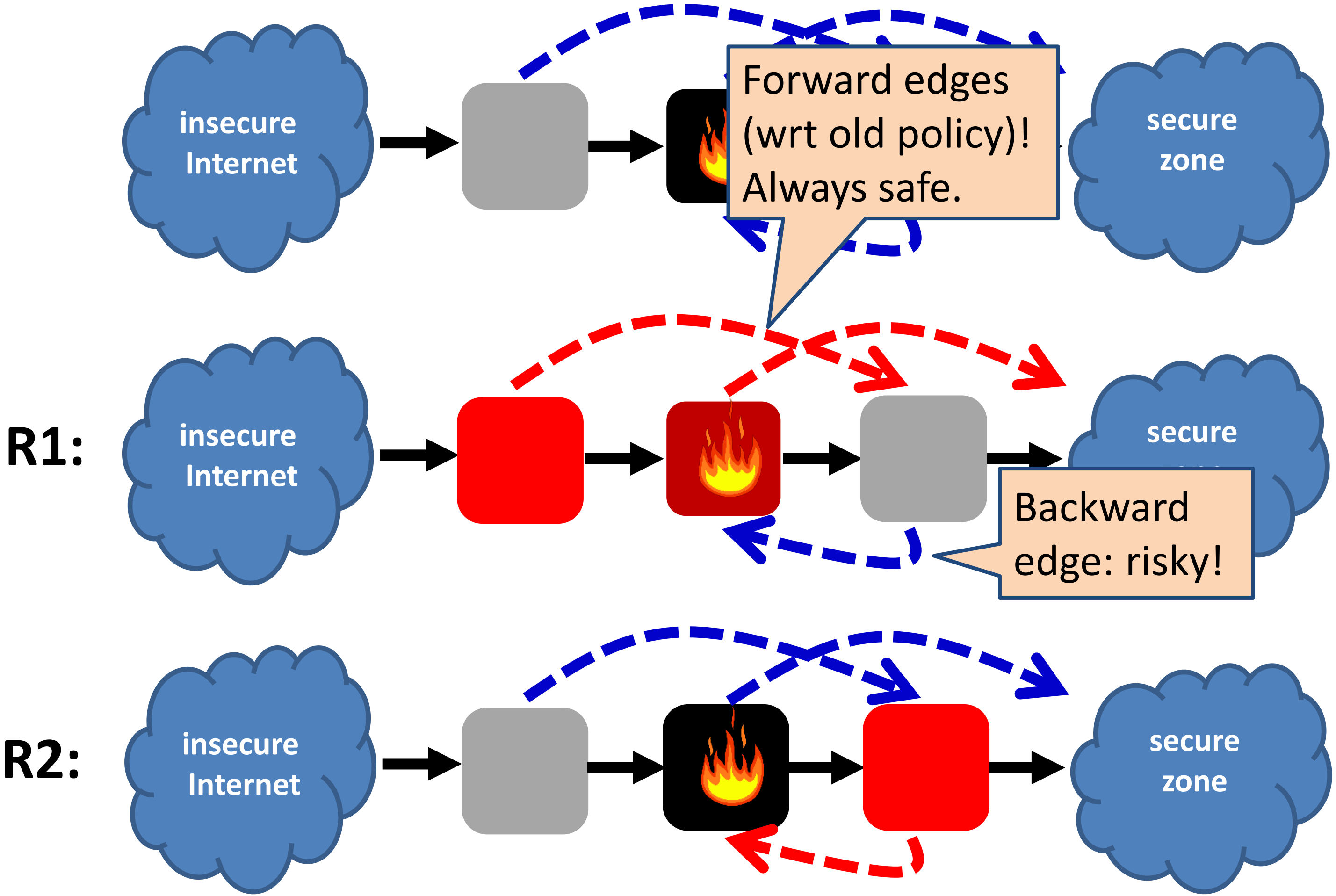
Loop-Free Update Schedule



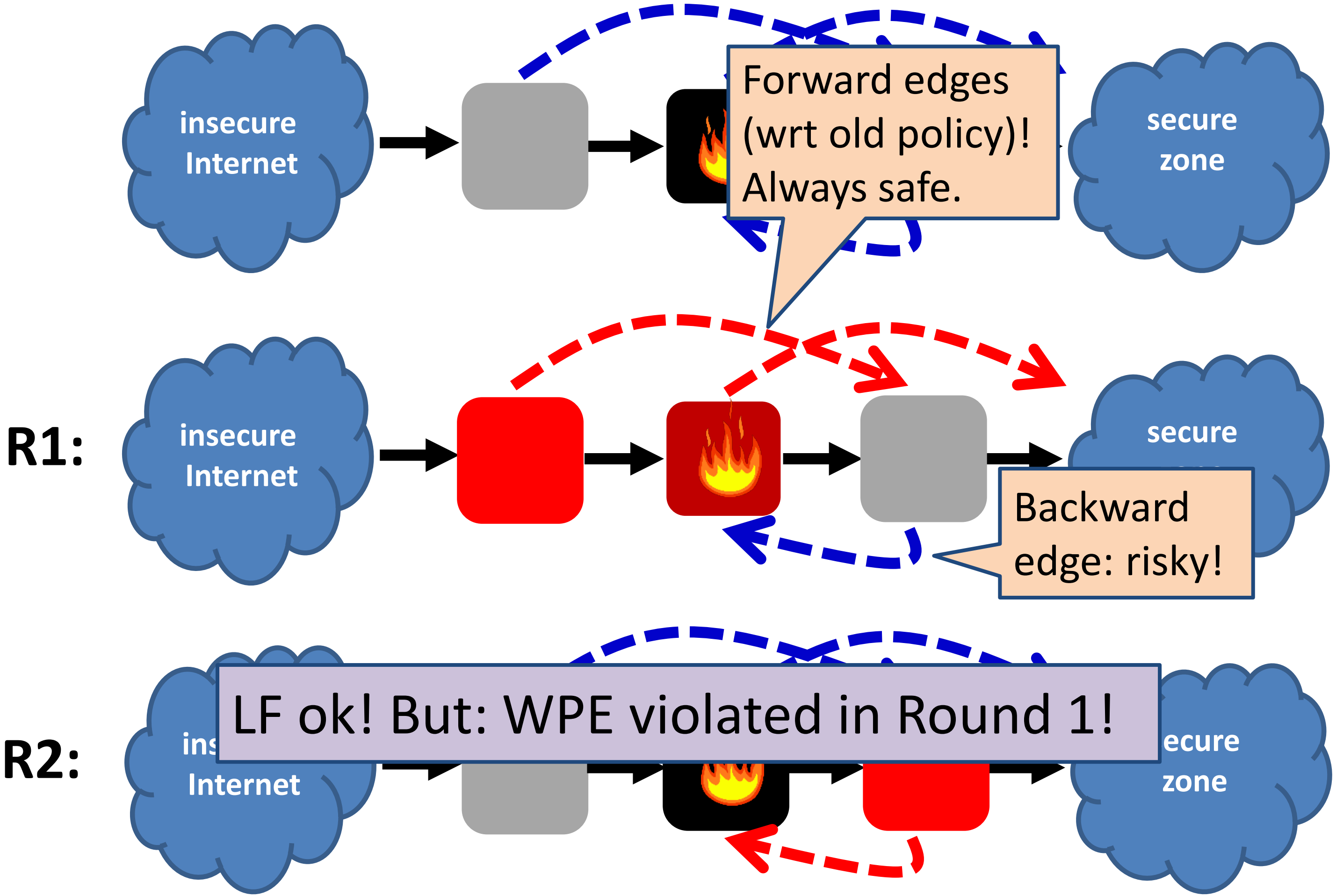
Loop-Free Update Schedule



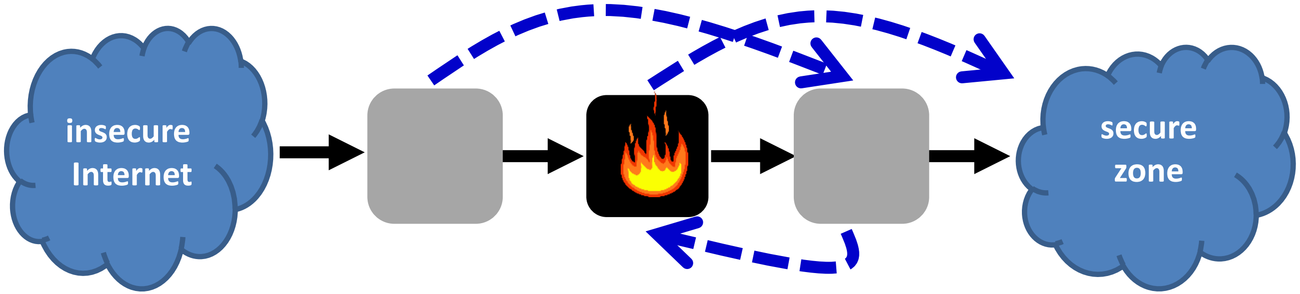
Loop-Free Update Schedule



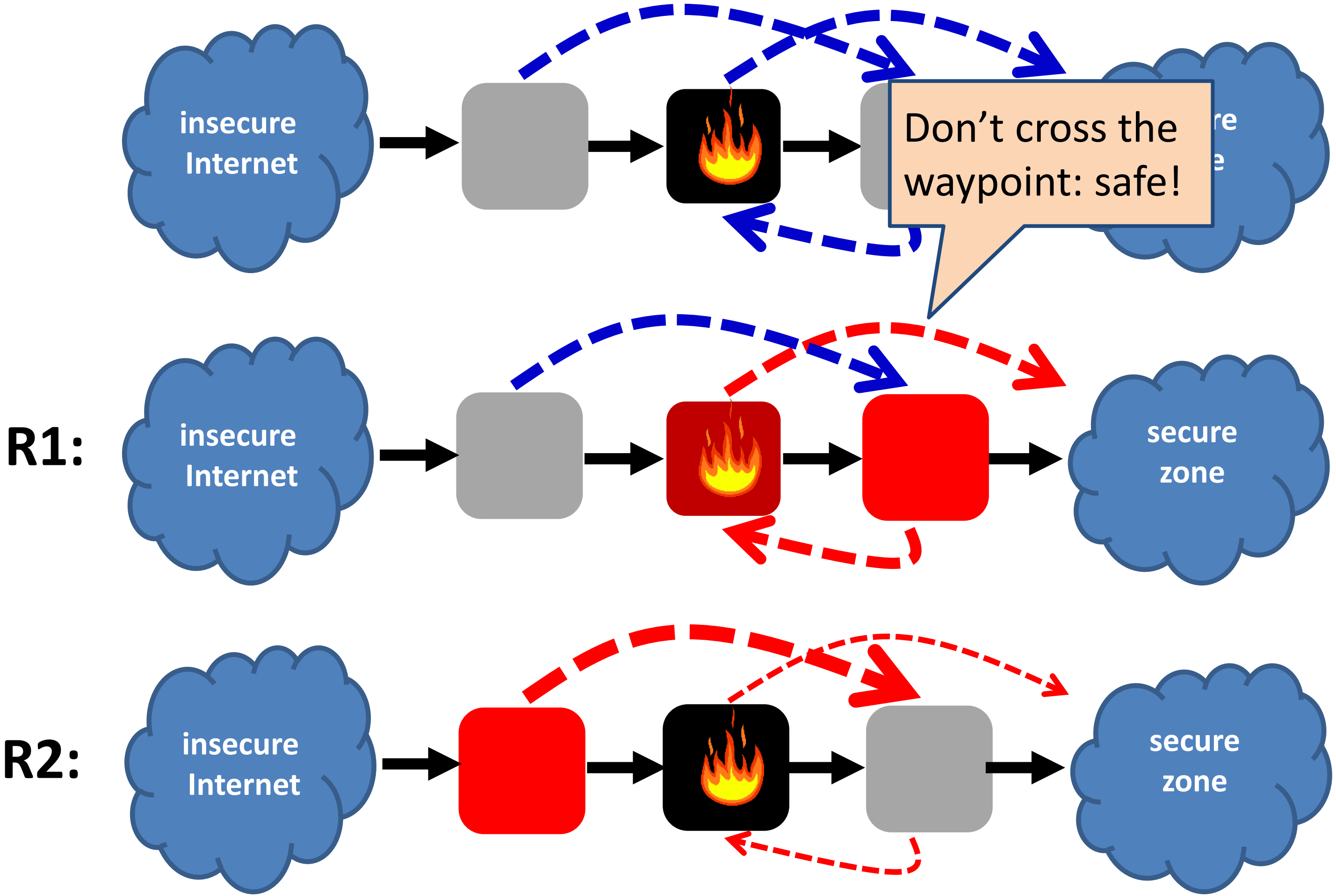
Loop-Free Update Schedule



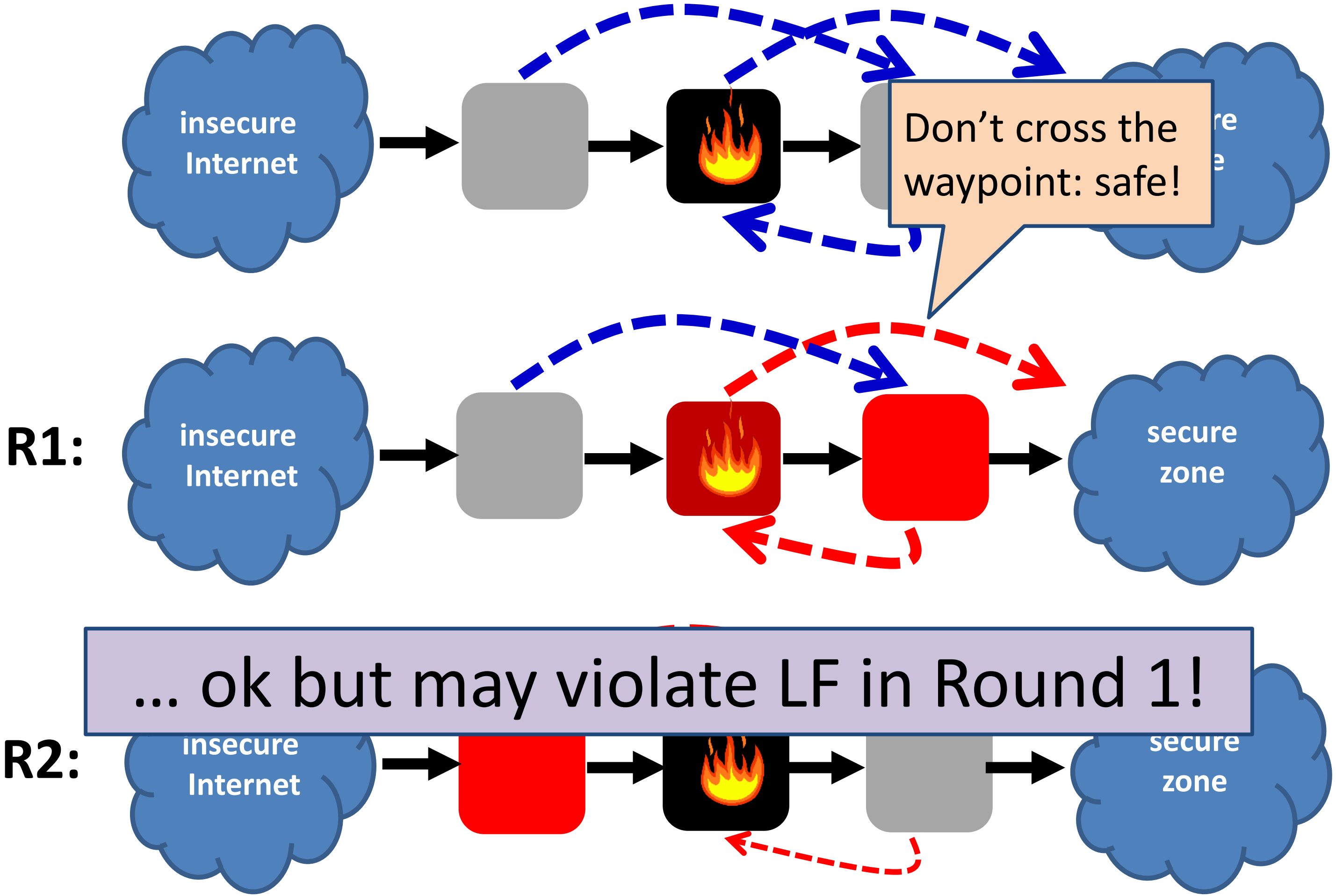
Waypoint Respecting Schedule



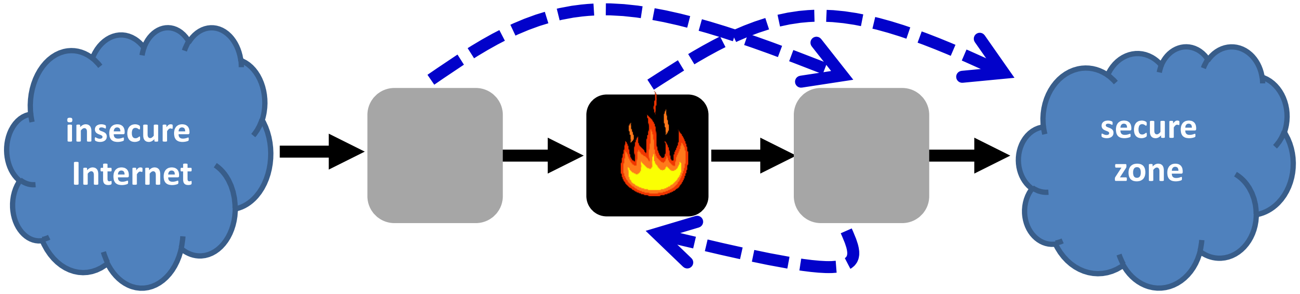
Waypoint Respecting Schedule



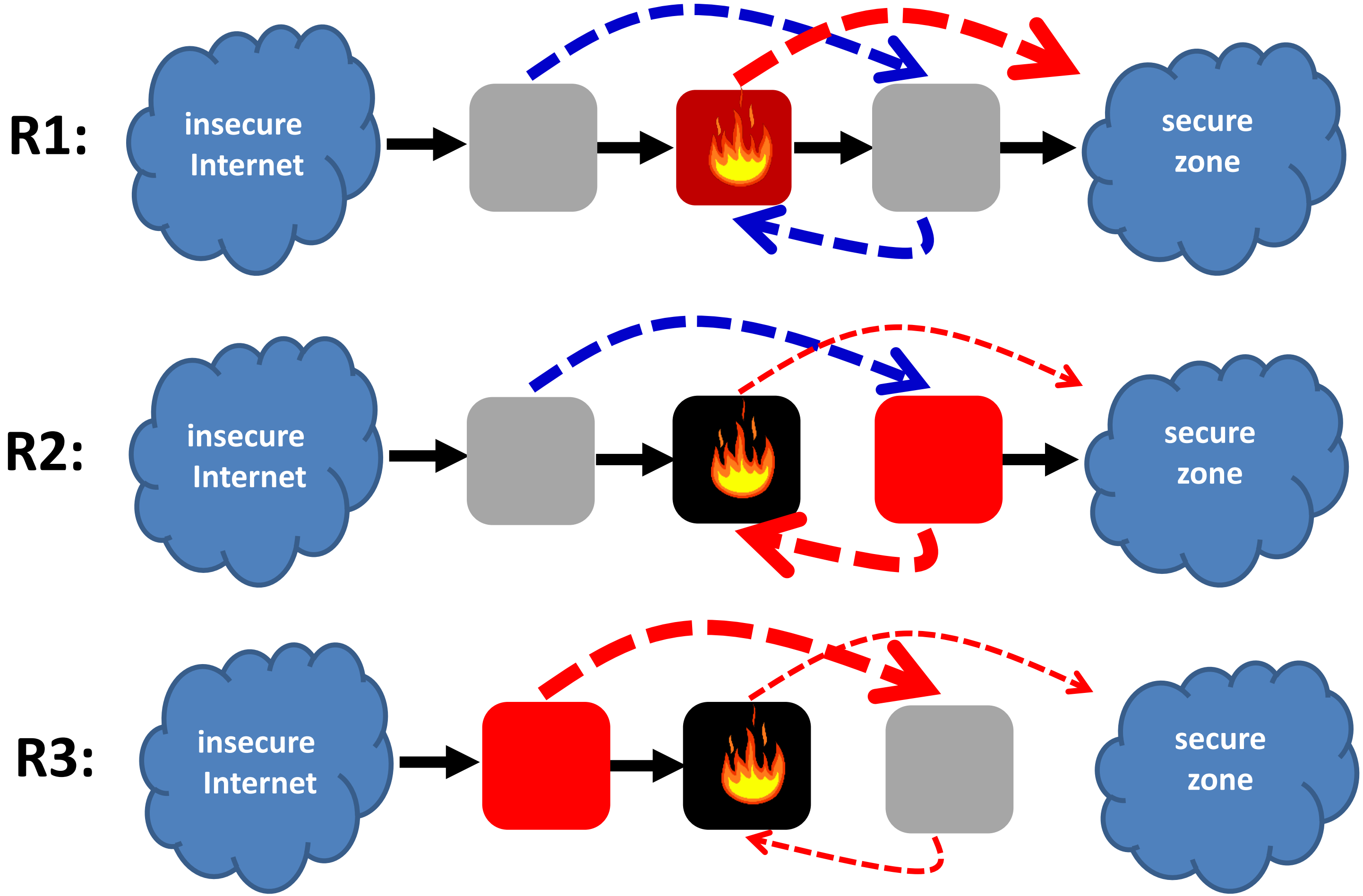
Waypoint Respecting Schedule



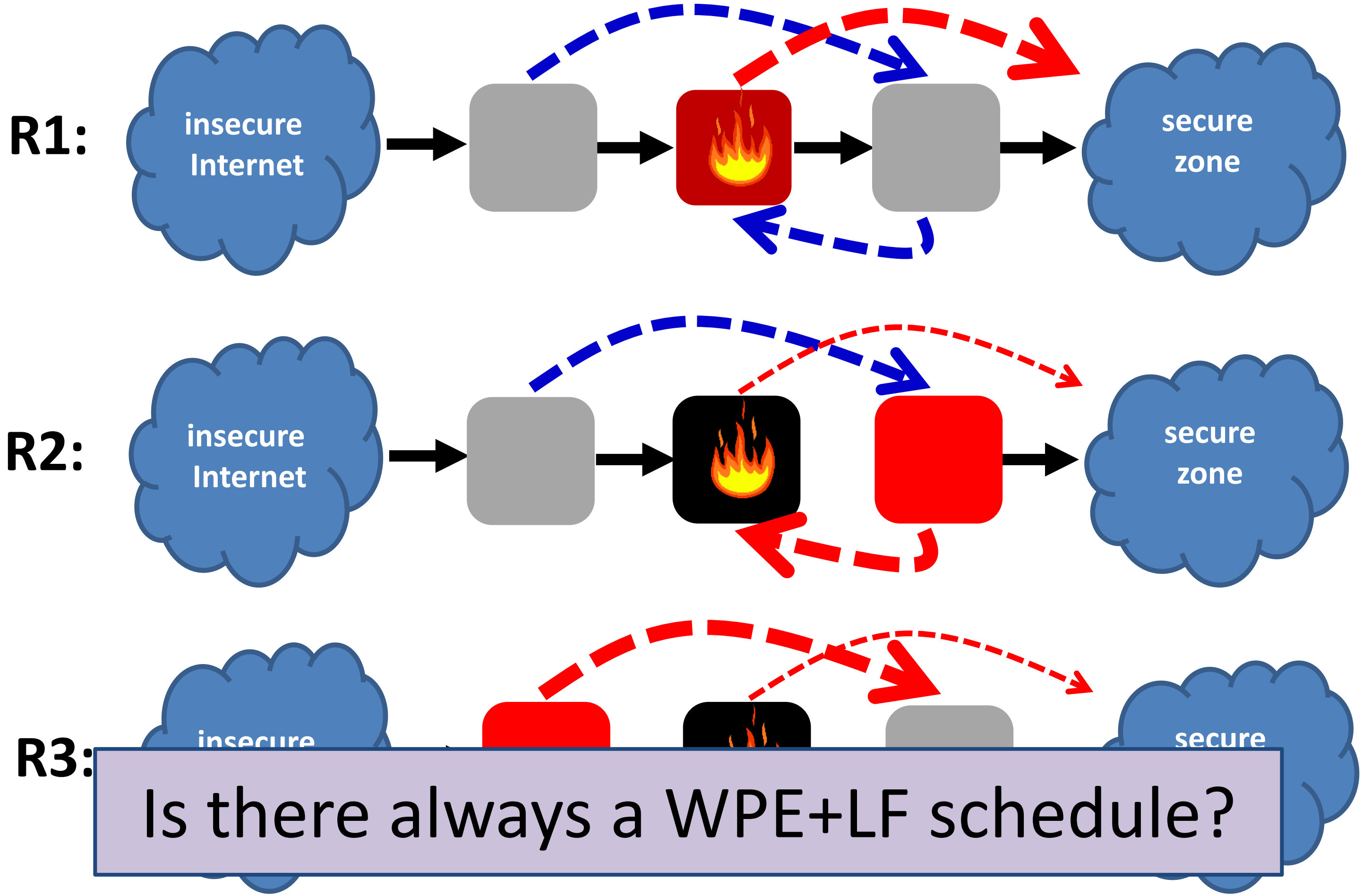
Can we have both LF and WPE?



Yes: but it takes 3 rounds!

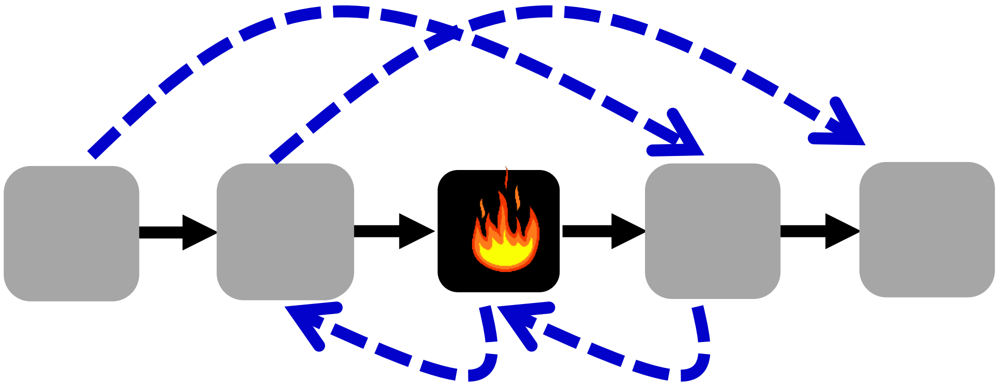


Yes: but it takes 3 rounds!

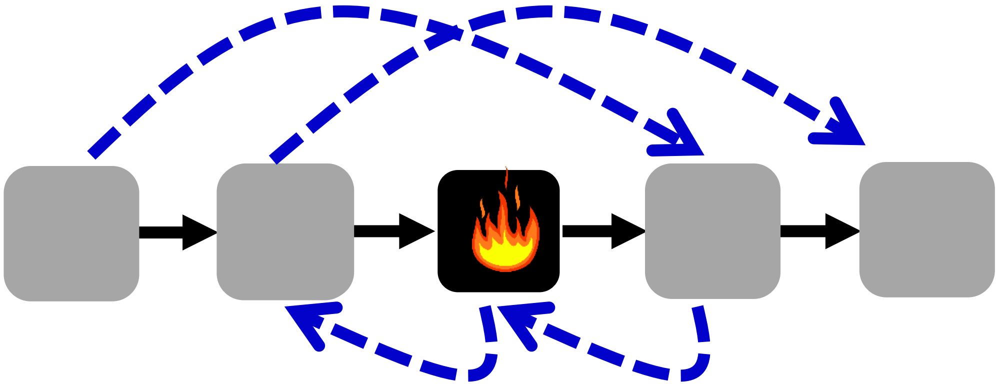


Is there always a WPE+LF schedule?

What about this one?



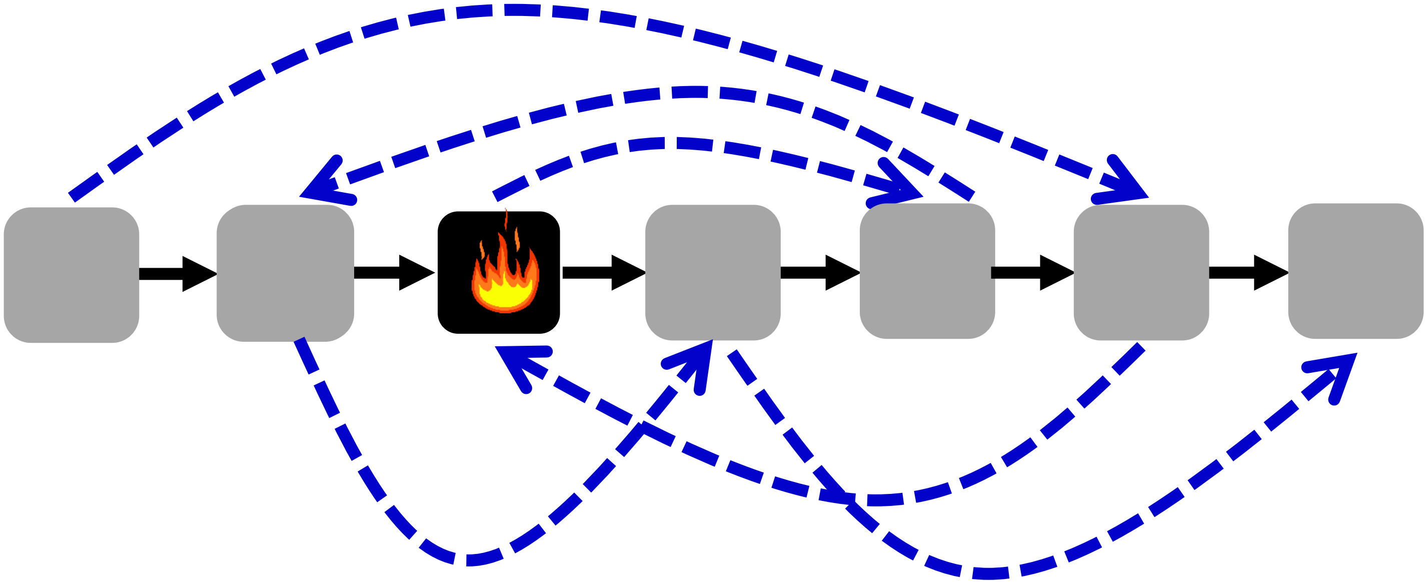
LF and WPE may conflict!



- ❑ Cannot update any **forward edge** in R1: WP
- ❑ Cannot update any **backward edge** in R1: LF

No schedule exists! Resort to tagging…

What about this one?

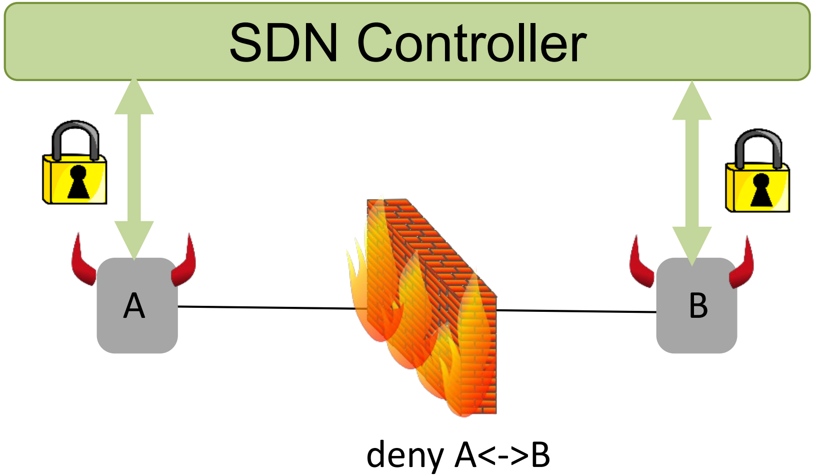


Further reading:
Survey of Consistent Software-Defined Network Updates
Klaus-Tycho Foerster, Stefan Schmid, and Stefano Vissicchio. IEEE Communications Surveys and Tutorials (**COMST**), to appear.

Example: New Threats

New Types of Attacks: Via SDN Controller

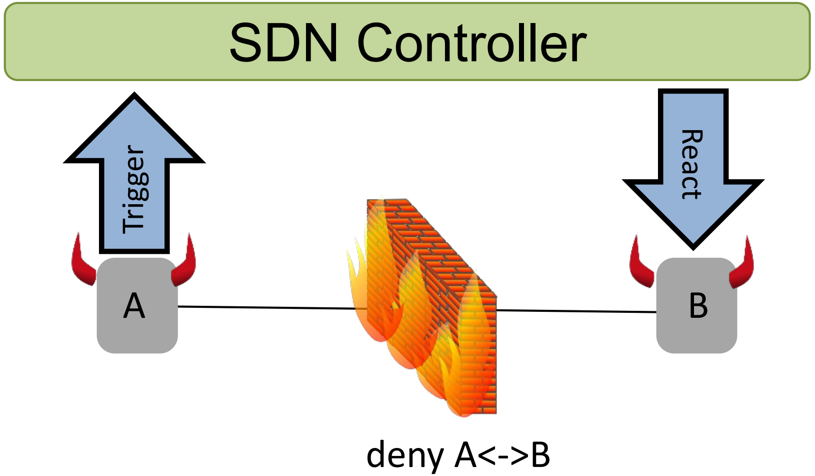
- **Controller** may be attacked or exploited



Outsmarting Network Security with SDN Teleportation
Kashyap Thimmaraju, Liron Schiff, and S.
EuroS&P, Paris, France, April 2017.

New Types of Attacks: Via SDN Controller

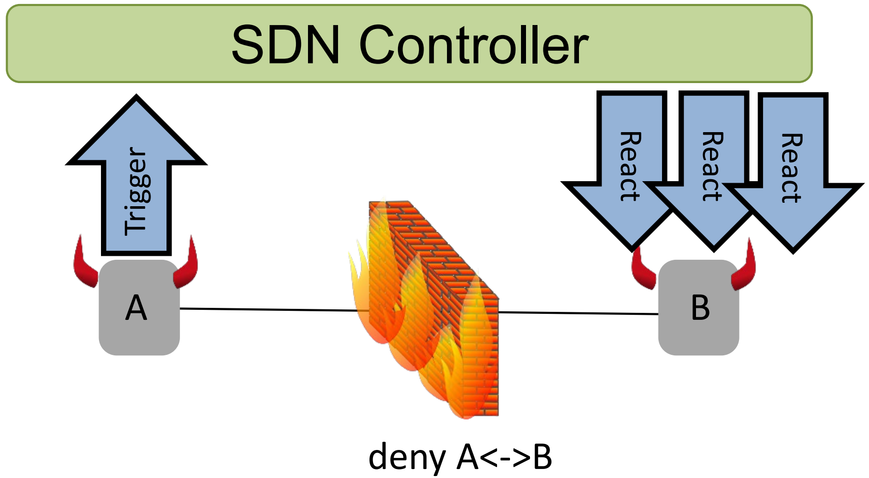
- **Controller** may be attacked or exploited
  - By design, *reacts* to switch events, e.g., by packet-outs



Outsmarting Network Security with SDN Teleportation
Kashyap Thimmaraju, Liron Schiff, and S.
EuroS&P, Paris, France, April 2017.

New Types of Attacks: Via SDN Controller

- **Controller** may be attacked or exploited
  - By design, *reacts* to switch events, e.g., by packet-outs
  - Or even *multicast*: **pave-path technique** more efficient than hop-by-hop

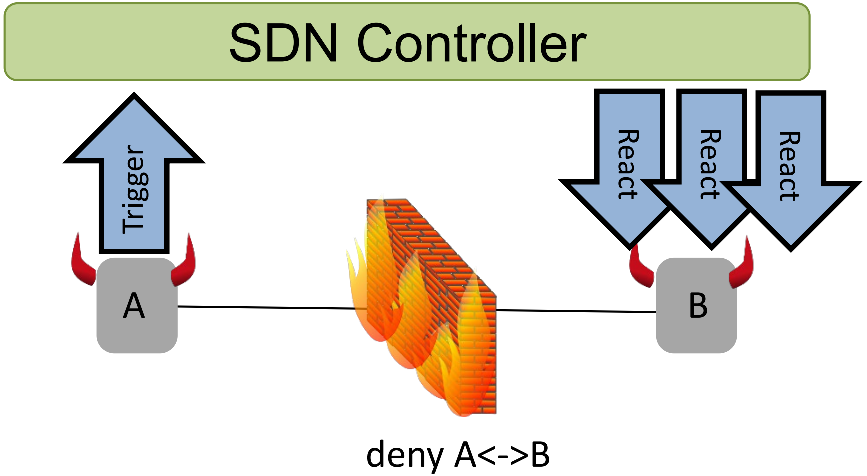


Outsmarting Network Security with SDN Teleportation
Kashyap Thimmaraju, Liron Schiff, and S.
EuroS&P, Paris, France, April 2017.

New Types of Attacks: Via SDN Controller

- **Controller** may be attacked or exploited
  - By design, *reacts* to switch events, e.g., by packet-outs
  - Or even *multicast*: **pave-path technique** more efficient than hop-by-hop

May introduce ***new communication paths*** which can be used in unintendend ways!



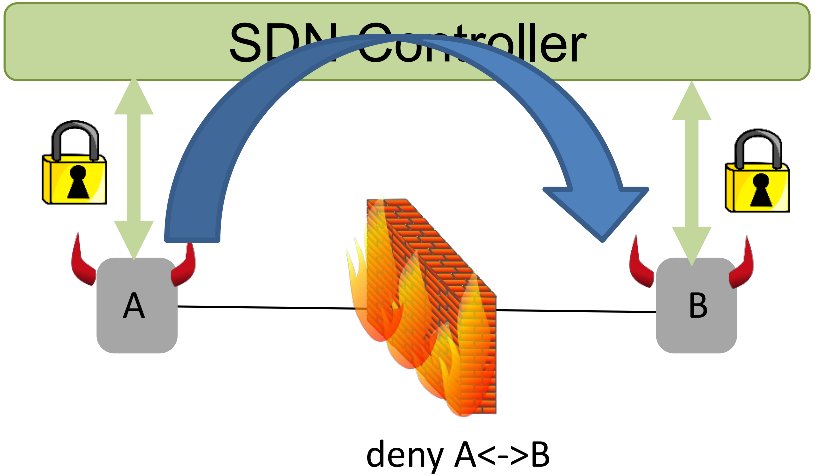
Outsmarting Network Security with SDN Teleportation
Kashyap Thimmaraju, Liron Schiff, and Stefan Schmid.
EuroS&P, Paris, France, April 2017 + ***CVEs***.

New Types of Attacks: Via SDN Controller

- In particular: new **covert communication** channels
  - E.g., exploit MAC learning (use codeword „0xBADDAD“) or modulate information with timing

- May *bypass security-critical elements*: e.g., firewall in the dataplane

- *Hard to catch*: along „normal communication paths“ and encrypted

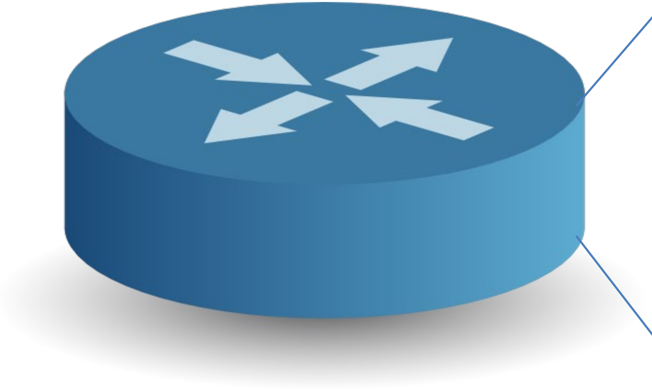


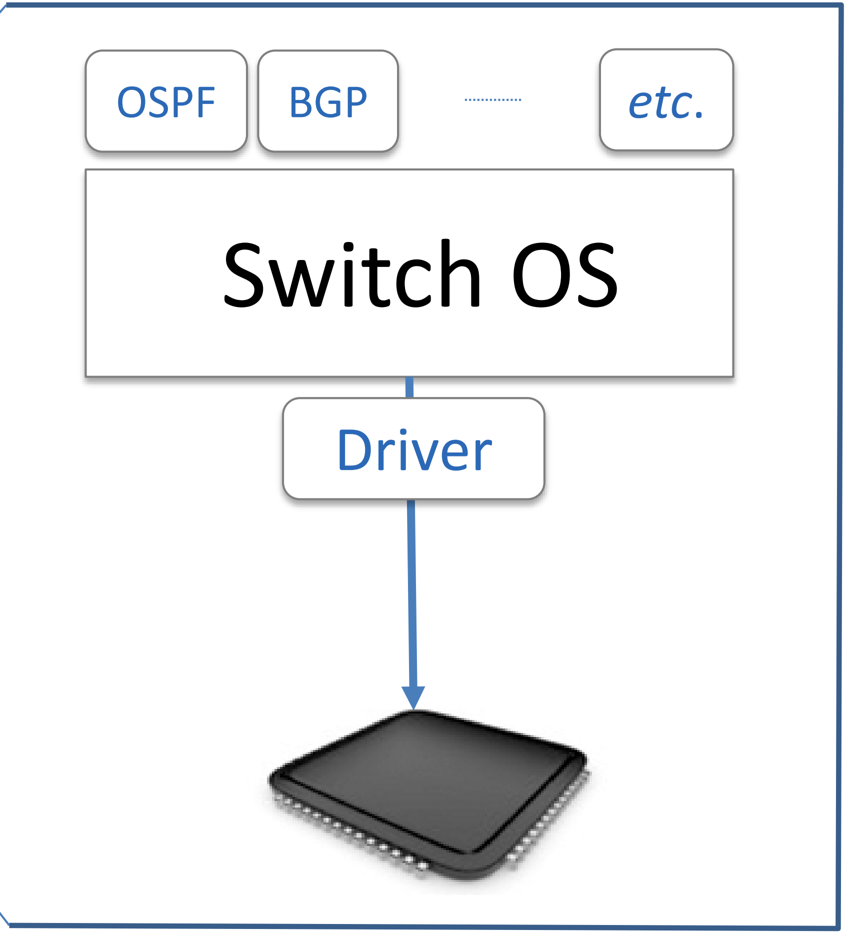
Outsmarting Network Security with SDN Teleportation
Kashyap Thimmaraju, Liron Schiff, and Stefan Schmid.
EuroS&P, Paris, France, April 2017 + ***CVEs***.

Roadmap

- Software-defined networks
- **Programmable dataplanes**
- Network virtualization

Innovation is Slow: Example VxLAN

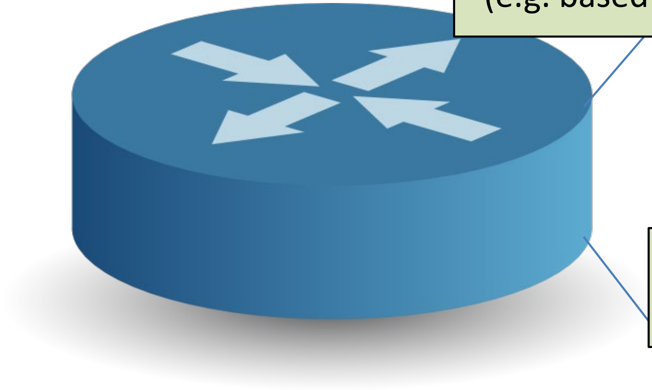


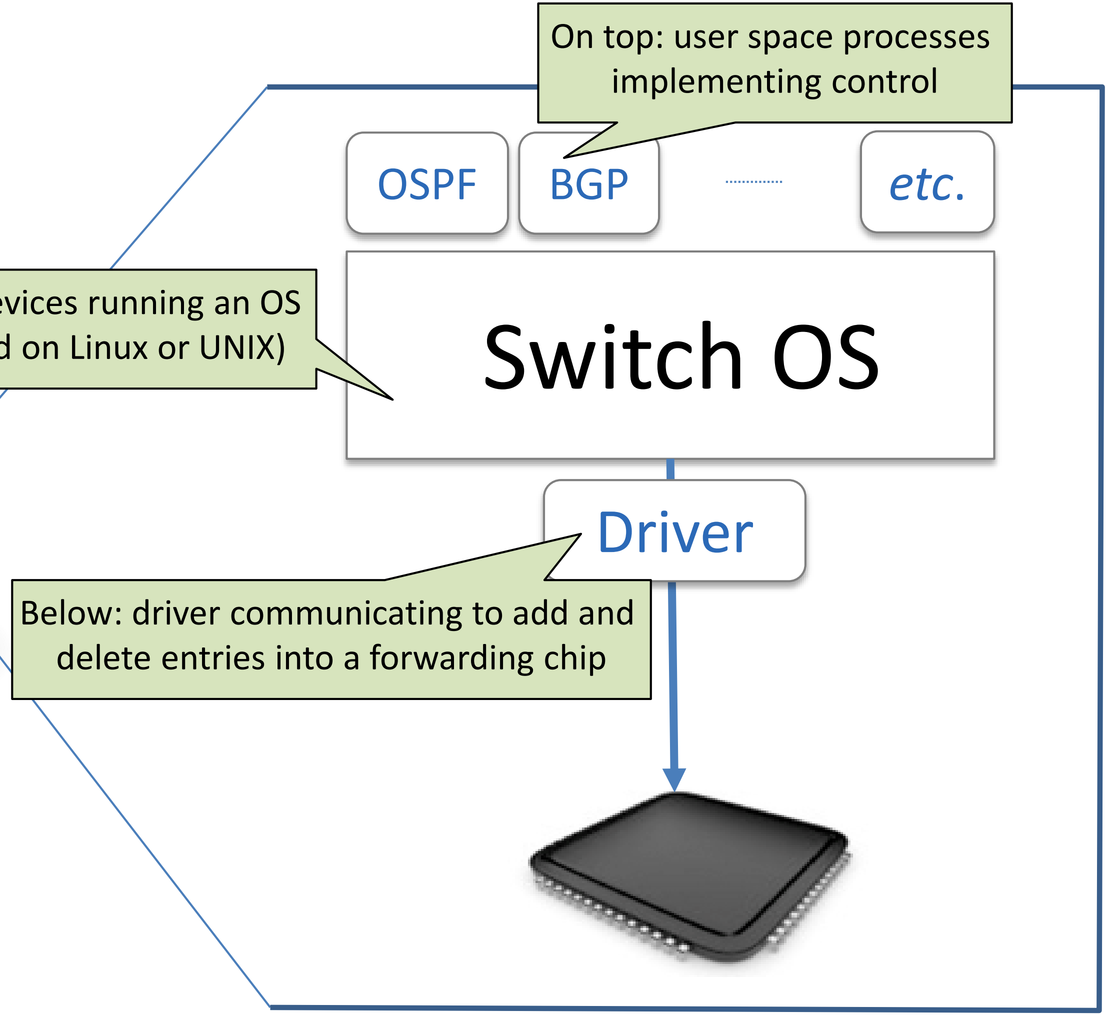


VxLAN: In principle, addition of a *simple function* to be added to switches and routers

- Defined 2010 by Cisco and Vmware

Innovation is Slow: Example VxLAN

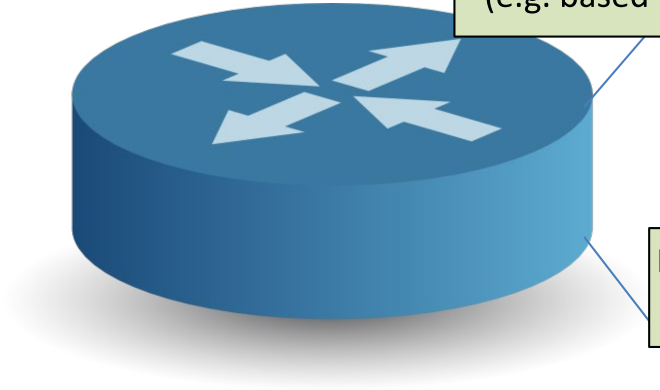


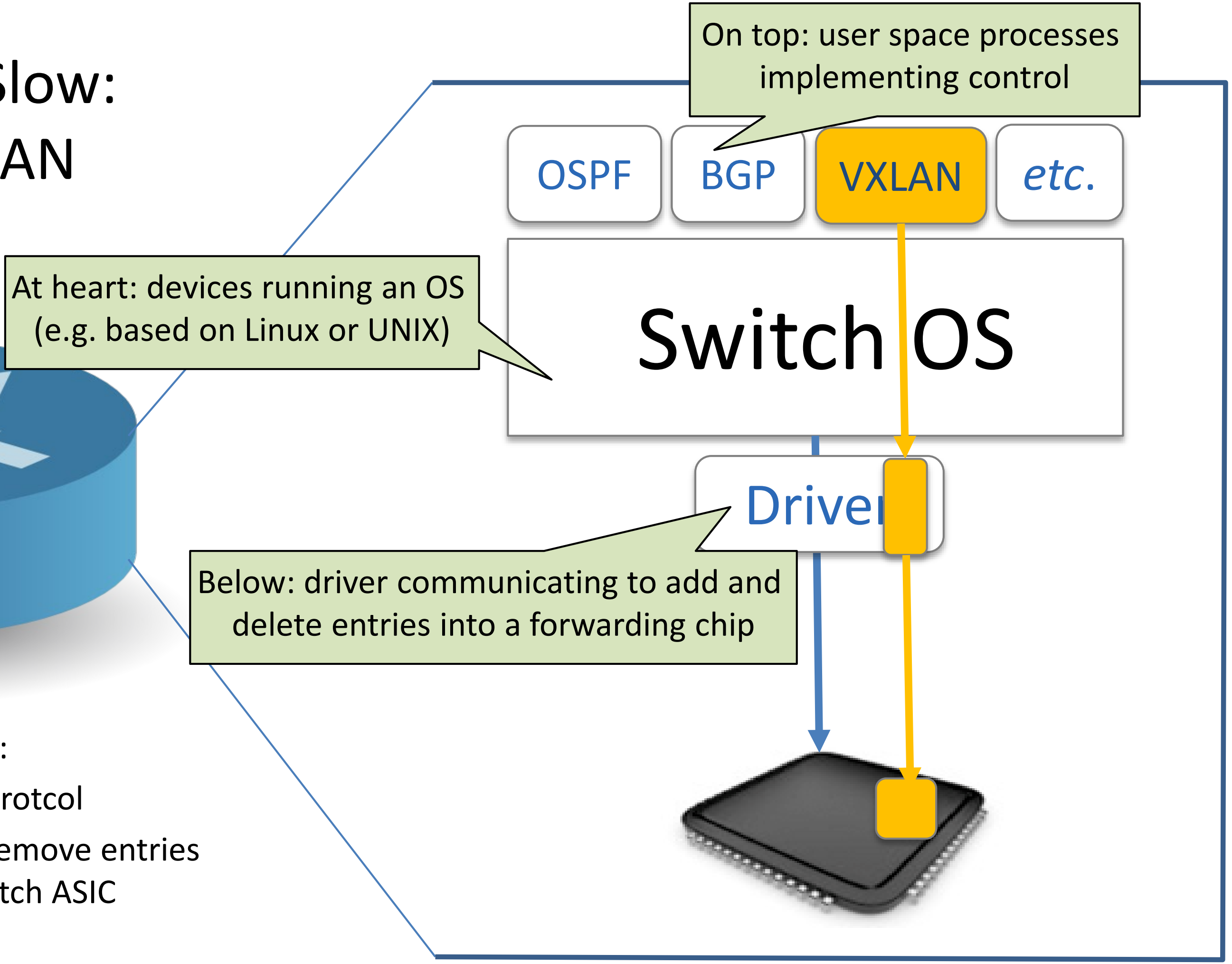


VxLAN: In principle, addition of a *simple function* to be added to switches and routers

- Defined 2010 by Cisco and Vmware

Innovation is Slow: Example VxLAN

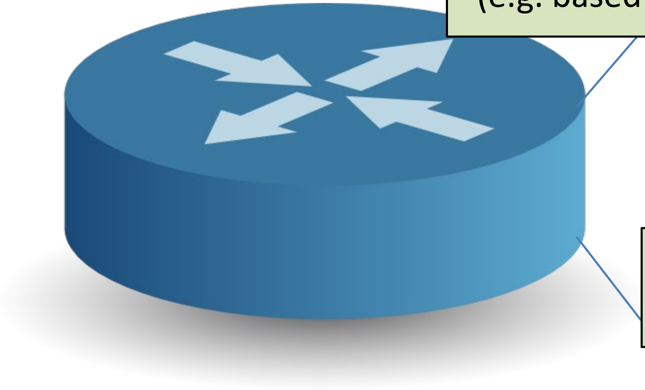


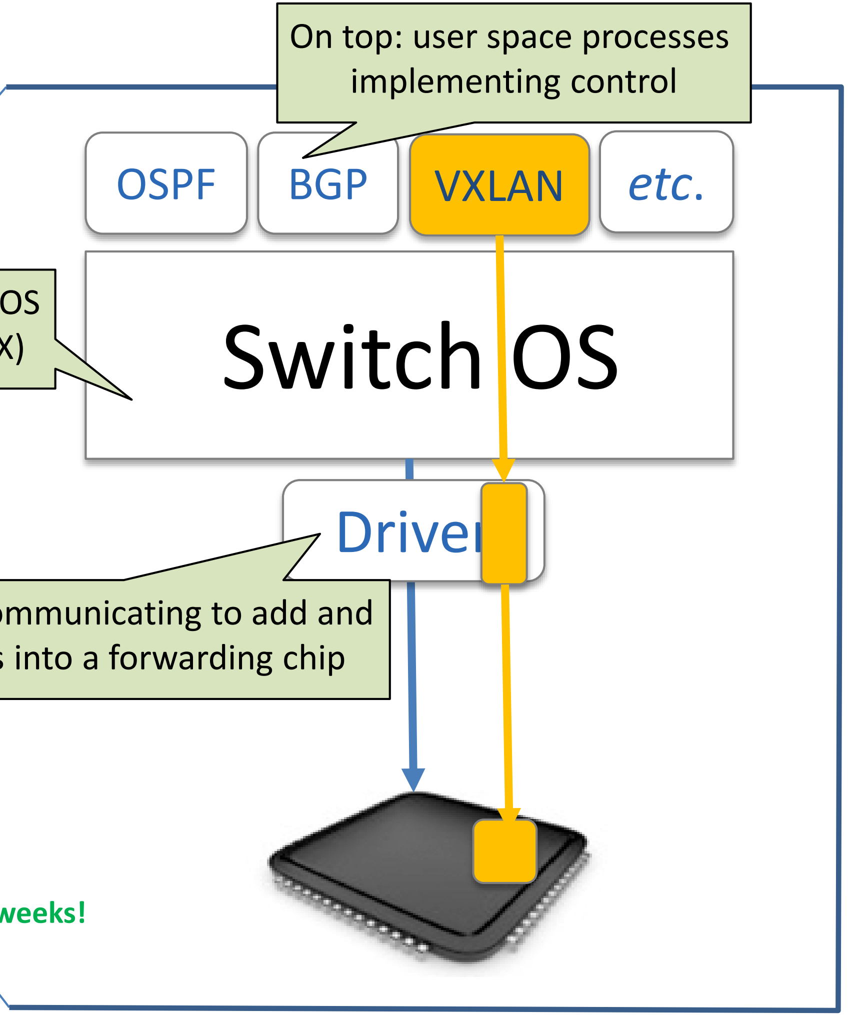


Needed steps to add VxLAN:

- *Add control* of VxLAN protcol
- *Change driver* to add/remove entries into VxLAN table in switch ASIC
- *Update ASIC*

Innovation is Slow: Example VxLAN

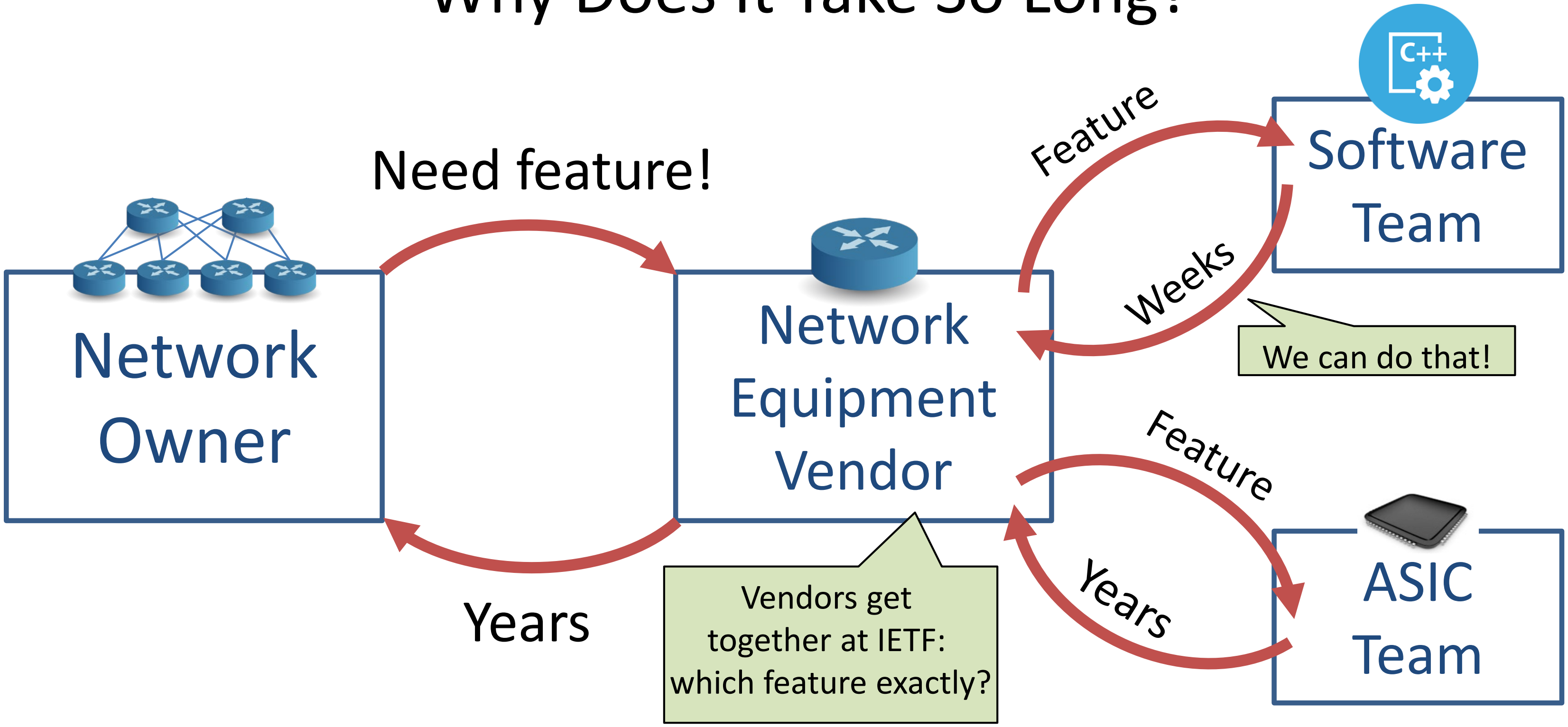




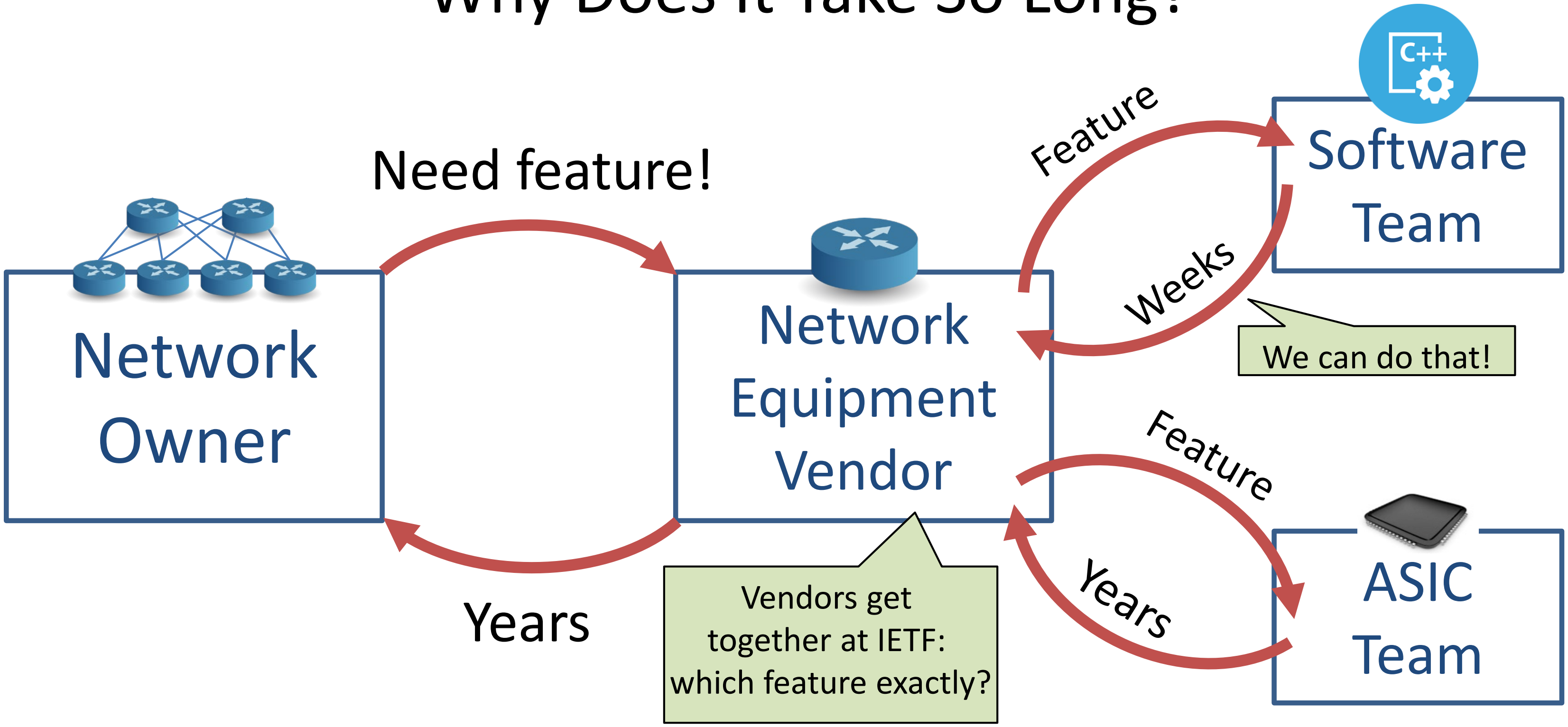
Needed steps to add VxLAN:

- *Add control* of VxLAN protcol 👍 **Doable in weeks!**
- *Change driver* to add/remove entries into VxLAN table in switch ASIC 👍 **Doable in weeks!**
- *Update ASIC* **Took 4 years to add feature to ASIC! ☹**

Why Does It Take So Long?



Why Does It Take So Long?



In the meantime, owners probably figured out a workaround making network more complex and brittle.

Besides Slow Innovation: Process is Inflexible and Expensive

Operator says:

I need extended VTP (VLAN Trunking Protocol) / a 3rd spanport etc. !

Vendor's answer:

Buy one of these!

Besides Slow Innovation: Process is Inflexible and Expensive

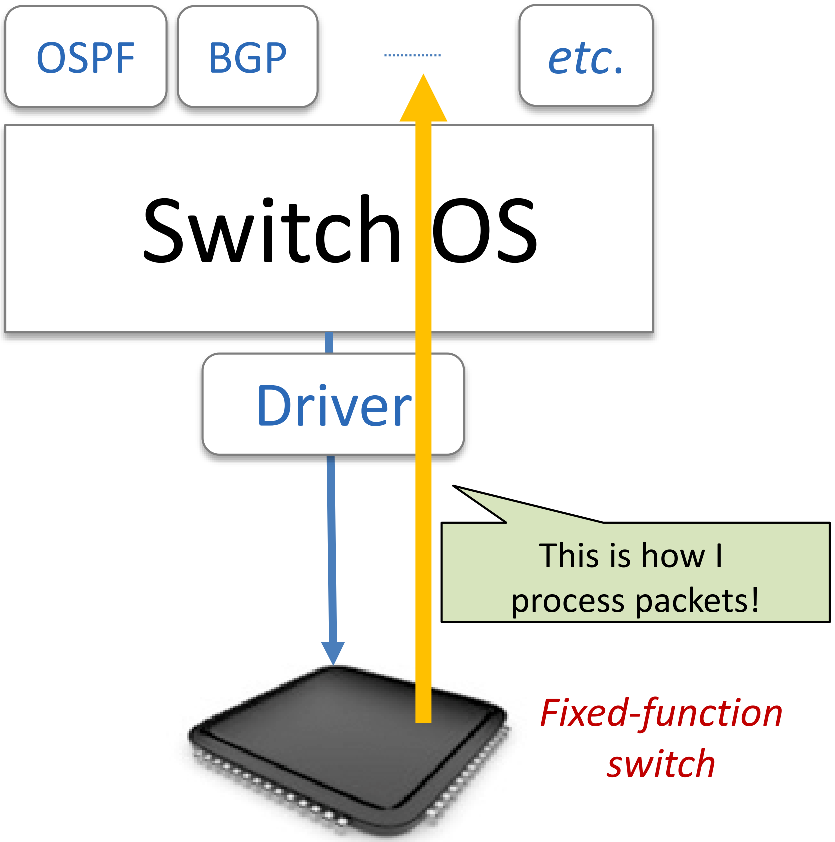
Operator says:

I need something better than STP for my data-center...

Vendor's answer:

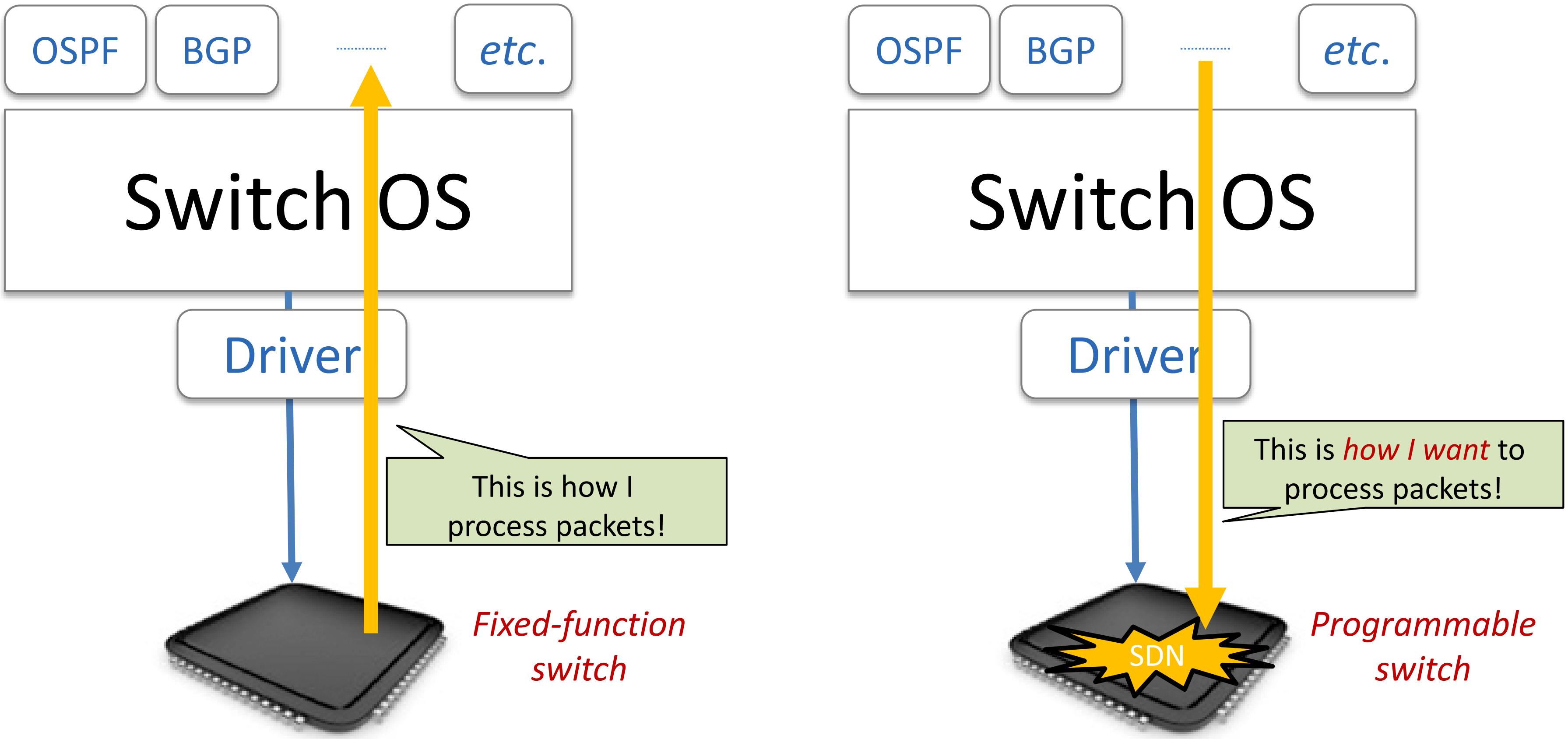
We don't have that!

Programmable Networks



Traditionally: features defined *by chip designers*, defines what can be done.

Programmable Networks



Traditionally: features defined *by chip designers*, defines what can be done.

Future? Features defined *by operator*, tells switch what we really want!

Networking is Catching Up: Happening in Other Domains

Domain specific processors are a trend:

Computers	Graphics	Signal Processing	Machine Learning	Networking
Java	OpenCL	Java	TensorFlow	P4
Compiler	Compiler	Compiler	Compiler	Compiler

CPU

GPU

DSP

TPU

PISA/Tofino

What About Performance?

- Are programmable switches not much *slower* than fixed-function switches?
  - And *cost* more and consume more *power*?
- As data models, ASIC technology etc. are evolving: no!
- Tofino chip: operates at **6.5 Tb/s** (fastest in world!)
  - Can switch entire Netflix catalogue in ***20sec***
  - While running a ***4000 line program*** on any packet...
  - ... and not being more costly or consume more power

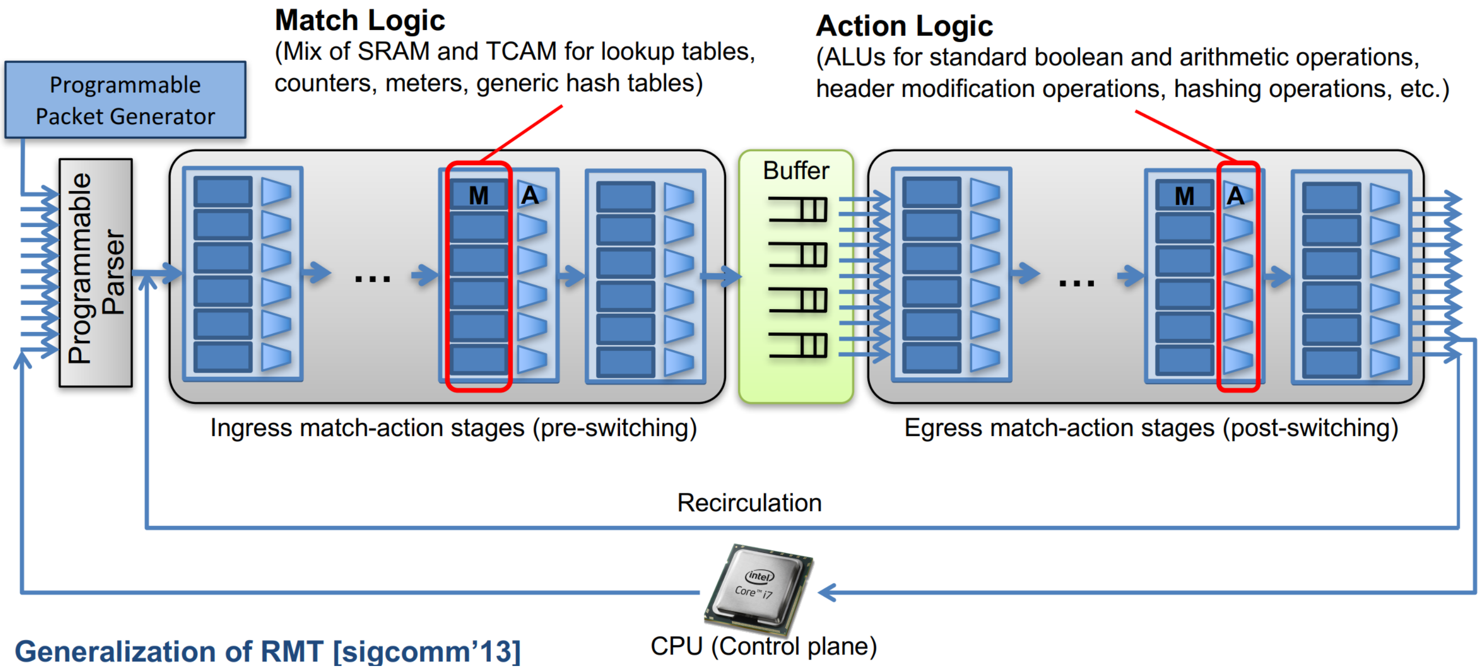
What About Performance?

- Are programmable switches not much *slower* than fixed-function switches?
  - And *cost* more and consume more *power*?
- As data models, ASIC technology etc. are evolving: no!
- Tofino chip: operates at **6.5 Tb/s** (fastest in world!)
  - Can switch entire Netflix catalogue in ***20sec***
  - While running a ***4000 line program*** on any packet...
  - ... and not being more costly or consume more power

Another Takeaway

Programmable networks can enable faster *innovation* without decreasing performance or increasing cost.

The Protocol Independent Switch Architecture (PISA)



Generalization of RMT [sigcomm'13]

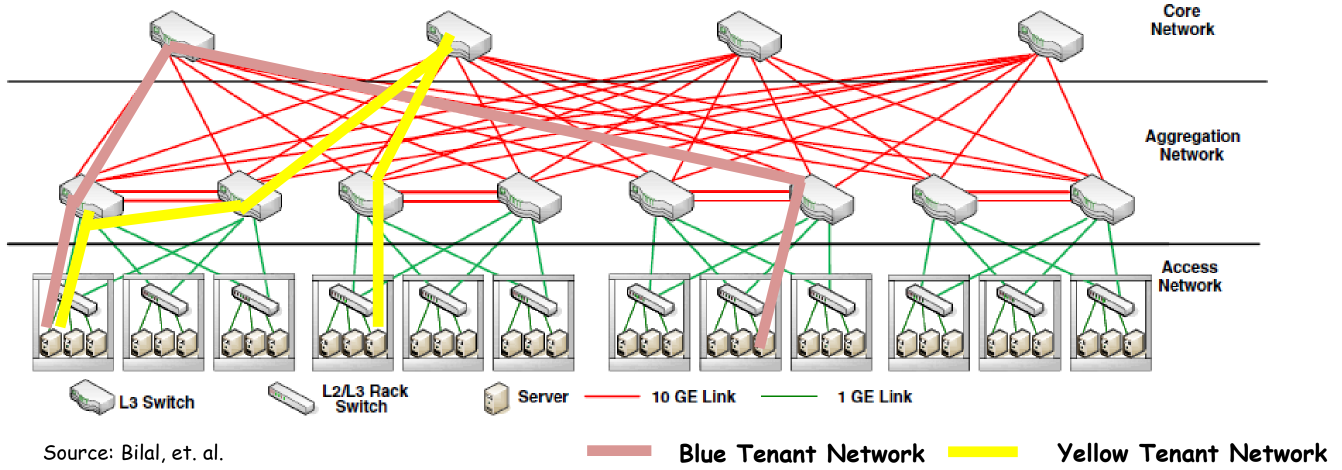
Roadmap

- Software-defined networks
- Programmable dataplanes
- **Network virtualization**

Network Virtualization:
A Killer Application for SDN

Virtual Networks through Overlays

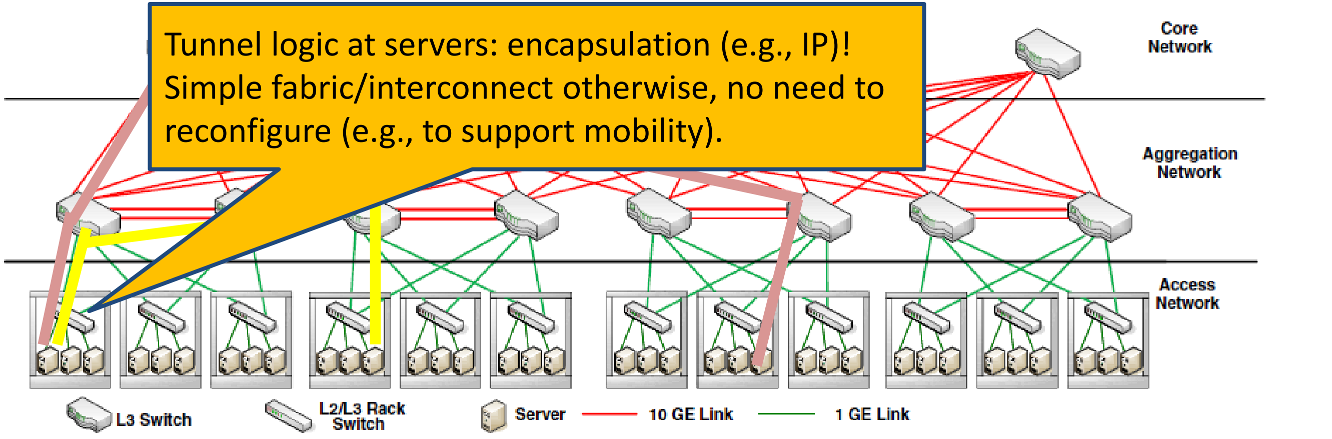
- Recall basic idea of an overlay:
  - Tunnel (e.g., using IP) tenant packets through underlying physical Ethernet or IP network
  - Overlay forms a conceptually separate network providing a separate service from underlay
- L2 service like VPLS or EVPN
  - Overlay spans a separate broadcast domain
- L3 service like IP VPNs
  - Different tenant networks have separate IP address spaces
- Dynamically provision and remove overlay as tenants need network service
- Multiple tenants with separate networks on the same server



Source: Bilal, et. al.

Virtual Networks through Overlays

- Recall basic idea of an overlay:
  - Tunnel (e.g., using IP) tenant packets through underlying physical Ethernet or IP network
  - Overlay forms a conceptually separate network providing a separate service from underlay
- L2 service like VPLS or EVPN
  - Overlay spans a separate broadcast domain
- L3 service like IP VPNs
  - Different tenant networks have separate IP address spaces
- Dynamically provision and remove overlay as tenants need network service
- Multiple tenants with separate networks on the same server



Source: Bilal, et. al.

Blue Tenant Network   Yellow Tenant Network

Advantages of Overlays

- Overlays can potentially support large numbers of tenant networks
- Virtual network state and end node reachability are *handled in the end nodes* (the servers, “fabric”)
- Tenant addresses hidden from other tenants
  - Multiple tenants with the same IP address space
- Addresses in underlay are hidden from the tenant
  - Inhibits unauthorized tenants from accessing data center infrastructure
- Tunneling is used to aggregate traffic

Challenges of Overlays

- Efficient *multicast* is challenging
- Management tools to *co-ordinate overlay and underlay* and performance
  - Overlay networks probe for bandwidth and packet loss, which can lead to inaccurate information
  - Lack of communication between overlay and underlay can lead to inefficient usage of network resources
  - Lack of communication between overlays can lead to contention and other performance issues
- Overlay packets may fail to traverse *firewalls*
- Path MTU limit may cause fragmentation
- …

VxLAN: Virtual eXtensible Local Area Network

- Virtual Extensible LAN (VXLAN) is an evolution of efforts to standardize on an ***overlay encapsulation protocol***, increasing scalability up to 16 million logical networks

- Concretely: ***VLAN-like*** encapsulation technique to encapsulate MAC-based Layer-2 frames with Layer-4 UDP

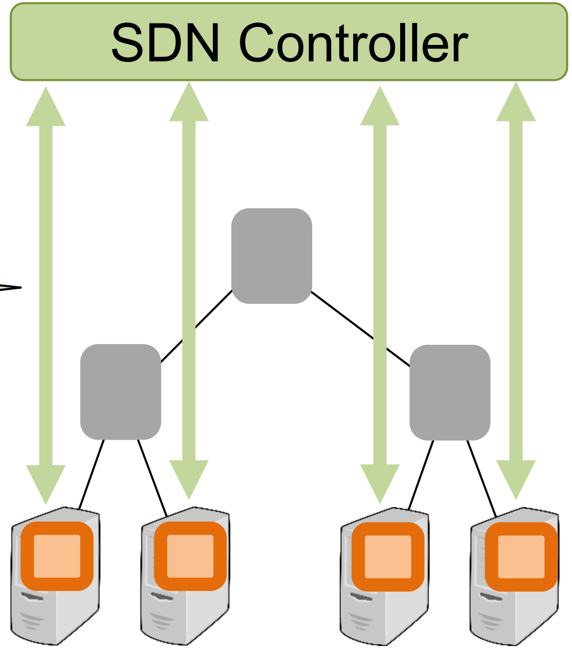
- VxLAN segments constitute a ***broadcast domain***

- VxLAN endpoints terminate tunnels and may be both virtual or physical switch ports
  - E.g., Open Vswitch (OVS)

Another Trend: Virtualization of Switches

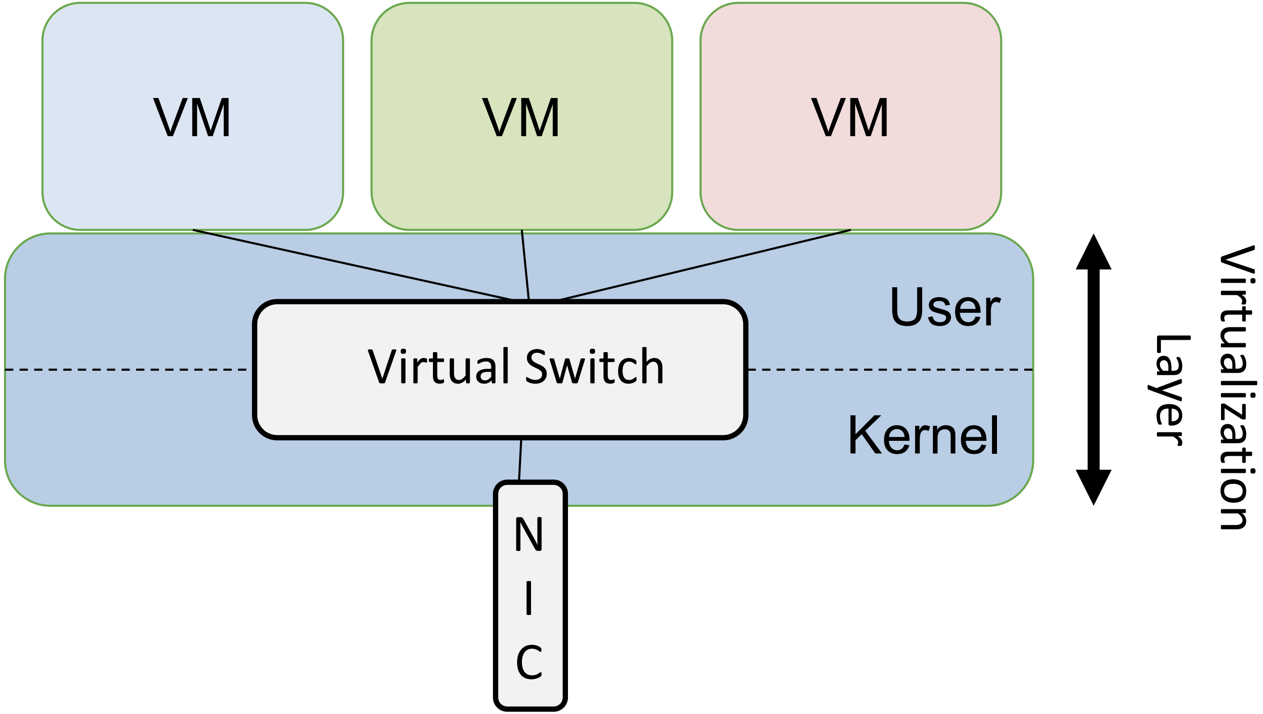
Trend in Datacenter Networks: Virtual Switches

Simple in core: hardware switches.



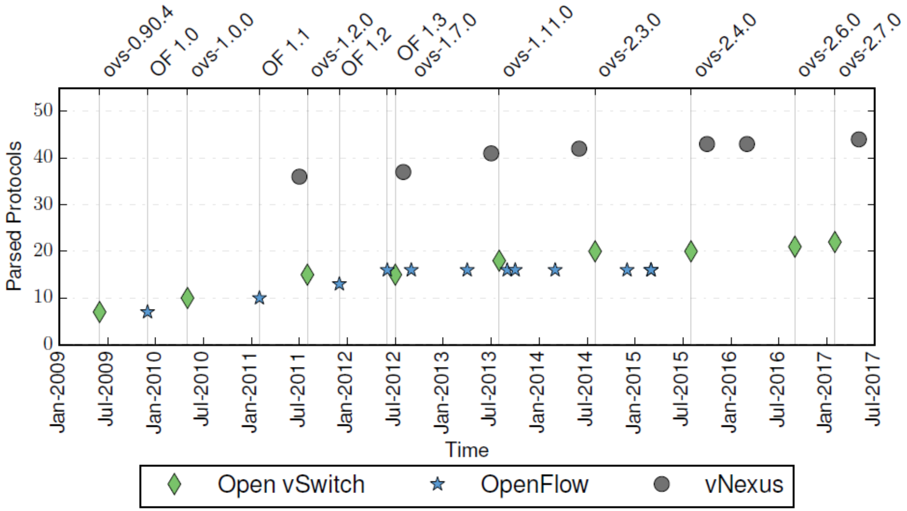
Clever at edge: virtualized and Programmable.

Another New Vulnerability: Virtual Switch



Virtual switches reside in the **server's virtualization layer** (e.g., Xen's Dom0). Goal: provide connectivity and isolation.

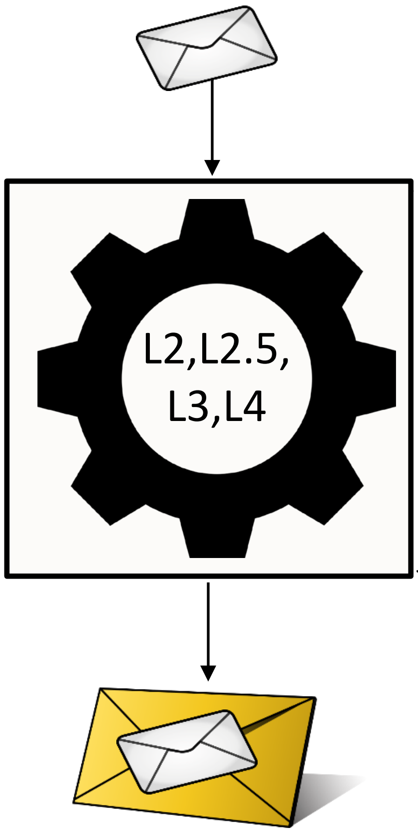
The Underlying Problem: Complexity

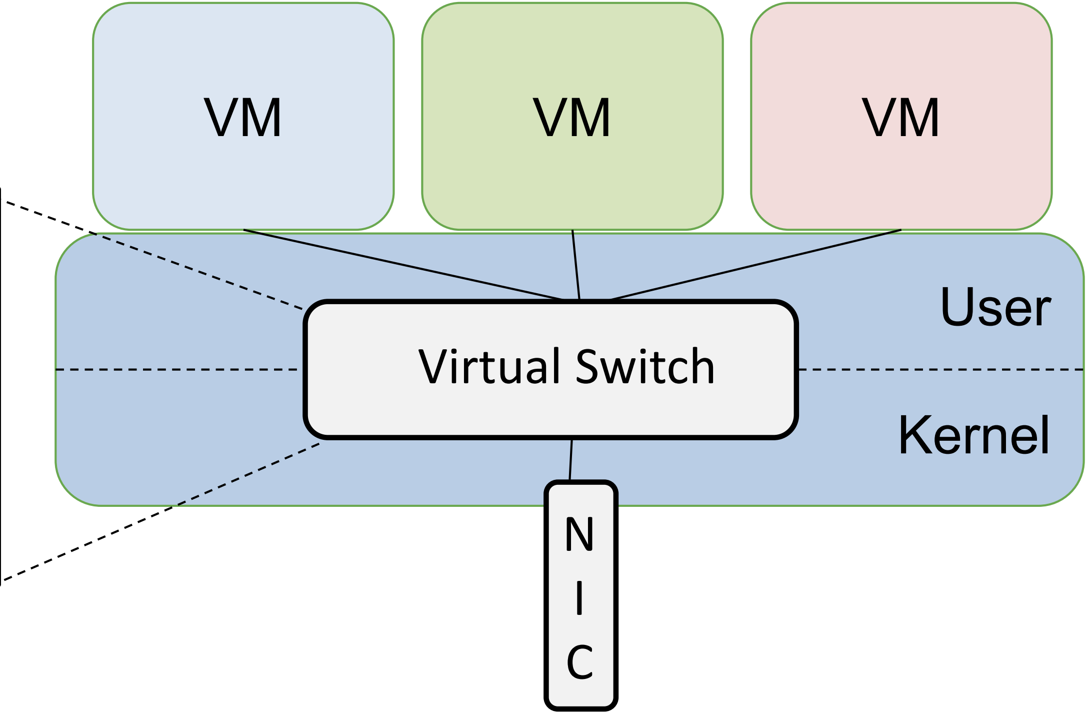


Number of parsed high-level protocols constantly increases…

Complexity: Parsing

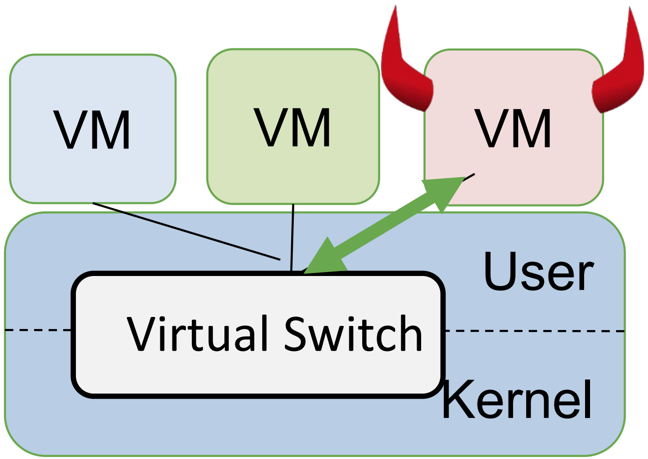
Ethernet
LLC
VLAN
MPLS
IPv4
ICMPv4
TCP
UDP
ARP
SCTP
IPv6
ICMPv6
IPv6 ND
GRE
LISP
VXLAN
PBB
IPv6 EXT HDR
TUNNEL-ID
IPv6 ND
IPv6 EXT HDR
IPv6HOPOPTS
IPv6ROUTING
IPv6Fragment
IPv6DESTOPT
IPv6ESP
IPv6 AH
RARP
IGMP



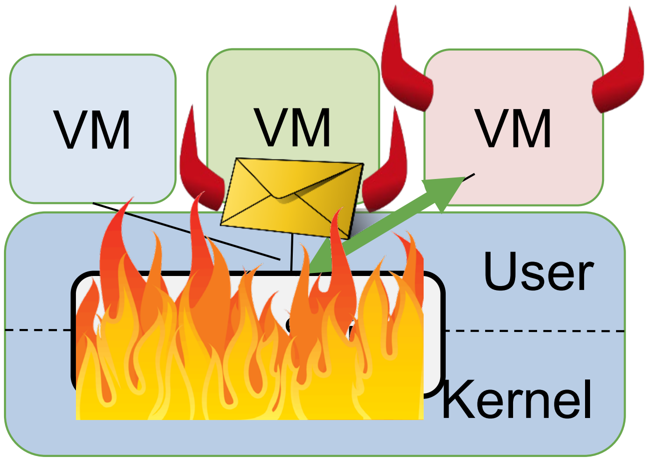


Parser directly faces attacker and vSwitch runs with high security privileges.

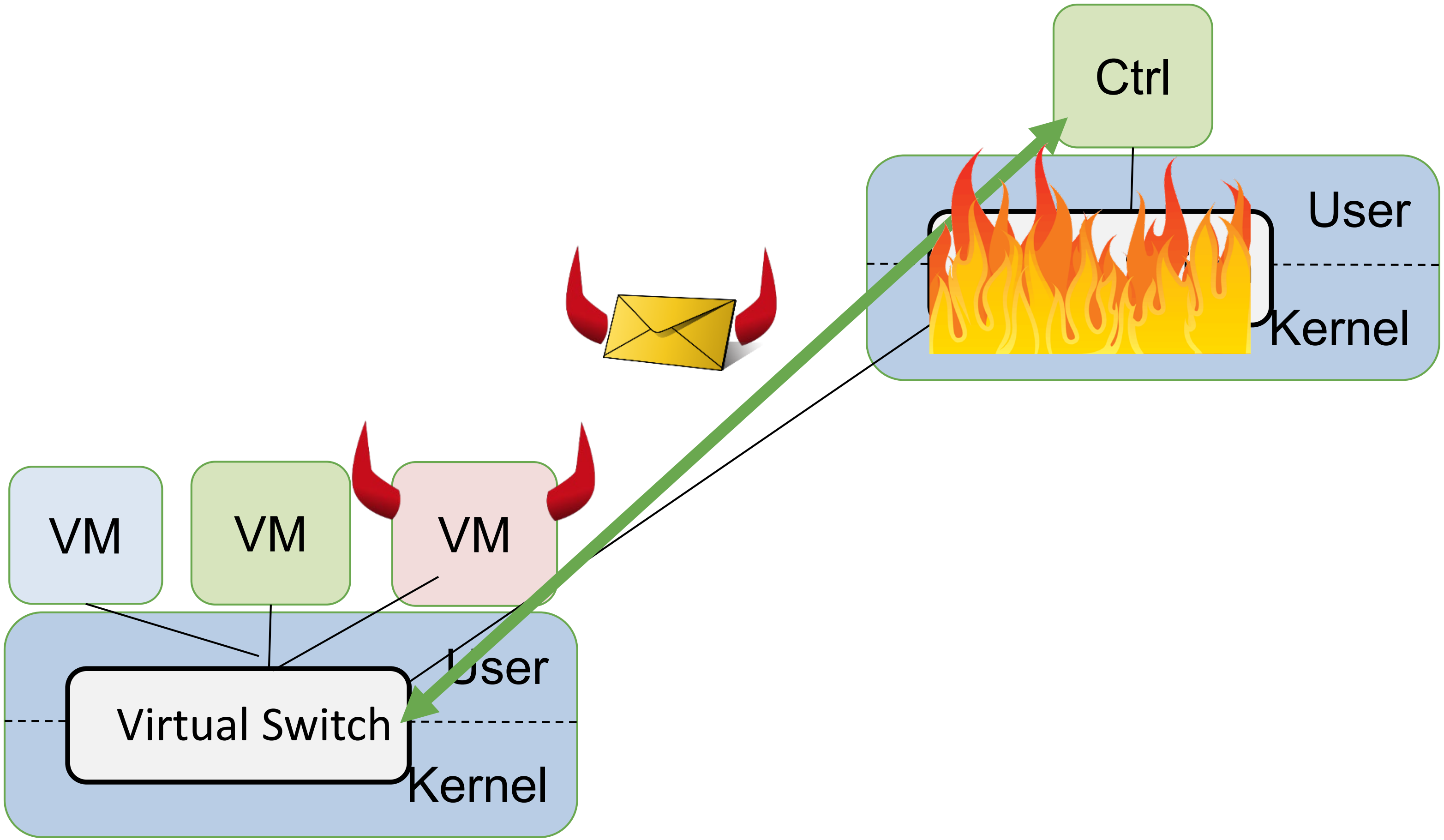
Enables Very Low-Cost Attacks



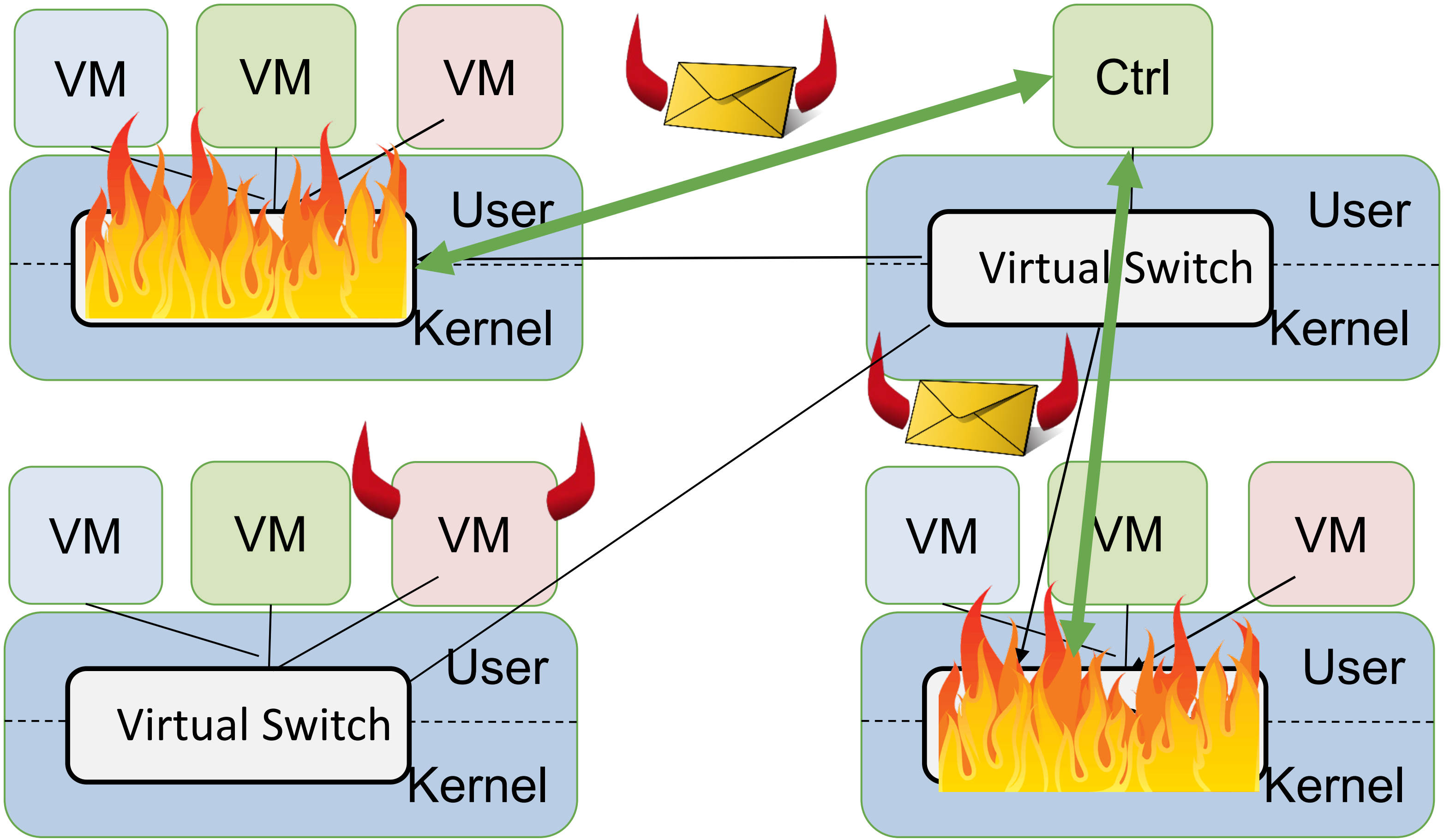
Enables Very Low-Cost Attacks



Enables Very Low-Cost Attacks



Enables Very Low-Cost Attacks

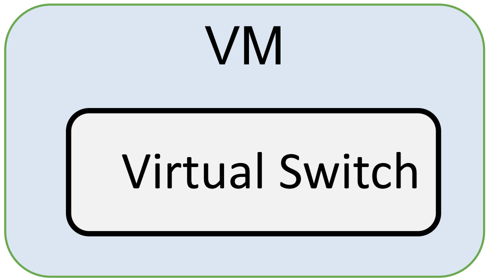


Further Reading

Taking Control of SDN-based Cloud Systems via the Data Plane (Best Paper Award)
Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid.
ACM Symposium on SDN Research (SOSR), Los Angeles, California, USA, March 2018.

Challenge: How to provide better isolation *efficiently*?

- Idea for better *isolation*: put vSwitch in a VM
- But what about *performance*?
- Or container?

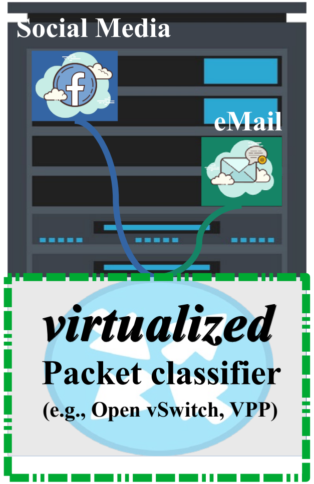


MTS: Bringing Multi-Tenancy to Virtual Switches
Kashyap Thimmaraju, Saad Hermak, Gabor Retvari, and S. USENIX ATC, 2019.

Another Threat:
Algorithmic Complexity Attacks

Algorithmic Complexity Attacks

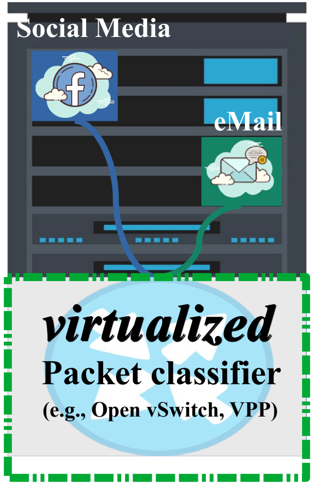
- Network dataplane runs many **complex algorithms**: may perform poorly under specific or *adversarial inputs*
- E.g., packet classifier: runs **Tuple Space Search** algorithm (e.g., in OVS)
- Can be exploited: adversary can *degrade performance* to ~10% of the baseline (10 Gbps) with only <1 Mbps (!) attack traffic
- Idea:
  - Tenants can use the Cloud Management System (CMS) to set up their **ACLs** to access-control, redirect, log, etc.
  - Attacker's goal: send some *packet towards the virtual switch* that when subjected to the ACLs will *exhaust resources*



Tuple Space Explosion: A Denial-of-Service Attack Against a Software Packet Classifier. Levente Csikor et al. ACM CoNEXT, 2019.

Algorithmic Complexity Attacks

- Network dataplane runs many **complex algorithms**: may perform poorly under specific or *adversarial inputs*
- E.g., packet classifier: runs **Tuple Space Search** algorithm (e.g., in OVS)
- Can be exploited: adversary can *degrade performance* to ~10% of the baseline (10 Gbps) with only <1 Mbps (!) attack traffic
- Idea:
  - Tenants can use the Cloud Management System (CMS) to set up their **ACLs** to access-control, redirect, log, etc.
  - Attacker's goal: send some *packet towards the virtual switch* that when subjected to the ACLs will *exhaust resources*



How to find such attacks?!

Tuple Space Explosion: A Denial-of-Service Attack Against a Software Packet Classifier. Levente Csikor et al. ACM CoNEXT, 2019.

Conclusion

- Why networks need more innovation
- Programmable control and data planes
- Network virtualization and datacenters

Further Reading

Toward Active and Passive Confidentiality Attacks On Cryptocurrency Off-Chain Networks
Utz Nisslmueller, Klaus-Tycho Foerster, Stefan Schmid, and Christian Decker.
6th International Conference on Information Systems Security and Privacy (**ICISSP**), Valletta, Malta, February 2020.

NetBOA: Self-Driving Network Benchmarking
Johannes Zerwas, Patrick Kalmbach, Laurenz Henkel, Gabor Retvari, Wolfgang Kellerer, Andreas Blenk, and Stefan Schmid.
ACM SIGCOMM Workshop on Network Meets AI & ML (**NetAI**), Beijing, China, August 2019.

MTS: Bringing Multi-Tenancy to Virtual Switches
Kashyap Thimmaraju, Saad Hermak, Gabor Retvari, and Stefan Schmid.
USENIX Annual Technical Conference (**ATC**), Renton, Washington, USA, July 2019.

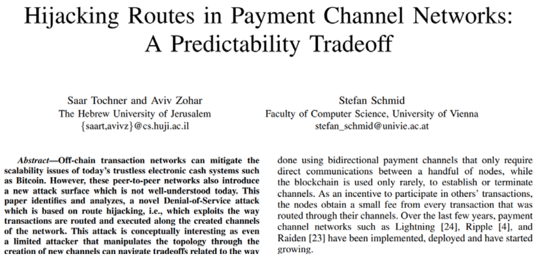
Taking Control of SDN-based Cloud Systems via the Data Plane (Best Paper Award)
Kashyap Thimmaraju, Bhargava Shastry, Tobias Fiebig, Felicitas Hetzelt, Jean-Pierre Seifert, Anja Feldmann, and Stefan Schmid.
ACM Symposium on SDN Research (**SOSR**), Los Angeles, California, USA, March 2018.

Outsmarting Network Security with SDN Teleportation
Kashyap Thimmaraju, Liron Schiff, and Stefan Schmid.
2nd IEEE European Symposium on Security and Privacy (**EuroS&P**), Paris, France, April 2017.

Preacher: Network Policy Checker for Adversarial Environments
Kashyap Thimmaraju, Liron Schiff, and Stefan Schmid.
38th International Symposium on Reliable Distributed Systems (**SRDS**), Lyon, France, October 2019.

P-Rex: Fast Verification of MPLS Networks with Multiple Link Failures
Jesper Stenbjerg Jensen, Troels Beck Krogh, Jonas Sand Madsen, Stefan Schmid, Jiri Srba, and Marc Tom Thorgersen.
14th International Conference on emerging Networking EXperiments and Technologies (**CoNEXT**), Heraklion, Greece, December 2018.

And



Hijacking Routes in Payment Channel Networks: A Predictability Tradeoff

Saar Tochner and Aviv Zohar
The Hebrew University of Jerusalem
{saart,avivz}@cs.huji.ac.il

Stefan Schmid
Faculty of Computer Science, University of Vienna
stefan_schmid@univie.ac.at

Abstract—**Off-chain transaction networks can mitigate the scalability issues of today's trustless electronic cash systems such as Bitcoin. However, these peer-to-peer networks also introduce a new attack surface which is not well-understood today. This paper identifies and analyzes, a novel Denial-of-Service attack which is based on route hijacking, i.e., which exploits the way transactions are routed and executed along the created channels of the network. This attack is conceptually interesting as even a limited attacker that manipulates the topology through the creation of new channels can navigate tradeoffs related to the way**

done using bidirectional payment channels that only require direct communications between a handful of nodes, while the blockchain is used only rarely, to establish or terminate channels. As an incentive to participate in others' transactions, the nodes obtain a small fee from every transaction that was routed through their channels. Over the last few years, payment channel networks such as Lightning [24], Ripple [4], and Raiden [23] have been implemented, deployed and have started growing.